Payment Card Industry (PCI)

Contactless Payments on COTS (CPoC™)

Security and Test Requirements

Version 1.0

December 2019

Document Changes

Date	Version	Description
December 2019	1.0	Initial Publication

Table of Contents

Introduction

Purpose

The purpose of Contactless Payments on COTS (CPoC™) Security and Test Requirements (hereinafter referred to as the "CPoC Standard"), is to provide a set of principles and requirements for a mobile payment-contactless acceptance solution, referred to as "CPoC solution" or the "solution," where the contactless read functions are performed using the NFC interface that is native to and embedded in a commercial off-the-shelf (COTS) device (e.g., smartphone or tablet) and an associated secure contactless payment-acceptance application on the same COTS device. Software-based PIN entry is not permitted in CPoC solution.

> ***Note:*** *This standard does not supersede other PCI standards, nor do these requirements constitute a recommendation from the Council or obligate merchants, service providers, or financial institutions to purchase or deploy such solutions. As with all other PCI standards, any mandates, regulations, or rules regarding these requirements are provided by the participating payment brands.*

Such solutions are intended for a merchant-attended environment, and address only contactless chip-based transactions that support dynamic transaction data and are processed online. Offline payment transactions, such as EMV offline transaction authorizations or deferred authorizations, are prohibited. Other uses may introduce risks that are out of scope for this Standard.

The security requirements described in this document provide a security framework to protect the confidentiality and integrity of sensitive payment information captured and processed in CPoC solutions. The test requirements outlined in this document provide greater detail and visibility into the testing processes performed by the evaluation laboratories that will perform the validation testing of the solutions.

Audience

The security requirements and test requirements outlined in this document apply to organizations developing, managing, or deploying CPoC solutions, evaluator labs, and assessors.

Usage Conventions

This document has been prepared with certain conventions. Within this document, the following terms have a specific meaning when used:

- Must: Defines a mandatory requirement.
- Should: Defines a recommendation.

Glossary

In addition to terms defined in the PCI DSS *Glossary, Abbreviations and Acronyms*[1], the following terms are used throughout this document. Throughout this document, terms defined in this Glossary are shown in green.

Term	Definition
Account data	Account data consists of cardholder data and/or sensitive authentication data. See cardholder data and sensitive authentication data.
AES	Abbreviation for "Advanced Encryption Standard." Block cipher used in Symmetric Key cryptography adopted by NIST in November 2001 as *FIPS PUB 197* (or *FIPS 197*).
Asymmetric encryption	Also known as public key cryptography, asymmetric cryptosystems are based on the intractability of certain mathematical problems. A cryptographic technique that uses two related transformations, a public transformation (defined by the public key) and a private transformation (defined by the private key). The two transformations have the property that, given the public transformation, it is not computationally feasible to derive the private transformation.
Attestation	The act of attestation in this standard is the interaction between a verifier (possibly server-based) and a prover (possibly client-based) to determine the current security state/behavior of the prover based on predefined measurements and thresholds provided by the prover.
Attestation component	An element of the solution that performs attestation processing.
Attestation system	The set of components that perform attestation processing for the solution. The implementation may be shared across different execution environments, which provides a level of validation and assurance of the execution environment in which the CPoC application executes, providing a level of software-based tamper detection and response. Its components include the CPoC application attestation component and the back-end attestation component—the latter works in close association with the back-end monitoring system.
Authentication	Process of verifying identity of an individual, device, or process. Authentication typically occurs through the use of one or more authentication factors such as: Something you know, such as a password or passphrase Something you have, such as a token device or smart card Something you are, such as a biometric
Back-end systems	The set of systems providing the server-side functionality of the solution. These functionalities include monitoring, attestation, and transaction processing. In addition, the back-end systems include the IT environments necessary to support the functionalities of the solution.
Cardholder	Non-consumer or consumer customer to whom a payment card is issued to or any individual authorized to use the payment card.

[1] https://www.pcisecuritystandards.org/pci_security/glossary

Term	Definition
Cardholder data	At a minimum, cardholder data consists of the full PAN. Cardholder data may also appear in the form of the full PAN plus any of the following: cardholder name, expiration date, and/or service code. See sensitive authentication data for additional data elements that may be transmitted or processed (but not stored) as part of a payment transaction.
Cardholder Data Environment (CDE)	The people, processes and technology that store, process or transmit cardholder data or sensitive authentication data.
Cleartext	Intelligible data that has meaning and can be read or acted upon without the application of decryption. Also known as plaintext.
Commercial off-the-shelf (COTS) device	A mobile device (e.g., smartphone or tablet) that is designed for mass-market distribution.
Compiling	Translation of computer code from one format into another format. Usually used to transform human-readable “source” code into a format that can be executed by a specific platform or execution environment.
Consumer	Individual purchasing goods, services, or both.
Contactless kernel	A software that processes contactless transactions. The kernel is selected by the CPoC application based on the characteristics of the transaction and the consumer device (e.g., credit card) supporting the contactless transactions. The kernel contains interface routines, security and control functions, and logic to manage a set of commands and responses to retrieve the necessary data from the consumer payment device to complete a transaction.[2]
Contactless magnetic-stripe mode	A mode of contactless payments based on Track 1 and/or Track 2 data obtained from the contactless interface. This data includes a dynamic card verification code/value for transactional security purposes. Also, referred to as contactless mag-stripe or magnetic stripe data.
Contactless Payments on COTS (CPoC) API	An optional software component, developed and provided by the solution provider, to allow third-party developers to interface with the CPoC solution.
Contactless Payments on COTS (CPoC) application	All parts of the code, regardless of the execution environment, that is installed and executed on the merchant COTS device for the purposes of accepting and processing account data associated with a contactless transaction. The CPoC API, attestation component, and/or a payment application may be incorporated into the CPoC application or may be separate.
Contactless Payments on COTS (CPoC) solution	The set of components and processes that supports the contactless read and protection of account data into a COTS device. At a minimum, the solution includes the CPoC application, attestation system, and the back-end systems and environments that perform attestation, monitoring, and payment processing.
Correlatable data	In the context of this standard, this is data that would facilitate the correlation of data including PAN and other account data with a separate transaction or database that contains account data such that interception of this data and other transactional data could reasonably lead to the association of such data with full track magnetic stripe data with or without PIN.
COTS platform	The hardware and operating system of the COTS device.

[2] EMVCo (https://www.emvco.com/)

Term	Definition
COTS system baseline	A measurable configuration reference point of the COTS device attributes and COTS OS, on which the CPoC application may be executed. The COTS system baseline is used for periodic comparative analysis by the back-end attestation system to determine changes that would impact the overall security of the COTS device to continue to process transactions.
Cryptographic material	All materials involved in the implementation of a cryptographic algorithm or process including keys, entropy seeds, nonces, and lookup tables involved in the execution of the algorithm, etc.
Deterministic Random Number Generator (DRNG)	A deterministic algorithm to generate a sequence of numbers with little or no discernible pattern in the numbers, except for broad statistical properties. Also referred to as Pseudo Random Number Generator (PRNG). See Random Number Generator (RNG).
Dual control	A process of using two or more separate entities (usually persons), operating in concert, to protect sensitive functions or information. Each entity is equally responsible for the physical protection of materials involved in vulnerable processes. No single person is be able to access or to use the materials (e.g., cryptographic key). For manual key generation, conveyance, loading, storage and retrieval, dual control requires split knowledge of the key among the entities. No single person can gain control of a protected item or process. See split knowledge.
Elliptic Curve Cryptography (ECC)	Approach to public-key cryptography based on elliptic curves over finite fields.
EMV®	A payment standard that implements cryptographic authentication, published by EMVCo.
EMVCo	A privately-owned corporation. The current members of EMVCo are JCB International, American Express, Mastercard, China UnionPay, Discover Financial and Visa Inc.
Encryption	Process of converting information into an unintelligible form except to holders of a specific cryptographic key. Use of encryption protects information between the encryption process and the decryption process (the inverse of encryption) against unauthorized disclosure.
Environment	The IT environment supporting one or more functionalities of the solution—such as the IT environment hosting the back-end monitoring system.
Execution environment	The set of hardware and software on which a program is executed. This may be provided through hardware alone, include a combination of hardware and software elements, or be virtualized and implemented in software such that the execution environment can be similarly executed on different hardware platforms.
Hash	A (mathematical) function that is a non-secret algorithm, which takes any arbitrary-length message as input and produces a fixed-length hash result.
Hash-based message authentication code (HMAC)	A code that is produced using hash algorithms rather than a symmetric cryptographic algorithm. Defined in *FIPS 198-1.* See Message authentication code (MAC).
Integrity	Ensuring consistency of data; in particular, preventing unauthorized and undetected creation, alteration, or destruction of data.

Term	Definition
Key block	A format for storage and transmission of symmetric cryptographic keys that embeds metadata about the key type and use, as well as providing cryptographic authentication across the encrypted key and this metadata to ensure that the key and its purpose cannot be altered.
Key Check Value (KCV)	A value used to identify a key without directly revealing any bits of the actual key itself.
Key generation	Creation of a cryptographic key either from a random number generator or through a one-way process utilizing another cryptographic key.
Key management	The activities involving the handling of cryptographic keys and other related security parameters (e.g., initialization vectors, counters) during the entire life cycle of the keys, including their generation, storage, distribution, loading and use, deletion, destruction, and archiving.
Key variant	A new key formed by a process (which need not be secret) with the original key, such that one or more of the non-parity bits of the new key differ from the corresponding bits of the original key.
Key verification checks (KVC)	See Key Check Value (KCV).
Key wrapping	A process, by which a cryptographic key is protected in integrity, confidentiality or both, by the generation of a key block to encapsulate (encrypt) the cryptographic key material for transport or storage.
Mandatory access control	Access control by which the operating system constrains the ability of a process or thread to access or perform an operation on objects or targets such as files, directories, TCP/UDP ports, shared memory segments, IO devices, etc., through an authorization rule enforced by the operating system kernel.
Man-in-the-middle (MITM attack) attack	An attack method where a malicious third party interposes between two other communicating parties and modifies the data sent between them.
Message authentication code (MAC)	In cryptography, an acronym for "message authentication code." A small piece of information used to authenticate a message. See Hash-based message authentication code (HMAC).
Mobile device	In the context of this Standard, see COTS device.
M-of-N	An M-of-N scheme is a key component or key share allocation scheme, where m is the number of shares or components necessary to form the key, and n is the number of the total set of shares or components related to the key. Management of the shares or components should be sufficient to ensure that no subgroup of less than m persons can gain access to enough of the components to form the key.
Monitoring system	Monitors and provisions security controls to detect, alert, and mitigate suspected or actual threats and attacks against the CPoC solution.
NFC Interface	The subsystem in the COTS device that is used by the COTS platform to access data, including account data, read from contactless cards or devices. The main physical components are the NFC antenna and the NFC controller.
Nondeterministic Random Number Generator (NRNG)	A random number generator that has access to an entropy source and (when working properly) produces output numbers (or bit strings) that have full entropy. Contrast with a Deterministic Random Number Generator (DRNG).

Term	Definition
Obfuscation	Protection applied to a process or data through increasing the complexity of interpreting that data. Obfuscation methods may include, but are not be limited to, control-flow and data obfuscation, execution of code sections in remote environments, and API renaming.
Offline payment transaction	In an offline EMV transaction, the card and terminal communicate and use issuer-defined risk parameters that are set in the card to determine whether the transaction can be authorized. Offline transactions are used when terminals do not have online connectivity—e.g., at a ticket kiosk—or in countries where telecommunications costs are high.
Operating system (OS)	System software that manages the underlying hardware and software resources and provides common services for programs. Common operating systems in a COTS environment include, but are not limited to, Android and iOS implementations.
OS store	A digital distribution service operated by the COTS OS vendor or by the COTS device manufacturer.
PCI DSS	The Data Security Standard published and maintained by the Payment Card Industry Security Standards Council. PCI DSS provides a baseline of technical and operational requirements designed to protect account data.
Perfect forward secrecy	Also known as "Forward Secrecy." A protocol has Perfect Forward Secrecy if a compromise of long-term keys does not also compromise past session keys.
Physically Unclonable Function (PUF)	Also known as "Physical Unclonable Function." An intrinsic value or transformation that can be provided by a system that is a function of some physical process, such that it cannot be replicated or altered.
Private key	A cryptographic key used with a public-key cryptographic algorithm that is uniquely associated with an entity and is not made public. In the case of an asymmetric signature system, the private key defines the signature transformation. In the case of an asymmetric encipherment system, the private key defines the decipherment transformation.
Public key	A cryptographic key used with a public-key cryptographic algorithm that is uniquely associated with an entity and may be made public. In the case of an asymmetric signature system, the public key defines the verification transformation. In the case of an asymmetric encipherment system, the public key defines the encipherment transformation. A key that is "publicly known" is not necessarily globally available. The key may only be available to all members of a pre-specified group.
Public key cryptography	See asymmetric encryption.
Random Number Generator (RNG)	The process of generating values with a high level of entropy and that satisfy various qualifications, using cryptographic and hardware-based "noise" mechanisms. This results in a value in a set that has equal probability of being selected from the total population of possibilities, hence unpredictable.
Replay attack	A replay attack (also known as playback attack) is a form of network attack in which a valid data transmission is maliciously or fraudulently repeated or delayed.
Rich execution environment (REE)	Refers to an execution environment where COTS device resources are shared by OS applications.

Term	Definition
RSA	Algorithm for public-key encryption described in 1977 by Ron Rivest, Adi Shamir, and Len Adleman at Massachusetts Institute of Technology (MIT); letters RSA are the initials of their surnames.
Secure boot	See trusted boot.
Secure channel	A cryptographically protected connection between two points.
Secure cryptographic device (SCD)	A physically and logically protected hardware device that provides a secure set of cryptographic services. It includes the set of hardware, firmware, software, or some combination thereof that implements cryptographic logic, cryptographic processes, or both, including cryptographic algorithms. Examples include *ANSI X9.24* part 1 and *ISO 13491*.
Secure Element (SE)	A tamper-resistant platform (typically a one-chip secure microcontroller) capable of securely hosting applications and their confidential and cryptographic data (e.g., key management).
Secure reading and exchange of data (SRED)	Module 4 of the PCI PTS POI Standard, detailing the requirements for devices that protect account data.
Security processor	Within a COTS device, a security processor is a separate processor or co-processor with its own dedicated memory running separate operating system, applications and data on these processors are not accessible by the COTS device's main operating system.
Sensitive authentication data	Security-related information (including but not limited to card validation codes/values, full track data (from the magnetic stripe or equivalent on a chip), PINs, and PIN blocks) used to authenticate cardholders and/or authorize payment card transactions.
Sensitive data	For the purposes of this standard, sensitive data is cryptographic materials—e.g., keys, certificates, or account data.
Sensitive services	A sensitive service is any service that may affect the security of the overall system, as well as those functions that affect underlying processes that support the protection of sensitive data—e.g., cryptographic keys and account data. Common examples are key management, modification or update of attestation services, or remote component of contactless kernels (or other remote processing components) and cryptographic signing of assets to allow their authenticity to be verified.
Software protection mechanisms	Methods and implementations used to prevent the reverse-engineering and modification of software, including, but not limited to, hooking, rooting, emulation or debugging detection, verification and validation of software.
Solution	See Contactless Payments on COTS (CPoC) solution.
Solution provider	An entity that develops, manages, and/or deploys CPoC solutions.
Split knowledge	A condition under which two or more entities separately have key components or key shares that individually convey no knowledge of the resultant cryptographic key. The information needed to perform a process such as key formation is split among two or more people. No individual has enough information to gain knowledge of any part of the actual key that is formed.
Symmetric key	Same symmetric key that is used for encryption is also used for decryption. Also known as "secret key."
Tamper-detection	The automatic determination by a cryptographic module that an attempt has been made to compromise the security of the module.

Term	Definition
Tamper-resistant	A characteristic that provides passive physical protection against an attack.
Tamper-responsive	A characteristic that provides an active response to the detection of an attack, thereby preventing a successful attack.
Third-party app stores	App stores that are not supported by the COTS OS vendor and are not pre-installed by the device manufacturer. See OS store.
Trusted boot	Also known as Verified Boot and secure boot. A cryptographic process where the bootloader verifies the integrity of all components (e.g., kernel objects) loaded during operating system startup process, prior to loading.
Trusted Execution Environment (TEE)	A trusted execution environment provides hardware-based security features such as isolated execution environment for Trusted Applications. It protects security assets from general software attacks, defines safeguards as to data and functions that a program can access, and resists a set of defined threats.
User interface (UI)	The set of the human-machine interfaces that allows for interaction between a person and a computerized system.
White-box cryptography	A method used to obfuscate a (mostly symmetric) cryptographic algorithm and key with the goal of making determination of the key value computationally complex.

Publications and References

PCI SSC Standards[3]	
DSS	Data Security Standard
DESV	Designated Entities Supplemental Validation (Appendix A3 in PCI DSS v3.2)
HSM	PIN Transaction Security (PTS) Hardware Security Module Security Requirements
PA-DSS	Payment Application Data Security Standard
POI	PTS Point of Interaction Module Security Requirements
SSF	Software Security Framework

Other Industry Security Resources	
AIS 31	Physical Random Number Generator exploiting quantum physics
ANSI X9.24	Retail Financial Services Key Management
ANSI X9.79-4	Public Key Infrastructure (PKI) - Part 4: Asymmetric Key Management
ANSI X9.80	Prime Number Generation, Primality Testing, and Primality Certificates
ANSI X9.102	Symmetric Key Cryptography for the Financial Services Industry - Wrapping of Keys And Associated Data
ANSI X9.111	Penetration Testing within the Financial Services Industry
CERTSECCODE	SEI CERT Coding Standards – https://www.securecoding.cert.org
Control Objectives for Information and Related Technology (COBIT)	Information Systems Audit and Control Association framework for IT governance and management
Define, Measure, Analyze, Improve and Control (DMAIC)	Data-driven improvement cycle used for improving, optimizing, and stabilizing business processes and designs
Deming cycle, Plan-Do-Check-Act (PDCA)	Continuous quality-improvement model consisting of a logical sequence of four repetitive steps for continuous improvement and learning: Plan, Do, Check (Study), and Act
FIPS 140-2	Federal Information Processing Standard, Security Requirements for Cryptographic Modules
FIPS 140-3	Security Requirements for Cryptographic Modules
FIPS 186-4	Federal Information Processing Standard, Digital Signature Standard (DSS)
FIPS 198-1	Federal Information Processing Standard, The Keyed-Hash Message Authentication Code (HMAC)

[3] https://www.pcisecuritystandards.org/document_library

Other Industry Security Resources	
FIPS PUB 197 (or "FIPS 197")	Federal Information Processing Standards Publication 197 ADVANCED ENCRYPTION STANDARD (AES)
ISECOM	Institute for Security and Open Methodologies
ISO 11568	Financial Services, Key Management
ISO 13491	Financial Services, Secure Cryptographic Devices
ISO 16609	Banking Requirements for message authentication using symmetric techniques
ISO 20038	Banking and Related Financial Services - Key Wrap Using AES
ISO 27001	Requirements for an Information Security Management System
ISO 97971	Information technology – Security techniques – Message Authentication Codes (MACs)
ISO/IEC 11770	Information technology -- Security techniques -- Key management
ISO/IEC 18031	Information technology -- Security techniques -- Random bit generation
ISO/IEC 18032	Information technology — Security techniques — Prime number generation
ISO/IEC 18033-3	Information technology – Security techniques – Encryption algorithms – Part 3: Block ciphers
ISO/IEC 21827	Federal Information Processing Standard, Advanced Encryption Standard
ISO/IEC 27034	Information Technology - Security Techniques - Application Security
ISSEA SSE-CMM	International Systems Security Engineering Association Systems Security Engineering Capability Maturity Model
ECSG SCS Volume	European Cards Stakeholders Group (ECSG) SEPA Cards Standardisation Volume
EMVCo SBMP	EMVCo Software-Based Mobile Payments Security Requirements
MITRE's Common Vulnerabilities and Exposures list	MITRE Common Vulnerabilities and Exposures database
NIST SP 800-22 *NIST SP 800-22 revision 1a*	National Institute of Standards and Technology, Special Report on A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications
NIST SP 800-38B	National Institute of Standards and Technology, Special Report on Recommendation for Block Cipher Modes of Operation: the CMAC Mode for Authentication
NIST SP 800-38F	Recommendation for Block Cipher Modes of Operation: Methods for Key Wrapping
NIST SP 800-52	National Institute of Standards and Technology, Special Report on Guidelines for the Selection, Configuration, and Use of Transport Layer Security (TLS) Implementations

Other Industry Security Resources	
NIST SP 800-56A	Recommendation for Pair-Wise Key Establishment Schemes Using Discrete Logarithm Cryptography
NIST SP 800-56B	Recommendation for Pair-Wise Key-Establishment Using Integer Factorization Cryptography
NIST SP 800-57	National Institute of Standards and Technology, Special Report on Recommendation for Key Management, Part 1: General (Revision 3)
NIST SP 800-90A	Recommendation for Random Number Generation Using Deterministic Random Bit Generators
NIST SP 800-90B	Recommendation for the Entropy Sources Used for Random Bit Generation
NIST SP 800-90C	Recommendation for Random Bit Generator (RBG) Constructions
NIST SP 800-115	Technical Guide to Information Security Testing and Assessment
NIST Special Publication 800-57pt2-r1	National Institute of Standards and Technology, Special Report on Framework for Designing Cryptographic Key Management Systems
NIST Special Publication 800-64	Security Considerations in the System Development Life Cycle
NIST Special Publication 800-130	National Institute of Standards and Technology, Special Report on Framework for Designing Cryptographic Key Management Systems
NIST Special Publication 800-90A	National Institute of Standards and Technology, Special Report on Recommendation for Random Number Generation Using Deterministic Random Bit Generators
NIST's National Vulnerability Database	National Institute of Standards and Technology National Vulnerability Database
Open Source Security Testing Methodology Manual (OSSTMM)	http://www.isecom.org/research/osstmm.html
OWASPMOB10	*OWASP Mobile Security Project – Top Ten Mobile Risks* https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks
OWASPMOB2015	*OWASP Mobile Security Project – 2015 Scratchpad* https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-2015_Scratchpad
SCSECSWDEV	*The Ten Best Practices for Secure Software Development* https://www.isc2.org/uploadedFiles/(ISC)2_Public_Content/Certification Programs/CSSLP/ISC2_WPIV.pdf
Six Sigma	A set of techniques and tools for process improvement
U.S. Department of Homeland Security's US-CERT	United States Computer Emergency Readiness Team

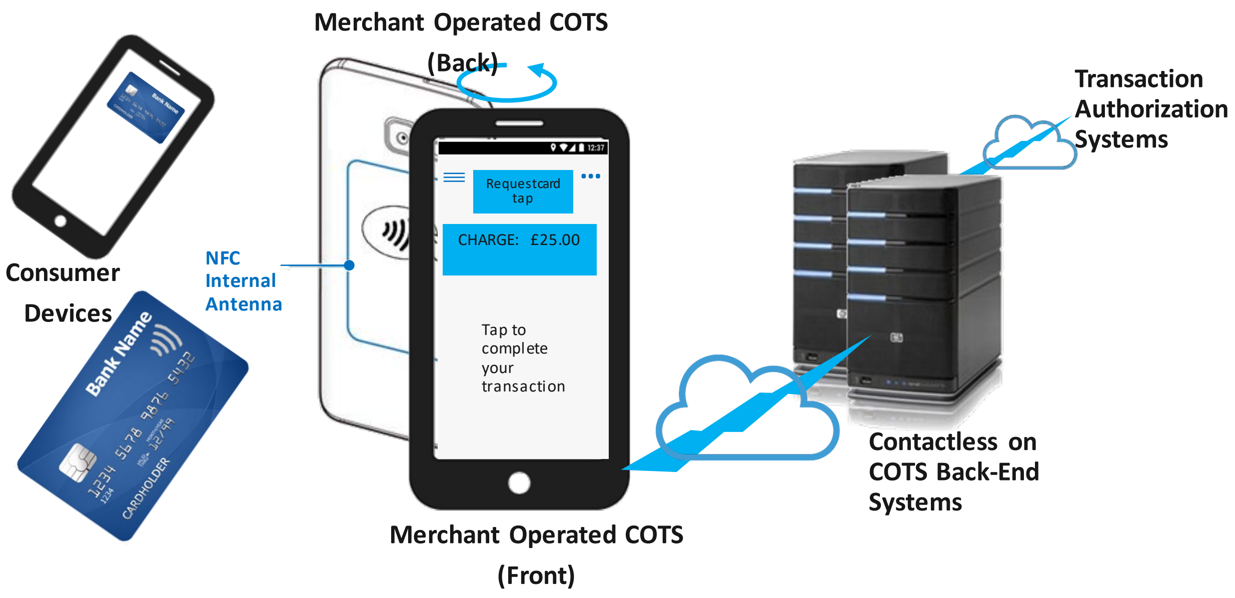
Overview

Background

The solution provides an alternative acceptance mechanism for contactless EMV® markets by enabling a secure software application on a merchant's COTS device to capture and process account data read by the native and embedded NFC capabilities. The solution enables a cardholder to pay with a contactless-enabled card or device (e.g., wearable, phone, tablet) at a merchant using a COTS device and associated CPoC application for authorization of contactless chip-based card payment transactions. To secure account data, these solutions rely on a combination of mechanisms and security controls including, but not limited to, application software and monitoring, and attestation of the entire process.

Figure 1 shows the functional model of the solution that uses a software application only without additional hardware for the protection of account data.

Figure 1: Functional Model of a CPoC Solution Implementation

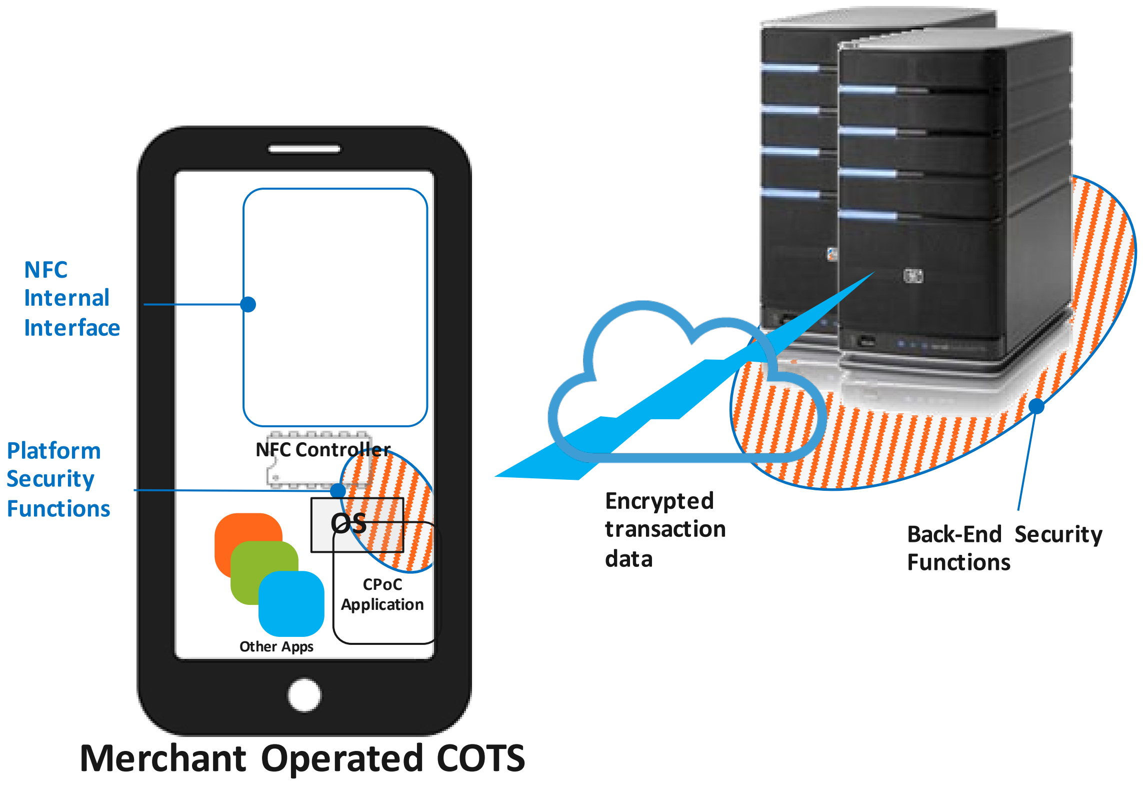

Merchant Operated COTS (Back)
Transaction Authorization Systems
Consumer Devices
NFC Internal Antenna
Requestcard tap
CHARGE: £25.00
Tap to complete your transaction
Contactless on COTS Back-End Systems
Merchant Operated COTS (Front)

Contactless Payments on COTS— Security Model

The architecture of the solution is based on the components in Figure 2:

- A CPoC application
- COTS Devices
- Set of back-end systems

Figure 2: Security Elements of a CPoC Solution

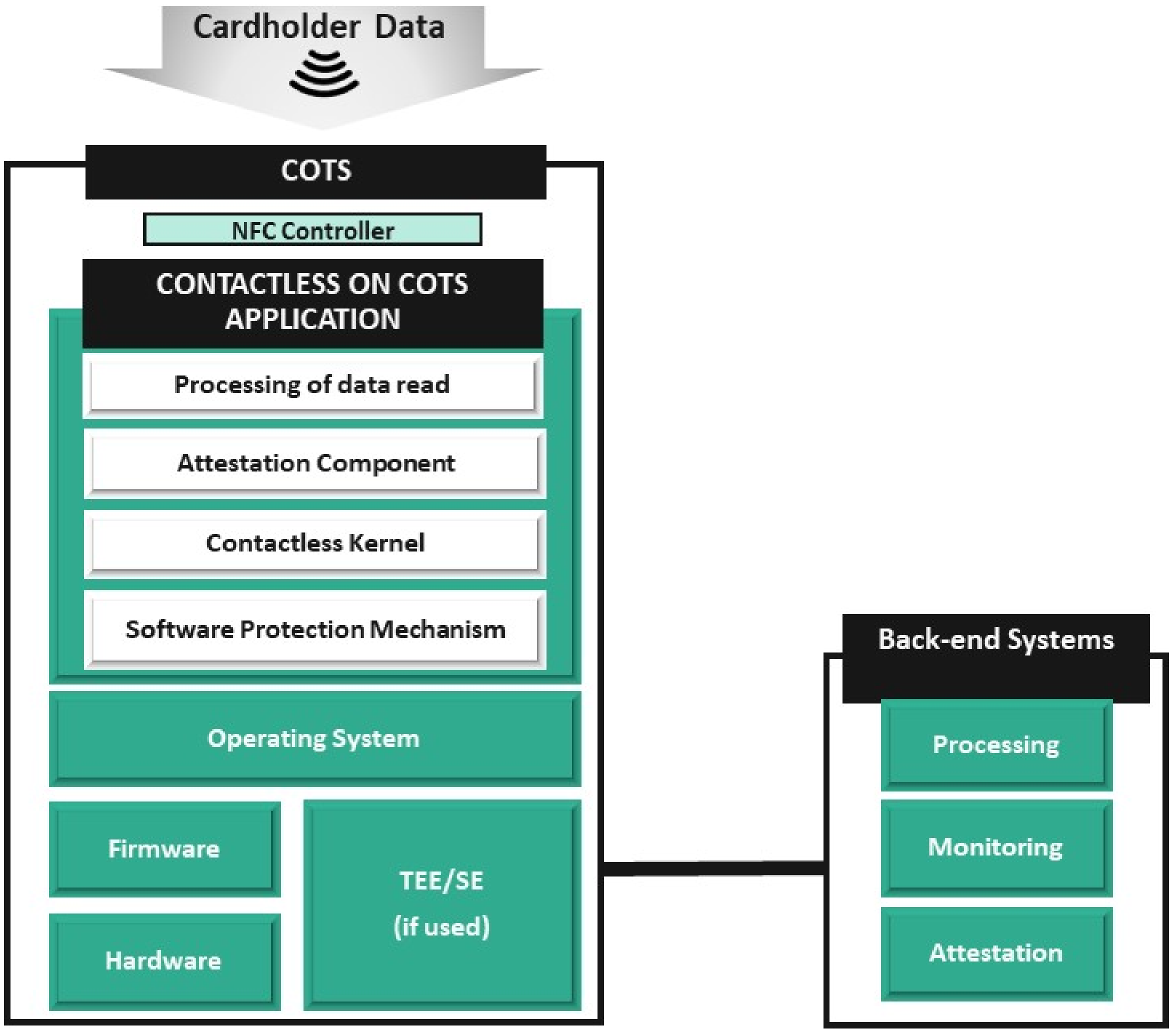


In the COTS device, the NFC interface typically is controlled natively through the COTS operating system (OS). The security model calls for a set of protection mechanisms that span the NFC interface, COTS OS, and CPoC application. This set of protection mechanisms may be implemented in a variety of architectures:

- Rely on security functions offered by the COTS platform, such as a combination of the rich operating system, NFC interface or SE, and protection mechanisms within the CPoC application,
- Be implemented solely within rich OS execution environment as a software-based protection mechanism within the CPoC application on the COTS device, or
- Be provided by a hardware-based secure execution environment such as TEE or SE.

The back-end system provides transaction processing, security monitoring and attestation services for the COTS system baseline, which work in tandem with the COTS device security mechanisms.

Figure 3: An Example of CPoC Solution Architecture



The solution architecture in Figure 3 relies upon the following components to provide for protection, attestation, detection, and response controls:

- A COTS device that is operated by the merchant to run the CPoC application.
  - The COTS device must have online connectivity to support close interaction with the set of back-end systems.
  - The COTS device may have a TEE or SE built in that could be used to perform cryptographic operations, key management and host trusted applications.

- A CPoC application that resides on the COTS device, which:
  - Provides a *channel* to the embedded NFC interface within the COTS device (using rich OS security controls when not implemented within TEE or SE) to initiate the contactless transaction read.
  - Performs initial encryption of the account data.
  - Passes attestation health-check data about the COTS platform and CPoC application to the back-end monitoring system.
  - Contains software protection mechanisms to maintain its own integrity against attack.
  - Was developed with security concepts and activities throughout the entire software life cycle, including inception through development, deployment, operation, maintenance, and decommission.
  - Delivers the encrypted account data through a secure channel to the back-end processing environment to be decrypted in order for it to be passed for subsequent transaction processing.
- A set of back-end systems that performs functions for the CPoC application that include:
  - Attestation**:** Processes attestation health-check data from the CPoC application and enforces pre-established security policies.
  - **Monitoring:** Monitors and provisions security controls to detect, alert, and mitigate suspected or actual threats and attacks against the CPoC application and the COTS device.
  - **Processing:** Processing system that receives encrypted account data from the CPoC application.

Contactless Payments on COTS Solution Overview

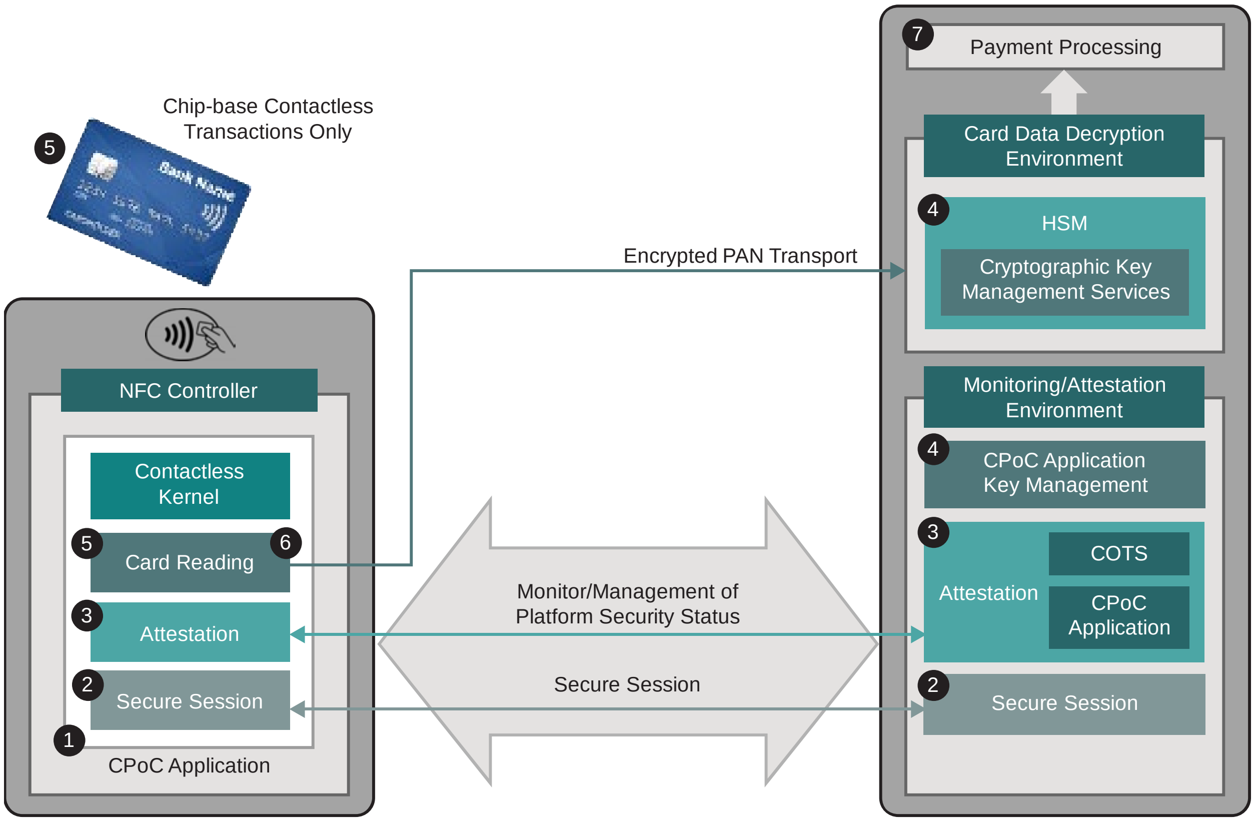
Figure 4 shows an example flow of CPoC application enablement and a subsequent payment transaction in the solution.

1. The CPoC application is downloaded.
2. A secure communication channel between the CPoC application and the back-end attestation system is established.
3. The back-end monitoring system determines the security status of the mobile payment platform (COTS platform and CPoC application) using the attestation component.

Note: *The attestation process is performed as required by the standard.*

4. The CPoC application is initialized with its financial cryptographic keys.
5. A payment transaction is initiated and a contactless card or a contactless-enabled device is presented to the COTS device for reading.
6. The CPoC application reads and encrypts the account data, and constructs a payment transaction message.
7. The payment transaction is processed.

Figure 4: An Example of CPoC Solution Enablement and Transaction Flow


7
Payment Processing
Chip-base Contactless Transactions Only
5
Bank Name
Card Data Decryption Environment
4
HSM
Cryptographic Key Management Services
Encrypted PAN Transport
NFC Controller
Contactless Kernel
5
Card Reading
6
3
Attestation
2
Secure Session
1
CPoC Application
Monitor/Management of Platform Security Status
Secure Session
Monitoring/Attestation Environment
4
CPoC Application Key Management
3
Attestation
COTS
CPoC Application
2
Secure Session

Security Objective and Assets

Security Objective

The objective of these security requirements is to ensure the integrity of the NFC interface and contactless kernel on the COTS device, and to reasonably ensure that the solutions provide adequate security mechanisms, controls, and mitigations to protect the consumer's account data and other assets, such as cryptographic keys. These requirements ensure protection from unauthorized disclosure, modification, or misuse by restricting the available attack surface and make it cost prohibitive to attack.

It is recognized that an attacker may have other objectives, such as self-promotion or nation-state attack, and may expend more resources to circumvent established controls than is warranted by the direct financial rewards.

For the COTS platform components, the objective of these security requirements is to provide reasonable assurance that these components are kept up-to-date and have not been tampered.

Security Assets

There is a number of assets to be protected within the payment ecosystem, such as PAN and full-track-equivalent data. The Payment Card Industry Security Standards Council (PCI SSC) defines security requirements for protecting this data and other sensitive assets, such as cryptographic keys used to protect account data. There may be additional assets that require protection as determined by national or regional regulations, such as personally identifiable information (PII). Protection of these additional assets is beyond the scope of these requirements.

Assets are categorized as requiring one or more security services: Confidentiality (C), *Integrity* (I), and *Integrity* with auditability/authentication (I+). The solution provider may identify other security services, such as confidentiality of binaries.

The following assets are protected by the solution. The solution provider may implement additional controls for other assets not on this list.

- Account data (C and I)
- Attestation data (I+)
- Private and secret keys or session keys and related parameters (static and ephemeral) used during account data encryption. (C and I)
- Cryptographic keys and related parameters (static and ephemeral) used for communications processing and to secure transport to/from the CPoC application resident within the COTS device. (C and I+)
- Contactless kernel (I)
- EMV Protocol/Transaction Process (I)
- CPoC application (C and I+)
- CPoC solution source code (I)
- CPoC application source code (I+)

Security Requirements, Test Requirements and Guidance

The security requirements address known attack scenarios at time the Standard was published. As detailed in the requirements, solution providers have ongoing responsibility to proactively perform risk assessments to identify potential security flaws in CPoC transaction scenarios that were introduced by changes in technology or by the identification of new threats and vulnerabilities.

The requirements defined in this standard (described in Table 1) have been categorized to align with major components that support the security of the overall solution and to support security evaluations if components are the responsibility of different organizations.

Table 1: Requirement Modules

Module	Title	Description
1	Core Requirements	General requirements defining security controls that apply to the overall solution, including requirements for a general oversight and governance of the solution to ensure that all security controls are in place and functioning as intended. The solution provider is responsible for ensuring that these requirements are in place.
2	Contactless Payments on COTS Application	The CPoC application requirements apply to the software applications that reside on the COTS device and communicate with attestation components, back-end monitoring systems, and back-end processing systems. The CPoC application is responsible for initial and any subsequent encryption of the NFC read account data and collecting and reporting attestation information.
3	Back-end systems – Monitoring/Attestation	The back-end monitoring system supports the management of the solution. It interacts with the CPoC application on the COTS device and detects anomalous and potentially fraudulent activity, including suspicious transactions. The back-end attestation ensures that the required security controls and mitigation mechanisms on the COTS device, and functions within the CPoC application that are necessary to protect account data are intact and functioning as intended.
4	Back-end systems – Processing	The back-end payment processing systems are the processes/environments that perform and complete payment processing.
5	Contactless Kernel	The card acceptance protocols and transactions must be EMV contactless. Contactless magnetic stripe data (MSD) transactions that use a dynamic transaction verification code can also be supported by the solution.

Scope of the Target of Evaluation

The following security requirements apply to the individual components and processes that make up the solution. The functional requirements of the Target of Evaluation (TOE) can be further divided into the following physical and logical components of the solution that are also assessed under this Standard:

- Merchant payment processing applications, referred to as the CPoC applications, which are involved in contactless payment acceptance and/or may influence the security of payment data processing. These applications may be executed, in part or as a whole, on a COTS device or executed remotely and rendered on the COTS device through other means. The CPoC application could include multiple libraries and SDKs, but is validated in its entirety. The CPoC application interfaces with the NFC interface on the COTS device, which in turn interfaces with and performs payment transactions with the customer contactless payment mechanism.
- An optional CPoC API provided by the CPoC application that allows other third-party developers to interface with the CPoC solution.
- Contactless kernel, regardless if implemented within CPoC application, residing on the COTS device or implemented as a remote component of a contactless kernel (e.g., cloud-based).
- The back-end attestation and monitoring systems that cannot be entirely (logically) accessed from the merchant environment and that must be capable of being regularly/rapidly updated to respond to new threats and to apply changes or updates to the solutions.
- A back-end processing system that performs the transaction processing. This system is logically different from the monitoring system, but may be integrated with the back-end monitoring system.
- Assessment of the integration of the previously listed components when the solution is not provided by a single solution provider and/or integration of the components may lead to potential security threats.

It is expected that the monitoring and attestation systems will integrate both local and remote features to allow for the identification of new types of attacks, rapid response, and deployment of updated mitigations against such threats.

Reporting Requirements for Contactless Payments on CPoC Laboratories

Evaluation reports must present evidence of the solution compliance with the security requirements. Before releasing a new test report, a delta report, or any updated report version, the lab must perform a thorough technical and quality assurance review to ensure that the report:

- Provides accurate information specified in this document without omissions, ambiguities, or inconsistencies.
- Includes information relevant to any applicable FAQs.
- Conforms to Laboratory General Requirements and any other related documentation in the CPoC Program Guide such as, but not limited to, Reporting Guidance and Templates.

Minimal Contents of Reports and Minimal Test Activities

The beginning of all reports must include overview and summary sections. At a minimum, this summary shall include the following:

- A system overview that summarizes the design, hardware, and software architectures, security functions, and any other relevant attributes, features, or functions including, but not limited to:
  - Any back-end or "cloud-based" components that are part of the system
  - Test COTS devices that were used in the evaluation of the system
- A summary list of security requirements that were tested and of those, indication about which were compliant.
- A summary of any assistance the solution provider provided to the laboratory.

To avoid prohibitively lengthy testing, the solution provider, as directed by the test laboratory, must support the laboratory in tasks, such as code review, fuzzing interfacing, and analysis of white-box cryptographic implementations.

The solution provider shall make source code available to the lab and help in the systematic review of relevant security functions.

The evaluation report document shall demonstrate compliance with security requirements. For all test requirements, the tester shall present sufficient information on direct tests and theoretical claims to validate conclusions by demonstrating how any conclusions are derived. The tester shall determine the appropriate tests and shall document why the test evidence and methods are valid based on test requirements, FAQs, the Program Guide, and any other related documents. Evaluation conclusions stated in the report for each test requirement should be supported by sufficient evidence so as to be understood and confirmed. This evidence includes, but is not limited to:

- References to relevant information in the overview sections of the evaluation report and to other test requirements where appropriate.
- Descriptions of the solution provider's evidence of compliance with security requirements, including information and assistance that the solution provider offered to support the evaluation.
- Accurate descriptions of relevant system attributes, including, but not limited to, physical and logical protections, software architecture, and OS.
- Detailed explanations of the scope and focus of test activities and attack hypotheses, including white-box or black-box approaches that were used and the reasons why they were used.
- Details of decisions made to perform penetration testing, the methods used, and the results of penetration testing.
- Justification for reliance on test evidence not derived directly from the evaluation activities.

- Explanations for any conclusions based on theoretical analysis instead of applied testing.

The tester shall detail where conclusions or evidence are based on laboratory testing or solution provider documentation/assertions, and provide justification for any use of solution provider data rather than actual testing by the laboratory. Test evidence supplied by solution providers that is the sole source of data for any requirement without any laboratory testing may result in the rejection of the report by PCI SSC.

The tester shall justify any deviation from the prescribed routines. The tester is not limited to presenting information specified by test requirement text/guidance/FAQs/Program Guide. The tester shall expand upon the evidence, as needed, to support the conclusions.

In many cases, the test requirement text is insufficient without including screenshots and/or other graphic illustrations that explain the evaluation. Images shall be of sufficient quality for relevant details to be legible, such as clear identification of a screen image, values clearly discernible in a graph, images capturing displays and other outputs, and source code fragments.

All test requirements must include references to documents and any other relevant sources of information. References must indicate information sources sufficiently to enable PCI SSC to identify test evidence following device approval.

Detailed Security Requirements and Test Requirements

The following defines the column headings for the CPoC Standard:

- Security Requirements – This column defines the CPoC Standard requirements. Compliance with the CPoC Standard is validated against these requirements.
- Test Requirements – This column shows processes to be followed by the tester to validate that CPoC requirements have been met.
- Guidance – This column describes the intent or security objective behind each CPoC requirement. This column provides guidance only and is intended to assist understanding of the intent of each requirement. The guidance in this column does not replace or extend the CPoC Security Requirements and Test Requirements.

Note: *CPoC requirements are not considered to be met if controls are not yet implemented or are scheduled to be completed at a future date. After any open items are addressed by the CPoC solution provider, the PCI-recognized laboratory will reassess to validate that the remediation is completed and that all requirements are satisfied.*

Module 1: Core Requirements

Control Objective: All solution security requirements must work in concert to protect account data and support a secure mobile payment-acceptance transaction.

The entire solution must be assessed against these Core Requirements. All parties involved in the solution, including third-party service providers, are required to adhere to the requirements in this module. Solution providers are ultimately responsible for ensuring that all the requirements are met.

1.1 Protection of Sensitive Services

All sensitive services that support the confidentiality, integrity, and availability of the solution and its components are to be identified.

Security Requirements	Test Requirements	Guidance
1.1.1 Documentation detailing all sensitive services implemented by the solution and its components must exist, be reviewed at least annually and be updated as necessary. This must include, but is not limited to, key management, signing of applications, and signing of updates to the attestation services or configuration. ***Note:*** *This includes sensitive services on the COTS platform and the back-end systems.*	**1.1.1.a** The tester must confirm that solution provider documentation exists, is reviewed at least annually, and is up to date. **1.1.1.b** The tester must confirm that the documentation is consistent with the CPoC solution architecture. **1.1.1.c** The tester must confirm that documentation details all sensitive services implemented by the solution.	Proper identification of processes and functions that are fundamental to the security of the solution ensures common understanding, and assists with identifying roles and responsibilities for proper management and security of these processes and/or functions. Without this information, implementation of security controls may be overlooked, which could lead to unauthorized disclosure or compromise of the solution. It is expected that security vulnerabilities will be discovered throughout the year, and that the solution documentation must be updated to address them.
1.1.2 All sensitive services must be protected against unauthorized modification (i.e., integrity protection) and against unauthorized access (i.e., access control).	**1.1.2.a** The tester must confirm that all identified sensitive services are protected from unauthorized modification. An assurance that modification will be detected and processing suspended is sufficient if prevention is otherwise infeasible; however, the tester must confirm the assertion. **1.1.2.b** The tester must confirm that all identified sensitive services are protected from unauthorized access. The testing must include testing of the access control mechanism(s).	Misuse of sensitive services, whether through modification or unauthorized use, could lead to disclosure of cryptographic materials or account data. It could also result in failure of the attestation process or in manipulation of authentication results.

1.2 Random Numbers

Random numbers are relied upon by many security processes and secure communications methods. Generation of random numbers with insufficient entropy has been the cause of many high-profile vulnerabilities. This makes the quality of the random numbers generated by the solution vital. Random numbers are to be generated using a process that ensures sufficient entropy (for example, as defined in *NIST SP800-22*) and lack of statistical correlation within the set of solution components.

Any random numbers used on the COTS device for security purposes must be seeded from a value that comes from a trusted source. A COTS device that has a TEE or SE evaluated as NRNG can be used as a source of entropy. Otherwise, an external trusted source must be used.

The COTS platform should maintain an entropy "pool" that is updated regularly from the trusted source and other sources on the COTS platform. This pool data is sensitive and should be protected.

This applies to all components and parts of the solution where random numbers are generated for security functions.

***Note:** Random numbers that are not relied upon directly for security of the account data or attestation data are exempt from this requirement. Examples include random values used in TLS sessions, where transmitted data is otherwise protected using application-level cryptography,*

Security Requirements	Test Requirements	Guidance
1.2.1 Documentation to identify all random number generation functions and reliance on random data used in the solution must exist and be maintained.	**1.2.1.a** The tester must detail all random number generation functions used in the solution implementation. This must include random numbers generated on all COTS platforms supported by the solution (assessed under *Security Requirement 3.1.1*) and random numbers generated in any back-end systems that provide security to the solution. ***Note:** The intent is not to check the NRNG process used by the operating system of each. Rather it is to ensure that the CPoC application implements methods ensure robust random data generation when COTS platforms can have inherently potentially poor random data processes of their own. These details must include any seed values used, hardware systems, and software-based DRNG.* **1.2.1.b** The tester must confirm that the documentation is being reviewed, at least annually, and updated as required.	Identification of all security functions implemented within the solution that require or rely upon the use of random data is necessary to ensure that the process is sound and cannot be circumvented. Cryptography depends on the creation of secret data that is known to only those with a need to know while remaining unknown and unpredictable to others. Random number generation sets the security baseline upon which other security controls rely. This may include the generation of padding data for use in certificates, key bundles, and EMV unpredictable numbers, as well as the generation of cryptographic keys. Random generator attacks by malicious users exploit weak random number implementations and have been the cause of several high-profile vulnerabilities. Therefore, the quality of the random numbers generated by the solution is vital.

Security Requirements	Test Requirements	Guidance
1.2.2 Any random numbers used on the COTS device for security purposes must be seeded from a value provided from a trusted source combined with input from the Random Number Generator (RNG) on the COTS platform or within the CPoC application, and at least two other sources of non-deterministic data (such as user input timing and values collected from lowest bits of on-device analog sensors). ***Note:*** *Random numbers that are not relied upon directly for security of the account data or attestation data, such as random values used in TLS sessions where the data being transmitted is otherwise protected using application level cryptography, are exempt from this requirement.*	**1.2.2.a** The tester must confirm that the random values generated on the COTS devices require seeding from at least 256 bits of entropy sent from a trusted source of random data, such as a PCI- or FIPS-approved HSM, Secure Element, or other such components. The random values must be generated using a function approved for generating random data with equal input from the COTS platform or CPoC application RNG, and at least two different entropy sources from the COTS device. The tester must detail the random number generation method used on the COTS device. **1.2.2.b** The tester must detail the trusted source used and confirm the testing performed on this component to validate the entropy output for the functions used. Where previous testing includes options for externally provided entropy to this component, such as with FIPS140-2 (or equivalent in FIPS 140-3) approval, the tester must confirm the source of this external entropy. **1.2.2.c** The tester must detail the random number generation method used on the COTS device and confirm that the solution creates an entropy pool upon initial seeding that maintains the state of the DRNG between uses. The tester must confirm that this pool is maintained securely by the application, and that the pool is updated with the COTS platform or CPoC application RNG before each use. **1.2.2.d** The tester must confirm that the method used to combine the collected entropy inputs into a single seed for the DRNG maintains the entropy of each seed, such as by XORing each seed value into a single entropy pool.	Sufficient entropy is not assured in COTS devices. Therefore, confirmation is necessary that the secret value (seeded value) is from a trusted and tested source, and that the seed length is at least 256 bits. Trusted sources may include a hardware security component that is resident in a back-end platform or the COTS device that has been independently tested to ensure it meets the most stringent requirements. If unevaluated hardware-based random sources are used on COTS devices (e.g., Android StrongBox) as a trusted source, the lab should test these sources to ensure they meet NRNG requirements. Combining seeds should ensure that entropy of the seeds is maintained. An example of how this might be done is by XORing the two seed values. This may be combined with an additional seed generated on the COTS platform. The additional entropy sources is not used to increase the entropy provided by the NRNG, but rather to act as a safeguard if the NRNG fails. The DRNG can use the random number seed obtained from outside the COTS device to generate any white-box cryptographic keys or random numbers required by the CPoC application.
1.2.3 The DRNG must be reseeded each time the CPoC application launches.	**1.2.3.a** The tester must review the source code to confirm that the DRNG is reseeded every time CPoC application launches. **1.2.3.b** If DRNG is seeded from an external source, the tester must confirm that a secure channel is established between the COTS platform and the trusted external source.	DRNG will generate the same sequence of random numbers if the same seed is used. Hence, it is important to reseed the random number with a different random seed value to ensure that it produces a different random sequence of numbers each time the application launches.

Security Requirements	Test Requirements	Guidance
1.2.4 The seed value must never be stored in non-volatile memory.	**1.2.4.a** The tester must confirm that random number seeds are not stored in non-volatile memory. **1.2.4.b** The tester must confirm that seed values are cleared under each of the following minimum set of conditions: • The CPoC application pauses or stops executing • The CPoC application loses its Foreground focus **1.2.4.c** The tester must confirm that all clearing methods are robust and does not rely solely on "garbage collection."	Storing the seed value for the random number generator in non-volatile memory exposes the value to being compromised by a malicious application or process on the COTS device. This compromises cryptographic keys created from the output of the random number generator because the sequence becomes predictable. Furthermore, there is no reason to store the seed value after the random number generator is seeded with it.
1.2.5 The CPoC application must use an assessed RNG where cryptographic algorithms require the use of random numbers.	**1.2.5.a** The tester must list all security services implemented in the solution that require or rely on the use of random data. This may include generation of padding data, such as for use in certificates or key bundles, and generation of cryptographic keys. **1.2.5.b** The tester must review the source code of each security service and confirm that it uses properly the RNG reviewed in this requirement. This evaluation should focus on relevant security-critical sections of the source code to provide an optimal balance between the use of evaluation resources and overall evaluation goals. **1.2.5.c** The tester must detail how the EMV Unpredictable Number is generated, and confirm that the number uses either the approved RNG or the algorithm specified by EMV for this purpose.	There are two types of random number generators (RNGs): DRNG and NRNG. A DRNG uses an initial seed value provided by an NRNG to generate deterministic random numbers. Where the COTS device has a TEE or SE that were evaluated as NRNG, these can be used as a source of entropy. Otherwise, the seed value comes from an HSM external to the COTS device that is at least a *FIPS 140-2* Level 3 (or equivalent in FIPS 140-3) or PCI-approved HSM. The solution provider should avoid implementing its own DRNG and, when possible, implement well-known sources or algorithms specified in *NIST SP800-90a*. The RNG used by the CPoC application should be tested for fitness of purpose against industry-recognized test suites, such as *NIST SP800-22* or *AIS 31*.
1.2.6 Any random numbers used on a back-end system for security purposes must be seeded from a value provided initially from the NRNG on at least a *FIPS 140-2* Level 3 (or *FIPS 140-3* equivalent) or PCI-approved HSM.	**1.2.6.a** The tester must confirm that the NRNG in the back-end system is implemented using a random number generator approved by *FIPS140-2 Level 3* (or equivalent in *FIPS 140-3*) or *PCI PTS-* HSM. **1.2.6.b** If *FIPS140-2 Level 3* (or equivalent in *FIPS 140-3*) approval is used, the tester must validate from the approval that the entropy is not supplied externally.	NRNG generation performed in a *FIPS 140-2* Level 3 (or equivalent in *FIPS 140-3*) or in a PCI-approved HSM that has been independently tested ensures that the process meets the most stringent requirements.

1.3 Acceptable Cryptography

Cryptography is an important factor to ensure confidentiality and integrity of data and processes that support the solution. Therefore, it is important that only industry-recognized standard cryptographic algorithms and implementation methodologies be the basis for any security services used in the solution.

The account data should be encrypted on the COTS device for transporting to other components of the solution using cryptographic algorithms and modes of operation known to provide suitable levels of security.

Acceptable hash functions use SHA256 or stronger. This does not preclude the use of other hash types, such as digital fingerprinting or file comparison, when collision resistance is not required as a security feature.

All cryptographic keys should be used for a single specific purpose. For example, a key used to encrypt account data should not be used to protect the integrity of the tamper-detection data.

This requirement applies to all components and processes used in the solution.

Security Requirements	Test Requirements	Guidance
1.3.1 Documentation must exist that identifies cryptographic processes and operations used by the solution for security services. Documentation must include, but not be limited to, the following: • Cryptographic algorithms used and where they are used • Identification of all keys, the complete key hierarchy, their purposes, and their crypto periods • Key-generation or key-agreement processes • Description of cryptographic key protection mechanisms	**1.3.1.a** The tester must provide a list of all cryptographic operations used by the solution for security services, and specify the cryptographic algorithm used for each of these cryptographic operations. **1.3.1.b** For all cryptographic operations used by the solution, the tester must confirm that documentation includes: • Cryptographic algorithms used and where they are used • Identification of all keys, the complete key hierarchy, their purposes, and their crypto periods • Key-generation or key-agreement processes • Description of cryptographic key protection mechanisms	Information that identifies cryptographic operations used in the solution helps ensure that controls are tested appropriately. It also helps to identify areas where cryptography may increase the solution's security protection. Comprehensive documentation usually contains cipher suites or other cryptographic algorithms, including transport layer protocols, that are used for secure channels.

Security Requirements	Test Requirements	Guidance
1.3.2 All cryptographic processes provided by the solution must adhere to Appendix C Minimum and Equivalent Key Sizes and Strength for Approved Algorithms.	**1.3.2.a** Referencing the key table produced in Security Requirement 1.4.1, the tester must confirm that all algorithms meet the minimum requirements outlined in Appendix C Minimum and Equivalent Key Sizes and Strength for Approved Algorithms.	To withstand modern-day attacks and ensure future support, the solution should use the most robust and current encryption algorithms and key sizes. Legacy algorithms have a limited shelf life and may lose some of their effectiveness over time. The solution provider should consider adopting principles of cryptographic agility or cryptographic agnosticism to allow for evolution and adoption of alternative cryptographic algorithms and key sizes when those originally used can no longer meet current or future data security requirements. Use of recognized cryptographic methods ensures that the solution adheres to industry-tested and accepted algorithms and appropriate key lengths that deliver effective key strength and proper key-management practices. Proprietary or "home-grown" algorithms do not provide this assurance and are not permitted. For information about cryptography and secure protocols, see the industry standards and best practices, such as *NIST SP 800-52* and *SP 800-57*.
1.3.3 Hash functions must be implemented in accordance with Appendix C Minimum and Equivalent Key Sizes and Strength for Approved Algorithms.	**1.3.3.a** The tester must detail all instances where hash functions are used in the solution, including in certificates, storage of sensitive data for comparison, and for use in general-integrity or authenticity purposes. The tester must confirm that each of these uses implements hash functions that are approved for use in this Standard.	Use of hash functions ensures the integrity of data. Because hash functions are based on mathematical and cryptographic functions, use of recognized, accepted methods is necessary. This does not preclude the use of other hash types, such as MD5, where collision resistance is not required as a security feature (fingerprinting or file comparison).

Security Requirements	Test Requirements	Guidance
1.3.4 Security to any cryptographic key must not be provided by a key of lesser strength.	**1.3.4.a** The tester must confirm that security to any key is not provided by a key of lesser strength, such as by encrypting a 256-bit AES key with a 128-bit AES key.	Cryptographic keys lose their strength when protected by a key of lesser strength, making it easier for malicious users to attack weaker keys to reveal their content. For example, assume a data-encryption key is created with 256-bit AES security, which is considered a strong key and can likely withstand current data attacks. However, if that same data-encryption key is transported with a key-encryption key (KEK) that has only a 128-bit AES key strength, the data-encryption key has lost its strength and is providing only 128-bit security—making it much more vulnerable to attack. Organizations should always consider this when managing keys to avoid reducing the key strength inadvertently.
1.3.5 Public keys and certificates used by the solution must be authenticated up the entire chain to the root certificate authority (CA) at install.	**1.3.5.a** For any public keys used by the solution, the tester must confirm how the authenticity of that public key is maintained. Use of public keys that are not signed, or that are maintained in self-signed certificates is prohibited. ***Note:** Self-signed certificates that are part of the base COTS platform on which the CPoC application is executed are excluded from this requirement. However, the self-signed certificates must not be relied upon for security services by the CPoC application.*	Due to the nature of their use, public keys and certificates are public and may be sent and received in cleartext. Requiring encryption to preserve the confidentiality of the public key and certificate is not mandatory. However, it is important to protect the public key to ensure that it is genuine (authenticity) and has not been substituted or altered (integrity). Digital signatures or message authentication codes, such as MAC, are examples of methods that provide this assurance. The use of manual authentication in distribution systems (e.g., checking fingerprint) and authentication based on secure storage (HSM) is allowed.
1.3.6 Self-signed certificates are prohibited. ***Note:** Self-signed certificates that exist as part of the base COTS platform are excluded from this requirement.*	**1.3.6.a** The tester must confirm that self-signed certificates are not used by the CPoC application.	Self-signed certificates should not be relied upon for security services by the CPoC solution.

Security Requirements	Test Requirements	Guidance
1.3.7 Mechanisms must be in place to identify expired and revoked certificates, and to prevent continued processing when certificates have expired or been revoked.	**1.3.7.a** The tester must confirm that mechanisms are in place to identify both expired and revoked certificates. **1.3.7.b** The tester must detail the mechanism to prevent continued processing when a certificate has expired or been revoked.	Expired certificates introduce unacceptable risk to the solution. CPoC application functions should not be available when such certificates are detected. Expired certificates could be an indication of a malicious user acting as an imposter of a legitimate organization or process who is phishing for sensitive information. The requirement is applicable to all components of the solution, and includes only the certificates upon which the solution relies for security purpose. For example, certificates that exist on the COTS device as part of the COTS OS could be out of scope if the solution is not using these certificates.
1.3.8 Each key must have a single unique purpose, and no keys may be used for multiple purposes, such as both signing and encrypting data.	**1.3.8.a** The tester must confirm that each key has a unique purpose, no keys are used for multiple purposes (such as both signing and encrypting data), and that keys used to encrypt account data are not used for any other operation (such as general-purpose data encryption or attestation component message encryption).	Keys used to encrypt account data should not be used for any other operation, such as general-purpose data encryption or attestation message encryption. Using unique keys for dedicated purposes ensures that exposure is limited if a key is compromised.
1.3.9 Keys used to validate message authenticity must be unique to each endpoint so that signature generated at one end would always be different if generated at the other endpoint.	**1.3.9.a** The tester must confirm that keys used to validate the authenticity of a datagram are unique to each endpoint. This means that a signature generated at one endpoint would always be different than the signature generated by the other endpoint.	Digital signatures protect against message tampering and impersonation, so there is assurance that the information in the message is intact and has not been intercepted or altered in transit. Message examples include attestation data and control messages. Should a key be compromised, using unique keys for each endpoint further enhances protection by limiting exposure to the affected endpoint only and not to the entire population.
1.3.10 Any key signature or digital fingerprint values must not reveal any bits of the key itself.	**1.3.10.a** The tester must confirm that any key signature or digital fingerprint values do not reveal any bits of the key itself.	A key signature or digital fingerprint is the value that is shared between organizations to ensure the key that was conveyed is the key that was received. Keeping the encryption key secret is critical to overall security. However, key signatures or fingerprints require this shared value to be public; therefore, the key signature and fingerprint are created in a way that would not disclose information about the underlying bits of the encryption key or that would allow the information to be used to reverse-engineer the value of the key itself.

Security Requirements	Test Requirements	Guidance
1.3.11 Key Check Values (KCVs) must be limited to five bytes (10 hexadecimal digits). In addition, hash algorithms used for key fingerprints on secret or private keys must implement SHA256 or stronger, or be truncated to no more than five bytes.	**1.3.11.a** The tester must confirm that KCVs are limited to five bytes, and that hash algorithms used for key digital fingerprints on secret or private keys must use SHA256 or stronger. The tester must detail the methods used to enforce this requirement.	KVCs, also known as *key verification checks*, are values used to identify a key without directly revealing any bits of the actual key itself. Because encryption keys cannot be exposed in cleartext, KVCs provide a way to validate the authenticity of a key without disclosing any bits of the key itself. Limiting the number of bytes for the KVC ensures that the ability to retrieve the underlying cleartext key value from the KVC is greatly reduced.

1.4 Key Management

Successful key management is critical to the security of cryptographic systems. It is a fundamental factor for ensuring the confidentiality and integrity of data and processes that support the solution. Key-management practices are to conform to the industry-accepted practices described in this section. Cryptographic keys are managed securely using recognized industry requirements throughout the cryptographic lifecycle including, but not limited to:

- Generation
- Distribution/conveyance
- Storage
- Established crypto periods
- Replacement/rotation when the crypto period is reached
- Escrow/backup
- Key compromise and recovery
- Emergency procedures to destroy and replace keys
- Accountability and audit

Secret and private cryptographic keys that are relied upon for security should be unique per device/application. Shared public keys are acceptable, but methods and procedures for revoking compromised public/private key pairs should be implemented. For additional information about Public Key Infrastructure (PKI), refer to X9.79-4.

Operations that involve secret or private cryptographic keys are to be performed using split knowledge. Split knowledge requires that no one person can determine any single bit of a secret or private cryptographic key. Split knowledge can be provided in the following ways:

- Storing keys on secure cryptographic devices (SCD) that will not output the cleartext key
- Two or more full-length components during key loading
- An M-of-N secret-sharing scheme

Security Requirements	Testing Procedures	Guidance
1.4.1 All key lifecycle management functions and procedures used by the solution must be documented.	**1.4.1.a** The tester must provide a separate "key table" for every in-scope area of the current assessment that outlines all cryptographic keys used in the solution for the security of the transaction data or overall solution security. The key table should have at least the following columns: • Key name • Key purpose • Key size • Key location (e.g., device/system/memory location, as appropriate) • Algorithm • Method used to load/generate key • Whether the key is unique to transaction/device/overall solution (or other) • How often the key is changed • How the key is secured (e.g., in an HSM, using white--box cryptography, by encryption with another key, etc.) • How the key is erased/destroyed ***Note:*** *Applicable to all solution components, such as CPoC applications and back-end systems including the back-end monitoring system and the attestation system, as well as any remote functional services, such as a remote component of contactless kernel.* **1.4.1.b** For each key in the key table, the tester must detail the key use, algorithm, storage/method to secure the key, method of loading/generation, key expiry, and method of destruction. **1.4.1.c** The tester must confirm that the solution provider documentation covers all keys, as determined by the tester during the evaluation and listed in the key table. **1.4.1.d** The tester must confirm for each key in the key table that there is sufficient documentation to cover	A good key-management process, whether manual or automated, is based on industry standards and addresses all elements of the key lifecycle that include: • Distribution/conveyance • Storage • Established crypto periods • Replacement/rotation when the crypto period is reached • Escrow/backup • Key compromise and recovery • Emergency procedures to destroy and replace keys • Accountability and audit For example, key generation should conform to industry-recognized procedures that ensure the confidentiality of the underlying key. Keys should be distributed only in a secure manner, never in the clear, and only to designated custodians or recipients. Procedures for distribution apply both within the entity and outside it. Secret and private keys should be encrypted with a strong key-encrypting key that is stored separately, stored within a secure cryptographic device (such as an HSM), or stored as at least two full-length key components or key shares in accordance with an industry-accepted method. A crypto period should be identified for each key based on a risk assessment, and keys should be changed when this period is reached. Additionally, keys should be destroyed and replaced immediately upon suspicion of a compromise. Secure key-management practices include: • Minimizing access to keys to the fewest number of custodians necessary. • Enforcing split knowledge and dual control for activities involving cleartext keys or key components.

Security Requirements	Testing Procedures	Guidance
	all lifecycle functions for that key—from generation through use and destruction/decommissioning.	• Defining roles and responsibilities for Key Custodians and Key Managers.
1.4.2 Cryptographic key-management processes must conform to industry-accepted key-management standards. ***Note:*** *White-box cryptographic keys are covered in Module 2.*	**1.4.2.a** The tester must confirm that the key management implemented by the solution meets industry standards such as *ISO11568*, or *ANSI X9.24*.	Alignment with industry-accepted key-management practices provides reasonable assurance that required due diligence and execution reduce risk of unauthorized disclosure and support the overall security of the processes. Applicable standards include *NIST Special Publication 800-57* (all parts), *Special Publication 800-130*, *ISO 11568*, *ISO/IEC 11770*; *ANSI X9.24 Key Management Techniques*; and *NIST Special Publication 800-90A*, *ISO/IEC 18031* including associated normative references cited within as applicable.
1.4.3 Key-management techniques must protect the integrity and purpose of all keys used in the solution.	**1.4.3.a** The tester must confirm how key purpose and integrity are ensured for all keys used in the solution; for example, preventing a key of one purpose (such as account data encryption) from being replaced with a key of another purpose (such as general data encryption). ***Note:*** *This requirement does not mandate the use of key wrapping techniques for symmetric key encryption. The use of PCI-recognized standard methods, such as those outlined in ISO 20038 and X9.102, meet this requirement.*	To provide reliable security, cryptographic require mechanisms that associate the key type/purpose to ensure that the key is used only as a key-encrypting key. Symmetric key encryption methods may require additional key blocks or equivalent authentication methods[4] to prevent known attacks that weaken the underlying key's security. These attacks result in key recovery, and thereby compromise the encrypted data. Asymmetric keys are usually protected using certificates (e.g., X.509).
1.4.4 Secret cryptographic keys and private cryptographic keys that are used as part of account data security must be maintained in one or more of the following approved forms: • Encrypted by a key of equal or greater strength • Stored within a secure cryptographic device • Managed as two or more full-length components	**1.4.4.a** The tester must confirm that secret cryptographic keys and private cryptographic keys are always maintained in one of the following approved forms: • Encrypted by a key of equal or greater strength • Stored within a secure cryptographic device • Managed as two or more full-length components	Cryptographic keys should be protected to prevent unauthorized or unnecessary access that could result in the exposure of encrypted data. Cryptographic keys should never be stored in cleartext on a persistent storage. For information about acceptable storage methods, refer to industry-accepted practices.

[4] Additional information is available in *Cryptographic Key Blocks Information Supplement* (https://www.pcisecuritystandards.org/documents/Cryptographic_Key_Blocks_Information_Supplement_June_2017.pdf).

Security Requirements	Testing Procedures	Guidance
• Managed as an M-of-N secret-sharing scheme ***Note:*** *These approved methods do not apply when storing secret and/or private cryptographic keys within the CPoC application. For the CPoC application (white-box cryptography), refer to Section 2.2 Software-Protected Cryptography. These requirements do not to apply to TLS-negotiated sessions.*	• Managed under a valid M-of-N secret-sharing scheme	
1.4.5 Both secret cryptographic keys and private cryptographic keys must be unique per device and/or application. ***Note:*** *White-box cryptographic keys are covered in Module 2: Contactless Payments on COTS Application.*	**1.4.5.a** The tester must confirm that all secret cryptographic keys and private cryptographic keys, excluding any white-box keys, are unique.	Using a unique cryptographic key for each device or application limits exposure if the key is compromised.
1.4.6 Key management processes for cleartext secret or private keys or key components must ensure split knowledge and dual control principles are enforced. ***Note:*** *White-box cryptographic keys are covered in Module 2: Contactless Payments on COTS Application.*	**1.4.6.a** Referencing the key table produced in Security Requirement 1.4.1, the tester must provide details about each key-loading method and procedure used in the solution. This includes, but is not to be limited to, loading of keys into back-end HSMs, use of white-box cryptography keys, loading keys into any separate execution environments (such as a TEE or remote host/kernel) and initial provisioning of keys into the CPoC application. **1.4.6.b** For each key in the key table, the tester must confirm that loading and other operations performed for all secret and private cryptographic keys conform to the principles of split knowledge; specifically, that no one person is able to know any single bit of the resultant cleartext key. **1.4.6.c** For each key-loading method used in the solution, the tester must confirm that the method enforces dual control across the entire process, such that no one person is ever solely responsible (or can be held liable) for the security of the cryptographic key-loading or key-injection operation.	Keys are fundamental to the cryptographic process. Encryption keys should never be disclosed so that the entire key is available in cleartext or in a form where the possibility of its disclosure in its entirety exists. This is especially important during the generation and loading of the cleartext secret, private keys, or key components. The security principles of dual control and split knowledge require at least two entities be involved in the process to prevent a single entity from having access to the entire process. Split knowledge requires at least two individuals who have only partial information about the key. Dual control requires at least two individuals to perform a process. It is more difficult to establish a breach of process or information when multiple entities are required to conspire to misuse. There several ways to implement dual control and split knowledge using logical mechanisms, physical mechanisms, or both.

Security Requirements	Testing Procedures	Guidance
	1.4.6.d For each key in the key table, the tester must confirm that the method of loading the key ensures that no one person is able to subvert, alter, or otherwise compromise the value of the key. ***Note**: This requirement ensures that collusion between at least two trusted people is required to alter the value of any secret, private, or public key. Key secrecy is covered under a security requirement 1.4.4.* **1.4.6.e** Where security is provided for loading of keys by encrypting those keys, the tester must confirm that the key used to perform the encryption is of at least equal or greater strength than the key it is encrypting. Furthermore, the tester must confirm that a dedicated key-encrypting key is used. The use of transport-layer cryptography, such as TLS, is not permitted.	
1.4.7 Methods and procedures to revoke compromised public/private key pairs must be documented and implemented.	**1.4.7.a** For each of the asymmetric keys used in the solution, the tester must detail how to revoke the key.	The ability to identify and revoke compromised public keys/certificates and private keys are necessary to respond to confirmed or suspicious activity that could lead to misuse or data breaches. Ensuring that procedures are documented and implemented supports common knowledge and sets expectations.
1.4.8 All symmetric key derivation functions must implement one-way functions or other irreversible processes.	**1.4.8.a** The tester must confirm that no reversible key calculation modes, such as key variants, are used to create new keys directly from an existing key. All key-generation methods must use one-way functions or other irreversible key-generation processes. **1.4.8.b** Where key derivation methods are used to create the transaction unique key, the tester must verify that the method is not reversible so that the compromise of any individual working key does not allow the compromise of any past or future derived keys in the same space.	One-way functions or other irreversible key-generation processes produce keys that are impossible to revert to the original data that formed the key. Adhering to industry-accepted cryptography algorithms and techniques supports these concepts.
1.4.9 Audit logs must be generated and maintained for all key-management activities and all activities involving cleartext key components. The audit log includes:	**1.4.9.a** The tester must verify that a mechanism exists to generate audit logs for all key management activities and all activities involving cleartext key components.	Recording the function or key-management activity being performed (for example, key loading) and the purpose of the affected key (for example, data encryption) provides the entity with a complete and concise record of key-management activities. Identifying the activity success or

Security Requirements	Testing Procedures	Guidance
• Unique identification of the entity that performed each function • Date and time • Function being performed • Purpose • Success or failure of the activity	**1.4.9.b** The tester must review sample audit logs and confirm that secret keys, private keys, and cryptographic material are not stored. **1.4.9.c** The tester must verify that the audit log mechanism includes integrity protection against unauthorized modifications or deletions.	failure confirms the status upon conclusion of the activity. By recording these details for the auditable events, a potential compromise can be identified quickly, with sufficient detail to know who, what, where, when, and how. To protect the security of cryptographic material, the audit logs should not include secret keys, private keys, or key materials that might be used to recreate those keys.
1.4.10 Retention of key-management audit logs must align with the organization's record retention policies and, at a minimum, be retained for two years subsequent to key destruction.	**1.4.10.a** The tester must detail the retention practices and periods used by the solution provider for key management audit logs. **1.4.10.b** The tester must confirm that the retention period and practices used for the logs are sufficient to enable post-compromise audits that may be required for determining access and use of cryptographic keys. **1.4.10.c** The tester must review a sample audit log and confirm that the logs are being maintained according to solution provider policies.	Retaining key management audit logs allows activity surrounding anomalies and investigation of breaches to be reviewed. Sufficient log history is required to determine the potential breach time span and the systems affected. Log-retention policies should include storage and retrieval procedures. If logs are stored in offline locations, procedures should include assurance that log data can be retrieved in a timely manner. Examples of industry accepted practices include: • *NIST Special Publication 800-57pt2-r1* Recommendation for Key Management Part 2: Best Practices for Key Management Organizations • *ANSI X9.24* (all parts) Retail Financial Services Symmetric Key Management • *ISO 11568-2 Financial services*—Key management (retail)—Part 2: Symmetric ciphers, their key management and life cycle
1.4.11 The integrity of audit logs containing key-management activities must be protected from unauthorized modifications or deletion. Key management audit logs that are retained in a solution provider's CDE must be stored in accordance with PCI DSS. Otherwise, the environment must comply with the logical and physical security requirements defined in Appendix A Monitoring and Attestation Environment Basic Protections.	**1.4.11.a** The tester must detail the methods used by the solution provider to protect the integrity of the key management audit logs. The tester must confirm that these methods are sufficient to protect the logs from unauthorized modifications or deletion. **1.4.11.b** The tester must detail the methods used by the solution provider to ensure that the audit logs are being maintained in a way that protects them from unauthorized modification or deletion. **1.4.11.c** The tester must review a sample audit logs and confirm that the methods for protecting the logs are being used. **1.4.11.d** When key management audit logs are retained in an organization's CDE, the tester must	The integrity of the audit logs is essential to support audit or forensic investigations required by the solution. Examples of mechanisms that protect the integrity of audit logs include cryptographic hash functions and digital signatures.

Security Requirements	Testing Procedures	Guidance
	obtain and review the Attestation of Compliance (AOC) outlining compliance of the environment with the PCI DSS requirements. Otherwise, the tester must confirm that the environment is compliant with the logical and physical security requirements defined in Appendix A Monitoring and Attestation Environment Basic Protections.	
1.4.12 Incident response procedures must exist and include activities for reporting and responding to suspicious or confirmed key-related issues, including key compromises.	**1.4.12.a** The tester must verify that incident response procedures exist, and that they include activities for reporting and responding to suspicious or confirmed key-related issues and key compromises. **1.4.12.b** The tester must confirm that these procedures name the personnel, positions, or groups who must be notified immediately of any breach impacting keys.	Documented procedures contribute to the rapid and efficient execution of those procedures. Procedure documentation should explain how to escalate the issue for further investigation and resolution, including how to initiate incident response procedures. Appropriate personnel should be notified immediately about any breach that impacts the keys.

1.5 Secure Channels

A secure channel is a communications connection between two points that is secured using encryption. Secure channels are to be established to protect all communications between the solution components including, if present:

- Between the CPoC application and the back-end monitoring system and attestation systems.
- Between the CPoC application , back-end systems and other services on which the solution relies, such as remote contactless kernel (when the contactless kernel is split across several subcomponents residing outside of the CPoC application) or remote key-loading services.

The secure channels should provide mutual authentication and the ability to identify each component uniquely, so that back-end monitoring system can detect and flag component changes as potential tamper events.

Where standard protocols provide secure channels, such as TLS, are used, certificate pinning (or other applicable methods) should be implemented to ensure that only authorized connections are possible. Cryptographic methods should be limited to those accepted under this Standard.

Security Requirements	Test Requirements	Guidance
1.5.1 Connections between different physical and logical components of the solution must be secured.	**1.5.1.a** Using the block diagram that illustrates the software and hardware components of the solution, the tester must confirm that a secure channel exists between all physically disparate components. **1.5.1.b** The tester must detail the implementation for each secure channel in use. Where different components are collocated physically in the same data center or COTS device, the tester must detail how the secure channel is achieved. **1.5.1.c** The tester must detail the communications between each of the logically disparate aspects of the solution and confirm the methods that secure the communications. Where security methods rely on properties of the COTS platform instead of a secure channel, the tester must detail these methods, and confirm that they are suitable for use and that they are present on the platforms being used. **1.5.1.d** The tester must detail what methods are used to prevent traffic analysis on each of the paired interfaces in the solution for the purposes of determining sensitive information. This will require the interface to be monitored during operation.	The various components that make up the solution exchange information. Where components are physically separate, a secure channel is required. Such secure channels should demonstrate data confidentiality and authenticity during the establishment and subsequent use of the channel to ensure that data sent is the data received, and that data is sent to the intended recipient. No secret or sensitive data should be transferred between the devices prior to the establishment of the secure channel. The contactless kernels may be contained within a single location, module, or library. However, the implementation of the contactless kernel can be split across several subcomponents, with some residing outside the CPoC application or COTS device, such as back-end processing or "cloud" environments. In addition, it is important that all solution components, including different software modules that exist on the same COTS device or server, such as attestation components and contactless kernel components, establish secure connections such that the communication cannot be tampered with or accessed without proper authorization. These secure connections may be implemented through cryptographic means using

Security Requirements	Test Requirements	Guidance
		a secure channel as defined in these requirements, or using properties provided by the COTS platform or server platform for the protection of the communications.
1.5.2 Each secure channel must provide mutual authentication to uniquely identify each component prior to exchanging sensitive data and protect against MITM and replay attacks. ***Note:*** *Mutual authentication between the communicating components must be based on cryptography that aligns with Appendix C Minimum and Equivalent Key Sizes and Strength for Approved Algorithms.*	**1.5.2.a** The tester must confirm that each secure channel uses mutual authentication and provide details on how this is implemented. **1.5.2.b** The tester must confirm that the cryptography used in all secure channels complies with the minimum cryptographic requirements of this Standard. **1.5.2.c** For each secure channel, the tester must detail how MITM and replay attacks are prevented. At a minimum, this requires the use of unique keys for each secure channel session.	It is important to uniquely identify each channel established between the various solution components so that the back-end monitoring system can detect and potentially flag changes to these components as tamper events. The specific mechanism used to establish a mutual authentication between each endpoint depends on the technique used to establish a secure channel. Examples of methods include digital certificates, key derivation techniques, and key negotiation techniques.
1.5.3 Cryptographic keys used to establish secure channels between the solution components and for data encryption must be unique, except by chance.	**1.5.3.a** The tester must confirm that any cryptographic keys used to secure the channels between the various components are unique to that component, except by chance. The tester must specifically detail how unique connections are achieved between the COTS device and any other component.	A different set of cryptographic keys is required for channel encryption versus data encryption to ensure key separation. The use of channel encryption through a secure channel does not meet the data-encryption requirements for account data. Account data should be encrypted separately using application and datagram level keys.
1.5.4 Use of standard protocols must prevent downgrade attacks.	**1.5.4.a** Where standard protocols are used, the tester must confirm that downgrade attacks are not possible, and that the solution does not permit the use of algorithms or key sizes that do not meet strong cryptography requirements. **1.5.4.b** The tester must perform a public-vulnerability search on the implemented protocol and version, and confirm that there are no exploitable vulnerabilities.	Many communication protocols support backward compatibility with previous protocol versions that may have security vulnerabilities. Therefore, solutions should ensure that downgrades to non-secure versions of the protocol cannot happen.

1.6 Correlatable Data

The term "correlatable" refers to a mutual relationship or association between one or more data elements, when the value of one data element can be inferred from the other. In the context of security, the ability to correlate data across platforms can produce a combination of data elements that invalidates the security assumption of data isolation (an important control in most security architectures). For example, if a data element from the CPoC solution has a high correlation with the consumer's PAN, but is itself not treated as confidential, (e.g., an email address), then someone with access to previously compromised PAN and associated email addresses could capture the email address from the CPoC solution as a way to confirm that the PAN remained valid for unauthorized use. A more serious case would be the correlation of data sufficient to reconstruct track data or sufficient to correlate PAN and PIN.

Since the solution provider cannot assure that databases outside of its control do not exist, the best the solution provider can do is identify what information created by or input into the CPoC application will have a high correlation to account data. From that, the solution provider can identify the countermeasures it has adopted to minimize the capture or leakage of such data elements.

Security Requirements	Testing Procedures	Guidance
1.6.1 Solution provider must maintain documentation that includes any data elements created or accepted as input that the solution provider reasonably believes could be used for correlation of account data to CPoC application transactions.	**1.6.1.a** Tester must confirm that solution provider has documented correlatable data elements. **1.6.1.b** The tester must confirm that the solution provider performs periodic, at least annual, review to confirm the documented correlatable data elements are accurate.	The intent is for the solution provider to demonstrate an awareness of this security risk and to have taken steps to minimize the amount of, or access to, any data that might facilitate out-of-band correlations that could result in reconstruction of tracked data or association of sensitive authentication data (SAD) to PAN.
1.6.2 Solution provider must document the countermeasures incorporated in the solution to minimize the potential for correlatable data.	**1.6.2.a** Tester must confirm that the solution provider has documented countermeasures for correlatable data.	

1.7 Operational Management

Oversight, governance, and responsibility for the solution are necessary to ensure all security controls are in place and functioning as intended.

Sound physical and logical security procedures and practices, and trained personnel responsible for the management of the solution are fundamental to the security of the solution. Incident response procedures should be in place that detail how to respond to any fraudulent or malicious activity, and these processes should be maintained, tested, and updated as required.

The solution provider should maintain a risk assessment and vulnerability-management program, and should document processes for determining and processing new vulnerabilities in the solution.

The intent of these requirements is to ensure that there are sufficient detail and processes in place to provide unfamiliar employees with the context to do their jobs executing and protecting the solution. The standard does not mandate any particular method or process for creating or managing this documentation, but it is not sufficient to rely on training alone without the backup or support of written documentation.

Organizations that manage the monitoring, attestation, and back-end processing environment are responsible for the implementation and ongoing maintenance of these requirements.

Security Requirements	Test Requirements	Guidance
1.7.1 Documented procedures to support the operation of the back-end environments must exist and be demonstrably in use.	**1.7.1.a** The tester must confirm that documented procedures for the operation of the back-end environments exist. **1.7.1.b** The tester must interview staff who are responsible for these operations and confirm that the documents are distributed, and processes are understood and implemented.	This requirement supports controlled operations and management of the environment and ensures common understanding for those involved in the operations of the environment.
1.7.2 The solution provider must have documented risk-assessment policy and procedures that are demonstrably in use and that provide details on: • Methods used to assess on-going risk of the solution • Thresholds for minimum acceptable CPoC application versions • How and when updates to the COTS system baseline are performed • How such changes are communicated to affected merchants The risk-assessment policy and procedures must be reviewed at least annually and when there are significant changes to the solution.	**1.7.2.a** The tester must obtain and review the solution provider's risk-assessment policy, update procedure documents, and confirm that these documents contain the following: • How to assess whether newly exposed vulnerabilities pose a risk to platforms • A requirement to reassess all supported COTS platforms at least every year and the reassessment method used • How and when updates to the COTS system baseline are performed **1.7.2.b** Where possible, the tester must compare the information in the policy with actual changes made to the	The solution provider should have documented risk-assessment policy and procedures, which is reviewed at least annually. This policy should include the methods used to assess on-going risk of the solution, how and when updates to the COTS system baseline are performed, and how such changes are communicated to affected merchants. • It is not acceptable for the policy to specify a minimum number of CPoC applications that can use a vulnerable COTS platform before it is removed from the COTS system baseline.

Security Requirements	Test Requirements	Guidance
	COTS system baseline to confirm that the policy is being followed. **1.7.2.c** The tester must confirm how merchants are informed when changes to the COTS system baseline affect their COTS platforms. **1.7.2.d** The tester must confirm that the risk-assessment policy has been reviewed within the previous 12 months. **1.7.2.e** The tester must review changes made to the solution within the previous 12 months (e.g., by reviewing change management tickets or committed source code changes), and confirm that risk-assessment policy and procedures have been reviewed when a significant change was implemented.	
1.7.3 A threat-management process must be established to monitor newly discovered vulnerabilities that may impact the security of the solution. A risk assessment of these vulnerabilities must be performed against currently implemented security and attestation controls to: • Determine the residual risk • Ensure that the vulnerability does not change the baseline integrity of the solution	**1.7.3.a** The tester must confirm that the solution provider has a documented threat-management process to discover new vulnerabilities that may impact the security of the solution. **1.7.3.b** The tester must confirm that the threat-management process includes methods for identifying new vulnerabilities, and specifies groups or individuals within the organization who are responsible for this process. **1.7.3.c** The tester must confirm that the threat-management process includes a method for assessing the risk of any vulnerability, and the tester must detail how this risk affects the solution. **1.7.3.d** The tester must detail how a new vulnerability affects the COTS system baseline and when a change to the COTS system baseline is required.	As vulnerabilities are being announced and discovered constantly, a process with resources to monitor proactively and evaluate each vulnerability as it is announced ensures that the organization is able to modify its detection and response process to accommodate new vulnerabilities.

Security Requirements	Test Requirements	Guidance
1.7.4 The solution provider vulnerability management process must be integrated with the threat-management process.	**1.7.4.a** The tester must confirm that the vulnerability-management program is finding, accepting, and remediating security vulnerabilities. The solution provider must demonstrate that any such vulnerabilities are processed through its risk and update process, and patched accordingly. **1.7.4.b** The tester must obtain a list of vulnerabilities, and validate that these vulnerabilities have been identified and addressed according to the solution provider documented vulnerability-management program.	With the assessment of new vulnerabilities, the solution provider should ensure that mitigation controls are in place to address any incremental risks, and that the new vulnerability is taken into consideration quickly in the development of the next product release. This allows accelerated integration of critical security patches that can be deployed through software updates to the solution in the field. Given the dynamic nature of the software risk/vulnerability discovery process, the solution provider should have a tightly integrated process in which the development team and the threat management team collaborate closely to identify and mitigate any threats to their solution and environment.
1.7.5 The solution must be tested least annually.	**1.7.5.a** The tester must confirm that there is a documented policy requiring the solution to undergo penetration testing at least every year, and that the scope of this testing includes all aspects of the solution, including the CPoC application and back-end systems (attestation, monitoring, and processing components). **1.7.5.b** The tester must obtain a list of vulnerabilities reported through the penetration testing, and validate that these vulnerabilities have been addressed according to solution provider documented vulnerability-management program.	In the face of new vulnerabilities and patches, regular testing of the solution and its components is necessary to confirm an ongoing security posture.
1.7.6 A public vulnerability-management program must be securely implemented and provide confidentiality of the reported vulnerability.	**1.7.6.a** The tester must confirm that the solution provider has a procedure that governs the acceptance and processing of new vulnerabilities through external communications. This requirement does not mandate the implementation of a "bug bounty" program, but does require that all reported vulnerabilities to be registered and processed according to a documented procedure. **1.7.6.b** The tester must confirm that there is a public-facing procedure for securely reporting vulnerabilities in the solution. This procedure must use methods to ensure the confidentiality of the vulnerability as it is reported. **1.7.6.c** The tester must obtain a list of vulnerabilities reported through the public vulnerability reporting program, and validate that these vulnerabilities have been	Public vulnerability-management programs are important to ensure the reporting of security vulnerabilities found by security researchers and members of the general public. This requirement does not specify the need for a “bug bounty” program; however, it does require that the solution provider have a secure process in place for accepting and reviewing submissions from sources outside their own company and direct-contract parties regarding the security of their system. For example, a procedure that requires the reporting of a vulnerability to a shared "info@[company]" e-mail address without additional encryption, would not meet this requirement. Use of a specific web portal secured with TLS (using acceptable cipher suites) and/or e-mails secured with strong cryptography are examples of acceptable

Security Requirements	Test Requirements	Guidance
	addressed according to solution provider documented vulnerability-management program.	methods for securing the confidentiality of vulnerability reporting.
1.7.7 Plans and procedures must be defined, and tested at least annually, to address interruptions to the solution due to unplanned business disruptions, major disasters or failures of service.	**1.7.7.a** The tester must confirm that the solution provider has a documented policy and procedures covering service disruption, and that this plan is tested at least annually. **1.7.7.b** The tester must confirm that the service disruption plan does not include bypassing controls that have been implemented to meet the requirements of this Standard, such as disabling or bypassing attestation controls.	Adequate preparation should be made for business continuity of the solution processing and security.
1.7.8 Changes and updates of any of the solution components, such as the back-end monitoring system, attestation system, or a remote component of contactless kernel, must follow a formal change-control process.	**1.7.8.a** The tester must confirm how modifications and/or updates to the back-end systems are performed, and confirm that changes follow a formal change-control procedure. This must be done before the update is applied in a production environment. **1.7.8.b** The tester must confirm that the documentation required by the change control process exists and has been followed for previous changes.	All changes to the solution components should be defined, documented, approved, and tracked so that any vulnerabilities or security weaknesses introduced by a change can be identified and resolved as quickly as possible. Problems related to undocumented changes take longer to trace and resolve, and thus places the solution at greater risk of attack or compromise.
1.7.9 Reviews must be performed at least quarterly to verify that operational procedures are being followed. Reviews must be performed by personnel assigned to security governance and include the following: • Confirmation that all operation-management processes are being performed • Confirmation that personnel are following security policies and operational procedures, such as daily log reviews, firewall rule-set reviews, and configuration standards for new systems	**1.7.9.a** The tester must confirm that the solution provider has a documented policy requiring quarterly reviews (at a minimum) of operational procedures. **1.7.9.b** The tester must read the results from past reviews of operational procedures to confirm these are being performed. ***Note:** For the initial compliance evaluation, results from a single complete quarterly review are sufficient. Where compliance is being re-evaluated for the purposes of a delta or ongoing recertification, the tester must confirm that the results dating back to the initial certification exist and comply with this requirement.*	Business-as-usual (BAU) procedures and processes should be followed to ensure continued security of the environment. BAU breakdowns lead to non-compliance but, more importantly, risk of security exposure. The intent is to provide evidence as requested for audits.

1.8 Secure Software Development Practices

Software is to be developed and maintained according to a defined software-security development process. Software developers require knowledge to address software vulnerabilities and emerging risks.

Development of secure software requires knowledge of common attack techniques and vulnerabilities. These vulnerabilities can change over time; therefore, a continuous process to inform software developers about these changes is vital. It is not sufficient to confirm that the programmers have been provided with documentation, books, and/or training on secure software development. There should be auditable confirmation that the developers have knowledge of common vulnerabilities in the language and environment in which they develop the applications.

These requirements address the development of software for all aspects of the solution, including the CPoC application, and the back-end systems (processing, monitoring, and attestation). Additional requirements specific to the CPoC application are stated in Module 2: Contactless Payments on COTS Application.

Security Requirements	Testing Procedures	Guidance
1.8.1 All software must be developed in accordance with the development process and to the development requirements described in Appendix D Software Security Requirements.	**1.8.1.a** The tester must provide a list of all areas of software used by the solution, identifying those areas developed by the solution provider and those that are provided by external parties, such as commercial OS, third-party libraries, and open source software. **1.8.1.b** The tester must confirm that all software developed by the solution provider is in accordance with the development process and the development requirements outlined in the Appendix D Software Security Requirements.	The application software should be developed and maintained in accordance with secure coding standards and best practices to reduce the risk of vulnerabilities that result from poor coding techniques. Knowledge of additional industry application development standards and best practices provides information on current exploits and trends, such as the following: • CERT secure coding standards • *Institute for Security and Open Methodologies (ISECOM)* • *Open Source Security Testing Methodology Manual* (OSSTMM) • *International Systems Security Engineering Association (ISSEA) Systems Security Engineering Capability Maturity Model (SSE-CMM)* • *ISO/IEC 21827*

1.9 Development, Maintenance and Dissemination of Solution User Manual

Merchants are to be clearly informed in the proper use of the solution to support a secure payment acceptance environment.

A solution user manual should be created and maintained. This document should describe how to implement payments using the solution, and detail any required processes or implementation specifics that are delegated to the merchant using the solution. Such information is contained in the CPoC solution *user manual* that is to be produced by the solution provider.

This manual should be available to any merchant using the solution, and any updates to this manual should be communicated to those merchants.

Security Requirements	Test Requirements	Guidance
1.9.1 A user manual that provides information about the solution, including identifying control points and security responsibilities for the merchants, must exist and be made available to the merchant.	**1.9.1.a** The tester must verify that a solution user manual exists and contains the following details: • How to implement and perform payments with the solution. • Any administrative tasks necessary for the solution. • Any security responsibilities that belong to the merchant when using the solution.	Documentation to support the use and management of the solution provides for common knowledge and definition of procedures to ensure that security controls are in place and functioning as intended. User manual information should address implementation and administration tasks associated with the solution, including guidance on maintenance for technical controls and processes, and procedures for downloading and verifying an authorized CPoC application.
1.9.2 The solution user manual must be disseminated to merchants who are using the solution at the time of onboarding or upon request.	**1.9.2.a** The tester must detail how the solution user manual is disseminated to merchants, and confirm that this method ensures that the merchant receives the manual at the time of onboarding or upon request.	Proper dissemination of the solution user manual to merchants is imperative to ensure that they have the information they need for a secure implementation of their solution. This is best handled at the time of onboarding of the merchant.
1.9.3 The solution user manual must be reviewed at least annually and upon changes to the solution, including the COTS device, the contactless COTS application, and the back-end systems. Any changes to the manual must be disseminated to merchants.	**1.9.3.a** The tester must confirm that the solution provider has a policy to review the solution user manual at least annually and update it as required. **1.9.3.b** The tester must note how any updates to the solution user manual are communicated to existing merchants, and how this ensures that they are aware of the update and have the ability to request, or otherwise obtain, and review the updated manual.	Merchants should have the latest version of the solution user manual, so that they have the information necessary to properly configure and implement the solution.

Module 2: Contactless Payments on COTS Application

Control Objective: The control objective is designed to ensure the COTS environment meets the security baseline, and that sensitive data is protected and not retained in the COTS environment for longer than necessary.

These requirements apply to the CPoC application and attestation components residing on the COTS device (if those components are not part of the CPoC application). The CPoC application may be instantiated in various parts or components from the following:

- Code that executes in the COTS rich execution environment
- Code that runs within a TEE or security processor
- Web or cloud-based functions, such as remote component of contactless kernel

To ensure that the security requirements are assessed properly, the entire scope of account data processing is to be understood and mapped. These requirements do not prohibit the use of remote component of contactless kernels or other non-local processing components for the developing of the CPoC application. However, the solution provider is to have clear documentation describing how the application is instantiated, and how each component is maintained and updated. Communications between each component is to be secured, and the sharing of cleartext account data between the various elements is to be minimized to reduce the risk of exposure.

2.1 Tamper and Reverse-engineering Protection

It is assumed that an attacker has full access to the software that executes on any unknown or untrusted platform, where that software may be a binary executable, interpreted bytecode, or other form as it is loaded onto the platform. Therefore, the software is to provide inherent protections that resist reverse-engineering of and tampering with the code execution flow. These protections may include, but are not limited to, the use of code obfuscation, internal integrity checks for code and processing flows, and code segment encryption.

Obfuscation can reduce the efficacy of common code decompilation tools. Obfuscation methods may include, but are not be limited to, control-flow and data obfuscation, execution of code sections in remote/cloud environments, and API renaming. Where the application is provided as a number of files (libraries, SDKs, etc.), note what protections are provided for the calls between the files.

These protections are not required across all code, but should be used to protect all code that provides CPoC application security features in the following ways:

- Increasing code complexity in a significant and demonstrable way
- Making execution possible only on unmodified environments, such as through use of a device physically unclonable function to encrypt data/execution
- Implementing the CPoC application code in a trusted application that can be executed only in a secure TEE or SE

This requirement is not intended to cover any tamper-detection or response features that may be provided by the combination of the CPoC application, attestation components and the back-end monitoring system, or inherently provided by the COTS platform itself; that is when the platform on which the CPoC application executes provides some form of tamper-responsive feature.

Security Requirements	Test Requirements	Guidance
2.1.1 Documentation must exist and be maintained on how tamper resistance is achieved for each of the supported COTS platforms, including, but not limited to: • Code obfuscation • Protections provided by specific platforms • Reliance on TEE, security processor, or other security features of the COTS devices used	**2.1.1.a** The tester must confirm that the solution provider maintains documentation covering the tamper-resistant features and the implementation of each supported COTS platform.	The solution provider and the various parties involved should document how tamper resistance is achieved for each supported COTS platform, where it leverages protections offered by the platform, and how it compensates for any security gaps. This information is critical for the labs to review and validate.
2.1.2 The CPoC application must be protected by tamper-resistance measures to protect its code, application, attestation interface code, and any code involved in the use or security of cryptographic keys (both public and private/secret keys) for all the supported COTS platform and protection methods (such as TEE, secure enclave, and *white-box cryptography*). ***Note:*** *Tamper-resistant measures can be implemented in the CPoC application, provided by the COTS platform, or a combination of both.*	**2.1.2.a** The tester must review documentation and other evidence provided by the application developer, and use this information to detail the protections provided to the solution against tampering and reverse-engineering as attested by the solution provider. **2.1.2.b** The tester must provide details of all areas where application-provided functions are executed. This includes the rich execution environment of the COTS device, but may also include other local or remote execution environments, such as a TEE, embedded secure processor, or remote component of contactless kernel. **2.1.2.c** The tester must outline all areas of the CPoC application that are protected by the tamper-resistant measures. This must include the contactless reading and interface code, memory and storage, and any code that is involved in the use or security of cryptographic keys (both public and private/secret keys) and the contactless kernel code. **2.1.2.d** The tester must document any protections provided by the CPoC application. These protections include, but are not to be limited to, compile-time protection options used, virtualized execution, or other hardware-level abstraction to the application operation. The tester must confirm that these protections apply across all supported COTS platforms (as assessed under Security Requirement 3.1.1), or detail any gaps in the coverage of these protections and justify why these gaps do not increase the risk posed by those platforms. **2.1.2.e** Where protections are provided (partially or wholly) through code obfuscation, the tester must:	Obfuscation should reduce the efficacy of common code decompilation tools. Obfuscation methods may include, but may not be limited to, control-flow and data obfuscation, execution of code sections in remote/cloud environments, and API renaming. Where the application is provided as a number of files (libraries), note the protections provided for the calls between the libraries. These protections are not required across all code, but should be implemented to protect all code that provides account data security features. These protections should increase code complexity, or limit execution on only unmodified environments through the use of a device physically unclonable function to encrypt data/execution or by implementing the CPoC Application code in a trusted application that can be executed only in a secure TEE or SE.

Security Requirements	Test Requirements	Guidance
	• Examine provided application installation files in which the protection methods have been applied. Compare these files with files in which protections have not yet been applied. Based on this comparison, comment on the validity of the solution provider attestations and documentation regarding implemented protection methods. • Comment on the comparative file sizes between unprotected and protected application samples. Also, comment on the comparative relative compression ratios of each file type when standard compression functions are applied directly to each obfuscated code segment. • Attempt extraction of data objects (such as ASCII strings and symbols) and functional linking/interface tables (such as PLT/GOT), and observe any differences between code sets before and after the obfuscation process. • Analyze and comment on the comparative code flow and linkage between the code before and after the obfuscation process. • Observe and comment on any areas of non-traditional execution, where the obfuscation relies on techniques such as virtualized/interpreted commands, non-deterministic operations, or polymorphic processes. It is expected that the applied tamper-resistance features will use one or more of these techniques, and the use of the obfuscation of the standard code execution flow will not be sufficient to meet these requirements. ***Note:*** *The intent of this test item is for the tester to detail the obfuscation security features implemented within the CPoC application code, so this information can be used during the analysis attempt to break the obfuscation.* **2.1.2.f** Where protections are partially provided by the COTS platform, the tester must: • Detail any public vulnerabilities that exist for the protection mechanism.	

Security Requirements	Test Requirements	Guidance
	• Confirm that the supported COTS platforms (as assessed in Security Requirement 3.1.1) provide the required tamper-resistance features and can be used for the CPoC application. • Detail the protections that are provided by the COTS platform, confirm how they provide the required tamper-resistance features, and determine whether the protections can be disabled or deactivated by the user or applications resident within the platform. • Detail any formal evaluation process that has been used to assess the security and efficacy of the tamper-resistance features of the COTS platform, such as the validation of a TEE implementation through industry-best-practice testing and/or a certification program. Such a TEE implementation need not be assessed through an external approval process, but the assessor must be aware of what testing has been performed to create a valid test process for this requirement. Where previous evaluation results are to be accepted and/or re-used, the tester must validate and provide evidence of their equivalency. ***Note:*** *The intent of this test item is for the tester to detail the CPoC application tamper-resistance security features as implemented and relied upon by the COTS platform, so this information may be used during the analysis attempt to break the obfuscation.* **2.1.2.g** The tester must attempt to circumvent the tamper-resistance protection and subvert the normal operation of the CPoC application to capture or compromise the account data entry and processing. This must be done without the tamper-detection and response features of the back-end monitoring system to prove the tamper-resistant protection alone. The tester must consider the use of state-of-the-art malware reverse-engineering techniques and attacks on DRM systems. The tester may use the code outside the COTS device on which it is normally targeted for execution.	

Security Requirements	Test Requirements	Guidance
	2.1.2.h The tester must document the process and attack results, and provide a costing of this attack based on the method outlined in Appendix B Software Tamper-responsive Attack Costing Framework.	
2.1.3 The attestation component must have the least privilege required to access proprietary APIs to determine the COTS platform state.	**2.1.3.a** The tester must confirm that all aspects of the attestation process can be implemented with the least privilege applied to the attestation component.	Attestation information and measurements reflect the state of the COTS platform running the CPoC application. For example, the COTS device vendor-supplied Google SafetyNet or third-party proprietary tools used to obtain measurement information should not require privilege escalation on the COTS platform.
2.1.4 Attestation code implemented in the CPoC application must be protected by tamper-resistance features.	**2.1.4.a** The tester must detail how the attestation components are implemented within the COTS platform, including what components and functions are provided by the solution provider code and what components and functions are provided by other code, such as third-party libraries or COTS platform functions. **2.1.4.b** The tester must confirm that the attestation code implemented in the CPoC application is protected by tamper-resistant features. **2.1.4.c** For parts of the attestation functions that are implemented by third-party systems outside of the control of the CPoC application, the tester must detail how these attestation functions are protected to prevent tampering.	Any attestation code in the CPoC application should be protected from reverse-engineering and any static or dynamic attacks that could subvert attestation processing.

Security Requirements	Test Requirements	Guidance
2.1.5 The contactless kernel, including any configuration files, optional settings, or payment brands public keys embedded or associated with CPoC application, must be protected by tamper-resistant methods to guarantee its integrity.	**2.1.5.a** The tester must provide details of all areas where the contactless kernel is implemented, including local and remote processing, configuration files, and public keys. **2.1.5.b** For each area or dataset, the tester must detail the methods that have been implemented to protect the integrity of the data and its use. ***Note:*** *These methods may involve use of File Integrity Monitoring (FIM), cryptographic signatures, or execution environments protected beyond the standard level of the OS implemented on the COTS platform.* **2.1.5.c** The tester must confirm that any inputs to the contactless kernel, such as configuration files or public keys, cannot be altered by other applications on the COTS platform.	Contactless kernels often allow the application of configuration options or different operating modes through files or system settings. Such changes can directly affect the security of the contactless kernel operation and should be controlled. Additionally, payment brands public keys form the root of trust of a contactless transaction and therefore should be protected.
2.1.6 The contactless kernel operation must be immutable, such that transaction processing cannot be interfered by other applications or users on the COTS device.	**2.1.6.a** The tester must attempt to interfere with the contactless kernel operation and confirm that unauthorized modification is not possible without detection.	Unauthorized modification of the contactless kernel may result in disclosure of sensitive information.

Security Requirements	Test Requirements	Guidance
2.1.7 The CPoC application must implement methods for detecting and reporting the following to the back-end monitoring system: • COTS devices that have been rooted, jailbroken, or in developer mode. • CPoC application that has been "side loaded" outside normal channels. • Status of COTS device sensors and hardware, as allowed by the COTS OS, that can be used to leak sensitive data. All these events must be reported under conditions that include, but are not limited to: – When the CPoC application is executed – As requested by the CPoC application and back-end attestation components – When white-box cryptography or obfuscation methods, implementations, or instantiations are updated	**2.1.7.a** The tester must document the detection and reporting methods provided by the CPoC application that detect rooted or jailbroken devices, devices in developer mode and application "side-loading." This may rely on external attestation components, either as part of the solution provider attestation system or as part of a solution provided by the COTS OS vendor. **2.1.7.b** The tester must review documentation and other evidence to confirm that the solution provider performs periodic analysis of possible side-channel attacks that use COTS device sensors and hardware (such as cameras and microphones) to leak sensitive data. **2.1.7.c** The tester must attempt to bypass the detection and reporting methods by gaining root on the COTS platform, installing root kits, unlocking the COTS bootloader, or any other method or combination of methods. **2.1.7.d** The tester must confirm that attestation components collect and report the status of COTS device sensors and hardware that may be used for side-channel attacks to the back-end monitoring system. **2.1.7.e** The tester must confirm that the detection methods are operable: • Upon execution of the CPoC application • As requested by the back-end monitoring system • When white-box cryptography or obfuscation methods are updated	Although a COTS platform may offer many security and/or tamper-protection mechanisms, rooting or jailbreaking a device could impact and weaken the overall security controls. Rooting and jailbreaking can also open avenues for a malicious user to install malware or exploit other vulnerabilities to harvest sensitive data or affect the integrity of the solution. Side-loading of applications (i.e., loading an application though channels other than the OS stores) may expose the COTS device to additional risks. For example, OS stores usually verify the identity of the software developers before allowing to publish their applications, and often conduct regular reviews of the applications before these are accepted. Moreover, to be able to install an application from a third-party app store, the COTS device often needs to be rooted or the enable "Unknown Sources" (allow installation from untrusted sites). Several side-channel attacks are using COTS device sensors and hardware (camera and microphone) to leak sensitive data. For example, a camera can be used by a malicious background process or application to capture account data read by the NFC interface. Motion and orientation sensors can be used in a side-channel attack to reveal user passphrases and PINs. The requirement does not prescribe what COTS device sensors and hardware can or cannot be used to leak sensitive data; the solution should account for specifics of the supported COTS platforms. The detection of attacks should enable attestation components to trigger a tamper-response mechanism, such as clearing internal buffers and prohibiting acceptance of account data.

Security Requirements	Test Requirements	Guidance
2.1.8 A CPoC application that fails tamper checks must be prohibited from accepting account data.	**2.1.8.a** The tester must confirm that the CPoC application does not accept account data if any tamper checks fail. **2.1.8.b** The tester must detail the methods used to prevent the acceptance of account data, and note if acceptance is permitted later if the tamper check has not failed. For solutions that allow for temporary suspension of account data acceptance, the tester must explain how this deters attacks that attempt to gain advanced privileges for a short period of time, and then return to normal operation after security changes have been made by the attacker.	If such security issues are detected, a tamper-detection response should be triggered to the back-end monitoring system, and the application should not be allowed to accept account data.

2.2 Software-Protected Cryptography

Protection of cryptographic operations and assets has been traditionally provided by tamper-responsive hardware devices such as HSMs or tamper-resistant hardware devices such as secure elements. Although these types of internal hardware systems are becoming more common in COTS devices, support for them is not universal. Frequently, their design and features may not allow secure implementation for a CPoC application.

Another method of protecting cryptographic operations and sensitive data is through software protections, such as white-box cryptography, where the cryptographic functions and storage methods used to protect the cryptographic keys are obfuscated such that extraction of the sensitive data, or tracing of the execution flow of the cryptographic process, is rendered computationally expensive.

Where purely software methods are used for the protection of these keys, such as with white-box cryptography, the specific instance used in a CPoC application is to be changed periodically, where the period is less than the estimated time it would take to reverse-engineer the software protections. At a minimum, changes to the white-box instance are to occur at least once per month. It is acceptable to have two sets of keys during the changeover period, but all old keys are to be invalidated when the new keys are installed. This requirement does not imply that different implementations or different algorithms are to be used each month.

Use of protection mechanisms that are inherent to the platform on which the CPoC application runs, such as hardware-backed keystore, may be acceptable in place of white-box cryptography. However, where such device-dependent security is relied upon, testing is to be performed on all types of protection provided. For example, where protection by a TEE is claimed for key security, each platform on which the CPoC application runs is to be confirmed to provide a secure TEE implementation. Combined approaches that use different protection methods for different platforms (depending on the security features provided by that platform) or hybrid implementations that use combinations of hardware- and software-based protections are also acceptable.

There are many different methods of software protection that can be applied to a cryptographic process; this Standard does not mandate, require, or endorse any particular method. However, the tester is expected to examine the methods used, including review of the implementation source code, to ensure that the specific methods provide robust protection of the cryptographic process and sensitive data it is using.

Security Requirements	Test Requirements	Guidance
2.2.1 Cryptographic methods protected primarily by software-based methods must be protected against analysis and abuse.	**2.2.1.a** The tester must detail how keys used with the software protection methods (such as white-box cryptography) are generated. Keys must have sufficient entropy, and the key value must not be exposed during the generation process. Generation of software-protected keys must use entropy from an RNG that is approved under this Standard. Where the white-box key-generation process is implemented outside of a secure cryptographic device, this device must be: • A stand-alone device; that is, without modems, not connected to a LAN or WAN, and not capable of wireless connections	This standard does not require the use of software-based methods to protect cryptographic keys, such as white-box cryptography. However, where software-based methods are used, the implementation should be validated to be robust against attacks and abuse. The methods of evaluation of the software-based protection mechanisms can include a full source code review (including code of the software-based protection mechanism) and should utilize a side-channel analysis such as monitoring the behavior of the application during the execution.

Security Requirements	Test Requirements	Guidance
	• Dedicated only to the generation function used for the software protection keys; that is, there is no other application software installed • Located in a physically secure room that is dedicated to software-protected key-generation activities **2.2.1.b** The tester must confirm that these code areas/functions are integrated into the overall CPoC application and are protected by the tamper-resistance features to prevent easy extraction of the code and/or use of this code as an encryption oracle. **2.2.1.c** The tester must evaluate the robustness of the implemented software-based protection mechanism. The tester must detail: • How the software protection method is implemented, including security for the cryptographic keys • Whether the protection method has undergone any formal certification or evaluation process • Whether the protection method differs between platforms due to differences in COTS OS or other COTS platform-specific features • How the protection method prevents side-channel and algebraic attacks, such as DCA or BGE. This must consider virtualized environments, monitoring of program address/data access, and execution flow using methods, such as statistical analysis, code lifting, and exploitation of cryptographic oracles/APIs, to recover information about the cryptographic keys. **2.2.1.d** The tester must detail any cryptographic implementation features that protect against fault injection attacks. At a minimum, the tester must consider fault injection through, but not limited to, the following: • Requiring local physical interaction with the device running the CPoC application, such as through the use of EM Fault Injection or CPU cache timing attacks • Direct injection of faults into the code execution through manipulation of the execution environment as made possible through direct control of that execution environment, such as in a virtualized context	

Security Requirements	Test Requirements	Guidance
	• Manipulation of the execution environment through an interface to hardware features, such as clock/voltage settings and direct/indirect memory access using "RowHammer-like" attacks **2.2.1.e** The tester must confirm what physical test, debug, or in-circuit emulation features exist on the supported COTS platforms, and consider these in any attack methods and costings. **2.2.1.f** The tester must provide a costing of an attack to determine, extract, or modify the cryptographic keys used by the CPoC application for security services. The tester must perform this costing using the method outlined in Appendix B Software Tamper-responsive Attack Costing Framework. This costing must take into account the difficulty in bypassing all tamper-resistance and attestation component features of the solution where applicable.	
2.2.2 The robustness of the software-based protection mechanisms must be evaluated, at least annually, against current attack scenarios and vectors.	**2.2.2.a** The tester must confirm that the solution provider analyzes the software-based mechanisms for robustness at least annually. **2.2.2.b** The tester must review the result of the software-based protection mechanisms analysis and confirm that the security controls (such as frequency of change of cryptographic material) are adequate.	The solution provider should evaluate the design and robustness of the software-based protection mechanisms to identify the applicable attack scenarios, and the results of that analysis should be documented. The frequency of the evaluation needs to be at least annually, or more frequently based on the *CPoC* solution provider threat-risk management. Documentation should describe the following: • Aspects of the code that could be attacked, including tasks or actions that frameworks and libraries conduct on behalf of the software-based protection mechanisms • The difficulty in mounting a successful attack • How widely the program will be distributed • The mitigation techniques that are used, such as how the operating system security functions are leveraged • Ways to measure the likelihood and impact of an exploit
2.2.3 The cryptographic material used in software-based protection mechanisms, such as white-box keys, entropy seeds and nonces, must be changed	**2.2.3.a** The tester must detail the solution provider's application update policy and confirm that this includes at	Frequently changing the white-box implementation and encryption keys used to protect data increases the security of the solution substantially. When encryption is

Security Requirements	Test Requirements	Guidance
periodically (at least monthly) to prevent cryptographic key compromise.	least a monthly update of the CPoC application white-box keys. **2.2.3.b** Where white-box cryptography is used, the tester must confirm that the implementations are recompiled at least monthly using new entropy seeds, nonces, and transformation tables for the generation of new keys. Implementations can be recompiled more frequently per analysis required in *Security Requirement 2.2.2*.	performed in software, it is critical to change frequently the white-box key and any tables that perform the encryption to prevent unauthorized disclosure. The cryptographic material change frequency should reflect the risk analysis and the costing of an attack to determine, extract, or modify the cryptographic keys used by the CPoC application for security services.
2.2.4 Retired cryptographic material used in software-based protection mechanisms must be securely deleted no later than six months after initial deployment of CPoC application versions using those keys.	**2.2.4.a** The tester must detail what methods are implemented to securely delete retired software-protected cryptographic keys and confirm that the solution provider policy ensures that any keys more than six months old are not used.	At a minimum, changes to the cryptographic material used in software-based protection mechanisms are to occur at least once per month. It is acceptable to have two sets of keys during the changeover period, but all previous keys and other cryptographic material are to be invalidated when the new keys are installed. To prevent misuse, white-box keys, transformation tables, and other cryptographic material are to be retired no later than six months after the initial deployment to OS stores of the application versions using those keys.
2.2.5 The cryptographic material used in software-based protection mechanisms must not be used directly for account data or attestation data encryption.	**2.2.5.a** The tester must detail the exact use of any cryptographic material used in software-based protection mechanisms, and confirm that these keys are not used directly for the encryption of account data or for directly securing any other communications that are part of the overall solution security, such as attestation data.	Given that identical account data will be encrypted each time a particular card is used in any given instantiation of the solution, not using a static white-box cryptography key reduces the likelihood of same-text attacks. Additionally, the effective security life of a white-box key is likely to be much shorter than the useful life of account data. Therefore, using such keys increases the risk that encrypted account data would be vulnerable to future decryption.
2.2.6 Cryptographic keys that are protected primarily with software-based methods must be unique per CPoC application version and instance of the OS store.	**2.2.6.a** The tester must outline how cryptographic keys are unique per CPoC application installation instance, and how the use of the common white-box keys is minimized after the secure provisioning process. The details must include how each of the cryptographic keys protected with software-based methods are used, how they are involved in the secure provisioning process, and what use the cryptographic keys have (if any) after the secure provisioning process is complete.	To reduce the possibility of re-using a compromised white-box key, white-box keys should be unique for each type of OS store (such as App Store and Google Play) and geographical region.

Security Requirements	Test Requirements	Guidance
2.2.7 Cryptographic algorithms and keys used in software-based protection mechanisms must meet security requirements in Section 1.3 Acceptable Cryptography.	**2.2.7.a** The tester must confirm that cryptographic processes and cryptographic material used in software-based protection mechanisms meet the security requirements in Section 1.3 Acceptable Cryptography.	Software-based protection mechanisms, such as white-box cryptography, should use the most robust and current encryption algorithms and key sizes to withstand modern-day attacks and ensure support into the future. Legacy algorithms have a limited shelf life and may lose effectiveness over time. Use of recognized cryptographic methods provides assurance that industry-tested and accepted algorithms with appropriate key lengths provide effective key strength and proper key-management practices. Proprietary or “home-grown” algorithms do not provide this assurance and are not permitted. Cryptographic algorithms used in software-based protection mechanisms should be obfuscated. Within white-box cryptography, the allowance of algorithm reformatting and obfuscation techniques of algorithms may impact the flows, but not the results. Furthermore, additional steps, extra rounds, and calculations are allowed to produce noise and further obfuscate the algorithm.
2.2.8 Cryptographic keys used in software-based protection mechanisms must meet the key management requirements described in Section 1.4 Key Management.	**2.2.8.a** The tester must confirm that key management processes used in software-based protection mechanisms meet the security requirements Section 1.4 Key Management.	White-box cryptography processes should use the most appropriate key management techniques, as defined in Section 1.4 Key Management. Techniques that protect white-box cryptography algorithms and keys vary greatly and are implementation dependent. Within white-box cryptography solutions, entropy and diversification strategies should incorporate randomness, as defined in Section 1.2 Random Numbers. Key diversification is a core key management technique for white-box solutions. The diversification process should be protected leveraging various obfuscation and white-box cryptography techniques. Within white-box cryptography solutions, there may be higher reliance on automated key generation. white-box cryptography key generation can further leverage device or specific user account information to support key generation. There is a number of ways to implement dual control and split knowledge through logical mechanisms, physical mechanisms, or both for the solution generating the automated keys.

Security Requirements	Test Requirements	Guidance
		Keys in white-box cryptographic solutions should be protected and not directly accessible to an attacker. Keys should be wrapped or protected prior to entering the white-box solution. Where possible, storing keys should be reduced or eliminated; the use of key diversification techniques can help. When stored, keys are often embedded or merged into the algorithm in such a way that they do not appear in memory as a simple string of bytes. There are various techniques and mechanisms that can be used to further protect strings of code that represent a key in a white-box solution, and developers should leverage those capabilities. The use of cleartext keys stored in memory or code when the algorithm is executed is not recommended. Traditional side-channel and fault-injection countermeasures can be used to further protect the keys in white-box cryptographic solutions. White-box cryptography solutions should not allow key exports; that is, cryptographic keys should only be imported.

2.3 Online Processing

All transactions are processed “online.” Online refers to the transmission of the financial message to the remote host during the performance of the transaction. Where online connectivity is not available, the transaction processing is to be prevented. If connectivity drops during a transaction and the transaction must be reversed, all customer data is to be erased from the COTS device immediately.

Security Requirements	Test Requirements	Guidance
2.3.1 All payment processing must be performed online.	**2.3.1.a** The tester must confirm that the solution provider documentation indicates that the solution accepts and processes only contactless transactions when there is online connectivity enabling the transaction to be authorized. **2.3.1.b** The tester must disable network connectivity on the COTS device and attempt to perform a transaction. To satisfy this requirement, the transaction must not be permitted. Disabling of the network connectively must consider the following: • Deactivating data connectivity for a cellular network. • Connecting to a local network, such as Wi-Fi or Bluetooth, that is not connected to the Internet. • Disabling all network connections. **2.3.1.c** The tester must start a transaction where network connectivity is available, and then disable this connectivity after initializing the transaction but before it completes. The exact time at which to disable the online connectivity may vary based on the solution; the tester must detail and justify the testing process used.	To help prevent fraud, it is important that each transaction is sent online and validated. This involves validating the cryptogram for contactless EMV-based transactions, or the dynamic card verification code for magnetic-stripe data (MSD) transactions, and possibly performing other checks at the issuing side.

Security Requirements	Test Requirements	Guidance
2.3.2 All components of the solution must be online and in an operational state before initiating any contactless transactions.	**2.3.2.a** The tester must confirm that the CPoC application establishes online connectivity to the back-end systems before allowing contactless card entry. **2.3.2.b** The tester must disconnect the COTS device from the Internet while a transaction is in progress. Start the application and allow initial connections/checks to complete, perform a transaction, and then disconnect the COTS device from the Internet using each of the following methods: • Disconnect the network on the COTS device, such as by disabling the Wi-Fi. • Disconnect the network gateway so the COTS device remains connected to the local network, but is unable to connect to the back-end systems. • Set specific rules in the local network connection to drop or incorrectly route packets to back-end systems, but continue to allow other connections. **2.3.2.c** The tester must document the processes and results of the tests, and confirm that contactless transactions are not accepted during the tests.	As most of the security controls and mechanisms involve the ability of the solution to monitor and mitigate through controls located and coordinated by the back-end systems, the ability to be connected to the back-end environment is critical. Therefore, the CPoC application and the back-end systems should be connected to execute the various controls, such as attestation functions.

2.4 Application Authenticity

The authenticity of the CPoC application is of paramount concern when securing account data entry.

Security Requirements	Test Requirements	Guidance
2.4.1 The CPoC application must include methods to allow for the merchant to validate the authenticity of the application through separate channels.	**2.4.1.a** The tester must detail the methods implemented by the CPoC application to allow the merchant to authenticate the application. **2.4.1.b** The tester must confirm that application authentication method implements a message that is unique to the merchant. **2.4.1.c** The tester must confirm that the solution provides guidance to the merchant that this message should be checked upon installation of the CPoC application. **2.4.1.d** The tester must detail how this information is conveyed to the merchant and how this ensures that fraudulent applications do not simply remove this message from their display.	Because the CPoC application can be downloaded easily from OS store and used without any further hardware, it is important that merchants are able to validate the application. This may involve directing the merchant to the phone help desk or webpage of the payment service provider and quote a uniquely generated value to confirm the authenticity of the application. This validation method should be detailed clearly in the solution user manual.
2.4.2 Mechanisms must exist to uniquely identify and authenticate each instance of the CPoC application to the back-end monitoring system and the back-end attestation component.	**2.4.2.a** The tester must confirm that mechanisms exist to uniquely identify and validate the CPoC application as authorized for use with the solution and accepting customer cards. The tester must detail these mechanisms and the criteria to authorize their use. **2.4.2.b** The tester must detail what methods are used to ensure that the CPoC application provides this unique ID upon installation.	The solution provider should ensure that each instance of the CPoC application is uniquely identified and authenticated to the back-end monitoring system. The solution provider should conduct a periodic risk assessment to ensure the adequacy of the authentication mechanisms and that controls remain in place.
2.4.3 The CPoC application must be able to display the current version of the application software on startup and upon request.	**2.4.3.a** The tester must detail the versioning method used by the CPoC application and confirm that there is a unique value for each released version of the application. **2.4.3.b** The tester must detail the methods provided by the CPoC application to display or provide the application version, and confirm that these methods are detailed in the solution user manual. **2.4.3.c** The tester must demonstrate that this method is implemented and displays the correct application version number.	The solution provider should ensure that the version of the CPoC application is uniquely identified, and that the CPoC application can display this version number to the merchant.

2.5 Secure Application

For the CPoC application to be considered secure, it is to be designed, developed, and maintained in a manner that protects the integrity of payment transactions and the confidentiality of all sensitive data collected, stored, or processed in association with payment transactions.

Security Requirements	Test Requirements	Guidance
2.5.1 Documentation must exist and be maintained to detail the following: • Protections provided to the application against tampering, side-channel attacks, fault injection, and reverse-engineering for the various supported COTS platform and protection methods, such as TEE and white-box cryptography. • Details of all areas where functions provided by the application are executed. This should include the *rich* execution environment of the COTS device, but may also include other local execution environments, such as a TEE or embedded security processor. • Data-flow diagrams that show how the account data is entered, processed, encrypted, and validated within the application, where the data is transmitted outside of the scope of the application and any assumptions made about these external connections. • Block diagram that indicates where all sensitive data is available in cleartext on the merchant COTS platform. This includes, but may not be limited to, the COTS OS and any TEE or physically separate security-processing elements used. This diagram must indicate the flow of sensitive data through the various elements. • Identification of where internal buffers are used and cleared when collecting sensitive data.	**2.5.1.a** The tester must confirm that the solution provider has identified all sensitive data used by the solution, and that this list is complete and accurate given the tester's understanding of the solution. **2.5.1.b** The tester must confirm that solution provider documentation exists that provides details about the operation, location, and security of all the identified sensitive data in the CPoC application. **2.5.1.c** The tester must confirm that the solution provider documentation includes data-flow and block diagrams that show where all sensitive data is transmitted throughout the solution. These diagrams must show from acceptance into the contactless reader of the COTS device to finalizing the transaction. **2.5.1.d** The tester must confirm that the documentation details where internal buffers are used and the measures that ensure that the buffers are cleared of sensitive data upon completion of the transaction process or application execution, whichever comes first. **2.5.1.e** The tester must confirm that the documentation is consistent with the CPoC application architecture.	The solution provider and the various parties involved should document their solution components sufficiently so that labs, assessors, and other entities can understand the security around the various components individually and as a complete solution. Where the solution includes API to allow other applications to interface with it, the documentation has to provide guidance on how to securely invoke CPoC API exposed by the solution.
2.5.2 Documentation must exist and be maintained to identify logical connections between the CPoC application and other components of the solution.	**2.5.2.a** The tester must confirm that documentation exists for each secure channel that is required. **2.5.2.b** The tester must confirm that the secure channel documentation is complete and accurate based on the tester's understanding and testing of the solution.	Documentation of the connections between the various components that make up the solution will assist in testing the solution. Documentation helps identify where each security control exists and how it has to be managed.

Security Requirements	Test Requirements	Guidance
2.5.3 The CPoC application must clear sensitive data automatically from the internal buffers and working memory it controls when any one of the following occurs: • The transaction completes • The transaction terminates for any reason during its normal execution • The CPoC application times-out waiting for the response from the cardholder or merchant • The CPoC application or back-end monitoring system signals a tamper-detection event • The CPoC application pauses or stops executing • The CPoC application loses its foreground focus.	**2.5.3.a** The tester must confirm that internal buffers are cleared under each of the following minimum set of conditions: • The transaction completes • The transaction terminates for any reason during its normal execution • The CPoC application times-out while waiting for the response from the cardholder or merchant • The CPoC application or back-end monitoring system signals a tamper-detection event • The CPoC application pauses or stops executing • The CPoC application loses its foreground focus **2.5.3.b** The tester must confirm that the internal buffer clearing method is robust and does not rely solely on "garbage collection." **2.5.3.c** The tester must examine and cite any relevant documentation to verify support for solution provider responses. Relevant documentation includes test results for inspections of internal buffers, user guides, the software specification or the software implementation submitted by the solution provider.	Sensitive data should not be retained any longer, or used more often, than necessary. Regardless of whether the sensitive data exists in memory, in cleartext, or encrypted, it should be cleared from the internal buffers that CPoC application controls. Account data should be encrypted within the CPoC application immediately after the NFC read is complete. The device supports the encipherment of the account data as part of a transaction flow; transferring it between the CPoC application and the payment back-end. Implement each merchant-side instance to clear the buffers as soon as practical after use. The buffer clearing mechanism should be robust against compiler optimization. Solely relying on "garbage collection" functions to clear buffer data is not sufficient to meet this requirement.
2.5.4 The COTS platforms supported by the CPoC application must provide for secure compilation and/or execution of software applications.	**2.5.4.a** The tester must identify any sections of the supplied application that are precompiled. For each precompiled section, the tester must confirm that compile-time protections are implemented according to best practices for the target COTS platform. **2.5.4.b** Where the precompiled code does not include protection against buffer and stack overflows, or where the COTS platform may be responsible partially or completely for the compilation of the code, the tester must confirm that the target COTS platform provides secure methods to protect the executing application.	Many COTS platforms supported by applications provide their own compilation or optimization of applications, or require that the application is submitted as source code to be compiled by the OS store. Supported COTS platforms should ensure that compilation is performed securely with appropriate best-practice measures, such as address space layout randomization (ASLR) and data-execution protections.

Security Requirements	Test Requirements	Guidance
2.5.5 The CPoC application must use a validated RNG function.	**2.5.5.a** The tester must confirm that CPoC application only uses validated RNG functions that meet the security requirements in Section 1.3 Random Numbers.	Random numbers are used in numerous software applications, including cryptography, to protect sensitive information. encryption keys and initialization values (seeds) are examples of random numbers commonly used in applications. It is important to have a good understanding of the installation, initialization, configuration, and usage - for example, initial seeding of the random function - of the RNG mechanisms to ensure that the implementation can meet the effective security strength required for the intended use. It is not a trivial endeavor to design and implement a secure random number generator. Software vendors are required to use only validated random number generator algorithms and libraries that pass NIST STS test program (defined in *NIST Special Publication 800-22*).
2.5.6 The CPoC application must access only those COTS platform resources required to perform its transaction processing.	**2.5.6.a** The tester must list all COTS platform resources used by the CPoC application and detail justifications for each. **2.5.6.b** The tester must confirm that each COTS platform resource has a functional or business purpose in processing contactless transaction.	Software is often used to execute functions on the underlying operating systems or accessible external resources. When software requires excessive permissions, those permissions could be exploited by a malicious user. To minimize its attack surface, the software should request and be granted the minimum required privileges for transaction processing.
2.5.7 The CPoC application must access only those information repositories required for transaction processing.	**2.5.7.a** The tester must list all information repositories used by the CPoC application and detail justifications for each. **2.5.7.b** The tester must confirm that each repository has a functional or business purpose in processing contactless transactions.	The application should not access any information resources other than those essential to complete transaction processing. Information repository could be on COTS platform or remote. For example, a CPoC application could require an access to contact information on the COTS device, or perform an API call to retrieve contact information from a remote (e.g., cloud-based) information repository.

Security Requirements	Test Requirements	Guidance
2.5.8 The CPoC application must not disable or interfere with any security features provided by the COTS platform.	**2.5.8.a** The tester must confirm that the CPoC application does not require the disabling or bypassing of any security features of the COTS platform on which it executes. **2.5.8.b** The tester must specifically note that the CPoC application does not require special developer or administrative privileges to execute on the platform or to meet the requirements of this Standard.	This requirement ensures that platform security features do not need to be disabled for the application to run.
2.5.9 The CPoC application must initiate only those inbound and outbound network communications required to support the application's functions.	**2.5.9.a** The tester must detail all network connections made by the CPoC application, which of these are initiated by the application, and confirm that each network connection has a business purpose.	This requirement does not apply to network communications in which the application may generically access the file system, which may result in the platform accessing remotely mounted drives/shares.
2.5.10 The CPoC application must encrypt all sensitive data before transmission. ***Note:** A secure channel cannot be used as the sole security and encryption mechanism. Protocol-level encryption, such as TLS, does not meet this requirement, which is asking for application level encryption.*	**2.5.10.a** The tester must refer to the list of all sensitive data that can be communicated by the CPoC application and detail how this data is encrypted before transmission beyond the boundary of the CPoC application. **2.5.10.b** The tester must confirm that this encryption occurs at the application layer and that communications-layer security, such as TLS, is not relied upon solely for the protection of this data.	
2.5.11 Implementations must ensure that neither cleartext secret nor private cryptographic keys are exposed as cleartext in the COTS OS memory, except for the shortest feasible time while used for a cryptographic operation.	**2.5.11.a** The tester must detail the methods that secure the cryptographic keys that are used and operate within the COTS OS. **2.5.11.b** The tester must note if the cryptographic key security methods allows the keys to be exposed in cleartext in the COTS device OS, and if so, where this happens. The tester must detail how these exposed keys are permanently erased immediately after they are used. **2.5.11.c** The tester must specify the time period during which any cleartext secret or private key is exposed in the COTS OS, explain why this is the shortest feasible time, and explain why this time does not compromise secure operation of the CPoC application.	Use of TEE, hardware key stores, separate security processing environments, or white-box key obfuscation are examples of methods that may be sufficient to prevent exposure of cleartext keys. Storage of persistent credentials (e.g., secret keys, PKI private keys, or passwords) and any file that may potentially contain sensitive data (including temporary files) should be minimized, and be protected while stored.
2.5.12 The CPoC application must not support PIN or customer biometric entry on the merchant's COTS device.	**2.5.12.a** The tester must confirm that the CPoC application does not allow for the entry or use of customer PINs or biometric data on the COTS device.	For privacy and security reasons, biometric match-on-device or match-on-card, where the data is sent to the card by the terminal, is explicitly forbidden.

2.6 Secure Provisioning

As the CPoC application is downloaded from a single instance in an OS store, each application installation is initially identical to all others. Use of any common values, such as cryptographic keys stored in white-box cryptographic form, is to be minimized upon installation and first use. The CPoC application should use methods to ensure immediately upon installation that the instance is unlike any other and can be identified uniquely and secured through communication with the back-end systems. This is to be done so that such keys are not used to secure remote communications or expose data outside the application after installation and first use.

Loading applications from the OS store provides a level of confidence that the application has not been tampered prior to being installed on the merchant COTS device. The solution is to detect when the CPoC application has been side-loaded outside of normal channels and treat this as a tamper detection event.

Where the CPoC application allows or requires download of additional data from the back-end systems, such data should also be signed cryptographically and authenticated by the CPoC application before use or execution. Reliance upon the secure channel between the CPoC application and the back-end systems is not sufficient. This requirement implies that each datagram exchanged between the COTS device and the back-end systems has an individual signature or (*H*)MAC applied.

The scope of this requirement is the authentication of the application by the COTS OS. Additional authenticity checks are expected to be applied and checked by the back-end monitoring system or attestation system, or applied as part of the obfuscation methods; however, these are beyond the scope of this requirement.

An authenticated encryption mode that has been approved by NIST or other international standards bodies may be used instead of a discreet signature or (*H*)MAC.

Security Requirements	Test Requirements	Guidance
2.6.1 There must be a clear definition of all COTS platforms, including device types, hardware, and operating systems, on which the CPoC application can be executed. This definition is the COTS system baseline (see Section 3.1 COTS System Baseline).	**2.6.1.a** The tester must confirm that the solution provider has a clear definition of all COTS platforms on which the CPoC application is supported. ***Note:*** *The tester may refer to testing performed in Security Requirement 3.1 when responding to this item.* **2.6.1.b** The tester must detail all the OS stores supported by the *CPoC* solution provider.	Although these requirements are designed to allow for the NFC read of account data on a COTS platform that has not been assessed directly for security, it is expected that the solution will have criteria for which platforms are considered acceptable for reading account data through the NFC interface embedded in the COTS and which are not. For example, it is expected that solutions using older COTS OS that may contain unpatched vulnerabilities will not be acceptable for reading account data. The solution provider is required to have a clear method for determining the suitability of any COTS platform; this may be a whitelist, blacklist, or hybrid approach, but should clearly demonstrate a risk analysis of a COTS platform that accounts for any known or potential vulnerabilities in each merchant device. The COTS platforms requirements address the need for the CPoC application to be targeted to a limited subset of all available devices, and that the developer should have undertaken some risk analysis and mitigation steps to identify which platforms are suitable and secure.
2.6.2 CPoC applications must be developed only for supported COTS platforms—the COTS system baseline.	**2.6.2.a** The tester must confirm that any CPoC application is developed for use only on COTS platforms that meet the requirements of the COTS system baseline.	Where operating systems are no longer supported, security patches might not be available to protect the COTS platform from known exploits, which poses a significant risk. Unsupported operating systems expose the device, applications, and data on the device to unauthorized disclosure and modification. All new solutions should ensure they operate on supported COTS platforms only.

Security Requirements	Test Requirements	Guidance
2.6.3 The CPoC application must be installed and updated only through the OS store.	**2.6.3.a** For each OS store used for CPoC application deployment, the tester must confirm that signatures validate the integrity of the application in its entirety. **2.6.3.b** The tester must install the CPoC application from the OS store and note the web address from which it was installed. If the CPoC application has not yet been publicly released, then the OS stores beta testing features must be used (e.g., Apple TestFlight or Google Play Closed Release). **2.6.3.c** The tester must confirm that this signature is checked before installing the application. The tester must note any acceptable COTS system baseline where the signature is not (or cannot be) checked on executable code after installation onto the COTS device—for example, where Ahead-of-Time compile features are implemented, and the resulting compiled code is not provided with cryptographic authentication mechanisms that can be checked on execution. ***Note:** This requirement does not intend to prevent the use of such COTS platforms; however, it is expected that additional attestation component protections may be required in such instances.* **2.6.3.d** The tester must attempt to install the CPoC application using other mechanism (e.g., slide-loading) and confirm that it does not operate.	The authenticity of the CPoC application is a paramount concern in securing account data. Loading of applications from the OS store provides a level of confidence that the application has not been tampered before being installed on the merchant COTS device. Third-party OS stores are not allowed. Where the CPoC application allows or requires the download of additional data from the back-end monitoring and attestation systems, such data should also be signed cryptographically and authenticated by the CPoC application before use or execution. Reliance upon the secure channel between the CPoC application and the back-end monitoring system and attestation component is not sufficient. This requirement implies that each datagram exchanged between the device and the back-end monitoring system or the back-end attestation component has an individual signature or (*H*)MAC applied. The validation of this datagram signature/MAC is provided by back-end systems outside of the execution environment of the CPoC application.
2.6.4 CPoC application must be protected from unauthorized COTS OS or CPoC application rollback.	**2.6.4.a** The tester must detail the methods used by the solution to prevent roll-back of CPoC application versions. **2.6.4.b** The tester must confirm that attempts to perform COTS OS and CPoC application rollbacks are detected by the attestation system.	COTS OS rollback often used to revert to a more vulnerable version of the OS, which may permit compromise of cryptographic material.

Security Requirements	Test Requirements	Guidance
2.6.5 Methods must be implemented to protect the OS store interface used to upload the CPoC application and deployed CPoC applications from malicious alteration or misappropriation. ***Note:*** *Security of the OS store themselves are beyond the scope of these requirements.*	**2.6.5.a** The tester must detail the methods implemented by the solution provider to assist in securing or detecting the compromise of the OS store interface used to upload the CPoC application. **2.6.5.b** Where automated methods are implemented, the tester must confirm that these methods do not increase the risk of exposing the credentials used to upload content to the OS store, such as by storing the passwords in cleartext or by exposing the passwords to multiple personnel. **2.6.5.c** Where passwords alone are used as the authentication method for the OS store, the tester must confirm that the solution provider is implementing PCI DSS-*compliant* password controls for this purpose.	The OS store to which the CPoC application is uploaded, and from which all instances are downloaded to the COTS devices themselves, is a potential target for compromise. Although the local security of the OS store themselves are beyond the scope of these requirements, the *CPoC* solution provider should implement methods to prevent or detect any unauthorized changes to the assets deployed to the OS store. The Standard does not prescribe specific mechanisms to implement these controls. For example, a vendor could implement manual methods, such as two-factor authentication on the OS store login, adopt automated systems to check regularly for the assets loaded into the store, or implement a split knowledge for OS store passwords.
2.6.6 Any required cryptographic keys or other data necessary for first execution must be securely provided to the CPoC application and securely stored.	**2.6.6.a** The tester must detail the provisioning process using a message flow diagram. The tester must confirm that provisioning occurs over a secure channel that protects against MITM and replay attacks, and protects the confidentiality and integrity of the data communicated. **2.6.6.b** Where a standard protocol (such as TLS) is not used, the tester must identify which areas of the messages provide the MITM and replay protections. **2.6.6.c** The tester must confirm that only approved cryptography is used that meet the security requirements in Section 1.3 Acceptable Cryptography. **2.6.6.d** The tester must detail the methods used to ensure that any white-box cryptographic keys used meet the security requirements in Section 2.2 Software-Protected Cryptography.	As the CPoC application is downloaded from a single bundle on an OS store, each application installation initially is identical to all others. The solution should have methods to ensure that each application instance is unique, and can be identified and secured through communication with the back-end monitoring system and the back-end attestation components. This process should be performed on first execution.

Security Requirements	Test Requirements	Guidance
2.6.7 Secure provisioning must implement the principles of perfect forward secrecy.	**2.6.7.a** The tester must confirm that the secure provisioning process implements principles of perfect forward secrecy to ensure that any future compromise of the initial keys used during the provisioning process does not expose keys that already have been established and/or used.	Because each initial instance of the CPoC application is identical to all others, the application should undergo a process upon first execution to provision unique values so that the CPoC application can be identified to the back-end systems. The provisioning process also should deploy unique cryptographic keys that are stored securely, and used for the security of sensitive data managed and transferred by the CPoC application. It is expected that such keys can be discovered within a short time, and there should be perfect forward secrecy implemented in generating these keys such that discovery of a current or previous key does not enable a faster discovery of any future keys.
2.6.8 CPoC application executables and scripts must be digitally signed, and a signature must be provided to confirm the software author and to guarantee that the application (and any updates) from the OS store have not been altered or corrupted since it was last signed.	**2.6.8.a** The tester must document any additional scripts, data, executable files, interpreted commands, or other information downloaded by the CPoC application after installation.	Where the CPoC application allows or requires download of additional data from the back-end monitoring system and the back-end attestation components, such data should also be signed cryptographically and authenticated by the CPoC application before to use or execution.
2.6.9 Digital signatures used to sign CPoC application executables and scripts must be verified cryptographically prior to use of the application and at required attestation intervals.	**2.6.9.a** The tester must detail all cryptographic signatures implemented in the CPoC application and how they are applied. **2.6.9.b** For each signature contained in the CPoC application, the tester must confirm that the signature is authenticated cryptographically before using the signed data. **2.6.9.c** The tester must confirm that any signature can be validated by request through the back-end attestation systems.	The CPoC application may be authenticated through the use of multiple signatures. These signatures may be validated by the COTS OS and the back-end monitoring system.

Security Requirements	Test Requirements	Guidance
2.6.10 The process to generate digital signatures used to sign CPoC application executables and scripts must be performed using dual control on cryptographic keys that are secured within an HSM approved to at least *FIPS140-2* Level 3 (or equivalent in FIPS 140-3) or *PCI HSM*.	**2.6.10.a** The tester must confirm that the CPoC application signing must be performed using a dual-controlled process with cryptographic keys that are maintained in one of the approved forms (as defined in *Security Requirement 1.4.4*). ***Note:*** *This requirement does not apply to signatures that can be generated only by a third party, such as the OS store. However, all signatures that are generated by the solution provider must meet this requirement, even if available on the OS store.* **2.6.10.b** The tester must confirm that any non-application signatures, such as signatures applied to data sent to the application for processing, are also applied using secure hardware that satisfies this requirement. **2.6.10.c** For CPoC applications that rely in part on security provided by a separate execution environment, such as a TEE or remote host, the tester must confirm that any code or data loaded into this execution environment uses methods that permit the validation of its authenticity. **Note:** *File Integrity Monitoring (FIM) or other integrity monitoring solutions may be used to meet this requirement.*	Use of COTS devices introduces additional risks because it relates to privacy, unauthorized disclosure, and exposure to vulnerabilities. Therefore, it is imperative to establish trust through the use of digital signatures to ensure that the solution components know which other components are authentic, and that the data exchange to and from components is intact and has not been altered. Use of dual control and cryptographic keys further enhances the trust of digital signatures by ensuring that the processes creating the digital signatures conform to industry-acceptable practices. To ensure the digital signatures are authentic, authentication should be performed within an environment that meets industry standards. The signing of COTS applications may be performed without the use of an SCD, but tamper-resistant systems (such as smartcards) that store and use of cryptographic keys are to be used where direct SCD use is not possible. Any non-SCD used should be assessed as compliant to industry-standard security requirements, such as *FIPS140-2* (or equivalent in FIPS 140-3), Common Criteria or *PCI HSM*.
2.6.11 The CPoC application must be packaged such that its removal results in the deletion of the application and all associated data from the COTS device.	**2.6.11.a** The tester must confirm that CPoC application is using only COTS platform standard application package format. **2.6.11.b** Where CPoC application generates or downloads sensitive data, the CPoC application must provide a method to render unreadable any associated data. **2.6.11.c** The tester must detail the removal methods and perform a test removal of the CPoC application to confirm that this process works as documented.	The CPoC application should use COTS platform standard application package format, files, and directory structure (Android Package [APK] used by the Android operating system) for distribution and installation. Moreover, the CPoC application should refrain from downloading any other files, libraries, or other resources to the COTS device to ensure that their removal using the COTS device application management will result in the deletion of the CPoC application and all associated data from the COTS device.

Security Requirements	Test Requirements	Guidance
2.6.12 Code that handles, secures, or otherwise affects the security of the account data read through the NFC interface and processing on the COTS platform must be separated logically from code that is used for other purposes, such as general merchant UI.	**2.6.12.a** The tester must detail the code structure of the CPoC application and confirm that methods are provided to separate logically the account data processing code (or code that provides security to the account data processing code) from other parts of the code. This must include the following: • Data-flow diagrams that show how the account data is entered, processed, encrypted, and validated within the CPoC application. • Where the data is transmitted outside of the scope of the CPoC application and any assumptions made about these external connections. **2.6.12.b** The tester must confirm that this isolation is sufficient and explain how this separation allows for easy isolation of code segments, and therefore ensures that changes made to one area of the code do not affect areas that are isolated from it. ***Note:*** *Where the testers cannot provide this justification, or where code isolation is provided through a simple use of different functions or code files, this requirement must be marked as non-compliant.*	Areas of the CPoC application that are not involved directly with the processing or handling of the account data, or with providing security services and interface to the back-end monitoring system, can be updated without affecting the security code. This isolation may be achieved in many ways, but simply having different functions within the same body of code for security and non-security functions are not considered sufficient isolation. Multiple layers/levels of separation of the code may be implemented. For example, the code used for cryptographic key storage may be isolated logically from the code used for account data protection, and both of these code segments logically isolated from the overall merchant UI code. An example of how this can be met is to establish separate libraries that are signed individually and cryptographically.
2.6.13 Where third-party libraries are used, the CPoC application must be packaged with only those libraries that are used by the CPoC application. The libraries used must not have known and unpatched security vulnerabilities.	**2.6.13.a** The tester must detail any third-party libraries or code-sets used by the CPoC application and confirm that each library or code-set is up to date with the latest security patches. **2.6.13.b** The tester must confirm that the packages and code-sets in the CPoC application are only those required for the implementation. **2.6.13.c** The tester must confirm justification from the solution provider for each package based on the end-use operation of the CPoC application.	Third-party libraries should provide only those functions that actually are used by the CPoC application. Unnecessary functions may expand the available attack surface that can be exploited by a bad actor.

2.7 Audit Logs

To ensure that any specific actions or processes undertaken by the CPoC application can be validated or reviewed later by another party, it is vital to keep sufficient audit logs. The purpose of these logs is not to encapsulate and record every action taken by the CPoC application, but to ensure there is sufficient detail to reconstruct past events in response to review demands, such as audits or forensic examinations. Details should show what happened, what data was involved, who was involved in decision-making and actions taken, and when those decisions and actions transpired.

Security Requirements	Test Requirements	Guidance
2.7.1 The CPoC application must communicate securely the generated audit logs to the back-end monitoring system.	**2.7.1.a** The tester must detail the creation processes for all logs produced by the CPoC application and their contents. **2.7.1.b** The tester must confirm that these processes function during normal operation and cannot be disabled. **2.7.1.c** The tester must confirm that the logs are communicated securely to the back-end systems periodically; that is, at least once every 24 hours when the application is executing and always prior to processing any new transaction.	Application logs help provide individual accountability, reconstruction of events, intrusion detection, and problem identification. Logs should be transmitted frequently, as the COTS device cannot be relied upon for log storage.
2.7.2 Audit logs generated by the CPoC application must not contain sensitive data.	**2.7.2.a** The tester must confirm that audit logs do not contain sensitive data, including account data.	Sensitive information should not be included in audit logs because they may not be protected in the same manner.
2.7.3 Audit logs generated by the CPoC application must support reconstructing the following events: • All user access to sensitive data. • All activity that impacts security functions of the CPoC application, such as changes to cryptographic functions, changes to application permissions, and failure or success to establish secure channel with back-end monitoring system. • All access to the audit trail managed by or within the CPoC application. • Use of and changes to the CPoC application identification and authentication mechanisms. • Initialization, stopping, or pausing of the CPoC application logs.	**2.7.3.a** The tester must confirm that the audit logs produced by the CPoC application include sufficient information to reconstruct the following events: • All user access to sensitive data. • All activity that impacts security functions of the CPoC application, such as changes to cryptographic functions, changes to application permissions, and failure or success to establish a secure channel with the back-end monitoring system. • All access to the audit trail managed by or within the CPoC application. • Use of and changes to the CPoC application's identification and authentication mechanisms. • Initialization, stopping, or pausing of the CPoC application logs.	Logging of security events enables an organization to identify and trace potentially malicious activities. While the correlation and analysis of the event could occur on the back-end monitoring system, the CPoC application should be able to capture and communicate securely events that could be used by the back-end monitoring system.

Security Requirements	Test Requirements	Guidance
2.7.4 All recorded events must capture at least the following information: • User identification • Type of event • Date and time • Success or failure • Origination of event • Identity or name of affected data, system component, or resource	**2.7.4.a** The tester must confirm that the audit logs produced by the CPoC application include at least the following information: • User identification • Type of event • Date and time • Success or failure • Origination of event • Identity or name of affected data, system component, or resource	By recording these details for the auditable events, a potential compromise can be identified quickly and with sufficient detail to know the who, what, where, when, and how associated with the potential compromise.
2.7.5 All application audit logs must be time-synchronized with the back-end systems.	**2.7.5.a** The tester must detail the methods that ensure correlation between the timestamps on each aspect of the audit log to ensure that it is possible to align events between the disparate solution components in use.	The different components of the solution, such as COTS device, CPoC application, attestation components, back-end monitoring system, and back-end payment processing environments, may all operate in different time zones or have different time settings. The audit logs should be configured to ensure that it is possible to correlate the content with the events in each component. The CPoC application does not have to rely on the COTS device time, but instead could check and maintain an offset between the time on the COTS device and the time on the back-end systems.

2.8 Contactless Read of Account Data

The account data read process should be protected against manipulation or subversion. Attempts to modify, replace, or subvert the customer prompts, keyboard, or other UI features that are important for solution security should be prevented. Although these requirements consider the COTS platform beyond the scope of specific testing, some minimum requirements are considered for the NFC interface that are used to accept the customer account data. It is possible for the CPoC application to gain exclusive access to the NFC interface such that data passed between the card and application cannot be "sniffed" or otherwise monitored by other applications.

This requirement is to be assessed against all OS types and variants that are identified in Section 3.1 COTS System Baseline. The scope of this assessment should exclude malicious modification of these COTS OS to enable features specifically for collecting customer account data. However, modifications to a standard OS that may be performed by a COTS device vendor, distributor, or carrier that are not necessarily designed for account data capture, but may be repurposed maliciously or used for such purpose, are included in the scope of this assessment.

The intent of this requirement is to validate that CPoC application communication with the NFC interface *cannot be monitored* by another resident application.

Note: *The NFC interface is to be physically contained within the COTS device. This standard does not allow for the use of external contactless readers or antennas.*

Security Requirements	Test Requirements	Guidance
2.8.1 The CPoC application must reside and execute on the same COTS device as the NFC interface that is accessed to accept customer account data.	**2.8.1.a** The tester must confirm for all platforms supported by the CPoC application that the NFC interface used is physically integrated into the COTS device, and that there are no functions or methods that would allow for the use of an external NFC interface, even if the interface resides on another COTS platform running another instance of the CPoC application.	To ensure the security and protection of account data, CPoC applications should both reside and execute on the same COTS device as the NFC interface to prevent attack vectors that attempt to exploit vulnerabilities associated with any separation.

Security Requirements	Test Requirements	Guidance
2.8.2 The CPoC application must attempt to lock the NFC interface to make sure it cannot be used by other applications during the contactless read from a consumer card or device.	**2.8.2.a** The tester must confirm that CPoC application attempts to maintain exclusive access to NFC interface during the contactless read, and document the method used. **2.8.2.b** The tester must attempt to access the NFC interface during CPoC application contactless read and document the outcome of this test.	A COTS device can have more than one application that is authorized to use the NFC interface. To prevent other applications from monitoring, or "sniffing," data exchanged between the consumer's payment card or device, the CPoC application should use mechanisms and APIs made available by the COTS OS to gain an exclusive access to the NFC interface, or rely on the controls performed by the COTS OS. When a COTS platform allows only applications running in the foreground to send and receive NFC data, the CPoC application can meet the intent of the requirement ("locking" of NFC interface) by remaining in the foreground or reacting to its loss of foreground focus. The CPoC application can also attempt to intercept and claim a priority over other applications and processes that are registered to handle NFC data exchange, thus ensuring that account data cannot be read unintentionally.
2.8.3 The CPoC application must attempt to lock the COTS device camera to ensure that it cannot be used by other applications during the contactless payment transaction.	**2.8.3.a** The tester must confirm that the CPoC application provides methods to either detect or prevent the use of the camera by other applications. Where detection is used, the CPoC application must prevent the acceptance and processing of account data while another application is accessing the camera. **2.8.3.b** The tester must confirm that the CPoC application is able to either detect or prevent access of the camera when it is in the foreground, and document the method and outcome of this test. **2.8.3.c** The tester must confirm that the CPoC application is able to detect all COTS device cameras (e.g., when COTS device have multiple cameras).	To prevent unauthorized visual capturing of account data from a consumer payment card when it is in proximity to the COTS device, the CPoC application should attempt to prevent other applications and processes running on the COTS device from using the camera. The expectation is that other applications on the COTS device are not able to use the camera when the CPoC application prompts the consumer to initiate a contactless payment. The ability of the CPoC application and the method used to prevent other applications from using the camera device will depend on the COTS platform. For example, the CPoC application running in the foreground could invoke the COTS OS API to connect to a camera device that will result in any lower-priority (background) application to lose control and prohibit use of the camera. When the COTS platform does not allow the CPoC application to programmatically block other applications from using a camera, the CPoC application could prompt the user to disable the hardware manually and not permit to initiate a contactless read until the camera is disabled.

Security Requirements	Test Requirements	Guidance
2.8.4 If the CPoC application detects that it is being run in developer or emulator mode, the application must not be permitted to initiate a contactless payment transaction from a consumer card or device.	**2.8.4.a** The tester must detail the methods used by the CPoC application to detect developer mode, or that it is being run in an emulator. **2.8.4.b** The tester must confirm that if either of these developer modes is detected, the CPoC application is prevented from reading a contactless card or device through the COTS NFC interface. **2.8.4.c** The tester must attempt to enable developer mode on the COTS device and confirm that the CPoC application is able to detect these attempts. Where methods are found to circumvent the developer mode detection, the tester must detail these methods and confirm that the protections used by the CPoC application comply with industry best practices. **2.8.4.d** The tester must attempt to execute the CPoC application in an emulator and confirm that the CPoC application is able to detect these attempts. Where methods are found to circumvent the emulator detection, the tester must detail these methods and confirm that the protections used by the CPoC application comply with industry best practices.	Developer tools, emulator mode, and similar tools provide flexibility when developing applications, but may circumvent security controls required for production environments. Therefore, controls should be established to ensure that when these non-secure modes are present, sensitive information and functions are not displayed or performed. This is specific to the production-level CPoC application.
2.8.5 The following events must be detected by the CPoC application during contactless payment transaction , and must result in termination of the session and deletion of all data collected during the transaction, including account data: • The CPoC application or back-end attestation component signals a tamper-detection event. • The CPoC application detects that it is executing in developer or emulator mode. • Another application obscures the CPoC application. • The CPoC application pauses or stops executing. • The CPoC application loses its foreground focus.	**2.8.5.a** The tester must detail the methods used by the CPoC application, COTS OS, or a combination of both to detect switching between applications, losing focus, and access to the camera or NFC interface by any other application. **2.8.5.b** The tester must confirm through testing of representative samples of the supported COTS OS that tamper-detection event or switching context from the CPoC application to another application during contactless read of account data terminates processing and deletes any data collected during the transaction. Tests must include both manual change of focus (such as switching to another application) and automatic change of focus (such as during an incoming phone call or text message). **2.8.5.c** Where system messages provide pop-up, pop-in or pull-in dialogs that cannot be detected or disabled, the tester must explain why these notifications cannot be used to steal the account data as it is read.	The CPoC transaction process should be protected against manipulation or subversion. Attempts to modify or overlay the cardholder prompts (e.g., instructions to the cardholder) or other UI features that are important for the security of the solution should be prevented. Pausing of the application usually means that the application is still partially visible, but it is an indication that the user is interacting with a different dialogue or screen (e.g., in multi-screen environment). When user switches to a different application, application is considered "stopped" until either the user switches back or the system destroys the instance of the application. The security of the other applications on the COTS device is not known and therefore should not be trusted.

Security Requirements	Test Requirements	Guidance
2.8.6 The CPoC application must not store account data in persistent storage.	**2.8.6.a** The tester must confirm that the CPoC application maintains only account data in temporary storage for the time needed to perform the transaction. The tester must confirm that all locations that store account data are erased after the transaction completes or terminates, regardless of whether the transaction is successful. **2.8.6.b** The tester must confirm that the CPoC application does not use functions that return account data to the COTS device after the transaction terminates. **2.8.6.c** The tester must confirm that account data is not passed or otherwise made available to any other application.	The CPoC application should prevent storing account data in a way that allow other software to have access. These storage locations, depending on the COTS OS, include internal storage (private application storage), internal cache files, external storage (SD card), USB mass storage, preference and properties files, logs, and local databases. While some data storage is considered temporary (internal cache files) and can be cleared by the COTS device OS when it is low on internal storage space, those files can still be maintained by the CPoC application and can remain on the persistent storage for a considerable time.
2.8.7 The CPoC application must truncate PAN when providing customer receipts, either printed, electronic, or both using methods compliant with PCI DSS controls.	**2.8.7.a** The tester must identify any receipts provided by the solution and detail the methods by which they were provided, such as SMS, printed, or email. **2.8.7.b** For each receipt production method, the tester must confirm that the implemented PAN truncation methods comply with PCI DSS controls, and that no SAD is provided on the receipt. **2.8.7.c** The tester must confirm that these controls truncate the data before being communicated to the customer device or printer, and that the solution does not rely on methods in external devices to truncate data.	Customer receipts are necessary for customer validation of the transaction and as part of a formal payment challenge process. Where such data is sent from the back-end monitoring system for display or printing in the merchant environment, the receipt should be truncated to ensure that it cannot be uniquely correlated with the customer and that the full details are not available to the merchant. The intent of truncation is to remove a segment of PAN data permanently, so that only a portion (generally not to exceed the first six and last four digits) of the PAN is available on the COTS device.

2.9 Account Data Encryption

In many COTS platforms, the account data passes through several layers of software before being sent to the back-end systems. The CPoC application should be designed to accommodate specific methods of reading contactless data on each of the supported COTS platforms in a way that minimizes the exposure of the account data on those platforms.

Once read and processed, the account data should be encrypted as soon as possible, and always prior to any external transmission. This encryption should not be solely provided by the mechanisms of the secure channels used and should be applied at the application layer, specifically to the elements containing account data.

Where remote or split contactless kernel implementations are used, the account data should be protected during transmission between the contactless kernel subcomponents in the same way; that is, using application-layer cryptography with secure channels between each element.

Security Requirements	Test Requirements	Guidance
2.9.1 Account data must be encrypted within the CPoC application as soon as it is received by the application and always prior to transmission outside of the COTS device. Account data must remain encrypted when transmitted through a secure channel. ***Note:*** *A secure channel cannot be used as the sole security and encryption mechanism. Protocol-level encryption, such as TLS, does not meet this requirement, which requires for application-level encryption.*	**2.9.1.a** The tester must detail the process used by the CPoC application for handling account data from entry through transmission and to any subsequent return of that data from the remote host. **2.9.1.b** The tester must confirm that account data is encrypted as soon as practicable, and always before transmission outside the COTS device. **2.9.1.c** The tester must confirm that the encryption of the account data occurs before the transmission through the secure channel, and that the encryption inherent in the secure channel itself is not relied upon to secure the account data. **2.9.1.d** The tester must confirm that upon encryption of the account data, any remaining account data on the COTS device is deleted permanently, and that the transaction process does not return this data to the COTS device. Where a remote component of contactless kernel (i.e., split contactless kernel implementation) requires passing data back and forth between the COTS device to explicitly process the transaction, all remnants of the account data must be permanently deleted from the COTS device at the end of the transaction.	Encryption of the account data as it enters the CPoC application from the NFC interface and as it is transmitted to the back-end payment processing environment is essential, and sets the expectation of account data protection throughout the solution. Because account data encryption is performed in software within the CPoC application, additional measures are required to ensure confidentiality of the encrypted account data and the processes performing the encryption.

Security Requirements	Test Requirements	Guidance
2.9.2 Encryption used to protect account data must be performed using a key that is unique for each transaction/communication session.	**2.9.2.a** The tester must detail the key management used by the CPoC application and confirm that there is a unique key for each individual reading of account data through the NFC interface. **2.9.2.b** The tester must verify that the key associated with the account data reading is permanently deleted after the transaction terminates, regardless of whether the transaction was successful. **2.9.2.c** The tester must confirm that the key management used ensures that each unique key cannot be calculated from the previous key, such as through use of variants.	Requiring unique keys for each transaction and communication session ensures that compromised keys cannot be used in subsequent transactions. Examples of methods that ensure single-use symmetric keys include key derivation techniques and key negotiation techniques.
2.9.3 Encrypted account data must be protected from malicious activity.	**2.9.3.a** The tester must confirm that the cryptographic keys used to encrypt the account data are accessible and useable only by the CPoC application. **2.9.3.b** The tester must confirm that the use of unique keys per transaction also ensures that the replay of previously encrypted account data is not possible, and that any replay is logged as a security event by the attestation system.	The CPoC application should provide assurance that encrypted account data is not vulnerable to misuse, such as a replay attack or using building tables for space-time tradeoff attack to find the cryptographic key.
2.9.4 The integrity and confidentiality of the account data must be cryptographically protected wherever they are stored or processed.	**2.9.4.a** The tester must confirm that cryptographic processes and cryptographic material, such as random numbers, cryptographic algorithms, and keys used by the CPoC application to protect account data, meet the security requirements in Section 1.2 Random Numbers and Section 1.3 Acceptable Cryptography. **2.9.4.b** The tester must confirm that key management processes used by the cryptographic protection mechanisms, meet the security requirements in Section 1.4 Key Management.	Use of recognized cryptographic methods assure that industry-tested and accepted algorithms with appropriate key lengths provide effective key strength and proper key-management practices. Proprietary or "home-grown" algorithms do not provide this assurance and are not permitted.

Module 3: Back-end Systems—Monitoring/Attestation

Control Objective: Assurance that components in the solution are in a secure state, and the ability to react and address anomalies is fundamental to the overall security of the solution. Monitoring and attestation set the framework for this assurance.

Attestation is the interaction between a verifier (possibly server-based) and a prover (possibly client-based) to determine the current security state/behavior of the prover based on predefined measurements and thresholds provided by the prover. For the purposes of this document, attestation may be based on a hardware or software-based verification. Monitoring is the real-time interaction between the COTS device and CPoC application to the back-end monitoring and attestation systems.

Attestation may be demonstrated using a protocol between the prover and the verifier that provides the measurements to the verifier. The measurements may be determined in various ways, such as through a health-check interface that can be accessed by the prover. Attestation provides necessary assurance to the verifier that established and expected security controls at the prover are in an acceptable state and have not been modified. Organizations developing CPoC applications, designing or managing attestation systems and the solution providers are subject to these requirements.

The solution may implement various types of attestation. For the solution, the attestation health checks will be performed on varying components (provers): COTS platform and the CPoC application. Two verifier types and two prover types are presented in Table 2 corresponding to possible locations of the verifier.

***Note:** During attestation, the prover is assumed to be untrusted and the verifier is trusted. During Type 1 and Type 2 attestations, if the CPoC application attestation component has the role of the verifier, it may itself be compromised. Therefore, the security model is to account for this risk when using the results of Type 1 in Type 2 attestations provided by the CPoC application.*

Table 2: Contactless Payments on COTS Solution Attestation Types and Components

Type	Proven	Verifier	Purpose
1	COTS platform (through various sampled measurements)	• CPoC application attestation component • Back-end attestation component	Verifies that the COTS platform security model is intact. The assurance for Type 1 attestation relies on the inability of the attacker to spoof the measurements that are performed or, by the time it is possible for spoofing to be reliably performed, the presence of the attacker in the COTS platform has been detected by attestation systems and appropriate action taken. A CPoC application instantiated attestation and response may be limited due to limited processing availability and security afforded to local storage of measurement parameters. In contrast, an attestation call performed by the back-end attestation component (required) can be more robust because parameter checking is performed in close association by the back-end monitoring system.
2	CPoC application (attestation component)	Back-end attestation component	Verifies that both the security model of the CPoC application and its COTS platform are intact.

- The CPoC application attestation component is the process on the COTS platform used by the CPoC application to manage attestation. It may perform the role of the verifier and the prover. For example, in the role of verifier, it may perform an attestation of the COTS platform (as the prover) by taking measurements and comparing these with locally stored information (followed by any necessary action). In the role of prover, it may service a remote software attestation request sent from the back-end attestation component (as the verifier) and return the results to the server.
- The back-end attestation component (a server-based attestation component) is a process that manages attestation. It performs the role of verifier.

Figure 5 shows examples of attestation flows corresponding to each attestation type.

Figure 5: Attestation Flows

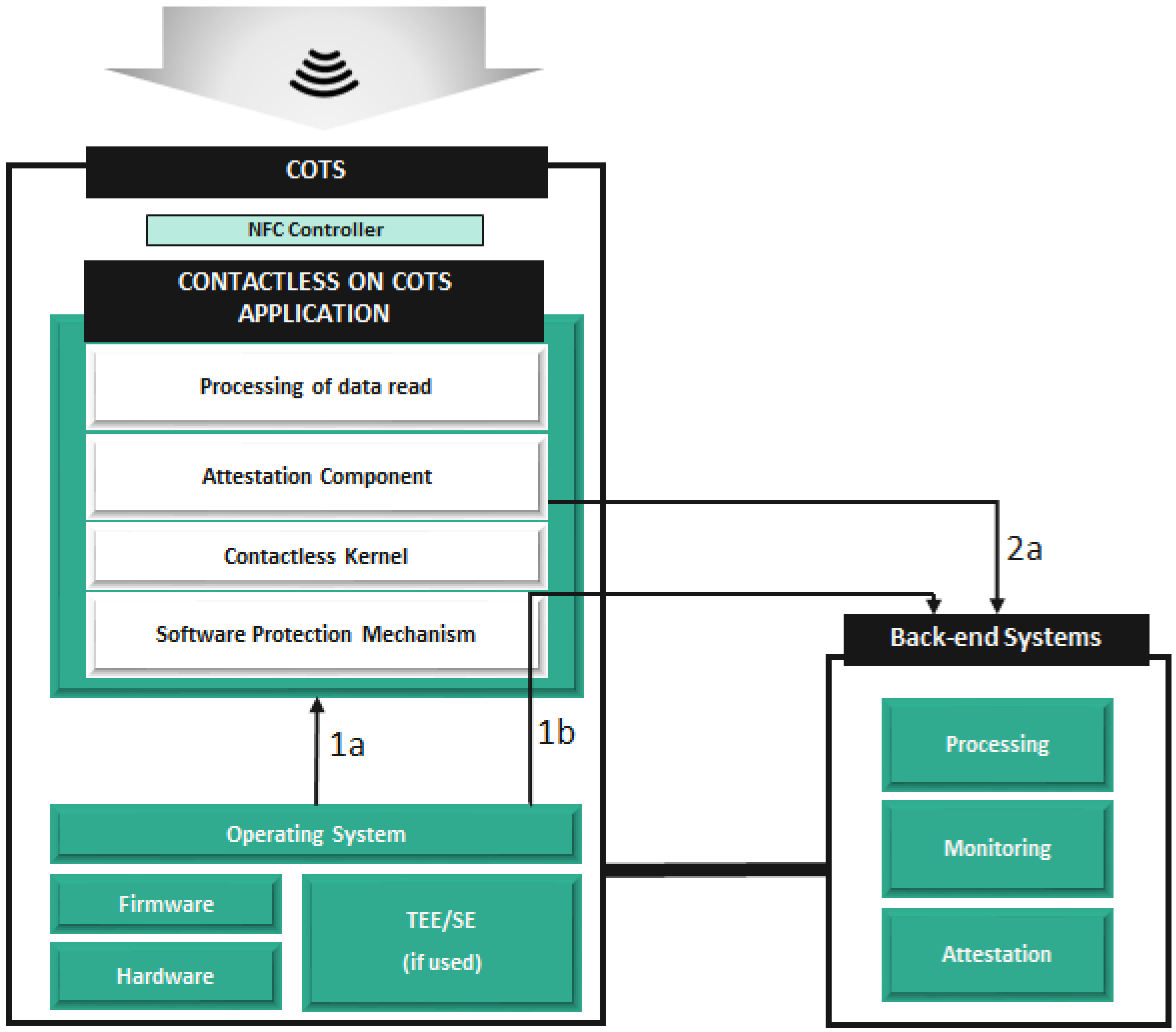


Figure 5 shows software attestation flows corresponding to each attestation type. For example, a Type 1a attestation is initiated by the CPoC application attestation component (as verifier) and sampled measurements from the COTS platform are returned to the CPoC application attestation component for local action according to the attestation policy. On the other hand, a Type 1b attestation request originates from the server (as verifier) and is processed by the attestation component of the CPoC application (protected by software protection mechanisms), which returns any required sampled measurements of the COTS platform (as prover) in the response to the back-end attestation component for further processing and action.

In Type 2a attestation, the CPoC application is the prover and the verification is performed using an external agency (such as back-end attestation component). This approach ensures that if the CPoC application execution environment is compromised completely, the evaluation of the attestation data collected cannot be manipulated.

3.1 COTS System Baseline

A defined set and state of COTS platforms, COTS devices, and COTS OS, on which the CPoC application may be executed is to be specified. The COTS system baseline is a subset of all currently deployed COTS platforms. The attestation process is required to define which of those COTS platforms is secure for use by the CPoC application based on data about current attack methods, new vulnerabilities, or other relevant information.

It is expected that this COTS system baseline will change over time. As a result, the process performed by the attestation component to determine the COTS system baseline will not be a single "point in time," but instead a process that assesses the threat environment continually and allows for changes to account data entry and processing.

Confirming that the COTS platform is in, and remains in, the COTS system baseline is part of the attestation process. The solution provider is responsible for establishing and maintaining COTS system baselines.

Security Requirements	Test Requirements	Guidance
3.1.1 Documentation must exist and be maintained for the following: • Implemented processes to determine the COTS system baseline for acceptance of COTS devices, such as whitelist, blacklist, or hybrid approach. • How implemented processes account for known and potential vulnerabilities in the COTS platform. • Clear identification of roles and responsibilities for which aspects of the COTS system baseline validation process are performed by the CPoC application itself and which are performed by other COTS platform components or execution environments.	**3.1.1.a** The tester must confirm that the solution provider must maintains a list of supported COTS platforms. **3.1.1.b** The tester must specify what security process is involved in the identification of supported COTS platforms and the method for selection, such as whitelist or blacklist. **3.1.1.c** The tester must detail the method employed by the solution provider for determining whether a COTS platform is acceptable or not based on known and potential vulnerabilities. **3.1.1.d** The tester must detail which aspects of the COTS system baseline validation process are performed by the CPoC application and which are performed by other COTS platform components or execution environments. The tester must confirm that these details are in the solution provider documentation.	Documentation helps to establish common knowledge of the security controls and COTS system baselines to understand how attestation is performed. Processes and risk management decisions that underlie the management of the COTS system baseline should be specified and comprehensive in the documentation.

Security Requirements	Test Requirements	Guidance
• Processes that are demonstrably in use for the discovery and remediation of bugs and vulnerabilities in the COTS platform.	**3.1.1.e** The tester must confirm that the solution provider has a documented process that is demonstrably in use for the discovery and remediation of security vulnerabilities for every supported COTS platform within the COTS system baseline.	
3.1.2 Documentation must exist and processes must be demonstrably in use that identify methods used for updating the COTS system baseline as new threats are identified.	**3.1.2.a** The tester must confirm that documented procedures exist and are demonstrably in use that manage changes to the COTS system baseline. For example, it is expected that such changes will effectively disconnect some merchants using COTS platforms that were previously acceptable, but now fall outside the acceptable COTS system baseline.	The COTS system baseline will change over time. This means the process performed by the attestation system will not be a single "point in time" check, but instead an on-going process that assesses the threat to the environment continually and allows for decisions to be made about the security of the solution at a platform level. The solution provider should be able to demonstrate a process for managing such instances and other events that may result from changes to the acceptable COTS system baseline. Procedures that rely on waiting for potentially vulnerable COTS platforms to become less common and unused by merchants are not satisfactory for this requirement.
3.1.3 The initial COTS system baseline must include only COTS OS versions that are supported by the OS vendor with security patches.	**3.1.3.a** The tester must confirm that the initial COTS system baseline developed by the solution provider includes only devices for which patches are available from the OS vendor. **3.1.3.b** Where the COTS system baseline accepts the use of COTS products for which security patches are no longer supported by the OS vendor, the tester must explain why the acceptance and use of such platforms for accepting account data does not increase the risk of account data exposure or subversion of the payment process beyond the use of devices that are supported by security patches.	While there is a large number of COTS device manufacturers, the number of COTS OS is much smaller, with iOS and Android being the two largest players. For this requirement, the minimum acceptable baseline is tied to the COTS OS where OS vendor maintains security patches for that particular version of the COTS OS. Exceptions to this requirement may be considered for COTS platforms where the supported COTS devices have changed over the period of deployment. However, brand-specific compliance rules may apply in this case.

Security Requirements	Test Requirements	Guidance
3.1.4 The COTS system baseline must only include COTS platforms that allow applications to maintain control over NFC interface, hardware, and sensors that can be used to read account data while in foreground.	**3.1.4.a** The tester must confirm that COTS system baseline includes only COTS platform that allow application to maintain control over NFC interface. **3.1.4.b** The tester must confirm that the CPoC solution identifies COTS device hardware and sensors that can be reasonably used for a side-channel attack to read account data, correlatable data, or both. **3.1.4.c** The tester must confirm that COTS system baseline include only COTS platforms that allow applications to control COTS device hardware and sensors that could be used for a side-channel attack.	Since the CPoC solution relies on the COTS OS security controls to prevent other applications from monitoring, or “sniffing,” data exchanged between the consumer’s payment card or device and the CPoC application, the COTS platform should have mechanisms that would allow a CPoC application to maintain control over NFC interface, hardware (e.g., camera) and COTS device sensors that could be used to read account data. For example, some COTS platforms do not allow background applications to initiate an NFC read or to use a camera hardware. Other COTS OS could expose API to allow applications running in the foreground to “lock” the use of NFC or camera, to prohibit use of these by other lower-priority (background) applications.
3.1.5 The COTS system baseline must include only COTS platforms that, at minimum, provide the following features: • An enforcing mandatory access control framework. • A trusted boot mechanism that validates the operating system’s authenticity. • Validation of an application cryptographic signature upon loading and execution of that application.	**3.1.5.a** The tester must confirm that the COTS system baseline includes only platforms that provide the following at a minimum: • An enforcing mandatory access control framework. • A "trusted boot" mechanism that validates the COTS OS’ authenticity. • Validation of an application signature upon loading and execution of that application.	To ensure the security of the solution, the CPoC application should be enabled on only a COTS device that meets the minimum acceptable criteria. The solution provider should undertake some risk analysis and mitigation steps to identify which platforms are suitable and secure.
3.1.6 The COTS system baseline must not include “rooted” or “jailbroken” devices.	**3.1.6.a** The tester must confirm that the COTS system baseline does not include devices that are rooted or jailbroken. **3.1.6.b** The tester must detail what protections are provided to detect rooted or jailbroken environments. **3.1.6.c** The tester must detail how effective these protections will be, perform tests that attempt to bypass the detections, and detail the results of the tests.	To provide reasonable assurance that the COTS system baseline reflects a secure, trusted state of the environment, the baseline should be free from influences that could negatively impact or affect the integrity of the baseline, such as devices that have been compromised.
3.1.7 The COTS system baseline must only include COTS platforms that support secure distribution of the applications.	**3.1.7.a** The tester must detail all supported methods for loading the CPoC application onto the supported COTS platforms.	The digital distribution service (OS store) used by the COTS platform should ensure the integrity and authenticity of the applications.

Security Requirements	Test Requirements	Guidance
	***Note:** This may include multiple methods such as OS stores, online stores and side-loading.* **3.1.7.b** The tester must confirm that only the OS store of the COTS system baseline COTS OS can be used for CPoC application deployment. **3.1.7.c** The tester must confirm for all supported COTS platforms that the OS stores enforce the use of methods that validate the authenticity and integrity of any connections between a device and the store, such as implementing recent and secure versions of TLS with cipher suites that enforce strong cryptography. This connection must prevent MITM and replay attacks. The tester must document the security controls and cryptography enforced by each supported COTS platform. **3.1.7.d** The tester must confirm that for each OS store used for CPoC application deployment, signatures validate the integrity of the application in its entirety. The tester must confirm that this signature is checked before the CPoC application is installed and must note any supported COTS platforms where the signature is not (or cannot be) checked on executable code after installation onto the COTS device (for example, where Ahead-of-time compile features are implemented, and the resulting compiled code is not provided with cryptographic authentication mechanisms that can be checked on execution). ***Note:** It is not the intent of this requirement to prevent the use of COTS platforms for which the signature cannot be checked, but it is expected that additional attestation component protections are required in such instances.* **3.1.7.e** The tester must document any additional scripts, data, executable files, interpreted commands, or other information that is downloaded by the CPoC application after installation. In each case, the tester must confirm that data is authenticated cryptographically and that the authentication is validated before use of the data.	

Security Requirements	Test Requirements	Guidance
3.1.8 The COTS system baseline must include only COTS platforms that support secure compilation and execution of the applications.	**3.1.8.a** For each supported COTS platform, the tester must detail how compilation is performed for the applications on that platform. Where multiple methods are possible on a platform, the tester must detail which methods are used (or possible) for the CPoC application. **3.1.8.b** For each compilation method used for CPoC application, the tester must detail the provided protections and confirm that they meet industry best practices for the protection of executables.	A CPoC application may be deployed to a COTS OS store in a pre-compiled form, it may be shipped as source code for the OS store to compile, or it may even be compiled on the COTS device itself. In all cases, the solution should ensure that the resulting executables are implemented using industry best practice security methods. When the solution provider is unable to control the application compilation themselves, the solution provider should ensure that only platforms that provide security features to the applications that the solution provider distributes are supported.
3.1.9 The COTS system baseline must be validated by the attestation process upon initial startup of the CPoC application.	**3.1.9.a** The tester must confirm that the COTS system baseline is validated through application of the attestation process during the initial startup of the CPoC application. The tester must confirm that the attestation process completes successfully before the CPoC application processes any transactions.	The solution should establish a trusted status (or baseline) upon initial startup for its components to provide meaningful and relevant information with which to make security decisions, identify anomalies, or take actions.
3.1.10 Validation of the COTS system baseline must be performed during each attestation check performed by the back-end attestation component.	**3.1.10.a** The tester must confirm that validation of the COTS system baseline is part of the attestation process, and that the validation involves the back-end components of the attestation system.	Ongoing verifications to the COTS system baseline by the back-end attestation component helps to identify deviations that could indicate unauthorized access or a compromise, that may need to be made available to the back-end monitoring system. Therefore, ensuring that validation is consistently performed is imperative to retain a trusted state.

Security Requirements	Test Requirements	Guidance
3.1.11 A documented policy and procedure for assessing changes to the COTS system baseline must exist and provide details on how: • COTS platforms are added to the COTS system baseline. • Decisions are made to remove previously acceptable COTS platforms from the COTS system baseline. • Such changes will affect the parties using these platforms. Therefore, the documentation must also include how communication is handled in these cases.	**3.1.11.a** The tester must obtain and review the solution provider's risk-assessment policy, update procedure documents, and confirm that they contain information on the following: • How to assess whether newly exposed vulnerabilities pose a risk to platforms. • The need to reassess all supported COTS platforms at least every year and the method used for reassessment. • How and when updates to the COTS system baseline are performed. **3.1.11.b** Where possible, the tester must compare the information in the policy with actual changes made to the COTS system baseline to confirm that the policy is being followed. **3.1.11.c** The tester must detail how merchants are informed when changes to the COTS system baseline affect their systems.	As the security landscape changes, platforms or operating systems that may be acceptable under the COTS system baseline may become vulnerable. A documented policy and procedure for assessing these changes should exist and provide details on how decisions are made to remove previously acceptable platforms from the COTS system baseline. Such changes will affect the parties using these platforms, so the documentation should also include how communication is handled in these cases.

3.2 Attestation Mechanism

The security of the solution is based largely on the protections provided by the attestation and monitoring systems. These systems should collect data about the individual platforms on which the CPoC application executes and be able to compare and contrast this data with data collected from other systems. These systems should also collect examples of malware and known attack methods.

Attestation components that gather attestation data on the COTS device should be protected from reverse-engineering and bypass. Attestation data transferred between the COTS device and the back-end attestation systems should be protected for both integrity and authenticity.

Processes for collecting data, analyzing data, and acting on the results of that analysis should be based on a documented attestation policy. This policy should detail clearly the responsible parties involved in rendering decisions and how those decisions are to be made.

Attestation determines whether the COTS device that hosts the CPoC application, or CPoC application is being, or has been, altered maliciously or fails to meet the specified criteria.

The solution provider is responsible for defining policies and procedures.

Security Requirements	Test Requirements	Guidance
3.2.1 A documented attestation policy that defines health-check rules for the COTS platform and CPoC application attestation component must exist and support the following: • Detailed response procedures for health-check results. • Health-check rules are maintained and strictly controlled. • Health-check rules are reviewed and updated as necessary, at least annually.	**3.2.1.a** The tester must confirm that a documented attestation policy exists and that it includes: • How data from the COTS environment is interpreted. • Procedures detailing when and how to escalate alerts. • Staff names or groups who are responsible for processing attestation alerts. • Staff names or groups who are responsible for maintaining the attestation systems. • Staff names or groups who are responsible for maintaining the attestation policy. • A requirement to review the policy at least annually and update it as required.	A policy that defines the specifics to support the attestation mechanisms is necessary for common understanding about how each attestation component works individually and together. The policy should explain the security trust model and the residual risk, how the attestation system protects the solution users, the thresholds used, triggers and acceptable errors, categorization of attestation findings, and response procedures and time frames for responses.
3.2.2 Implement controls to protect the attestation components and attestation system from reverse-engineering.	**3.2.2.a** The tester must confirm that the portions of the attestation process used on the COTS devices are protected from reverse-engineering. ***Note:*** *The tester may refer to details and testing performed in previous sections.*	It should be difficult for an attacker to learn details about the attestation components' design, construction, and operation. Use of obfuscation and native code are examples of techniques that can be used.

Security Requirements	Test Requirements	Guidance
3.2.3 The attestation component must not be interrupted by payment-transaction processing by the CPoC application.	**3.2.3.a** The tester must detail the mechanisms that ensure the attestation component cannot be interrupted by payment transaction processing. **3.2.3.b** The tester must confirm that the mechanisms cannot be exploited to prevent the execution or impede the integrity of the attestation process, such as by forcing rapid transactions to prevent the execution of the attestation process.	If the attestation is running when the solution is online, it should not be interrupted by transaction processing.
3.2.4 The integrity of the attestation data must be cryptographically protected wherever they are stored or processed.	**3.2.4.a** The tester must detail the data types that can be included in the attestation process and attestation messages. **3.2.4.b** For each attestation message or data type, the tester must confirm that cryptographic authenticity mechanisms are applied. **3.2.4.c** The tester must confirm that the applied authenticity methods ensure that the attestation data is not subject to replay, preplay, or MITM attacks. **3.2.4.d** When an attestation component on COTS device is implemented independently from CPoC application, the tester must confirm that any attestation message or data type sent between attestation component and the CPoC application is signed cryptographically or MAC'd before exchange. **3.2.4.e** The tester must confirm that any attestation message or data type sent from the attestation component on the COTS device (standalone or part of the CPoC application) to the back-end attestation system is signed cryptographically or MAC'd before transmission. **3.2.4.f** The tester must confirm that cryptographic processes and cryptographic material, such as random numbers, cryptographic algorithms, and keys used by the cryptographic authenticity mechanisms, meet the security requirements in Section 1.2 Random Numbers and Section 1.3 Acceptable Cryptography. **3.2.4.g** The tester must confirm that key management processes used by the cryptographic authenticity mechanisms, meet the security requirements in Section 1.4 Key Management.	If any attestation parameters or results of attestation can be altered maliciously, the integrity of the attestation system is affected. Measurement parameters can be static or behavior-based, such as privileges, intents, and system calls. Examples of attestation parameters and measurements include: • Nonces—require integrity-protected storage of the Hashes for previous nonces. • Counters—require integrity-protected storage for the counters. • Timestamps—require a trusted synchronized clock at prover side. Specialist attestation proxies may be used to collect measurements as part of a multi-layer approach.

Security Requirements	Test Requirements	Guidance
3.2.5 For any attestation data, the solution provider must be able to identify: • Where the attestation data originates (in the CPoC application, in a server-based attestation component, remotely, or locally to the consumer device) • Identification of the responsible entity or process that is to take action on the attestation data • Whether the process to address the attestation data is managed by a third-party provider API with no other privileged access ***Note:*** *If no action is available for any given attestation data, any security dependence on that attestation is considered a residual risk and must be accounted for by the solution provider.*	**3.2.5.a** The tester must detail how the origin of each attestation message or response is identified, and that this identification is part of the attestation processing. **3.2.5.b** The tester must confirm that there is clear identification of the entity or process that is responsible for taking action on any particular attestation data. Where no action is to be taken for a specific attestation data, the tester must further confirm that this is accounted for in the solution provider attestation policy. **3.2.5.c** The tester must confirm the use of any third-party mechanism in the attestation function and provide details of this mechanism. The tester must specify how the third-party messages/attestation data is received and processed by the solution provider.	The attestation data should provide sufficient detail to discern the correct action to be taken by the system or by its managing staff.
3.2.6 The attestation system must establish mechanisms to ensure attestation data is refreshed and up to date.	**3.2.6.a** The tester must confirm that the attestation system ensures that the latest data from the COTS device in use is obtained and used for any validation of the COTS platform being used.	It is important to maintain, and use refreshed and up-to-date attestation data to ensure the integrity of the COTS device and CPoC application.
3.2.7 A set of rules must be defined for analyzing the attestation data and assigning a risk-severity rating for the attestation data that aligns with the attestation policy.	**3.2.7.a** The tester must confirm that the solution provider has a documented process for assigning risk ratings to attestation results, and that this process is followed. The tester must detail whether this process is manual, semi-automatic, fully automatic, or a combination.	Analysis may be automatic, semi-automatic, or fully manual. Where machine learning or other methods are used to allow for the monitoring system to adapt automatically to changes in the risk landscape, protections should be adopted to prevent "data poisoning" or other types of adversarial manipulation of input data to cause invalid rules to be put in place by the system.

Security Requirements	Test Requirements	Guidance
3.2.8 Document and establish detailed procedures or automated responses for attestation data. Procedures must accommodate the following, at a minimum: • Send an alert to the monitoring system support personnel based on attestation-response severity. • Conduct corrective actions for false positives, such as modifying a configuration file hash. • Completely block transaction processing in the most significant cases as defined in the attestation policy. • Temporarily stop transaction processing.	**3.2.8.a** The tester must confirm that there is a defined process for handling all possible attestation data. ***Note:** This requirement does not imply that each response or data type must be specifically named, but that there are clear and defined processes for attestation datatypes.* **3.2.8.b** The tester must confirm that any attestation data deemed suspicious or indicating a potential compromise must result in either the automatic blocking of that device/merchant or escalation to a manual review. Where false positives are found, procedures must be in place to modify the attestation system to reduce false positives.	Defined and known procedures ensure that correct follow-up actions are performed.
3.2.9 Maintain up-to-date configuration measurements to support attestation criteria.	**3.2.9.a** The tester must confirm that there is a process to update the attestation system according to changes made to the supported COTS platforms. **3.2.9.b** The tester must select two different COTS platforms supported by the attestation system and detail how the system handles the differences between these platforms to maintain the same level of security validation.	Attestation measurements should reflect up-to-date information to ensure accurate responses to support attestation requests.
3.2.10 Establish controls to defend against attestation abuses to subvert the prover.	**3.2.10.a** The tester must detail the methods that are in place to protect the attestation system from Denial of Service (DoS) attacks.	The solution should provide mitigation again compromise of the attestation component that may result in DoS.
3.2.11 Escalation procedures must be defined for undocumented, unexpected, and unknown attestation data.	**3.2.11.a** The tester must confirm that there are defined escalation procedures for any undocumented, unexpected, or unknown attestation data.	The attestation policy should provide staff with escalation procedures for dealing with unexpected scenarios or results from remote attestation.

Security Requirements	Test Requirements	Guidance
3.2.12 If the attestation system triggers a response in the monitoring system which involves a manual process, such as for a potential tamper event, it must be escalated to the back-end monitoring staff to validate: • Written procedures for manually processed events must exist and be demonstrably in use. • These procedures must cover events when staff who are relied upon for such determinations are unavailable. • Events must be escalated immediately for manual review and then actioned within 48 hours. • Automated systems must be in place to disable any further payment processing from systems when an event has not been actioned for 48 hours.	**3.2.12.a** The tester must confirm that the attestation policy clearly defines what types of events or attestation data require manual intervention or oversight. **3.2.12.b** The tester must confirm that the manual intervention process provides for instances when staff is unavailable due to holidays, vacation, or after-hours events. **3.2.12.c** The tester must confirm that any event that is escalated for manual review requires action within 48 hours. The tester must detail the automated methods used to disable payment processing if the manual intervention has not been acted upon within the 48-hour window. **3.2.12.d** The tester must interview the staff responsible for the attestation manual response, and ensure that the staff understands these items and the escalation procedure.	Manual processes for managing attestation system responses, including escalation procedures, should be well documented to avoid possible errors in interpretation by operational staff.
3.2.13 Requisite qualified staff must implement and interpret attestation health-check rules, associated controls and findings, and the associated training.	**3.2.13.a** The tester must confirm that the staff to whom attestation data is escalated is capable of processing this data. **3.2.13.b** The tester must detail the training program for all staff responsible for responding to manual escalations. Staff must complete this training before being deployed into an active role in attestation data response.	Attestation results that do not have an automated response may require skilled staff to interpret specific attestation findings or to interpret them within a wider risk management framework, such as the use of telemetry and transaction heuristics. Staff who are responsible for supporting the monitoring environment have specific training needs that exceed those that are typically provided by general security-awareness training. To perform duties completely and correctly, additional specialized training should focus on skills, such as vulnerability management, monitoring/alerting, problem solving, and COTS Systems baseline.

Security Requirements	Test Requirements	Guidance
3.2.14 Retention policy and associated procedures must be defined, documented and implemented for attestation results.	**3.2.14.a** The tester must confirm the solution provider has a documented retention policy, and that this policy ensures that the attestation data is available for troubleshooting and investigation purposes. The tester must detail the retention period in the policy. **3.2.14.b** The tester must confirm that there are procedures for the recovery of stored attestation data and that these procedures are valid for the solution under test. **3.2.14.c** The tester must detail any aspects of the solution for which data cannot be stored or maintained: for example, where the attestation components or attestation system comes from a third-party provider.	Defining retention policy and associated procedures ensures attestation data is available for troubleshooting and investigation purposes. Local regulation may impact retention. The policy should provide mitigations to address such regulation.
3.2.15 Retained attestation results must have a unique ID, date and time stamp and sufficient description to identify the information, attestation component, and attestation system used at that time.	**3.2.15.a** The tester must confirm that any retained attestation results are stored with identifying data including, but not limited to, a unique ID, date and time stamp, and description. **3.2.15.b** The tester must confirm that any retained attestation data is stored with information linking that data to a specific build, version, or details of the attestation system used. This metadata may include references/versions of scripts or rules as required by the instantiation and operation of the attestation function. **3.2.15.c** The tester must confirm that the metadata stored with the attestation data is sufficient to ensure that each data element can be uniquely traced to a specific COTS device and attestation process.	Unequivocal identification of findings is required for subsequent audit and troubleshooting.
3.2.16 Attestation components and attestation system changes must adhere to formal change-control procedures.	**3.2.16.a** The tester must detail the formal change control process used by the solution provider and confirm that any changes to the attestation component and attestation system adhere to this process. The change control process includes all manual methods and changes to the network or the attestation system infrastructure.	All changes to the solution components require identification of changes, business justification, and testing and approvals. Without following fundamental change-control principles, changes can be omitted that would jeopardize the security and processing of the solution.

Security Requirements	Test Requirements	Guidance
3.2.17 Automated attestation component and attestation system changes must be performed using authorized processes.	**3.2.17.a** If automated methods are used to update the attestation component and attestation system, the tester must detail where and how these methods are used. This includes all update methods, including base code updates, script updates, or configuration updates. **3.2.17.b** For any automated update methods used, such as machine-learning processes, the tester must detail what protections are provided to prevent attempts to exploit automation through data poisoning attacks that supply invalid data into the learning process to alter the detection algorithms. **3.2.17.c** The tester must detail how these automated methods are authorized and how any changes are prevented from being applied by other (unauthorized) code or methods.	It should not be possible to circumvent or create false attestation results by unauthorized modifications to the attestation system. Where automated methods are implemented, it should be ensured that this can be performed only by authorized code, such that other (unauthorized) applications are unable to create such changes.
3.2.18 For manual updates of the attestation system: • There must be documented procedures. • Deployment of changes to the production environment must adhere to formal change control procedures with evidence that changes were performed as intended.	**3.2.18.a** The tester must confirm that any changes can be performed only as the result of a process that requires authorization of the people or systems involved. **3.2.18.b** The tester must note how people are confirmed to be suitable for performing updates to the attestation component and attestation system, and the tester must confirm that this does not include people who would not be reasonably expected or suitable to perform such updates.	Manual procedures should be documented to avoid ambiguity or misinterpretation, which could lead to misconfiguration or other non-secure practices.
3.2.19 The disabling of the attestation system or a significant loss of its function must result in the disabling of all transaction processing on all solutions that rely on that attestation component and attestation system.	**3.2.19.a** The tester must detail how the solution is designed to respond to any disablement or significant loss of function of the tested attestation system. **3.2.19.b** The tester must confirm that the designed response includes specific policy on managing significant loss of function, including the capability to disable the transaction processing on any device until attestation system operation has been restored.	As the security of the solution is largely dependent on a robust and frequent attestation process, the failure of this process should result in the cessation of transaction processing until the attestation system is restored. The solution provider should have a risk-management policy to address the loss of attestation functionality of the attestation component on the COTS device and back-end attestation component and develop a suitable response. A significant loss of function occurs when attestation data is no longer generated or processed in this way, such as in a situation that normally would result in escalation to manual review or automatic disabling of that CPoC application instance.

3.3 Type 1—Attestation of COTS Platform

The solution needs to establish a reasonable assurance that the COTS platform executing the CPoC application can be trusted. In Type 1 attestation, the COTS platform is the prover and the CPoC application and/or back-end attestation component is the verifier. The attestation system implements methods to detect and respond actively to events that indicate that the COTS platform is being, or has been, altered maliciously. This type of attestation is expected to allow for rapid decisions about the operating environment, such as determining the platform's suitability for operation or the detection of possible rooting or jailbreaking methods.

These tamper-detection and response methods cannot be wholly implemented in the same execution environment of the CPoC application.

Security Requirements	Test Requirements	Guidance
3.3.1 Controls must be in place to validate the integrity of the attestation results.	**3.3.1.a** The tester must confirm that controls are implemented to validate the integrity of the attestation results. **3.3.1.b** The tester must attempt to modify the attestation results, and to confirm that the attestation system is able to detect the modification.	Attestation measurements should be an accurate representation of the state of the COTS platform. There should be assurances in place that attestation results received from the COTS platform have not been altered or spoofed.
3.3.2 Attestation components and the attestation system must not be vulnerable to time-of-check, time-of-use (TOCTOU) attacks.	**3.3.2.a** The tester must detail the period between the attestation check (i.e., collection of attestation data) and attestation data use, and consider how it may be exploited to attack the attestation components and attestation system. **3.3.2.b** Where such attacks are possible, the tester must detail these vulnerabilities and detail any additional protections mechanisms are in place to prevent these attacks. **3.3.2.c** Where ongoing attestation checks are not used to determine the application of high privilege, developer, or debug features, the tester must explain why the absence of these checks does not constitute a TOCTOU concern for any of the supported COTS platforms.	It should not be possible for an attacker to influence COTS platform resources between the time the attestation measurements are made and the time they are checked. The intent is to protect the attestation data collected before it is used by the attestation system. One option is to implement the attestation mechanism as an atomic action that cannot be interrupted or tampered with. Another option is to cryptographically protect the attestation data to ensure it is not tampered between being collected and when attestation system uses that data.

Security Requirements	Test Requirements	Guidance
3.3.3 Attestation data must not leak information about attestation components and the attestation system.	**3.3.3.a** The tester must confirm that the attestation process is non-deterministic by evaluating different sets of data and encrypting transmissions to ensure that the process or data produced cannot easily be reproduced or replayed.	Attestation data sent to the verifier should not provide deterministic information about the attestation component or attestation system. If a malicious provider intercepted the attestation data, it should not be able to learn about the weaknesses of the attestation system to design attacks that would allow circumventing detection
3.3.4 Attestation data must be unclonable.	**3.3.4.a** The tester must detail the methods used to ensure that the attestation data are unclonable. This must include both messages containing attestation data from the COTS device and any enabling or limiting messages sent from the attestation system. **3.3.4.b** The tester must confirm that each attestation message contains a freshness indicator, which is included within the portion of the message that is authenticated cryptographically, and that the use of this freshness indicator is sufficient to prevent replay of that data.	Attestation data should be unclonable, such as by cloning part of the CPoC application configuration using an emulator and performing MITM attacks. Examples of mechanisms that can be used include digital signatures and challenge response mechanism where nonces are generated by the back-end attestation system. If digital signatures are used to ensure attestation data is not cloneable, the process should be implemented strong cryptography.
3.3.5 The back-end monitoring system must be capable of detecting all failures of COTS device attestation components.	**3.3.5.a** The tester must confirm that there is a defined maximum period of time permitted for a COTS device to respond to an attestation request from the back-end systems.	A malicious process may interfere with attestation processing, such as creating a DoS. The CPoC application attestation component should notify the monitoring system if the response timeout is exceeded.
3.3.6 The Type 1 attestation component must be provided and maintained to provide up-to-date information about the state of the COTS platform and known vulnerabilities. At a minimum, attestation must check and report the following: • Rooted or jailbroken devices, or devices in developer mode • Asynchronous rooting and unrooting of the COTS OS • COTS platforms support for secure compilation and execution of the applications • Modifications or tampering of the COTS OS • *COTS* OS or CPoC application rollback • Details on the access and use of the NFC interface for operating systems that allow for the collection of such data	**3.3.6.a** The tester must detail the attestation data (e.g., configuration/ operational information) that are provided to the attestation system about the COTS device being used. **3.3.6.b** The tester must confirm that the attestation component collects device-specific information that allows for unique identification of the system, such as processor/GPU/memory speed, thermal response under load, and memory usage **3.3.6.c** The tester must detail how the attestation method detects and provides up-to-date information about the state of the COTS platform and known vulnerabilities. The tester must confirm that this method is accurate, and that it will function as designed on all supported COTS platforms. **3.3.6.d** The tester must install a recent version of a common and respected rooting tool/hooking framework, and ensure that it is detected by the attestation system.	Specific data is required to ensure the security state of the COTS platform. Attestation parameters will vary depending on OS, but should include basic verification and be as comprehensive as possible. Attestation may not be able to detect all possible roots/jailbreaks. However, it should detect some common methods including, but not limited to: • Traditional rooting—Involves modifying the COTS platform image and permanently rooting the device • Temporary jailbreak—Involves no changes to the file system, but the jail break is lost upon reboot • Loading modified kernel images—With an unlocked bootloader, the device can be booted with a kernel and initramfs sent over the USB port. Upon reboot, the default kernel and initramfs from flash is used instead.

Security Requirements	Test Requirements	Guidance
• Emulator use • Use of a hooking framework	The tester must provide the rooting/hooking tool name and version used in this test with the associated results. **3.3.6.e** The tester must attempt to install and execute the application in an up-to-date emulator (where available). The tester must confirm that the emulated application is detected by the attestation component and prevented from processing payments. **3.3.6.f** The tester must confirm that the attestation component validates the execution environment and ensures that the execution environment meets the defined COTS system baseline requirements, including all secure compilation requirements. **3.3.6.g** The tester must attempt to install a previous version of the CPoC application and confirm that the application does not allow contactless transactions. **3.3.6.h** The tester must attempt to install a previous version of the COTS OS on the COTS device and confirm that the application does not allow contactless transactions. **3.3.6.i** For each supported COTS platform, the tester must detail how access to the NFC interface is achieved and if this access is detectable by other applications. Where access detection is possible, the tester must confirm that the attestation process collects and reports data on applications that have contactless permissions. **3.3.6.j** The tester must explain why attestation data is sufficient (or insufficient) to detect malicious tampering of the operational environment of the COTS device.	• Temporary root—Exploit is used to gain additional privileges, but is lost upon reboot (similar to temporary jailbreak).
3.3.7 The COTS platform attestation must be performed in accordance with the specified attestation policy. At a minimum, the attestation must occur: • Initial execution of the CPoC application • At CPoC application startup • If initiated by the back-end monitoring system or CPoC application attestation component • At unpredictable intervals, polled during an online session (at least every 30 minutes)	**3.3.7.a** The tester must confirm that the attestation function is performed on the COTS device as soon as practical upon initial execution of the CPoC application. **3.3.7.b** The tester must detail any keying material or other security sensitive data that is distributed or generated on the COTS device before the execution of the first attestation process. The tester must confirm that the distribution or use of this sensitive data does not reduce the process security, or expose or risk COTS platform or CPoC applications.	The attestation policy specifies when and how attestation should be performed: • At initialization, the solution should be in a trusted state; otherwise, it may not be possible to trust any subsequent attestations. • When the solution is about to commence transaction processing, it should establish a trusted status for its components. • The back-end monitoring system should have the ability to request attestation at any time as part of its

Security Requirements	Test Requirements	Guidance
• After changes have been made to the solution or to major configuration files • When the CPoC application loses and then regains its Foreground focus	**3.3.7.c** The tester must detail the conditions that initiate the attestation process on the COTS device and confirm that these include at a minimum: • At startup of the CPoC application • When initiated by the back-end monitoring system request or CPoC application attestation component • At unpredictable intervals (at least every 30 minutes) • After changes have been made to the CPoC application or any significant configuration files, such as those affecting the contactless kernel, security operation, transaction or attestation processing • When the CPoC application regains focus after having lost focus **3.3.7.d** The tester must note any attestation triggers that are configurable and detail how this configuration is managed and used.	responsibility to maintain overall security for the solution. • The attestation of the COTS platform should be part of a process that requests an attestation data at unpredictable intervals. Recurring attestations ensure real-time evaluation of the state of security and allows for intervention if anomalies are present. • The attestation of the COTS platform should be part of a continuous process that requests an attestation data at unpredictable intervals. • When the solution has undergone changes, it should re-establish a trusted status for its components. • If the Application has lost and regained its foreground focus, the solution may no longer be in a secure state.

3.4 Type 2—Attestation of the CPoC Application

In Type 2 attestation, CPoC application is the prover. Type 2 attestation is performed using an external agency (such as back-end attestation component) as the verifier. This ensures that if the CPoC application execution environment is compromised completely, the evaluation of the attestation data collected cannot be manipulated.

Type 2 attestation establish assurance that:

- The COTS platform is trusted.
- The CPoC application attestation component is trusted.
- The monitoring system is adequately prepared to take appropriate action.

Because two different types of attestation methods are used, some repetition of data collection may be noted in the Type 2 requirements below. However, this remains necessary due to the lack of trust that may be placed in a Type 1 attestation, where the CPoC application acts as the prover for the attestation data.

Security Requirements	Test Requirements	Guidance
3.4.1 Controls must be in place to validate the integrity of the attestation results.	**3.4.1.a** The tester must confirm that controls are implemented to validate the integrity of the attestation results. **3.4.1.b** The tester must attempt to modify the attestation results, and confirm that the attestation system is able to detect the modification.	Attestation measurements should be an accurate representation of the state of the CPoC application. There should be assurances in place that attestation results received by the back-end attestation component have not been altered or spoofed.
3.4.2 Attestation components and the attestation system must not be vulnerable to time-of-check, time-of-use (TOCTOU) attacks.	**3.4.2.a** The tester must detail the period between the attestation check (i.e., collection of attestation data) and attestation data use, and consider how it may be exploited to attack the attestation components and attestation system. **3.4.2.b** Where such attacks are possible, the tester must detail these vulnerabilities and detail any additional protections mechanisms are in place to prevent these attacks. **3.4.2.c** Where ongoing attestation checks are not used to determine the application of high privilege, developer, or debug features, the tester must explain why the absence of these checks does not constitute a TOCTOU concern for any of the supported COTS platforms.	It should not be possible for an attacker to influence COTS platform resources between the time the attestation measurements are made and the time they are checked. The intent is to protect the attestation data collected before it is used by the attestation system. One option is to implement attestation mechanism as an atomic action that cannot be interrupted or tampered with. Another option is cryptographically protecting the attestation data to ensure it is not tampered between being collected and when attestation system uses that data.

Security Requirements	Test Requirements	Guidance
3.4.3 Attestation data must not leak information about attestation components and the attestation system.	**3.4.3.a** The tester must confirm that the attestation process is non-deterministic, evaluating different sets of data and encrypting transmissions that ensure that the process or data produced cannot be reproduced or replayed easily.	Attestation data sent to the verifier should not provide deterministic information about the attestation component or attestation system. If a malicious provider intercepted the attestation data, it should not be able to learn about the weaknesses of the attestation system to design attacks that would allow circumventing detection
3.4.4 Attestation data must be unclonable.	**3.4.4.a** The tester must detail the methods used to ensure that the attestation data are unclonable. This must include both messages containing attestation data from the COTS device and any enabling or limiting messages sent from the attestation system. **3.4.4.b** The tester must confirm that each attestation message contains a freshness indicator, which is included within the portion of the message that is cryptographically authenticated, and that the use of this freshness indicator is sufficient to prevent replay of that data.	Attestation data should be unclonable, such as by cloning part of the CPoC application configuration using an emulator and performing MITM attacks. Examples of mechanisms that can be used include digital signatures and challenge response mechanism where nonces are generated by the back-end attestation system. If digital signatures are used to ensure attestation data is not cloneable, the process should be implemented using strong cryptography.
3.4.5 The back-end attestation components must be capable of detecting all failures of CPoC application attestation components.	**3.4.5.a** The tester must confirm that there is a defined maximum period of time permitted for a CPoC application attestation component to respond to an attestation request from the back-end monitoring systems.	A malicious process may interfere with attestation processing, such as creating a DoS. The back-end attestation components should notify the monitoring system if the response timeout is exceeded.
3.4.6 Type 2 attestation components must be provided and maintained to provide up-to-date information about the state of the CPoC application on the COTS device. At a minimum, attestation must include and report on the following: • COTS platforms and version • Instance of CPoC application • Current version of CPoC application • CPoC application and configuration modification • CPoC application and configuration tamper • CPoC application public key modification or tamper • CPoC application execution in developer mode • CPoC application execution in debug mode • Use of CPoC application code, or part thereof, within either another valid or invalid execution environment, such as through “code lifting” of the	**3.4.6.a** The tester must detail the attestation data (e.g., configuration/operational information) that are provided to the attestation system about the COTS platform and CPoC application being used. **3.4.6.b** The tester must confirm that the attestation component collects device-specific information that allows for unique identification of the system, such as memory layout/mapping features, process linking, versions/fingerprints of software libraries, and OS modules. **3.4.6.c** The tester must confirm that the attestation component reports the following information: • COTS platform and version • Version details of the CPoC application • Unique ID for the CPoC application	Attestation criteria determine the health of the COTS platform and the CPoC application that is deployed on the COTS device through interrogation of a “health-check” interface and access to any security service checks provided by the monitoring system. For example, the attestation system could use the number of successful or failed transactions to identify an anomaly within the solution.

Security Requirements	Test Requirements	Guidance
entire or partial application to another platform after initialization and personalization • State of contactless kernel • DRNG function health-check • Accessible hardware resources and information repositories • Number of transactions performed since the last attestation process	• Output from a DRNG known-answer test, where the seed is supplied by the external attestation system and is unique for each test • Permissions of the CPoC application and any open communication ports or system interfaces used by the CPoC application **3.4.6.d** The tester must confirm that the attestation component can detect and report changes or modifications to the CPoC application, or to any configuration files or additional executable files on which the application relies for secure operation. The tester must attempt to make such modifications, detail the testing process and confirm that all modifications were detected by the attestation function. **3.4.6.e** The tester must confirm that the attestation component can detect the execution of the CPoC application in developer or debug mode. **3.4.6.f** The tester must attempt to execute the CPoC application with developer and (separately) debug privileges enabled. The tester must detail the process used and confirm that the attestation system is able to detect this configuration. **3.4.6.g** The tester must confirm that the attestation system provides details on the state and integrity of the contactless kernel. Where the contactless kernel is partially instantiated outside of the rich execution environment of the COTS device (e.g., split contactless kernel), the attestation system must collect data from other aspects of the contactless kernel to confirm that it remains in an approved and operational mode. **3.4.6.h** The tester must detail the data collected by the attestation system on the CPoC application and confirm that data includes details on the integrity, tamper state, and any public keys or certificates managed or stored by the application. **3.4.6.i** The tester must confirm that the attestation component reports the number of transactions performed since the last attestation process.	

Security Requirements	Test Requirements	Guidance
	3.4.6.j The tester must explain why attestation data is sufficient (or insufficient) to detect malicious tampering of the COTS platform and CPoC application.	
3.4.7 Attestation must be performed in accordance with the specified attestation policy. At a minimum, the attestation must occur: • At initial execution of the CPoC application • At CPoC application startup • At unpredictable intervals polled during an online session (at least every 30 minutes) • If initiated by the back-end monitoring system or CPoC application attestation component • After changes have been made to the solution or to major configuration files	**3.4.7.a** The tester must confirm that the attestation function is performed on the COTS device as soon as practical upon initial execution of the CPoC application. **3.4.7.b** The tester must detail any keying material or other security sensitive data that is distributed or generated on the COTS device before the execution of the first attestation process. The tester must confirm that the distribution or use of this sensitive data does not reduce the process security, or expose or risk COTS platform or CPoC applications. **3.4.7.c** The tester must detail the condition that initiate the attestation process and confirm that these include at a minimum: • At startup of the CPoC application • When initiated by the back-end monitoring system request or CPoC application attestation component • At unpredictable intervals (at least every 30 minutes) • After changes have been made to the CPoC application, or to any significant configuration files, such as those affecting the contactless kernel, security operation, transaction or attestation processing • When the CPoC application regains focus after having lost focus **3.4.7.d** The tester must document if any of the attestation triggers are configurable and detail how this configuration is managed and implemented.	The attestation policy specifies when and how attestation should be performed: • At initialization, the solution should be in a trusted state. Otherwise, it may not be possible for the verifier to trust any subsequent attestations. • When the solution is about to commence transaction processing, it should establish a trusted status for its components. • Recurring, unpredictable attestations ensure real-time evaluation of the state of security, which reduces opportunity for spoofing attestation results by a malicious process and allows for intervention if anomalies are present. • The contactless attestation component may detect a local finding with the platform during a Type 1 attestation and request a Type 2 attestation. • The back-end monitoring system should have the ability to request attestation at any time as part of its responsibility to maintain overall security for the solution. • When the solution has undergone changes, it should re-establish a trusted status for its components.

3.5 Identification and Validation of Components

All solution components are to be uniquely identified and validated by the back-end monitoring system. This includes identifying merchants who use the CPoC application and the COTS platform being used.

Security Requirements	Test Requirements	Guidance
3.5.1 The back-end monitoring system must identify all components of the solution.	**3.5.1.a** The tester must provide a block diagram of the solution that clearly identifies all components and their locations, such as execution environment of the COTS, TEE of the COTS, cloud hosting provider, and solution provider hosted back-end. **3.5.1.b** The tester must confirm that this diagram is accurate and complete.	Proper documentation of all assets is essential to identify and mitigate risks in both the back-end system and the CPoC application. For COTS devices, protection mechanisms can rely on either the OS-provided functions or dedicated cryptographic algorithms implemented in the CPoC application. Appropriate device registration and linking devices to authorized processes and system components reasonably ensure that substitution of rogue devices is prevented. ***Note:*** *For some types of assets, this document requires specific protection mechanisms.*
3.5.2 The CPoC application and the attestation component must be identified as authorized and validated by the back-end monitoring system through cryptographic means.	**3.5.2.a** The tester must detail the methods used by the back-end monitoring system to validate the CPoC application, including any attestation components running on the COTS device. **3.5.2.b** The tester must detail the cryptography used to validate the CPoC application and ensure that the cryptography and key management meet the requirements of this Standard.	Verification of the correct and expected state of the solution components is necessary to ensure subsequent processing is secure. This process should include the establishment of a COTS system baseline that can be used to ensure that changes are expected or authorized.

Security Requirements	Test Requirements	Guidance
3.5.3 The solution must be able to associate the contactless transaction to a specific merchant/COTS device combination for tracking. If this association is not successful, the transaction must fail.	**3.5.3.a** The tester must confirm that the solution can associate a contactless transaction to a specific merchant and COTS device for tracking purposes. If the solution is unable to track the contactless transaction with the merchant and COTS device, the transaction must fail. **3.5.3.b** The tester must document the methods by which contactless transactions are associated with the merchant and COTS device. **3.5.3.c** The tester must document the outcome of attempts to impersonate the merchant on different COTS devices or otherwise bypass the association methods.	The solution should be able to uniquely identify all transaction details for tracking based on the following at a minimum: • COTS device used for the transaction • Merchant details for the transaction • Contactless transaction processing details If the back-end monitoring system fails to associate the merchant who initiated the contactless transaction to a COTS device, the transaction should fail. This ensures that transactions will not be manipulated by any malicious activity. When direct identification of a COTS device is not possible (e.g., COTS platform does not provide a mechanism to read IMEI or other device unique identifier), the solution could create a strong correlation between the CPoC application instantiation and the COTS device (e.g., Android SSAID and iOS Install ID). The solution should be able to detect these failures and take appropriate action to block transactions coming from the COTS device and associated CPoC applications where failures are occurring in real time.
3.5.4 The solution must be able to accept and process attestation data from the CPoC application and take appropriate action based on predefined rules (for example, suspending transactions).	**3.5.4.a** The tester must confirm that the solution validates the COTS device and CPoC application before communicating sensitive data or performing payment transactions	The solution should implement methods to detect and respond actively to events.
3.5.5 The solution must incorporate a detection system (or feed other detection systems) capable of detecting anomalous and potentially fraudulent activity, including suspicious transactions.	**3.5.5.a** The tester must detail how the anomaly-detection system is implemented and maintained by the solution provider. This must include how to escalate potentially fraudulent activity and how to respond to such activity. **3.5.5.b** The tester must detail what methods are used to correlate different fraudulent attempts or activities in an attempt to isolate commonalities. Where common data points such as geolocation of the merchant are not used, the tester must explain why the absence of common data points does not reduce the overall security of the system. At a minimum, data points should include:	Detection systems should assist with monitoring, detecting and blocking suspicious or fraudulent transactions and be capable of issuing timely alerts to responsible personnel upon detection. Data from the back-end monitoring system and attestation system (device parameters) should be included in the detection system. Alerts should be acted upon in accordance with documented investigation and response procedures. Examples of activity that should be monitored for relative to contactless transactions include: • Unusual transaction velocity at the merchant level

Security Requirements	Test Requirements	Guidance
	• Merchant-level transactional velocities that are statistically inconsistent with historical transaction volumes associated with contactless-based transactions. • Anomalous merchant activity related to areas of geographical use that are inconsistent with the historical activity associated with contactless transactions. • Individual PAN usage velocities for contactless transactions that may be associated with probing or testing detection capabilities in an attempt to circumvent such controls. • Suspicious activity associated with multiple transactions originating from individual PANs for contactless-based transactions within time frames inconsistent with merchant geographical locations. • Signals consistent with cardholder/merchant collusion associated with contactless based transactions. ***Note:*** *In all cases, it is not sufficient for the anomaly-detection mechanism to rely on attestation data from the COTS platform; it must always include analysis and consideration of merchant and transaction-based data that is separate from the technical data collected by the monitoring system.*	• Anomalous merchant activity related to geographic origin of transactions • Unusual individual contactless transaction authorization attempts that may be associated with probing or testing • Signals associated with cardholder/merchant collusion involving contactless transactions • Signals that the device is being used by unauthorized users through behavioral biometric analysis or other technologies

3.6 Security of Monitoring and Attestation Environment

The back-end environments used for monitoring and attestation should be secured sufficiently. Physical and logical security controls for the network and system components that make up the monitoring and attestation environment are important to ensure the integrity, availability, and confidentiality of information processed on those systems and networks.

The back-end monitoring system or attestation system that are present within the Cardholder Data Environment (CDE) should be assessed against the PCI DSS *DESV* requirements. Back-end systems that are isolated sufficiently from the CDE and cannot access cleartext PAN should be assessed against Appendix A Monitoring Environment Basic Protections.

Organizations responsible for the operation of the monitoring and attestation environment are responsible for the implementation and ongoing maintenance of these requirements.

Security Requirements	Test Requirements	Guidance
3.6.1 When the back-end monitoring system and attestation system reside in an organization's CDE, it must adhere to PCI DSS, including DSS Appendix A3: *Designated Entities Supplemental Validation (DESV)*.	**3.6.1.a** The tester must obtain and review the Attestation of Compliance (AOC) outlining compliance of the solution provider environment to the PCI DSS requirements. This AOC must cover the scope of the back-end attestation and monitoring environments. **3.6.1.b** Where the back-end monitoring system or back-end attestation system is implemented within, or directly connected to, the CDE, the tester must confirm that the monitoring and attestation environment has been assessed to the additional controls outlined in Appendix A3 of PCI DSS, "*Designated Entities Supplemental Validation*."	Implementation of industry-recognized logical and physical protections are necessary for the confidentiality, integrity, and availability of the solution back-end environments. Appropriate scoping and identification of controls assist with ensuring that the back-end monitoring system and attestation system environments are adequately protected.

Security Requirements	Test Requirements	Guidance
3.6.2 If PAN is not present in the back-end monitoring system and attestation system environment, and it is not part of an organization's existing CDE, the environment must comply with the logical and physical security requirements defined in Appendix A Monitoring and Attestation Environment Basic Protections.	**3.6.2.a** Where the back-end monitoring system and attestation system is not within the CDE, the tester must confirm that the solution design requires that PANs are never present in the monitoring and attestation environment. **3.6.2.b** Where encryption is relied upon to descope the presence of PANs in the back-end monitoring system and back-end attestation system environments, the tester must confirm that a PCI QSA has verified that the PAN decryption mechanism/cryptographic keys are not accessible from the monitoring and attestation environment. **3.6.2.c** Where the back-end monitoring system and attestation system are not assessed to the PCI DSS *DESV* requirements, the tester must confirm that the back-end monitoring and attestation environment comply with the logical and physical security requirements in Appendix A Monitoring and Attestation Environment Basic Protections.	PAN includes cleartext PAN and encrypted PAN. Encrypted PAN may be out of scope if a PCI QSA can verify that PAN decryption mechanisms or PAN decryption keys are not accessible from the monitoring and attestation environments.

Module 4: Back-end Systems—Processing

Control Objective: The environments that decrypt account data and process the payment transaction subsequent to the solution are to adhere to payment-industry requirements for the protection of account data processing. In addition, account data is to be protected while being processed within the boundaries of the Contactless COTS device and payment acceptance application, and when data is transmitted within the solution.

4.1 Security of Account Data Processing Environment

The back-end payment processing environment used for the solution should comply with the requirements of PCI DSS. The scope of the PCI DSS assessment should include all components and infrastructure used for the solution where those components are in scope for PCI DSS assessment.

Security Requirements	Test Requirements	Guidance
4.1.1 Decryption of all account data must occur only in back-end payment processing environments.	**4.1.1.a** The tester must confirm that account data is decrypted only in the back-end payment processing environments after it has been encrypted for transport to those environments within the COTS device. **4.1.1.b** The tester must confirm that account data is not returned to the COTS device after being decrypted, unless this occurs during a transaction implementing a remote component of contactless kernel.	The solution outlines specific technical and procedural controls to protect the secrecy of account data. Therefore, decryption of this information should be performed only in environments designated and authorized to perform these functions. The back-end payment-processing environments require security controls that are separate and distinct from this standard to address the risks of cleartext data in those environments.
4.1.2 The back-end payment processing environment must maintain and comply with PCI DSS requirements.	**4.1.2.a** The tester must obtain and review the Attestation of Compliance (AOC) outlining compliance of the solution provider payment processing environment with the PCI DSS requirements. This AOC must cover the scope of the payment processing environment as understood by the tester through the details obtained in the evaluation process.	To ensure the confidentiality and integrity of the account data, verification that data decryption is performed only in a PCI DSS-compliant environment is required. Environments that are PCI DSS-compliant demonstrate that the minimum set of industry-expected security controls have been applied to that environment, which reduces risk compared to environments that do not apply security controls. ***Note:*** For information about back-end monitoring and attestation environment requirements, see Section 3.6 Security of Monitoring and Attestation Environment.

Module 5: Contactless Kernel

Control Objective: EMV function and payment brand-approved contactless specifications and their associated security requirements are to be supported by the COTS device and CPoC application.

If applicable, all parties involved in the solution are to adhere to stated requirements in this section. Ultimately, the solution providers are responsible for ensuring the stated requirements are met.

5.1 Contactless Kernel Functionality

Many of the security controls within the solution rely on the security functions provided by the EMV specification, such as dynamic transaction data in the form of a cryptogram. Contactless magnetic stripe data (MSD) transactions that use a dynamic transaction verification code can also be supported by the solution.

The solution should be able to process contactless transactions as implemented through the payment mechanisms of one of the payment brands. This Standard does not mandate the acceptance of only EMV-based payment cards. However, it does require that the payment function be validated through, and accepted by, at least one of the payment brands before approval of the overall solution.

Security Requirements	Test Requirements	Guidance
5.1.1 The solution must use a payment brand-approved contactless kernel implementation.	**5.1.1.a** The tester must confirm that the contactless kernel implemented in the solution has been approved to process contactless transactions by at least one payment brand. **5.1.1.b** The tester must cite the EMV/brand approval version and number for the contactless kernel, and confirm that the scope of the approval appears valid, given the testers understanding of the solution under evaluation. **5.1.1.c** The tester must confirm that contactless kernel is configured to only operate in an online authorization mode, and that the merchant cannot change the mode of operation.	Each payment brand currently supports contactless payment function through its own specifications. As such, each payment brand may apply the EMV specification differently, such as functional options and data element differences. Critical and common contactless kernel security requirements that apply to the solution include: • The contactless kernel should support only online contactless transactions. • The solution should support only chip-based transactions with a cryptogram or a dynamic card verification code. • The solution should maintain contactless kernel integrity. • The solution provider should deliver contactless kernel function securely within the solution. This includes the design, development and maintenance of the software and the secure transport of the application to the COTS device. • The solution provider should identify and authenticate COTS devices for the purpose of payment brand

Security Requirements	Test Requirements	Guidance
		public key and application (contactless kernel) delivery. Contactless kernel development should be based on the approved and current EMV Contactless Specification for Payment System[5] specification when EMV mode is selected.
5.1.2 The contactless kernel must use a suitable entropy source through an approved RNG or by using the EMV Unpredictable Number (UN) algorithm.	**5.1.2.a** The tester must confirm that the UN generated by the contactless kernel is generated using a suitable entropy source through an approved RNG or using the EMV UN algorithm. **5.1.2.b** The tester must confirm that cryptographic processes and cryptographic material, such as random numbers, cryptographic algorithms, and keys used by the contactless kernel meet the security requirements in Section 1.2 Random Numbers and Section 1.3 Acceptable Cryptography.	The contactless kernel should support the creation and use of dynamic data in a transactional event, such as providing an RNG or EMV UN.

[5] https://www.emvco.com/emv-technologies/contactless/

5.2 Contactless Kernel Security Requirement

Contactless payment processing must be performed in a secure manner, and the contactless kernel should be protected against manipulation or subversion. Although these requirements do not mandate any specific method for instantiating the contactless kernel, the security and integrity of that kernel are vitally important. It should be possible to validate the contactless kernel version at any time, including any cloud-based functions. The contactless kernel should not expose security data, such as payment brands keys, internal or intermediate values, and card tags, to any other process or application. Configuration data and options that may affect the security or function of the contactless kernel are to be loaded into the COTS platform with authentication or managed with FIM at the server end.

Because this kernel may be implemented locally on the COTS device, remotely in the cloud, or a combination of both. If the contactless kernel is implemented remotely (partially or completely), the remote hosting environment (e.g., cloud or remote component of contactless kernel) should demonstrate the minimum set of industry-expected security controls.

Security Requirements	Test Requirements	Guidance
5.2.1 Contactless kernel implementation must include security controls to protect its integrity and confidentiality.	**5.2.1.a** The tester must detail how the contactless kernel and transaction processing is implemented, including details of any cloud or remote component of contactless kernels, and use of configuration files. **5.2.1.b** The tester must confirm that there are methods implemented to ensure the integrity of the contactless kernel and security assets. This may include methods such as signatures on the files, use of file integrity monitoring, and use of secured storage and execution environments. The tester must detail the methods that are implemented. **5.2.1.c** The tester must detail the methods used to protect the confidentiality of the contactless kernel. Where fully or partially remote component of contactless kernels are implemented, the tester must detail the controls implemented to protect against attacks that expose sensitive data during operation or storage. The test must consider the configuration used for the remote component of contactless kernel instance and any protections applied to prevent side channel leakage to other applications or systems resident on the same hardware.	A major security control for the solution is the chip-based transaction, which supports dynamic transaction data. However, the way this data is gathered, used, and processed by the contactless kernel is also important. Therefore, the contactless kernel should be specifically called out and validated as part of the solution testing. The requirements also go beyond just the contactless kernel. The payment brand root certificates are also important to prevent others from generating their own cards that are validated through the solution. Furthermore, many contactless kernels come with configuration options that can impact significantly the operation and security of the contactless kernel itself.

Security Requirements	Test Requirements	Guidance
5.2.2 The remote component of contactless kernel environment must maintain and comply with PCI DSS requirements.	**5.2.2.a** The tester must obtain and review the Attestation of Compliance (AOC) outlining compliance of the remote component of contactless kernel environment with the PCI DSS requirements. This AOC must cover the scope of the remote component of contactless kernel environment as understood by the tester through the details obtained in the evaluation process.	To ensure the confidentiality and integrity of the account data, in the remote component of the contactless kernel implementation, the remote environment that hosts the component of contactless kernel should comply with PCI DSS. Environments that are PCI DSS-compliant demonstrate that the minimum set of industry-expected security controls has been applied to that environment, which reduces risk compared to environments that do not apply security controls.

Appendix A Monitoring and Attestation Environment Basic Protections

PAN is the underlying factor for determining the applicability of PCI DSS security requirements. Recognizing that PAN may not exist in the back-end monitoring system and the back-end attestation system that support the solution, this appendix defines the minimum requirements to ensure fundamental security of the back-end monitoring system and back-end attestation component.

The PCI-recognized lab personnel must be physically on-site for each assessment of the back-end monitoring system and attestation system environment, though the duration of the on-site visit will vary.

A.1 Governance and Security Policies

Control Objective: Security policies set the security tone for the organization and inform personnel what is expected of them. All personnel are to be aware of the sensitivity of data and their responsibilities for protecting it.

Security Requirements	Guidance
A.1.1 Executive management must establish responsibility for the protection of sensitive data and system components within the back-end monitoring system and attestation system environment. Responsibilities include: • Overall accountability for maintaining compliance to all required standards • Implementing a security governance program • Providing updates to executive management on security initiatives and issues, at least annually	Executive management assignment of responsibilities ensures senior-management visibility into the security of the back-end monitoring system and attestation system environment. Informed management can ask questions to determine the effectiveness of the program and influence strategic priorities. Overall responsibility for the compliance program may be assigned to individual roles and/or to business units within the organization. An established governance program assists with the ongoing business-as-usual activities to maintain a strong security posture. Executive management may include C-level positions, board of directors, or equivalent. The specific titles depend on the particular organizational structure. The level of detail provided to executive management should be appropriate for the particular organization and the intended audience.
A.1.2 The security governance program must include: • Definition of activities for maintaining and monitoring overall standards compliance, including business-as-usual activities • Annual assessment processes • Processes for the continuous validation of security requirements, such as daily, weekly, and quarterly per the requirement • A process for performing business-impact analysis to determine potential security and compliance impacts for strategic business decisions	Establishing a governance program that monitors the health of its security controls allows the organization to be proactive should a control fail within the solution. Security governance supports effectively communicating activities and statuses throughout the organization. The program can be a dedicated program or incorporated into an over-arching compliance and/or governance program. It should include a well-defined method that demonstrates consistent and effective evaluation. Example methodologies include *Deming Cycle of Plan-Do-Check-Act (PDCA)*, *ISO 27001*, COBIT, DMAIC and Six Sigma.

Security Requirements	Guidance
A.1.3 Changes to organizational structure, such as a company merger or acquisition or a change or reassignment of personnel with responsibility for security controls, must result in a formal (internal) review of the impact to the environment scope and applicability of controls.	An organization's structure and management define the requirements and protocol for effective and secure operations of the back-end monitoring system and attestation system environment. Changes to this structure could have negative effects on the processing and security of the environment by reallocating or removing resources that once supported the solution. Therefore, it is important to revisit the back-end monitoring system and attestation system environment scope and controls when there are changes to ensure required controls are in place and active.
A.1.4 Documented polices must exist and be demonstrably in use that require background checks for staff who are involved with the back-end monitoring system and attestation system environment.	Performing background investigations helps to ensure the hiring of qualified staff who will be involved with the back-end monitoring system and attestation system environment and avoid problems with employee integrity.
A.1.5 Determine back-end monitoring system and attestation system environment impact for all changes to systems or networks, including additions of new systems and new network connections. Processes must include: • A formal impact assessment • Identifying applicable security requirements to the system or network • Updating back-end monitoring system and attestation system environment impact as appropriate • Documented sign-off of the results of the impact assessment by responsible personnel	Changes to systems or networks can have significant impact on the environment. For example, firewall rule changes can impact whole network segments, or new systems may be added to the back-end monitoring system and attestation system environment that were not protected to the same level previously. Organizations require processes to determine the potential impact that changes introduce to systems and networks within the back-end monitoring system and attestation system environment. This ensures that these changes do not impact the security of the back-end monitoring system and attestation system environment negatively.
A.1.6 Configuration standards must be defined and applied to system components within the back-end monitoring system and attestation system environment. Configuration standards must align with industry-accepted standards.	Configuration standards support approved software versions, updates, and security controls. These standards also assist with security management and baseline configurations that are approved by the organization.
A.1.7 Configuration standards must include: • Changing all vendor-supplied default accounts and system settings • Removing or disabling all unnecessary system or application function • Preventing functions that require different security levels from co-existing on the same system component	Requirements to harden IT resources provide reasonable assurance that malicious users cannot exploit well-known vulnerabilities.

A.2 Secure Networks

Control Objective: Businesses depend on the ability of their networks to operate. Protections to ensure network availability, security, and reliability reduce risk to the organization.

Security Requirements	Guidance
A.2.1 Network and data-flow diagrams must exist to support the back-end monitoring system and attestation system environment identifying architecture and security control points.	Network and data-flow information, such as diagrams or network-mapping tools, document how networks are configured, the identity and location of system components, and how systems are connected to each other and to other systems and all communication paths with trusted and untrusted networks. This information provides a common understanding and helps to identify where security controls could be overlooked.
A.2.2 Network configuration and controls must be reviewed at least quarterly to ensure they remain active and relevant.	Reviewing device configurations allows the entity to identify and remove any unneeded, outdated, or incorrect rules and confirm that only authorized connections, ports, protocols, services, and APIs are allowed and have not changed from the baseline. All other services, protocols, and ports should remain disabled or be removed through periodic reviews. Review processes may include real-time monitoring and analysis, periodic maintenance cycles to ensure the controls are accurate and working as intended, and periodic reviews of network traffic connectivity across ports, protocols, and services. For guidance on services, protocols, or ports considered to be non-secure, refer to industry standards and guidance, such as NIST, ENISA, and OWASP.
A.2.3 Alerts must be generated for action by responsible personnel upon detection of suspicious activity or anomalies. Establish and follow procedures for investigation and response.	An alert should be generated that is monitored actively and investigated immediately. Where suspicious traffic is blocked automatically, a record of the traffic should also be generated and investigated to determine whether action is needed to prevent further attack.
A.2.4 Controls must be implemented to detect and/or block network attacks.	Controls should be implemented at the perimeter and critical system points, and include consideration of both network-based and application-based attack vectors. Methods of detection may include signature-based, behavioral, and other mechanisms that analyze traffic flows. Examples of tools include IDS/IPS, host firewalls, and real-time traffic analysis tools. All mechanisms, such as detection engines, baselines, and signatures, should be configured, maintained, and updated per vendor instructions to ensure optimal protection.
A.2.5 Mechanisms must be implemented to detect and prevent cleartext data from leaving the back-end monitoring system and attestation system environment through an unauthorized channel, method, or process, including generation of audit logs and alerts.	Mechanisms to detect and prevent unauthorized loss of data may include appropriate tools, such as data loss prevention (DLP) solutions, and/or manual processes and procedures. Coverage of the mechanisms should include, but not be limited to, e-mails, downloads to removable media, and output to printers. Use of these mechanisms allows an organization to detect and prevent situations that may lead to data loss.

Security Requirements	Guidance
A.2.6 Penetration testing on segmentation controls must be performed at least every six months and after any changes to segmentation controls/methods to confirm back-end monitoring system and attestation system environment scope.	If segmentation is used to isolate networks, those segmentation controls should be verified using penetration testing to confirm they continue to operate as intended. Penetration testing techniques should follow the existing penetration method as specified in PCI DSS Requirement 11. For additional information about effective penetration testing, refer to the *PCI SSC's Information Supplement on Penetration Testing Guidance*, *X9.111 Penetration Testing*, *NIST SP800-115*.
A.2.7 File-integrity monitoring must be used to protect configuration files, executables, and public keys/certificates used for security services on any back-end components of the back-end monitoring system and attestation system environment.	Changes that impact file integrity or the security posture of the components of the back-end monitoring system and attestation system environment should be detected and handled.

A.3 Vulnerability Management

Control Objective: Identify security vulnerabilities to determine mitigating controls and security requirements.

Processes for detecting security vulnerabilities include confirming the security of the COTS system baseline and any vendor-developed code. Penetration tests may be performed by internal staff; however, any penetration testers must be able to demonstrate skill and knowledge in the art through formal accreditation to standards, such as CREST and/or OSCP. System vendors are to have an active vulnerability reporting and management program that is commensurate with industry best practices and are to be able to demonstrate remediation of security vulnerabilities reported through such public programs.

Security Requirements	Guidance
A.3.1 Implement controls to prevent and/or detect and remove malicious software. Controls must be active and maintained.	Controls should prevent the introduction and execution of malicious software (malware). A combination of methods, tools, and programs may be used, such as anti-malware software, application whitelisting, host-based and network-based intrusion prevention tools, and system instrumentation. A combination of real-time protection and periodic scans should be considered. The implemented controls should be kept current, such as updated signatures and baselines. Anti-malware controls should not be disabled unless specifically authorized by management on a case-by-case basis for a limited time period.
A.3.2 Procedures to identify and rate vulnerabilities based on their criticality must exist and be in use. Procedures must align with industry-accepted practices.	Not all vulnerabilities pose the same risk to an organization's environment. Vulnerabilities should be ranked and prioritized in accordance with an industry-accepted method or organizational risk-management strategy.
A.3.3 Internal and external vulnerability scans to the back-end monitoring system and attestation system environment must be performed at least quarterly to identify and address vulnerabilities.	Malicious users exploit vulnerabilities in systems and applications to gain unauthorized access to environments and sensitive information. Vulnerability scans provide a way for the organization to identify weaknesses that could be exploited and take corrective action to remove the risk. Rescans should be performed as needed to verify that vulnerabilities have been addressed. Sources for vulnerability information should be trustworthy and often include vendor websites, industry news groups, mailing lists or RSS feeds.
A.3.4 External scans must be performed by a PCI SSC Approved Scanning Vendor (ASV). Internal scans are performed by qualified personnel.	Internal vulnerability scans can be performed by qualified internal staff or outsourced to a qualified third party. For scans managed by the entity, the entity should ensure that scanning engines and vulnerability fingerprints are up-to-date and that the scanning engine is configured in accordance with vendor guidance documentation. Personnel should have sufficient knowledge to review and understand the scan results and determine appropriate remediation. Internal personnel that interact with the ASV also should be knowledgeable in the network architecture and implemented security controls to provide the ASV with information needed to complete the scan.

Security Requirements	Guidance
A.3.5 Penetration testing of the monitoring environment must be performed by qualified personnel at least annually.	Penetration tests identify weaknesses in an organization's security boundaries and controls to identify gaps and take corrective action. The penetration-testing method should be based on industry-accepted approaches and incorporate both application-layer and network-layer testing. The scope of testing should cover the monitoring environment perimeter and critical systems, and include testing from both inside and outside the network.
A.3.6 Penetration test findings must be remediated based on predefined criteria that align with industry-accepted practices.	Security patches and fixes should be implemented based on risk ranking. Where high-risk vulnerabilities cannot be addressed per defined criteria, a formal exception process should be followed, including approval by personnel with appropriate responsibly and accountability. After remediation activities have been performed, penetration tests should be performed as necessary to verify that the remediation is effective and that the identified vulnerability or security issue has been mitigated.

A.4 Access Controls

Control Objective: Access to information and security assets in the back-end monitoring system and attestation system environment is provided on least-privilege and need-to-know principles.

Security Requirements	Guidance
A.4.1 Access to system components and data must be based on least-privileges and need-to-know that is specific to job functions or processes being performed.	Access to system components should be appropriate for job functions to prevent misuse. Access to systems and data within the back-end monitoring system and attestation system environment is restricted based on business need, while also accounting for the sensitivity of the data being transmitted between the systems.
A.4.2 Documented procedures for granting and managing access must exist and be in use.	Users with special access to create or modify other user IDs should follow established procedures to prevent errors or inadvertently grant unauthorized access. Procedures should address the approval process for provisioning, monitoring, changing, and revoking of accounts used to access the back-end monitoring system and attestation system environment .
A.4.3 Individuals must be assigned a unique user ID.	Unique user IDs allow the organization to maintain individual responsibility and accountability for actions performed using the ID and is an effective audit trail.
A.4.4 Controls must be implemented to protect the confidentiality and integrity of accounts and credentials.	Implemented controls should protect the confidentiality and integrity of accounts for both local and remote users. The controls should include secure transmission and storage of account and credential information at all times.
A.4.5 Mechanisms must be established to support the organization's password-composition policies, session timeout, and inactivity rules.	Organizations should have rules that govern the protection and use of user IDs and passwords to protect the organization IT assets.
A.4.6 Controls must be defined and active for managing and monitoring third-party access to the back-end monitoring system and attestation system environment .	Third parties pose significant risks because they may be the "weak link" in the organization. Third parties' security posture may not be consistent with the back-end monitoring system and attestation system environment . Therefore, you should understand their security posture, and limit and control their abilities. Configuration and connection requirements should be defined and implemented for all access by third-party personnel, such as ensuring accounts are enabled only during the time needed and disabled when not in use, and monitoring account activity when in use.
A.4.7 All user access to system components in the back-end monitoring system and attestation system environment must use multi-factor authentication.	User access to sensitive resources and processes requires additional assurance and verification that individuals who are attempting access is who they claim to be. For more information, see *PCI SSC Information Supplement—Multi-factor Authentication*.

Security Requirements	Guidance
A.4.8 User accounts and access privileges must be reviewed at least every six months to ensure that user accounts and access are authorized and appropriate based on job function.	User access should remain appropriate for job functions. Bi-annual review of user accounts and access privileges ensures that user access remains appropriate for the user's job functions and identifies inactive accounts that could be used to gain unauthorized access by malicious users. Inactive accounts should be removed from the system.

A.5 Physical Security

Control Objective: Ensure the physical premises and associated assets are protected commensurate with the sensitivity and value of those premises and assets and the information they contained.

Security Requirements	Guidance
A.5.1 Documented policies and procedures must exist to physically protect the system components and limit access to the monitoring environment.	Documented policies and procedures ensure common understanding and communicate management's expectation for securing these resources. They should include defining the physical access controls required to prevent the monitoring environment from being physically accessed by unauthorized persons. The controls should cover all physical access points, and include procedures for managing onsite employees and third parties. Specific procedures should be defined for managing visitors, including a visible means for identification and escorts by authorized personnel.
A.5.2 Physical access to the back-end monitoring system and attestation system environment must be monitored to ensure access is authorized and based on business need.	The ability to oversee and review security controls assists with timely identification and the ability to address anomalies. Monitoring controls should include use of video cameras and/or access-control mechanisms. Data from video cameras and/or access-control mechanisms should be logged to provide an audit trail of all physical access to the environment. Monitoring and periodic reviews of physical access controls and audit logs should be performed to allow early identification of incorrect controls and for timely response to suspicious activities. Personnel should be trained to follow procedures at all times. All suspicious activity should be managed according to incident security procedures.
A.5.3 Procedures to remove access and return assets, such as keys and access cards for personnel who are terminated or have a change in job duties, must be defined and demonstrably in use.	Individuals leaving the organization or moving to a different position with access to security assets poses risk and may lead to unauthorized access. Procedures assist with defining actions required to remove access and security assets in a timely manner.

Security Requirements	Guidance
A.5.4 Media associated with the back-end monitoring system and attestation system environment must be protected to ensure secure storage, transport, and disposal of media.	Physical media containing information assets require the same level of protection as logical access to ensure consistent security protection. Controls and process should cover secure storage transport and disposal of storage media. Specific controls/rigor may vary for different levels of sensitivity of the data stored on the media.
A.5.5 Implement response procedures to be initiated upon the detection of attempts to remove cleartext data from the back-end monitoring system and attestation system environment through an unauthorized channel, method, or process. Response procedures must include: • Procedures for the timely investigation of alerts by responsible personnel • Procedures for remediating data leaks or process gaps, as necessary, to prevent any data loss	Defined and documented plans and procedures assist with responding to security incidents in a timely and efficient manner. Procedures should include response activities, escalation, and notification, and cover all assets and processes that could impact the back-end monitoring environment operations or data. The incident response plan should be comprehensive and include coverage of all systems. Communication and contact strategies should include required notifications. Incident response personnel/teams should be trained and knowledgeable in incident response procedures and be available to respond immediately to an incident.
A.5.6 System back-up requirements for the monitoring environment must be defined and address the following: • Back-up copies of information, software and system images must be created and tested regularly • The frequency and retention of backups must be adequate to support day-to-day production activities, and must be sufficient for recovery and to achieve recovery objectives associated with those systems that require a recovery capability • Back-up information must be stored securely, with appropriate physical and environmental controls • Duration and frequency must match documented retention policy	Information backups help maintain the integrity and availability of information. Backups from the monitoring environment support recovery of the monitoring environment in the event of a disruption of services. Also, backups provide a point-in-time snapshot for investigation and analysis purposes. Frequency and retention of backups should align with the organization's overall risk-management strategy.

A.6 Incident Response

Control Objective: Address non-standard processing or events to prevent losses and maintain continuity of processing.

Security Requirements	Guidance
A.6.1 Procedures must be defined, documented, and communicated to support incident response policies.	Procedure documentation ensures common understanding and defines a process to be followed to address non-standard processing or events.
A.6.2 A process must be implemented to detect and alert on critical security control failures immediately. Examples of critical security controls include, but are not limited to: • Firewalls • IDS/IPS • FIM • Anti-virus • Physical access controls • Logical access controls • Audit-logging mechanisms • Segmentation controls	The ability to identify and address quickly anomalies in processing or failures in security controls reduces risk of loss.
A.6.3 Respond to failures of any critical security controls in a timely manner, not to exceed 48 hours. Processes for responding to failures in security controls must include: • Restoring security functions • Identifying and documenting the duration (date and time, start to end) of the security failure • Identifying and documenting causes of failure, including root cause and documenting remediation required to address the root cause • Identifying and addressing any security issues that arose during the failure	Well-defined procedures and processes limit exposure.
A.6.4 Implement response procedures to be initiated upon the detection of attempts to remove cleartext data from the back-end monitoring system and attestation system environment through an unauthorized channel, method, or process. Response procedures must include: • Procedures for the timely investigation of alerts by responsible personnel • Procedures for remediating data leaks or process gaps, as necessary, to prevent any data loss	Data loss-prevention techniques assist with identification of suspicious activity and notification of support staff members.

Security Requirements	Guidance
A.6.5 Incident response procedures must be reviewed and tested at least annually.	Testing an organization's incident response procedures identifies inadequacies and required improvements that can be addressed.

A.7 Audit Logs

Control Objective: Audit logs accomplish several security-related objectives, including individual accountability, reconstruction of events, intrusion detection, and problem identification. Prompt review of logs permits early detection of hackers who otherwise might be encouraged by an apparent lack of monitoring.

Security Requirements	Guidance
A.7.1 Policies and procedures must exist and be demonstrably in use for generating and managing audit logs for all system components.	Audit logs support common understanding and set management requirements.
A.7.2 Audit logs must identify all security-related activity. At a minimum, they must include: • User-oriented security events, such as log-on and log-off • Successful and rejected network-access attempts • Successful and rejected data and system-access attempts • Changes to system and security configurations • System administrator and system operator activities • Use of administrative privileges • Use of system utilities and applications • Files accessed and the kind of access • Alarms raised by access-control systems • Activation and de-activation of protection systems, such as anti-virus systems and intrusion-detection systems (IDS)	Audit logs should be able to reconstruct activities and have sufficient detail to clearly identify events.
A.7.3 Time synchronization must be in place for audit logs.	Effective forensics require audit logs to be synchronized to correlate events adequately.
A.7.4 Audit logs and security events must be monitored to identify anomalies or suspicious activity.	Ongoing review ensures timely identification and response to prevent losses.
A.7.5 Audit logs must be protected to prevent modification or deletion.	Malicious users attempt to hide their presence and activity by changing audit log entries. It is imperative that the integrity of audit logs be preserved. Examples of mechanisms to protect the integrity of audit logs include cryptographic hash functions and digital signatures.
A.7.6 Audit logs must be retained for least one year with a minimum of three months immediately available for analysis.	Retention ensures access to audit logs for investigations.

Security Requirements	Guidance
A.7.7 A method must be implemented for the timely identification of attack patterns and undesirable behavior across systems. For example, consider a method using coordinated manual reviews and/or centrally managed or automated log-correlation tools that includes the following at a minimum: • Identification of anomalies or suspicious activities as they occur • Issuance of timely alerts upon detection of suspicious activity or anomalies to responsible personnel • Response to alerts in accordance with documented response procedures	Analysis of network activity assists with identification of non-standard processing that may be the result of malicious activity.

Appendix B Software Tamper-responsive Attack Costing Framework

***Note:** This appendix assumes that the reader is familiar with the concepts captured in Appendix B Physical Attack Potential Formula covered within the PCI PTS POI Derived Test Requirements document.*

There are differences between a hardware-based tamper-responsive system and a software-based tamper-responsive system that must be considered as we look at the attack-costing framework. The key differences are:

- Physical Attacker Present versus Non-Physical Attacker Remote

  Attacking a hardware tamper-responsive system such as a PCI PTS POI device generally requires an attacker to be physically present with the attack target. Hardware security controls are implemented to both detect and respond to a physical attack against the system. When detected, the typical response of a hardware tamper-responsive system is to delete all sensitive assets with no means to recover.

  Attacking a software-tamper-responsive system, on the other hand, does not generally require the attacker to be physically present with the attack target. While some attack vectors require physical access to the system, software attack vectors usually are executed remotely. For example, attacks may be executed with a piece of code or an application under a different execution or privilege context, such as an application with root privileges attacking the system in user mode. Hence, detection of the attack may not always be straightforward and would rely on indirect pieces of data points. Typical detection strategies rely on anomaly-detection algorithms, with the data points coming from the software application as input and monitored over time. When an attack is detected, the response options are more varied than those of a hardware tamper-responsive system.

- Stand-alone Detection versus Distributed Detection

  Detection mechanisms of a hardware tamper-responsive system typically are self-contained within the device, relying on a change in a physical property, such as temperature or voltage, to detect an attack. The available data for the attack detection engine typically is well defined. The decision-making process of the detection engine is binary in its conclusion as to whether the system is under some form of attack.

  Software tamper-responsive systems typically have tamper-detection capabilities distributed between the mobile application and the back-end monitoring system where the attack detection engine is located. The mobile application may also gather local system data that is then sent to the back-end, where it is used by anomaly-detection algorithms to decide whether there is actual attack on the local system. The back-end monitoring system then makes a corresponding response decision. As a result, the detection engine has the ability to make decisions that are not as binary as a stand-alone detection engine.

- Individually versus Collectively

Due to the nature of the attacks against hardware-tamper mechanisms, the attacker must attack one device at a time. Each device under attack is subjected to the same physical exploitation process in an attempt to compromise the hardware-based protection.

On the other hand, exploits against software tamper-responsive systems may target the entire collection of similar systems using malware or a virus as the distribution medium. Hence, attack vectors for software tamper-responsive systems have an additional risk dimension of scalability, where the full population of similar systems could be potential targets.

The objective of this framework is to model attack vectors that deployed solutions may encounter. The cost model will not include attack vectors that are not also considered by this Standard (for example, a direct hardware attack on the device during the exploitation stages).

Additional Considerations for Attack Cost Calculations

From the differences described at the beginning of this appendix, we derive the following factors that are considered when developing the attack cost framework for a software tamper-responsive system. The attack cost factors are as follows:

- Attacker Present
- Attacker Remote
- Back-end Monitoring

Attacker Present

Attackers typically have full access to the CPoC application downloaded from the web store and the COTS device on which the application is running. With the device in front of him/her, the attacker can proceed to identify and exploit both the CPoC application and COTS device. Factors below consider the attack cost where an attack vector requires physical access to the CPoC application and COTS device:

1. **Access to the COTS device**

   Attackers may not always have full access to the device. This attack cost factor considers attack vectors that require varying degree of access to the physical device under attack. Four types of access are identified.

   - **Remote, no user interaction:** The attack is executed remotely on a target device and no user interaction is expected for the attack to be successful. Example of such an attack vector is when malicious code is injected into the CPoC application.
   - **Remote, user interaction required:** The attack is executed remotely on a target device, but user action is required on the target device to allow the attack to be executed successfully. Example of such an attack vector is when the user clicks on a malicious URL.
   - **Local locked:** The attack is executed with physical access to the device, but without requiring that the device is unlocked. Example is when the attack vector requires connecting the device via USB to a computer to initiate the attack without requiring the user to first unlock the device.

- **Local unlocked:** The attack is executed with physical access to the device, but requires that the device be first unlocked by the user. Example is when the attacker executes a jailbreaking process on a user's phone that has been left unlocked.

2. **Equipment Required**

   Attackers may require the use of equipment in the execution of some attack vectors. The type of equipment required to execute an attack vector will determine the attack cost associated. The three levels are:

   - **Standard/software only:** Where the attacker uses equipment that easily is obtainable from a consumer electronics store or executes the attack vector using software only. Examples are USB cables and software to spoof geolocation reporting by the phone.
   - **Specialized:** Where the attacker uses specialized equipment that is not easily obtainable, but has to be either custom built or modified from a piece of standard equipment to serve a specialized attack function. Examples are IMSI tracker and SIM card seizure tools.
   - **Chip level:** Equipment that directly attacks the chipsets on the device in an attempt to compromise sensitive data. Examples include equipment to initiate an electromagnetic fault injection (EMFI) attack on the chip.

3. **Expertise to Execute the Attack Vector**

   Optimizing the attack vector to target the various combinations of system configurations would require different levels of expertise, depending on both publicly available information and the attacker's technical understanding of the systems in question. Identified levels of expertise are as follows:

   - **Layman:** Persons without professional or specialized knowledge in a particular subject. They are unknowledgeable compared to experts, proficient, or skilled persons with no particular expertise, but who are capable of implementing simple steps to optimize an attack vector. For the purpose of exploitation, they can implement an attack based on a script or a written procedure without requiring any particular skill.
   - **Skilled:** Persons able to perform more complex optimization to attack vectors without direction. They have the ability and training to perform a specific task well.
   - **Proficient:** Persons who are highly competent and have the necessary ability, knowledge and skill to perform complex customization of attacks successfully. They are familiar with the security functionalities and behavior of the underlying systems.
   - **Experts:** Persons who are extremely knowledgeable and skillful in one or more areas. They are very familiar with the underlying algorithms, protocols, hardware components, physical and logical architectures implemented in the device or system type, and the principles and concepts of security employed.

4. **Attack Time**

   The attack time factor is the amount of time that is required for the identification and exploitation of an attack vector. In calculating the attack cost for identification and exploitation, considerations for the other factors must be factored into the calculation of the time:

   - **Identification Costing**

As part of calculating the cost of attacking the solution, the lab should take into consideration reliance by the CPoC application on native COTS security features and any other security controls that the vendor has integrated into the solution, such as obfuscation and white-box crypto.

At the same time, the lab should also take into consideration the four factors above (Scalability, Expertise to Execute the Attack Vector, Quality of Attestation Data, and Knowledge of the back-end monitoring system) in the estimation of the attack time for the identification phase.

– **Exploitation Costing**

  In calculating the attack time exploitation cost, the operational quality of the back-end monitoring system must be factored in. The lab will be required to have access to the back-end monitoring system as it assesses the exploitation attack time factor. This is required to ensure that the attack-exploitation time takes into consideration how the back-end monitoring system will respond to the specific attack vector. The lab is expected to evaluate the operational quality of the back-end monitoring system (see below) at the same time it is making a determination of the attack exploitation time cost.

Attacker Remote

In this scenario, the attackers do not have the COTS device in front of them. As a result, the attacker may not have full access to the CPoC application downloaded from the web store and the COTS device on which the application is running. Factors below consider the attack cost where the attack is conducted by a remote attacker:

5. **Scalability Factor**

   While the time and resources required to identify and exploit a vulnerability are the same for both hardware and software tamper-responsive systems, we have to take into consideration that a software attack on a solution running on a COTS platform may be scalable to impact a population of similar systems within a very short time frame. The same exploit can be optimized to apply to different system configurations. Customization would then include consideration and identification of the deployment mechanism—e.g., malware, phishing, virus—used to distribute the exploit to the system population in question:

   – **No customization required:** The attack vector can be applied to the entire population of system configurations.

   – **Customized for each vendor**: The attack vector must be optimized based on the vendor/manufacturer of the solution.

   – **Customized for each device model**: The attack vector must be optimized to work for each model of device.

   – **Customized for each major OS release**: The attack vector must be optimized to work on each of the major OS versions supported by the solution.

   – **Customized for each minor OS release**: The attack vector must be optimized to work on each of the minor OS version supported by the solution.

   – **Customized for each instance**: The attack vector must be optimized for each instance of a system configuration.

6. **Expertise to Optimize the Attack Vector for Scalability**

Optimizing the attack vector to target the various combinations of system configurations would require different levels of expertise, depending on both publicly available information and the attacker's technical understanding of the systems in question. Identified levels of expertise are as follows:

- **Layman:** Persons without professional or specialized knowledge in a particular subject. They are unknowledgeable compared to experts, proficient or skilled persons with no particular expertise, but who are capable of implementing simple steps to optimize an attack vector. For the purpose of exploitation, they can implement an attack based on a script or a written procedure without requiring any particular skill.
- **Skilled**: Persons able to perform more complex optimization to attack vectors without direction. They have the ability and training to perform a specific task well.
- **Proficient:** Persons who are highly competent and have the necessary ability, knowledge, and skill to perform complex customization of attacks successfully. They are familiar with the security functionalities and behavior of the underlying systems.
- **Experts:** Persons who are extremely knowledgeable and skillful in one or more areas. They are very familiar with the underlying algorithms, protocols, hardware components, physical and logical architectures implemented in the device or system type, and the principles and concepts of security employed.

7. Quality of Attestation Data

To bypass the monitoring system, the attacker has to suppress or simulate the attestation data sent by the CPoC application to the back-end monitoring system. Hence, the ability of the CPoC application to send the attestation data and the quality of the attestation data are important in determining the effort required to simulate this data to subvert the back-end monitoring system. The levels of the attestation data can be differentiated as:

- **Low quality**, where the data is suppressed easily or can be simulated easily by another application, with the intent of fooling the back-end monitoring system that the system has not been modified. Examples of such data known to be easily spoofed include, but are not limited to, geolocation data and the device's IP address.
- **Medium quality**, where the data provides a high level of assurance that the information is authentic and has not been spoofed. An example is the integrity information from white-box crypto solutions.
- **High quality**, where the data cannot be easily spoofed or simulated. Examples are cryptographically based attestation data or attestation data that contains information obtained from the underlying hardware. Examples of such data include information from hardware-based modules like Secure Elements or secure processors.

The quality of the attestation data does not apply to any individual datum, but to the sum of all the attestation data provided by the CPoC application to the back-end monitoring system by which attack detection decisions are made. It is the responsibility of the lab to determine the rating of the quality of the combined set of attestation data.

8. Knowledge of the back-end monitoring systems

This refers to the information of the back-end monitoring systems. It includes information on its capabilities and behavior, possibly including the anomaly detection algorithms used to interpret the various attestation data that comes from the application. Identified levels are as follows:

- **Public information** about the back-end monitoring system (or no information): Information is considered public if it can be easily obtained by anyone (for example, from the Internet) or if it is provided by the vendor to any customer.
- **Restricted information** concerning the back-end monitoring system (for example, as gained from vendor technical specifications): Information is considered restricted if it is distributed on request and the distribution is registered—for example, the PCI PTS POI DTRs.
- **Sensitive information** about the back-end monitoring system—for example, knowledge of internal design, which may have to be obtained by "social engineering" or exhaustive reverse-engineering.

9. Attack Time

The attack time factor is the amount of time that is required for the identification and exploitation of an attack vector. In calculating the attack cost for identification and exploitation, considerations for the other factors must be factored into the calculation of the time:

- **Identification Costing**

  As part of calculating the attack cost of attacking the solution, the lab should take into consideration reliance by the CPoC application on native COTS security features and any other security controls that the vendor has integrated into the solution, such as obfuscation and white-box crypto.

  At the same time, the lab should also take into consideration the four factors above (Scalability, Expertise to Execute the Attack Vector, Quality of Attestation Data, and Knowledge of the back-end monitoring system) in the estimation of the attack time for the identification phase.

- **Exploitation Costing**

  When calculating the attack time exploitation cost, the operational quality of the back-end monitoring system must be factored into the cost. The lab will be required to have access to the back-end monitoring system as it assesses the exploitation attack time factor. This is required to ensure that the attack exploitation time takes into consideration how the back-end monitoring system will respond to the specific attack vector. The lab is expected to evaluate the operational quality of the back-end monitoring system (see below) at the same time it determines the attack exploitation time cost.

Back-end Monitoring

The back-end monitoring system is a critical component of the overall software tamper-responsive system. This component is especially critical in a software-based payment solution where, depending on security and risk management policies, the monitoring system may terminate the payment transaction capability of any COTS immediately when there are signs that the device may be compromised.

10. Operational Quality of Back-end monitoring system

It is important that the various rules used for anomaly detection, processes to update the monitoring systems, and detection data from the CPoC application are updated constantly based on the latest information available. This will in large part depend on proficient personnel trained to identify attack signatures and provide the relevant updates to the back-end monitoring systems.

The three levels are:

- Low operational quality, where the rules, policies, processes, and personnel involved in operating the back-end monitoring system are not able to demonstrate the proficiency required to ensure the timely identification of attacks attempts.
- Medium operational quality, where the lab was able to establish a level of comfort where the rules, policies, processes, and personnel operating the back-end monitoring system understand their roles and will ensure that the majority of attack attempts identified.
- High operational quality, where the rules, policies, processes, and personnel involved in operating the back-end monitoring system demonstrate a high level of proficiency that provide the assurances to the lab that any attack attempts will be promptly identified, and the system updated to respond appropriately to any future attempts.

It is expected that a lab will provide an initial identification of the operational quality of the back-end monitoring system when the solution is first presented for testing. Since the solution would not have been in production, the initial identification costing of the quality would provide only a baseline cost. Subsequent periodic testing will then be required to determine the exploitation attack cost depending on the quality of the back-end monitoring system.

Given the nature and importance placed on back-end monitoring systems, vendor solutions that are rated Low either during the initial evaluation or during subsequent periodic testing will fail the evaluation immediately.

An Approach to Calculation

The section above identifies the factors to be considered.

Table 3 provides guidelines for the individual factors.

For a given attack, it might be necessary to make several passes through the table for different attack scenarios (for example, trading off scalability for detection). The lowest value obtained for any of these passes should be retained. In the case of a vulnerability that has been identified and is in the public domain, the identifying values should be selected for an attacker to uncover that attack scenario in the public domain, rather than to initially identify it.

Table 3: Guidelines for Calculating Individual Factors

Attack Factors		Range	Guidelines	
			Identification	Exploitation
Attacker Present	Access to the COTS device	Remote, no user interaction	NA	0
		Remote, user interaction required	NA	1
		Local locked	NA	2
		Local unlocked	NA	3
	Equipment Required	Standard/software only	0	0
		Specialized	3	3
		Chip level	7	7
	Expertise to Execute the Attack Vector	Layman	NA	0
		Skilled	NA	1
		Proficient	NA	3
		Expert	NA	4
	Attack Time	≤ 12 hours	0	0*
		≤ 1 day	2	2*
		≤ 1 week	3	3*
		≤ 1 month	5	5*
		Beyond 1 month	8	8*
Attacker Remote	Scalability Factor	No customization required	1	NA
		Customized for each vendor	2	NA
		Customized for each device model	3	NA
		Customized for each major OS variant	5	NA
		Customized for each minor OS variant	8	NA
		Customized for each instance	13	NA
	Expertise to Optimize the Attack Vector for scalability	Layman	NA	0
		Skilled	NA	1
		Proficient	NA	3

Attack Factors		Range	Guidelines	
			Identification	Exploitation
		Expert	NA	4
	Quality of Attestation Data (Bypassing automated monitoring)	Low	0	0
		Medium	5	5
		High	10	10
	Knowledge of the back-end monitoring system	Public	0	0
		Restricted	5	5
		Sensitive	10	10
	Attack Time	≤ 12 hours	0	0*
		≤ 1 day	2	2*
		≤ 1 week	3	3*
		≤ 1 month	5	5*
		Beyond 1 month	8	8*
Back-end Monitoring	Operational Quality of back-end monitoring system	Low	0*	Costing to be provided by a periodically recurring process. (See below)
		Medium	5*	
		High	10*	

* Exploitation cost of attack time will be done in tandem with the calculation of the operational quality of the back-end monitoring system as the lab is expected to have access to the back-end monitoring system while attempting to exploit the system.

Operational Quality of Back-end Monitoring Systems—Exploitation

Unlike a hardware-based responsive system that has limited opportunity to be updated in the field, a software-based tamper-responsive system with a back-end monitoring component has the continued opportunity and expectation for the solution to be constantly updated and patched in response to newly discovered vulnerabilities. Hence, when evaluating the cost to exploit software-based tamper-responsive systems, it is important to include an ongoing evaluation of the operational quality of the back-end monitoring system beyond the initial identification of its quality.

Therefore, it is expected that a periodic testing of the back-end monitoring system be conducted to ensure its operational quality is being maintained. This testing will be executed in production like the current quarterly scan requirement under PCI DSS. The objective of this ongoing assessment is to ensure operational quality for the back-end monitoring system and how it has been updated to respond to newly identified attack vectors and vulnerabilities and provide in-field update of the CPoC application.

Table 4: Period In-field Test

Vendor Expectations	Lab Expectations
1. Provide information on the actual adoption rate of the various supported COTS platform. 2. Provide information on how the vendor has kept up with the latest attack vector in the industry. 3. Be prepared to provide supporting information from the monitoring system of the test results.	1. Provide tester with vendor information on the various attestation data and events that would result in back-end monitoring system alerts. 2. Sign up for a merchant account using different phones and potentially from different geolocations, initiate test to attempt bypass the controls. 3. Obtain monitoring system event log information from the vendor. 4. Taking the actual merchant distribution of the various COTS platform into consideration, calculate the exploitation cost by evaluating the operational quality of the back-end monitoring system.

An approach similar to this cannot consider every circumstance or factor, but should give a better indication of the attack potential. Other factors, such as the reliance on unlikely chance occurrences, are not included in the basic model, but can be used by a tester as justification for a rating other than those that the basic model might indicate.

Determining Applicable Time and Levels

For each phase, the testing laboratory shall document all necessary steps, including all the factors described above. This information is best summarized in a table containing all the items described above.

Attack Example

Table 5: Attack Example—Remote Side-channel Extraction of Keys

Conditions	Assumptions
1. The COTS systems are using a secret/private cryptographic system on the COTS device, where the key storage is vulnerable to some remote side-channel extraction, such as through a cache timing or speculative execution timing vulnerability. 2. The attack must be customized for each minor OS version, but can be deployed through JavaScript on the COTS browser (so no user interaction is required). 3. The attack does not interact with the attestation component, so attestation component operational quality is assigned a "low" value. 4. Triggers must be assigned a "low" value.	None

Table 6: Attack Example—Factor Calculation

Attack Factors		Range	Guidelines	
			Identification	Exploitation
Attacker Present	Access to the COTS Device	Remote, no user interaction	NA	0
	Equipment Required	Standard/software only	0	0
	Expertise to Execute the Attack	Skilled	NA	1
	Attack Time	≤ 12 hours		0*
		Beyond 1 month	8	
Attacker Remote	Scalability Factor	Customized for each instance	13	NA
	Expertise to Optimize the Attack Vector for Scalability	Expert	NA	4
	Quality of Attestation Data			
	(bypassing automated monitoring)	Low	0	0
	Knowledge of the back-end monitoring system	Public	0	0
	Attack Time	≤ 12 hours	N/A	0*
		Beyond 1 month	8	
Back-end Monitoring	Operational Quality of back-end monitoring system	Low	0*	Costing to be provided by a periodically recurring process.
		Attack Potential per Phase	29	5
		Total Attack Potential	**34**	

Appendix C Minimum and Equivalent Key Sizes and Strengths for Approved Algorithms

Table 7 lists the minimum key sizes and parameters for the algorithms used with key transport, exchange, or establishment and for data protection in connection with these requirements. Approved key establishment schemes are described in NIST SP800-56A (ECC/FCC6-based key agreement), NIST SP800-56B (IFC-based key agreement) and NIST SP800-38F (AES-based key encryption/wrapping).

Other key sizes and algorithms may be supported for non-payment brand relevant transactions; otherwise, these are the only encryption algorithms designated as Approved Algorithms.

Table 7: Minimum Key Size

Algorithm	IFC (RSA)	ECC (ECDSA, ECDH, ECMQV)	FFC (DSA, DH, MQV)	AES
Minimum key size in number of bits	2048	224	2048/224	128

Key-encipherment keys are to be at least of equal or greater strength than any key they protect. This applies to any key-encipherment keys used for the protection of secret or private keys that are stored, keys used to encrypt any secret, or private keys for loading or transport. For purposes of this requirement, the algorithms and key sizes by row in Table 8 are considered equivalent. In Table 8:

- RSA key size refers to the size of the modulus.
- Elliptic Curve key size refers to the minimum order of the base point on the elliptic curve. This order is to be slightly smaller than the field size.
- DSA key sizes refer to the size of the modulus and the minimum size of a large subgroup.

[6] IFC: Integer Factorization Cryptography; ECC: Elliptic Curve Cryptography; FFC: Finite Field Cryptography

Table 8: Equivalent Key Sizes

Algorithm	Effective Bit Strength	IFC (RSA)	ECC (ECDSA, ECDH, ECMQV)	FFC (DSA, DH, MQV)	AES
Minimum key size in number of bits	112	2048	224	2048/224	–
Minimum key size in number of bits	128	3072	256	3072/256	128
Minimum key size in number of bits	192	7680	384	7680/384	192
Minimum key size in number of bits	256	15360	512	15360/512	256

For implementations using Diffie-Hellman (DH) or Elliptic Curve Diffie-Hellman (ECDH):

- DH implementations entities must generate and distribute the system-wide parameters securely: generator g, prime number p and parameter q, the large prime factor of (p - 1). Parameter p must be at least 2048 bits long and parameter q must be at least 224 bits long. Each entity must generate a private key x and a public key y using the domain parameters (p, q, g).
- ECDH implementations entities must securely generate and distribute the system-wide parameters. Entities may generate the elliptic curve domain parameters or use a recommended curve (see *FIPS186-4*). The elliptic curve specified by the domain parameters must at least be as secure as P-224. Each entity must generate a private key d and a public key Q using the specified elliptic curve domain parameters. (See *FIPS 186-4* for methods of generating d and Q.)
- Each private key is to be statistically unique, unpredictable, and created using an approved RNG, as described in this document. See Table 10: Random Number Generators for more information.
- Entities are to authenticate the DH or ECDH public keys using DSA, ECDSA, a certificate, or a symmetric MAC (see *ISO 16609 – Banking – Requirements for* message authentication *using symmetric techniques*). One of the following should be used:
  - MAC algorithm 1 using padding method 3
  - MAC algorithm 5 using padding method 4

TLS implementations are to prevent the use of cipher suites that do not enforce the use of cryptographic ciphers, hash functions, and key lengths as outlined in this appendix.

Key Check Values (KCVs) are values that are used to identify a key without revealing any bits of the actual key itself. Some check values are computed by encrypting an all-zero block using the key or component as the encryption key, using the leftmost n-bits of the result; where n is 24 bits (10 hexadecimal digits or 5 bytes) for AES. Alternatively, AES uses a technique where the KCV is calculated by MACing an all-zero block using the CMAC algorithm as specified in ISO 97971 (see also NIST SP 800-38B). The check value will be the leftmost n-bits of the result, where n is 10 hexadecimal digits. AES is the block cipher used in the CMAC function. The key length of a key or component will be MAC'd using the AES block cipher with an equivalent length (e.g., AES-128 uses 128-bit for MAC, AES-256 uses 256).

For hash algorithms used for authentication or security purposes, only the algorithms and associated bit lengths in Table 9 are permitted.

Table 9: Hash Algorithms

Algorithm	Length
SHA2 family	>255
SHA3 family	>255

Random Number Generator (RNG) are either a Deterministic Random Number Generator (DRNG) or a Non-deterministic Random Number Generator (NRNG).

All DRNG must be seeded by an NRNG that provides sufficient authenticated entropy. The entropy required must be at least as many bits as the intended key strength and should be twice as many bits. Entropy sources are discussed in NIST SP800-90B.

Table 10: Random Number Generators

RNG	Requirement
DRNG	Tested and approved under NIST SP 800-90A or ISO/IEC 18031 [§9]
NRNG	Tested and approved under NIST SP 800-90C or ISO/IEC 18031 [§8]

Prime Number Generators

For cryptographic processes that require prime numbers, use prime number generators tested to ISO/IEC 18032 Information Technology--Security Techniques: Prime Number Generation or X9.80 Prime Number Generation, Primality Testing, and Primality Certificates.

Appendix D Software Security Requirements

Table 10 describes the secure software requirements and corresponding guidance that provide a baseline for software development activities that support the solution. Design, development and software maintenance used by a vendor affect the overall security of the solution. Therefore, it is important that these vendor processes adhere to industry-recognized and accepted practices.

Table 10: Software Security Requirements

Software Security Requirements	Guidance
D.1.1 The software development process must be based on a formal process for secure development of applications, which includes: • Development processes based on industry standards and/or best practices • Information security incorporated throughout the software development life cycle • Security reviews performed prior to release of an application or application update At a minimum, the documentation must include quality control standards and measurements and change-control practices to ensure oversight of the development processes.	Without the inclusion of security during the requirements definition, design, analysis, and testing phases of the software development process, security vulnerabilities can be introduced into application code inadvertently or maliciously. Examples of secure software development practices include: • *ISO/IEC 27034 Application Security Guideline* • *NIST Special Publication 800-64 Revision 2* • *SEI CERT Coding Standards* Documentation should include techniques and methods used, specific notes on how things should be done to ensure security controls are functioning, and how to prevent vulnerabilities through misconfigurations. The vendor document should include development process information that can be audited. Examples of such documentation include: • Software-quality procedures • Documentation and software-control procedures • Change forms • Change-control logs • Change records
D.1.2 Test data, accounts, user IDs, and passwords must be removed before release.	Test data and accounts should be removed from the software before it is released because inclusion of these items may expose information about key constructs within the application. Pre-release custom accounts, user IDs, and passwords could be used as a back door for developers or other individuals with knowledge of those accounts to gain access to the software, which could compromise the software and related account data.

Software Security Requirements	Guidance
D.1.3 Source code must be reviewed using manual or automated processes prior to release and after any significant change. Doing so helps to identify any potential coding vulnerability. The code review must include at least the following: • Code changes are reviewed by individuals other than the originating code author and by individuals who are knowledgeable in code-review techniques and secure coding practices. • Code reviews ensure code is developed according to secure coding guidelines. • Appropriate corrections are implemented prior to release. • Code-review results are reviewed and approved by management prior to release. • Documented code-review results include management approval, code author, code reviewer, and corrections that were implemented prior to release. ***Note:*** *This code review requirement applies to all application components (both internal and public-facing applications) as part of the system development life cycle. Code reviews can be conducted by knowledgeable internal personnel or third parties.*	Security vulnerabilities in the software code are commonly exploited by malicious individuals to gain access to a network and compromise sensitive data. To protect against these types of attacks, proper code-reviewing techniques should be used. Code-review techniques should verify that secure-coding best practices were employed throughout the development process. The application vendor should incorporate relevant secure coding practices as applicable to the particular technologies used. This may include the use of static and/or dynamic code analysis tools, and validation of any known vulnerabilities and weaknesses in third-party applications and libraries that are used. Reviews should be performed by an individual knowledgeable in the technology and experienced in code-review techniques to identify potential coding issues. Assigning code reviews to someone other than the developer of the code allows an independent, objective review to be performed. Correcting coding errors before the code is released prevents faulty code from exposing customer environments to potential exploit. Faulty code is also far more difficult and expensive to address after it has been deployed. Including a formal review and signoff by management prior to release helps to ensure that code is approved and has been developed in accordance with policies and procedures.
D.1.4 Secure source-control practices must be implemented to verify integrity of the source code during the development process.	Good source-code control practices help ensure that all changes to code are intended, authorized, and performed only by those with a legitimate reason to change the code. Examples of these practices include check-in and checkout procedures for code with strict access controls and a comparison, such as a checksum, immediately before updating code to confirm that the last-approved version has not been changed.

Software Security Requirements	Guidance
D.1.5 Software must be developed according to industry best practices for secure coding techniques, including: • Developing with least privilege for the software execution environment • Developing with fail-safe defaults: all execution is, by default, denied unless specified within initial design • Coding techniques include documentation of how sensitive information, such as cryptographic material, certificates, and account data, is handled in memory. • Developing for all access point considerations, including input variances such as multi-channel input to the software	Developing software with least privilege is the most effective way to ensure non-secure assumptions are not introduced and exploited in the execution environment. Including fail-safe defaults could prevent an attacker from obtaining sensitive information about a software failure that could then be used to create subsequent attacks. Ensuring that security is applied to all accesses and inputs into the software avoids the possibility that an input channel may be left open to compromise. Attackers use various tools to capture sensitive data from memory. Minimizing the exposure of sensitive information while in memory helps reduce the probability that it can be captured by a malicious user or be saved unknowingly to disk in a memory file and left unprotected. This requirement is intended to ensure the consideration of how sensitive information is handled in memory. Understanding when sensitive data is present in memory, for how long, and in what format helps application developers to identify potential insecurities in their applications and determine whether additional safeguards are needed. Failure to consider these concepts while developing code could result in the release of a non-secure application and potentially excessive remediation at a later time.
D.1.6 Up-to-date training must be provided in secure development practices for application developers at least annually, according to the developer's job function and technology used. Example training topics include: • Secure application design • Secure coding techniques to avoid common coding vulnerabilities • Managing sensitive data in memory • Code reviews • Security testing (penetration-testing techniques) • Risk-assessment techniques ***Note:*** *Training for software developers may be provided in-house or by third parties. Examples of how training may be delivered include on-the-job, instructor-led, and computer-based.*	Ensuring developers are knowledgeable about secure development practices helps minimize the number of security vulnerabilities introduced through poor coding practices. Trained personnel are also more likely to identify potential security issues in the application design and code. Software development platforms and methods change frequently, as do the threats and risks to software applications. Training in secure development practices should keep current with changing development practices.
D.1.7 All software must be developed to prevent common coding vulnerabilities in software development processes.	The application layer is high-risk and may be targeted by both internal and external threats. Without proper security, account data and other confidential company information can be exposed. As industry-recognized common coding vulnerabilities change, vendor coding practices should also change to match.

Software Security Requirements	Guidance
D.1.8 Software vendor must follow change-control procedures for all software changes. Change-control procedures must follow the same software development processes as new releases and include the following: • Documentation of impact • Documented approval of change by appropriate authorized parties • Functional testing to verify that the change does not adversely impact the security of the system • Back-out or product de-installation procedures	If not properly managed, the impact of software updates and security patches might not be fully realized and could have unintended consequences.
D.1.9 The software development process must document and follow a software-versioning method that includes: • The format of the version scheme, including number of elements, separators, and character set consisting of alphabetic, numeric, and/or alphanumeric characters • Definition of what each element represents in the version scheme; for example, type of change (major, minor), security, or maintenance release	Without a thoroughly defined versioning method, changes to applications may not be identified properly, and customers and integrators/resellers may not understand the impact of a version change to the application. The versioning method should include a defined version scheme that specifically identifies the elements being used, the format of the version, and the hierarchy of the different version elements. The version scheme should clearly specify how each of the various elements is used in the version number. The version scheme can be indicated in a number of ways; for example: **N.NN.NNA**, where **N** indicates a numeric element and **A** indicates an alphabetic element. The versioning scheme should identify the character set (0-9, A-Z, a-z ...) that can be used for each element in the version. Without a properly defined version scheme, changes made to the application may not be represented accurately by the version number format.
D.1.10 Risk-assessment techniques, such as threat-modeling, must be used to identify potential application security design flaws and vulnerabilities during the software development process. Risk-assessment processes include the following: • Coverage of all functions of the software, including but not limited to, security-impacting features and features that cross trust-boundaries • Assessment of decision points, process flows, data flows, data storage, and trust boundaries • Identification of all areas within the software that interact with sensitive information or the monitoring system • A list of potential threats and vulnerabilities resulting from data flow analyses and assigned risk ratings (high, medium, or low risk) • Implementation of appropriate corrections and countermeasures during the development process • Documentation of risk-assessment results for management review and approval	To maintain the quality and security of software, risk-assessment techniques should be employed by software developers during the development process. Threat modeling is a form of risk assessment that can be used to analyze constructs and data flows for opportunities where confidential information may be exposed to unauthorized application users. These processes allow software developers and architects to identify and resolve potential security issues early in the development process, improving software security and minimizing development costs.

Software Security Requirements	Guidance
D.1.11 A process must be established to identify and manage vulnerabilities, as follows: • Identify new security vulnerabilities using reputable sources for obtaining security vulnerability information • Assign a risk ranking to all identified vulnerabilities, including vulnerabilities involving any underlying software or systems provided with or required by the software • Test applications and updates for the presence of vulnerabilities prior to release • Perform application-layer penetration testing at least annually or whenever there is a significant change that modifies security function	Developers who are knowledgeable in vulnerabilities within their own software or in underlying components should be able to resolve those vulnerabilities prior to release, or implement other mechanisms to reduce the likelihood that the vulnerability may be exploited if a third-party security patch is not available immediately. Reputable sources should be used for vulnerability information and/or patches in third-party software components. Sources for vulnerability information should be trustworthy and often include vendor websites, industry news groups, mailing lists, or RSS feeds. Examples of industry sources include *NIST's National Vulnerability Database*, *MITRE's Common Vulnerabilities and Exposures list*, and the *U.S. Department of Homeland Security's US-CERT* websites. When a vulnerability that could affect the application is identified, the risk that the vulnerability poses should be evaluated and ranked. This requires a process to monitor industry sources actively for vulnerability information. Classifying the risks (high, medium, or low) allows vendors to identify, prioritize, and address the highest risk items (for example, by releasing high-priority patches more quickly). This reduces the likelihood that vulnerabilities posing the greatest risk to customer environments will be exploited. Finally, adequate testing by a qualified internal employee (with organizational independence) or external third party should be included in the application-vulnerability management process to ensure that any identified vulnerabilities have been addressed properly prior to release. The scope of the testing should include the CPoC application, the protocols used to communicate between the software components (e.g., remote component of contactless kernel) and back-end processing and monitoring systems. Without a formal review and acknowledgment from a responsible party, critical security processes may be missed or excluded, resulting in a faulty or less secure application. Additional information can be found in the *PCI SSC Information Supplement, Penetration Testing Guidance*.

Software Security Requirements	Guidance
D.1.12 A process must be established for timely development and deployment of security patches and upgrades as follows: • Patches and updates are delivered in a secure manner with a known chain of trust. • Patches and updates are delivered in a manner that maintains the integrity of the patch and update code. • Provide instructions for customers about secure installation of patches and updates.	To minimize the time frame and likelihood that the vulnerability could be exploited, software updates to address security vulnerabilities should be developed and released to customers as quickly as possible after a critical vulnerability has been identified. All software requires mechanisms to ensure its integrity and authenticity. Security patches should be distributed in a manner that prevents malicious individuals from intercepting the updates in transit, modifying them, and then redistributing them to unsuspecting customers. Distribution for the mobile application component relies typically on commercially hosted application repositories like the Google Play store or Apple App Store. While these repositories have mechanisms to ensure the integrity and authenticity of the software they distribute, it is expected that these cannot be relied upon to ensure an authorized software update. The application should contain built-in mechanisms to ensure that updates are authorized. Security updates should include a mechanism within the update process to verify that the update code has not been replaced or tampered. Examples of integrity checks include, but are not limited to, checksums and digitally signed certificates.
D.1.13 Release notes must be included for all software updates, including details, impact of the update, and how the version number was changed to reflect the application update.	Release notes with details about software updates, including which files may have changed, which application function was modified, and any security-related features that may be affected. Release notes should also indicate how a particular patch or update affects the version number associated with the patch release.
D.1.14 A process must be implemented to document and authorize the final release of the software and any updates. Documentation includes: • Signature by an authorized party to approve formally release of the software or updates • Confirmation that secure development processes were followed by the vendor	Without a formal review and acknowledgment from a responsible party, critical security processes may be missed or excluded, resulting in a faulty or less secure application.
D.1.15 Develop, maintain, and disseminate an implementation guide that must: • Provide relevant information specific to the application • Address all requirements in the Standard • Include a review at least annually and upon changes to the application, and is updated as needed to keep the documentation current with all changes affecting the application and the requirements in the Standard.	A well-designed and detailed implementation guide helps to implement appropriate security measures and configurations within the application and its underlying components to meet the relevant SBPE requirements for protecting sensitive information. With each application update, system function and, in some cases, critical application security mechanisms are modified or introduced. If the implementation guide is not kept current with the latest versions of the application, users could overlook or misconfigure critical application security controls that could ultimately enable an attacker to bypass such security mechanisms and compromise sensitive data.

Appendix E Security Evaluation Laboratory Requirements

This appendix focuses on the equipment, skills and experience requirements necessary to undertake a security evaluation against the PCI Contactless Payments on COTS Standard.

Equipment

The laboratory must have possession of, and/or access to, all equipment (physical equipment and software tools) necessary to execute all test activities. Laboratories are expected to have, on site, most equipment defined by PCI PTS as "standard," "specialized," and "bespoke."

The following list describes the types of equipment a laboratory should have, of an adequate quality and sufficient volume, for the most commonly performed test activities.

- PCs/workstations, data storage, and data-backup facilities
- Virtualization environments for dynamic analysis
- Interfaces for equipment and devices under test (for example, cables, communications software, smart card reader, etc.)
- Tools for code disassembly and decompilation
- Environmental chambers for variable temperatures, etc.
- Electronics testing tools (for example, variable voltage supplies, signal generators, amplifiers, digital storage oscilloscope, etc.)
- Signal-acquisition equipment (for example, antennae, probes, EM coils, microphones, etc.)
- Signal-analysis and signal-processing software capable of filtering, compressing, synchronizing, or otherwise operating effectively on acquired signals
- Side-channel analysis test tools including effective user interface and configurable collection and analysis components
- Fault-injection resources such as perturbation source and glitch control tools for these sources (for example, pulse generators, lasers, etc.)
- NRNG analysis software
- Tools and interfaces capable of communicating with devices over various protocols to investigate logical anomalies and error-exploitation attacks such as fuzzing (for example, protocol analyzers and sniffers, configuration and analysis software, test scripts, etc.)
- Use of web proxies and traffic-interception tools for client server traffic analysis

The laboratory must have effective arrangements for utilizing vendor test tools when necessary for device-specific testing.

Skills and Experience

The laboratory’s personnel must have, collectively, the necessary skills and experience to execute all test activities. PCI SSC expects personnel to include individuals having expert-level expertise for subjects directly related to evaluations. This expertise must include both the knowledge needed to perform vulnerability analysis and strong practical capabilities for applying that knowledge in "hands-on" testing. Laboratory personnel must also have overall, thorough knowledge in diverse subject areas not directly associated with evaluations.

The following list describes the types of subject areas (capabilities, knowledge, and skills) laboratory personnel should have:

- Familiarity with mobile execution environments and their security models (for example, Android and iOS)
- Familiarity with software-protection tools and their analysis, including obfuscation and white-box cryptography
- Familiarity with hooking techniques
- Device physical components: structure and materials, how multiple components combine to realize security objectives, and how flaws can be identified and/or exploited
- Device security-critical components design and functionality, such as (but not limited to) keypads, displays, and cases
- Schematic representations of hardware and logic (for example, Gerber files and block diagrams) and the ability to detect flaws or weaknesses from these
- Device logical components: architecture functions of how multiple components interact to realize security objectives, and how flaws can be identified and/or exploited
- Formulation and analysis of all aspects of monitoring, penetration, or modification attacks; for example, but not limited to, device modification (electronic, mechanical, and chemical), side-channel analysis, and glitching/fault injection
- Operating evaluation tools for activities such as, but not limited to, virtual environments, failure analysis, signal processing, vulnerability scanning, communications interfaces, fuzzing, side-channel acquisition, side-channel analysis, OS testing, source-code analysis, NRNG testing, TCP/IP testing, and mechanical lab equipment
- Source code review, including the detection of vulnerabilities
- Programming languages such as C, and other programming languages that may be in scope of a device evaluation
- Assembler language and security-relevant behaviors of compilers/interpreters. Security-relevant characteristics of operating systems and operating system resource management including I/O, memory management, displays, prompts, keyboards, and readers
- Linux-based operating systems and any other operating systems that may be in scope of a device evaluation, including proprietary operating systems, application separation, access permissions, and application loading and deletion
- Cryptology, key management, firmware loading, and PIN encryption with regard to cryptology and the EMV application layer
- Transport layer security in devices (for example, Bluetooth, Ethernet, smart cards, USB, etc.)
- Operating vendors' development test tools as part of an evaluation
- Entire device lifecycles and the relevance of fraud models and threats
- Hardware security
- Machine learning, including adversarial techniques

Documentation and Materials

Documentation and samples required for the assessment of the solution should be agreed upon between the application vendor and the evaluation laboratory. The laboratory may need to request additional evaluation material when necessary. Examples of vendor-provided materials are as follows:

- Supporting documentation that will aid the evaluation, such as block diagrams, schematics, and flowcharts
- Any necessary hardware and software accessories required to perform software-based CPoC application functionality
- Documentation that relates to the development process that can be audited by the laboratory. Examples of such documentation include:
  - Software quality procedures

  - Documentation and software control procedures
  - Change forms
  - Change-control logs
  - Change records
- The user guidance provided by the vendor that includes a description of system-level security mechanisms that mitigate vulnerabilities that may be present in the CPoC application and COTS device

Laboratory Evaluation of Product and Monitoring Environment Basic Protections

The magnitude and scope of the evaluation of the contactless payment acceptance solution will vary, depending on the solution architecture. The evaluation includes a threat and vulnerability assessment of identified security assets. The vulnerability analysis should include currently known logical and physical attacks (threats) that are applicable to the contactless payment acceptance solution. The laboratory performs the required evaluation and generates an evaluation report documenting the results.

Evaluation may include physical testing of product samples, assessment of the design documentation, or auditing of the vendor's development processes. See *Documentation and Materials*.

Laboratory Review of Software Protection Tools

When the CPoC application is protected using software-protection tools, the laboratory should apply evaluation techniques that are appropriate to those tools and try to bypass them or determine their effectiveness in protecting CPoC application assets, including:

- De-Obfuscation of protected code
- Bypassing any anti-tamper protections
- Identification of control flows and data flows
- Direct key recovery from memory dumps
- Bypass of applied "node locking" or Device binding such that the CPoC application cannot be executed from different platforms than intended
- Protection of cryptographic keys or cryptographic APIs against misuse
- Analysis of white-box cryptography-protected implementation to obtain the WBC key

If the tools have been provided by a third party and applied to the CPoC application by the application vendor, the laboratory may establish whether there is any available and applicable assurance for the tool

***Note:** The laboratory will have to be able to justify any re-use of assurance within the context of the CPoC application evaluation if the tool has a number of parameterization options.*

Laboratory Review of Source Code

All source code within scope of the evaluation must be made available to the vendor's chosen laboratory, either at the approved laboratory's premises or by secure remote access. Code review may be performed at the vendor location, but the code should still be available at the approved laboratory premises for the duration of the evaluation.

If the CPoC application delegates functionality to open-source libraries, these must also form part of the code review.

If the third-party vendor of such tools can provide independent assurance for their integration with the CPoC application, it may be useful for the laboratory assessment; however, if the tool is configurable, the final assurance will be qualified by the tools' level of configuration rather than the assurance provided by the tool itself.

Laboratory Review of Source Code Development Processes

The integrity of the CPoC application software development process contributes to the assurance provided by the final product. The laboratory will establish that the vendor has implemented a software development methodology that includes quality controls and change controls.

Server-based Security Mechanisms

The CPoC application relies on server-based mechanisms as part of its security support. For example, rooting detection may be implemented by a server that samples the execution environment of the COTS device, or white-box components may be implemented at the server. It is expected that the laboratory includes a full analysis of these mechanisms, as they provide a major part of the system security.

Residual Risk

If, at the end of the evaluation, there remain any unevaluated proprietary components that contribute to the CPoC application security functionality, these will be listed as a Residual Risk to the system.

Deliverables from the Laboratory

- The security evaluation report of both the solution and its associated user guidance
- A Residual Risk Analysis that describes any security user guidance

Evaluation Results

Evaluation reports should be constructed as follows:

- A description of what was provided to the evaluation laboratory by the vendor
- A description of any existing assurance that was used to support the evaluation
- Whether a full, delta, or other type of evaluation was performed
- A description of the product architecture with accompanying diagrams
- For the CPoC application on the COTS device:
  - A vulnerability analysis of the application security functionality
  - Detail of any residual vulnerabilities
- Sufficient reporting of penetration testing to prove that the tests were completed, as appropriate, in order to reach the conclusions on the assurance level
- A description of any restrictions that were placed upon the laboratory by the vendor and prevented the evaluation from being fully white-box (for example, restricted access to source code or documentation)
- Conclusions of the evaluation should be modeled on the PCI PTS assurance-rating methodology, where vulnerabilities are rated in terms of their attack potential
- The report should contain a summary identifying:
  - A list of identified highest-risk, complete attack paths
  - The associated attack potentials
  - An assessment of any vendor-provided user guidance
  - A calculation of the overall assurance provided by the CPoC application

Appendix F Configuration and Use of the STS Tool

The NIST STS (Statistical Test Suite) is a reference implementation of the statistical tests described in *NIST SP 800-22* Revision 1a.

The tester shall use NIST's STS tool, version 2.1.2 or later, or its mathematical equivalent. The tester shall verify that the compiled instance of the STS tool is operating correctly on the testing device by testing the NIST-provided sample data and comparing the results with those found in *NIST SP 800-22* Revision 1a (SP800-22r1a), Appendix B. This configuration guidance is for use with STS version 2.1.2, though it will likely continue to be applicable to future versions.

Note about STS Versions: *Prior versions of STS include bugs that have been fixed in the current version. Previous versions must not be used unless the critical fixes present in the current NIST tool have been backported. At a minimum, prior versions must disable the Lempel-Ziv compression test [Hamano 2009] and include fixes to the DFT (Spectral) test [Kim 2004], the Overlapping Template test [Hamano 2007], the Non-Overlapping test [NIST 2014], and the "Proportion of Sequences Passing a Test" test interpretation.*

The tester should request and obtain a sample of 2^{30} bits from the vendor. The tester should exercise care to verify that the vendor-supplied data is interpreted correctly by the STS tool (the STS tool assumes that binary data is in big-endian format on all devices).

The STS testing on the data shall be judged as a "pass" if it passes all the tests for both the "Proportion of Sequences Passing a Test" interpretation approach and "Uniform Distribution of P-Values" interpretation approach. If the data does not pass all tests, and the failure is marginal, the tester should acquire additional data from the vendor and repeat the testing, including both the initial data and the additional vendor-supplied data.

The STS tool should be configured as per guidance provided in *SP 800-22 Revision 1a*, which is summarized below.

The settings in Table 11 are consistent with the SP 800-22 Revision 1a document:

Table 11: STS Tool Settings

Configuration Item	Setting	Reference in Keyword Below
Length of bit streams (n)	1,000,000	[1]
Number of bit streams (sample size) (M)	1,073	[2]
Block Frequency block length	20,000	[3]
Non-Overlapping Templates length	9	[4]
Overlapping Template length	9	[5]
Universal block length (L), number of initialization steps (Q)	L=7, Q=1,280	[6]
Approximate Entropy block length	8	[7]
Serial block length	16	[8]
Linear Complexity block length	1,000	[9]

Key to Configuration Item Table Above

[1] *n* must be selected to be consistent with the requirements of all of the tests to be run. The Overlapping Templates, Linear Complexity, Random Excursions, and Random Excursions Variant tests all require *n* to be greater than or equal to 10^6 in order to produce meaningful results. The Discrete Fourier Transform (Spectral) test requires *n* to equal 10^6. (See SP 800-22r1a Sections 2.8.7, 2.10.7, 2.14.7, 2.15.7, and [NIST 2010].)

[2] The number of bit sequences (sample size) must be 1,000 or greater in order for the "Proportion of Sequences Passing a Test" result to be meaningful. (See SP 800-22r1a Section 4.2.1.) This value will be 1,073 for the first test, but any additional testing (for example, further testing to resolve test failures) will necessarily include more bit sequences.

[3] For the Block Frequency test, if $n=10^6$, the test block size should be set between 10^4 and 10^6. (See SP 800-22r1a Section 2.2.7.)

[4] The Non-Overlapping test requires selection of a template length of 9 or 10 in order to produce meaningful results. (See SP 800-22r1a Sections 2.7.7 and 2.8.7.) For a template length of 10, the MAXNUMOFTEMPLATES constant (in defs.h) should be set to at least 284 prior to compiling STS, otherwise most 10-bit aperiodic templates with a leading 1 bit are discarded.

[5] The Overlapping test requires selection of a template length of 9 or 10 in order to produce meaningful results. When n=10^6, the template size of 9 comes closest to fulfilling the parameter selection criteria. (See SP 800-22r1a Section 2.8.7.)

[6] The Universal test block length (L) and initialization steps (Q) must be consistent with the table in SP 800-22r1a Section 2.9.7. For n=10^6, the only acceptable values are (L=6, Q=640) and (L=7, Q=1280).

***Note:** Any parameters passed into this test are discarded, and reasonable values are internally set. For n=10^6, STS automatically uses the parameters recommended here.*

[7] For the Approximate Entropy (ApEn) test, SP 800-22r1a Section 2.12.7 requires the block length to be less than $\lfloor \log_2 n \rfloor - 5$. Other analysis [Hill 2004] has shown that for n=1,000,000, block lengths greater than 8 can cause failures more often than expected for large scale testing.

[8] The Serial Test block length is also set based on n. If n=10^6, the block length must be less than 17. (See SP 800-22r1a Section 2.11.7.)

[9] The Linear Complexity test block length is required to be set to between 500 and 5,000 (inclusive) and requires that $\frac{n}{M} \geq 200$. (See SP 800-22r1a Section 2.10.7.)

References

- [Rukhin 2010] Rukhin, Andrew, et al., "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," NIST SP 800-22, Revision 1a.
- [Kim 2004] Kim, Song-Ju, et al., "Corrections of the NIST Statistical Test Suite for Randomness."
- [Hill 2004] Hill, Joshua (InfoGard Labs), "ApEn Test Parameter Selection."
- [Hamano 2007] Hamano, K. and Kaneko, T., "Correction of Overlapping Template Matching Test Included in NIST Randomness Test Suite," IEICE Trans. Fundamentals, vol. E90-A, no. 9, pp. 1788-1792, Sept. 2007.
- [Hamano 2009] Hamano, Kenji, "Analysis and Application of the T-complexity." Ph.D. thesis, The University of Tokyo.
- [NIST 2010] STS Software Revision History.
  URL: https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software
  Internet Archive: http://web.archive.org/web/20150520193625/http://csrc.nist.gov/groups/ST/toolkit/rng/revision_history_software.html
- [NIST 2014] Current STS Release Notes.
  URL: https://csrc.nist.gov/projects/random-bit-generation/documentation-and-software.
  Ω
  Internet Archive: http://web.archive.org/web/20150103230340/http://csrc.nist.gov/groups/ST/toolkit/rng/documentation_software.html