Complying with the PCI CPoC Standard

Get your mobile contactless payment application to market faster with Zimperium

ZIMPERIUM®

Introduction

Demand for contactless payments continues to surge as both consumers and merchants seek faster, more convenient, and safer methods to conduct transactions. Advances in the field have turned commercial off-the-shelf (COTS) phones and tablets into fully enabled, contactless mobile point-of-sale (mPOS) systems.

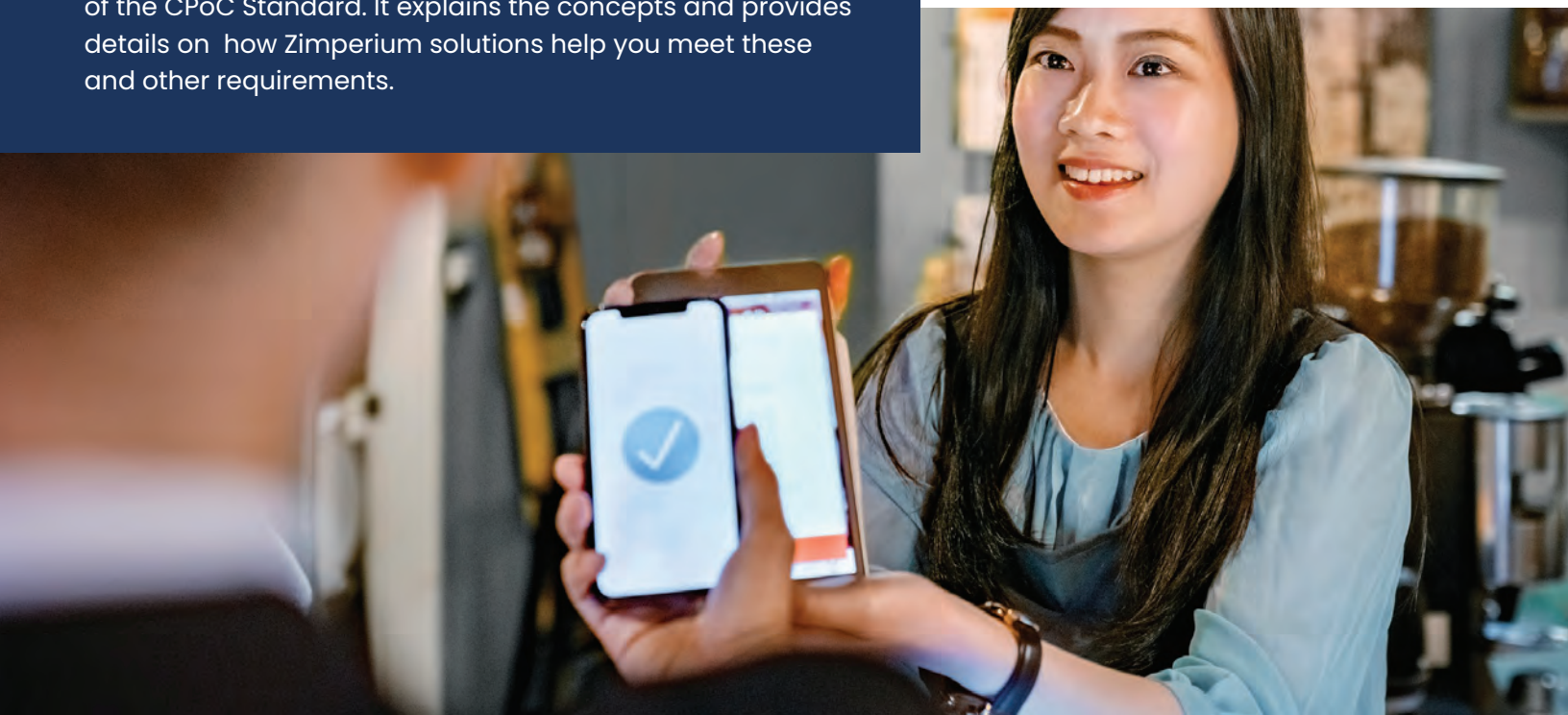
These Contactless Payments on COTS (CPoC) applications accept payments using the NFC interface in standard mobile devices, without requiring additional hardware.

To ensure payment security under this emerging technology, the PCI Security Standards Council published the CPoC Standard.[1] In addition to detailing app security requirements, it includes a stringent testing and certification program.

Zimperium's application protection solutions, zShield and zKeyBox, help developers and vendors secure their payment apps and comply with key elements of the CPoC Standard.

WHAT YOU'LL LEARN

The PCI CPoC guidelines cover multiple areas from general security components, to distinct security requirements for the application software, backend systems, and contactless kernel. This white paper takes an in-depth look at the requirements specifically related to protection of the application code and the use of white-box cryptography to secure encryption keys, which fall under sections 2.1 and 2.2 of the CPoC Standard. It explains the concepts and provides details on how Zimperium solutions help you meet these and other requirements.



CPoC Standard section 2.1:
Tamper and reverse- engineering protection

A primary requirement of the CPoC Standard, defined in section 2.1, is protecting the CPoC application from tampering and reverse engineering. Almost every attack on a software application begins with reverse engineering its code to understand the structure and logic, and identify avenues of attack.

For example, experienced hackers can look at assembly language code and recognize that it is performing cryptographic operations.

Once the inner workings of the target application are understood, attackers can tamper with the code, circumvent security restrictions, lift transaction-related code to perform unauthorized transactions outside of the application, change app behavior, and use other methods to steal secret keys, personal account data, payment card and PIN details, and funds.

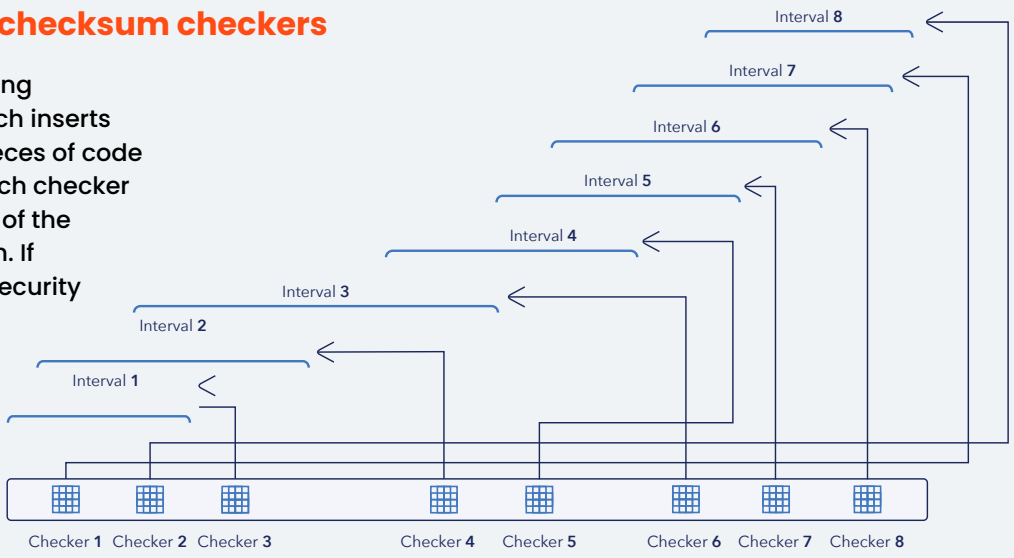
Since the application may be installed on a compromised or untrusted device, the software must contain internal protections including, but not limited to, the use of code obfuscation, internal integrity checks for code and processing flows, and code segment encryption.

zShield is a comprehensive solution that adds tamper-resistance, anti-reverse engineering, and obfuscation characteristics to software applications on all major platforms including Android and iOS. Its security features ensure not just compliance with the CPoC Standard, but also strong application-level protection against many software attacks.

In the next section, we examine the specific requirements and how Zimperium can help you achieve compliance and accelerate your time to market.

Integrity protection using checksum checkers

One of zShield's many anti-tampering measures is integrity checking, which inserts thousands of small, overlapping pieces of code called checkers. During runtime, each checker tests whether a particular segment of the executable has been tampered with. If tampering is detected, automatic security measures are triggered.



Section 2.1 Requirements

Section 2.1 of the CPoC Standard lists eight specific tamper-resistance requirements developers must implement for approval.

Req.	Description	Explanation and compliance
2.1.1	**Documentation must exist on how tamper resistance is achieved for each of the supported platforms of the CPoC application.**	Zimperium provides comprehensive documentation on the code obfuscation and anti-tampering methods used by zShield so that CPoC application development teams can readily comply with this requirement.
2.1.2	**The CPoC application must be protected by tamper-resistance measures to protect its code, including any code involved in the use or security of cryptographic keys.**	zShield transforms application source code by applying code and string literal obfuscation, code and data integrity protection, debugger detection, binary packing, and other application defense mechanisms. The end result is an application that looks and works the same as the original, but internally its code contains layers of self-defense mechanisms to prevent analysis and tampering. For additional cryptographic key protection, zKeyBox provides the highest available level of security for handling keys in software-only environments, protecting keys at all stages of their lifecycle.
2.1.3	**The attestation component must have the east privilege required to access proprietary APIs to determine the COTS platform state.**	zShield secures the attestation component or any code interacting with it, however, sound design principles must govern privilege implementation and handling. Any tools used to obtain attestation information and measurements of the execution platform must not require privilege escalation and should successfully run with the least privilege applied to the attestation component.
2.1.4	**Attestation code implemented in the CPoC application must be protected by tamper-resistance features.**	The strength of attestation code protections necessarily depends on whether or not the CPoC application developer has the source code of the attestation component. If so, the full power of zShield's advanced obfuscation and anti-tampering mechanisms can be applied. If the attestation process relies on a pre-compiled third-party toolkit or library, Code Protection still provides a number of features to help meet the attestation code protection requirement, such as signature checking of shared libraries, verification of libraries calling application functions, and binary packing (encryption of code segments) of the pre-compiled libraries.
2.1.5	**The contactless kernel of the CPoC application must be protected by tamper-resistant methods to guarantee its integrity.**	The contactless kernel, which is the software that processes the contactless transactions, must be strictly protected, including any configuration files, optional settings, and payment brands public keys. zShield allows you to split the application code in logical modules and apply different levels of security and sets of protection features to each module. This means that you can optimize security and performance, for example, applying no or low level protection to basic UI rendering routines and increasing the security for core components, such as the contactless kernel. zShield also adds extra layers of protection against code-lifting.
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.**	When the contactless kernel code is protected with zShield and zKeyBox, any attempt to interfere with the contactless kernel operation from outside will be blocked. For example, integrity checking mechanisms detect and stop a hacker from changing a single bit of application code. Similarly, attempts to impersonate the user and hijack communications are thwarted as the network module is protected against reverse engineering and cryptographic keys are never exposed.
2.1.7	**The CPoC application must implement methods for detecting a number of listed threats, including if the device has been rooted and jailbroken, and report them to a back-end monitoring system.**	zShield provides the ability to execute code in response to certain types of threats, such as when a rooted or jailbroken device is detected. Developers can leverage this callback mechanism to implement a threat reporting system in the CPoC application that reports any threats or attempts to attack the application.
2.1.8	**A CPoC application that fails tamper checks must be prohibited from accepting account data.**	zShield includes real-time intrusion detection and response. When a tampering attempt is identified, it can automatically shut down the application to prevent it from accepting account data. Alternatively, a custom callback function can be defined to prevent the acceptance of account data without fully disabling the app.

CPoC Standard section 2.2:
White-box cryptography

Of all the CPoC Standard requirements, the white-box cryptography section seems to generate the most questions. White-box cryptography provides developers software-based cryptographic security in open and untrusted execution environments, where secure hardware components are not available or acceptable.

Traditionally, protection of cryptographic operations is achieved by special-purpose tamper-resistant hardware components, such as secure enclaves (SE) and trusted execution environments (TEE). Although these types of internal hardware modules are increasingly common in COTS devices, they are not universal. Moreover, on some devices, their design or features do not allow a sufficiently secure or straightforward implementation for a CPoC application.

The PCI guidelines state that white-box cryptography can be used as a substitute for device-dependent cryptography where the underlying hardware support is lacking. In fact, software-only implementations that rely solely on white-box cryptography can qualify for a PCI Letter of Approval. This means that by using white-box technology alone, CPoC developers can skip customizing their application to support hardware-backed cryptography for each type of device.

What is white-box cryptography?

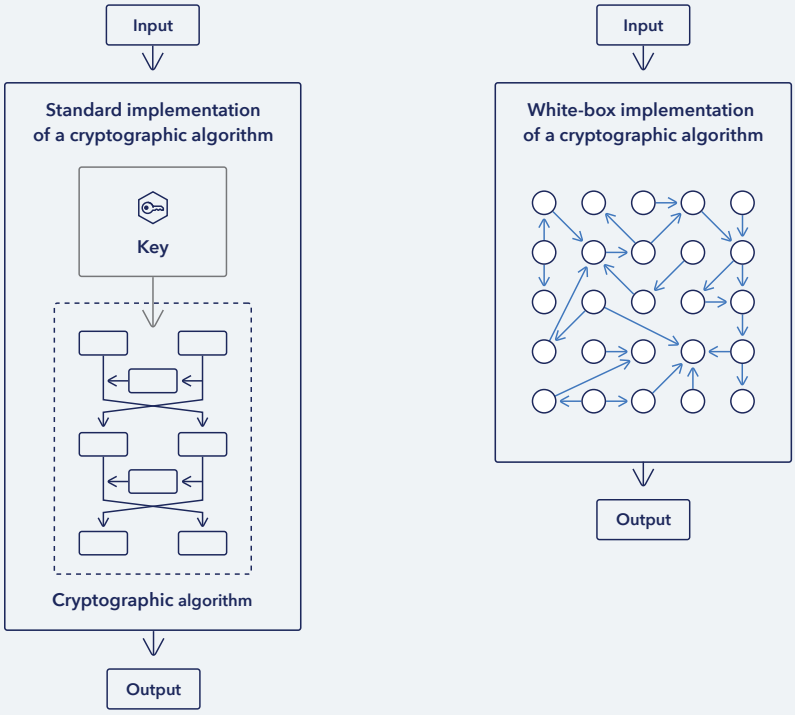
The white-box model assumes that the internal workings of an algorithm are going to be visible and modifiable. This means that a software-only implementation of cryptographic algorithms must be sufficiently secure against attack such that the secret cryptographic keys are protected and hidden at all times, at rest and at runtime. The CPoC Standard specifically defines white-box cryptography as "a method used to obfuscate a (mostly symmetric) cryptographic algorithm and key with the goal of making determination of the key value computationally complex."

For developers and vendors of contactless or mPOS based payment systems on COTS, the number and stringency of the security requirements expected by the CPoC Standard can quickly become overwhelming. For most, in-house development of a sufficiently secure white-box cryptography module is impractical if not impossible, especially those aiming to get to market swiftly. White-box cryptography is complex and requires significant cryptographic expertise and years of research and development to get right.

zKeyBox is an industry-leading white-box cryptography solution, providing the ability to use AES, RSA, ECC, 3DES, CMAC, RetailMAC (ISO/IEC 9797-1), and other ciphers with the highest level of security. The strength of Zimperium products is continuously tested by top third-party penetration testing and security labs.

Conceptual overview of white-box cryptography

White-box cryptography provides the equivalent functionality of standard algorithms without revealing the intermediate values. In a "standard" cryptographic implementation, the keys and execution logic are clearly distinguishable and easy to tamper with. In a white box implementation, the internal data and execution flow are obscured and inseparable—keys cannot be easily extracted and any attempted code modifications can break the entire executable.



Section 2.2 Requirements

White-box cryptography offers developers and vendors several advantages over hardware-based cryptography, including ease of deployment and upgrade, portability, and cross-platform support. However, the strength of its security depends on following strict implementation protocols. PCI has defined eight security requirements that applications using white-box cryptography must meet.

Req.	Description	Explanation and compliance
2.2.1	**Cryptographic methods protected primarily by software-based methods must be protected against analysis and abuse.**	The CPoC Standard details specific expectations regarding secure key generation, how the software-based protection should be implemented, and what attacks it must be ready to withstand. zKeyBox is designed and built from the ground up to be robust against analysis, tampering, side-channel attacks and reverse engineering. Its unique implementation encodes not only the cryptographic keys, but also the processing logic, which makes it extremely difficult for an attacker to determine what the library does.
2.2.2	**The robustness of the software-based protection mechanisms must be evaluated, at least annually, against current attack scenarios and vectors.**	This requires the ongoing involvement of security labs, researchers, and payment domain security experts, to test the solution and produce analysis documentation of potential threats including their execution difficulty, likelihood, mitigation techniques, and potential impact. zKeyBox and our own team of researchers constantly monitor and analyze its implementation in the context of existing and emerging attack vectors. Documentation required by the PCI or other entities is available.
2.2.3	**The cryptographic material used in software-based protection mechanisms, such as white-box keys, entropy seeds and nonces, must be changed periodically to prevent cryptographic key compromise.**	This provision requires the application developer to change the white-box implementation and cryptographic keys at least once a month to minimize the risk of breach. The simplest way to achieve this is to release an updated version of the application every month. zKeyBox's incorporation of strong cryptographic diversification ensures keys are never reused and internal data is encoded differently per delivery. SKB also makes it easy to update your cryptographic implementation monthly without changing the API.
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.**	Once a different edition of zKeyBox is deployed, existing key material is upgraded or reimported via the library's API. Retired keys can then be deleted. Moreover, even if old keys were leaked, they would no longer work with the updated SKB instances.
2.2.5	**The cryptographic material used in software-based protection mechanisms must not be used directly for account data or attestation data encryption.**	Each key used in white-box cryptography must be used for one purpose only, and they must not be 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. While much of this requirement depends on sound cryptographic architecture, zKeyBox ensures that cryptographic material used to protect keys cannot be used to encrypt data. It also has an internal security feature that prevents one type of key to be used in other contexts. For example, it does not allow a key used for decryption to be used for unwrapping of other keys.

Section 2.2 Requirements (continued)

Req.	Description	Explanation and compliance
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.**	To achieve this, each version of the application for each OS store type (such as Apple's App Store® and Google Play™) and geographic region must be unique. This involves creating and deploying a key provisioning system that allows effective and secure generation and distribution of keys for each unique global market. zKeyBox's diversification feature lets you link a different edition of the library, with a different binary footprint, into individual editions of applications for separate geographic markets. Moreover, it includes another powerful capability, device binding, which allows you to encode internal secrets differently depending on the device or operating system the application runs on.
2.27	**Cryptographic algorithms and keys used in software-based protection mechanisms must meet the security requirements of acceptable cryptography.**	Industry-recognized standard cryptographic algorithms, key lengths, and other implementation methodologies should form the basis for any security services used in the solution. These criteria are described in section 1.3 of the CPoC Standard. zKeyBox supports a wide variety of industry-accepted cryptographic algorithms and key lengths, including 128-bit, 192-bit, and 256-bit AES, 2048-bit RSA, ECC with P-256, P-384, and P-521 curves, SHA-256 and SHA-512, 3DES, DH, ECDH, ECDSA, HMAC, CMAC, RetailMAC (ISO/IEC 9797-1), and more. zKeyBox even provides a white-box implementation of the TLS protocol commonly used in network communication of payment systems.
2.2.8	**Cryptographic keys used in software-based protection mechanisms must meet the established key management requirements.**	Cryptographic keys must be managed securely using recognized industry practices throughout their cryptographic lifecycle, including generation, distribution, storage, replacement, revocation, deletion, and accountability. These criteria are described in section 1.4 of the CPoC Standard. zKeyBox ensures that keys are always encrypted and encapsulated into secure data objects whose plaintext content cannot be read, not even by the application itself. This is true at all stages of the key lifecycle, including generation, loading, processing, and exporting to storage.

Additional CPoC Compliance Support

Zimperium solutions aid in compliance with multiple other sections of the CPoC Standard.

Req.	Description	Explanation and compliance
Secure application	**CPoC Standard section 2.5 defines a secure application to be one that it is "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."**	While the majority of the provision deals with secure coding practices and documenting security measures, Intertrust aids in compliance with several of the individual requirements. Zimperium solutions harden the application against tampering, side-channel attacks, fault injection, and reverse-engineering, providing developer documentation on methods used. zKeyBox also ensures that secret or private cryptographic keys are never exposed as cleartext in the COTS memory.
Secure provisioning	**CPoC Standard section 2.6 concerns making sure that merchants only install an authentic version of the application.**	Zimperium solutions support compliance with this specification through several mechanisms. • Application integrity checking ensures that no modifications have been made to the mPOS application since it was built by the development team. If tampering is detected, the application will not run on the merchant's device. • zShield includes safeguards that protect the integrity of the Android APK package. It also prevents unauthorized re-signing and distribution of apps in the Mach-O file format used by iOS. • zKeyBox supports multiple methods, such as key unwrapping and computing a shared secret, that facilitate secure provisioning of cryptographic keys on first execution.
Account data encryption	**Section 2.9 of the CPoC Standard is concerned with securing account data.**	While much of it focuses on best practices for encrypting data and managing encryption keys, zKeyBox has built-in safeguards that can aid in compliance. For example, it employs industry-recognized key ciphers in a secure manner and prevents key discovery and re-use.
Contactless kernel security	**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.**	zKeyBox makes sure that keys and intermediate values are never exposed in the clear.

Zimperium helps you comply with the CPoC Standard

Zimperium provides advanced application protection and white-box cryptography solutions that protect apps with sensitive information, thwart attacks, and help you comply with industry regulations, including the PCI CPoC Standard.

Code Protection

zShield embeds advanced security defenses into your applications, enabling them to run securely in zero-trust environments. It uses multiple methods to prevent tampering and reverse engineering including advanced code obfuscation, jailbreak/rooting detection, and real-time intrusion detection and response.

Secure Key Box

zKeyBox is a state-of-the-art white-box cryptography library that keeps your cryptographic keys protected within the app code, even during runtime, and even if a device is rooted or jailbroken. Extremely easy to integrate and use, it provides an extensive set of high-level classes and methods for operating with the most popular cryptographic algorithms across multiple platforms in the mobile, desktop, cloud, and embedded space.

Partner with Zimperium

Zimperium helps the world's leading fintech vendors build and deliver the most secure and innovative financial applications. Our team has worked hand-in-hand with CPoC developers and accredited testing labs and can help guide you through the process.

KEY BENEFITS

- Protect your applications against reverse-engineering and tampering
- Secure encryption keys at all times even on compromised devices
- Comply with CPoC white-box requirement without specialized development work
- Meet CPoC penetration testing requirements
- Accelerate your time to market

To learn more about Zimperium's Mobile Application Protection Suite (MAPS) that provides protection across the entire application lifecycle, visit us here:

https://www.zimperium.com/mobile-app-protection

About Zimperium

Zimperium is a global leader in mobile device and app security, offering real-time, on-device protection against both known and unknown threats on Android, iOS and Chromebook endpoints. The company was founded under the premise that the then current state of mobile security was insufficient to solve the growing mobile security problem. At the time, most mobile security was a port from traditional endpoint security technologies.

Zimperium recognized mobile devices had unique characteristics needing a completely new approach. The team set to work to reimagine how to protect mobile devices and developed the award winning, patented z9 machine learning-based engine. z9 protects mobile devices from device, network, phishing and application attacks. And as first envisioned, z9 has detected 100% of zero-day mobile exploits in the wild without requiring an update or suffering from the delays and limitations of cloud-based detection —something no other mobile security provider can claim.

Sources

1 Payment Card Industry (PCI) Contactless Payments on COTS (CPoC™) Security and Test Requirements, Version 1.0, PCI Security Standards Council, December 2019

ZIMPERIUM®

Learn more at: zimperium.com
Contact us at: 844.601.6760 | info@zimperium.com
Zimperium, Inc
4055 Valley View, Dallas, TX 75244