

# YOUR CRYPTOGRAPHIC INVENTORY IS A JOIN, NOT ANOTHER SPREADSHEET

TCP 443 is a service fact. TLS enabled is a configuration claim. Cryptographic viability is a maintained decision. Stop letting one impersonate the others.

## PRACTITIONER POSITION

PCI DSS Requirements 1.2.5, 2.2.4, and 12.3.3 describe related but distinct propositions. The first governs permitted services, protocols, and ports with an identified business need. The second limits enabled services, protocols, daemons, and functions to those that are necessary. The third requires an inventory of cryptographic cipher suites and protocols used to meet PCI DSS requirements, their purpose and location, and at least annual review of their continuing viability. My practitioner position is that three isolated worksheets are the wrong operating shape. Build a governed join between the component, service, configuration, cryptographic implementation, requirement use, observation, viability source, owner, and change plan. The join may be rendered as a spreadsheet, graph, or database. Its value comes from the relationships it can prove and reopen-not from the number of columns it contains.

## THE FIELD PATTERN

## WHAT THIS LOOKS LIKE IN PRACTICE

A payment API is listed as HTTPS over TCP 443 in the network-control record. A hardening sheet says the web service is required. A cryptography sheet says TLS 1.2 and TLS 1.3 are used. Each row is green. None identifies whether TLS terminates at a CDN, WAF, load balancer, API gateway, service mesh, application process, or several of them. The sheets do not agree on the component identifier, configuration source, permitted cipher policy, observed negotiation, provider responsibility, viability authority, or affected change plan. The organization has three inventories and no reliable answer to a simple question: if one protocol, suite, library, certificate path, or managed-edge policy becomes unacceptable, which services, components, claims, tests, and owners must change?

## SIGNALS DETECTED

## WHAT SHOULD MAKE YOU STOP

- The service inventory, secure-configuration record, and cryptographic inventory use different names for the same component and cannot join on a durable identifier.
- TCP 443, HTTPS, TLS 1.2, a managed-provider policy name, and a cipher suite are treated as interchangeable facts.
- The inventory says TLS is used but does not identify each termination point, configuration source, or material path variant.

- A provider default is treated as the organization's approved configuration without current configuration evidence.
- The record contains an algorithm or protocol but not the PCI DSS purpose or requirement-dependent use it supports.
- The annual review date exists, but the viability source, reviewer, decision, affected population, and response strategy do not.
- The certificate inventory is presented as the cryptographic inventory even though it does not describe permitted protocols, suites, algorithms, purposes, or configuration control.
- One cryptographic row points to one port even when the implementation supports encryption at rest, application protection, authentication, hashing, signing, or key exchange without a meaningful port relationship.
- Observed handshakes are treated as the entire permitted configuration, or configuration policy is treated as proof of what every endpoint actually negotiates.
- A deprecation notice is assigned to the security team, but no dependency radius identifies services, providers, applications, evidence, or claims that must be reopened.

#### WHY IT MATTERS

## THE COST OF LEAVING IT UNEXPLAINED

Start with the authoritative floor. Requirement 12.3.3 addresses cipher suites and protocols used to meet PCI DSS requirements. The inventory identifies their purpose and where they are used, and the documented review occurs at least annually. That review includes active monitoring for changes in cryptographic viability and a documented strategy for responding to anticipated changes. Do not silently rewrite this as a requirement for one universal enterprise cryptography database.

Use a broader operating model deliberately and label it as practitioner method. Cryptography supporting storage, transmission, authentication, password protection, key exchange, digital signatures, hashing, token or application protection, and other PCI DSS-dependent uses can be governed together. The wider model improves change impact and evidence reuse; it does not enlarge the normative text by editorial enthusiasm.

Give every object a stable identity. Component, environment, service record, configuration baseline, termination point, provider service, cryptographic implementation, observation, viability decision, and change plan need identifiers that survive display-name changes. A row called Payment API is not a key.

Separate permission from operation. Requirement 1.2.5 establishes which services, protocols, and ports are permitted and why. Requirement 2.2.4 establishes which services, protocols, daemons, and functions are necessary and enabled. Requirements 1.2.6 and 2.2.5 address insecure items that must remain and their additional security features. None of those statements, by itself, establishes the complete 12.3.3 cryptographic inventory or viability review.

Model termination, not merely destination. For each material flow, identify where cryptography begins and ends: browser, device, edge, CDN, WAF, load balancer, API gateway, service mesh, workload, database driver, managed database, queue, storage service, or provider boundary. One friendly hostname can conceal several independently managed cryptographic control points.

Record configured and observed states separately. Preserve the approved policy or configuration source, the expected protocols and suites, the observed negotiation or effective configuration, the observation method, the population, and the time. A single successful TLS handshake proves one negotiated path at one moment; it does not enumerate everything the service permits.

Make provider responsibility explicit. A managed edge or cloud service may implement and maintain part of the cryptographic stack. The entity still needs to identify the exact service, configuration choice, shared-responsibility boundary, evidence source, monitoring path, and customer-controlled changes that support its claim.

Treat viability as a sourced decision. Record the source or monitoring feed, the affected protocol or suite, the date observed, the evaluator, the disposition, the rationale, the response trigger, and the next review. PCI SSC FAQ 1491 emphasizes appropriate cipher suites and key exchange, strong cryptography and authentication, and active monitoring of current industry best practices; a version label alone is not the whole decision.

Build many-to-many relationships. One service can terminate through several cryptographic implementations. One implementation or provider policy can protect many services. One deprecation can affect many components and claims. A forced one-row-to-one-port design hides exactly the dependency radius the inventory should expose.

Join evidence without collapsing its authority. Approved configuration, discovered endpoint behavior, provider documentation, scanner output, source code, certificate metadata, owner assertion, and assessor validation are different objects. They may support the same claim, but they do not become equivalent because a dashboard renders them in one color.

Turn the annual review into a change-impact drill. Select one weakening protocol, cipher family, library, provider policy, or cryptographic dependency. Trace every dependent service, component, data flow, owner, test, claim, and remediation record. If the inventory cannot produce a bounded response population, the review has inspected a list without testing its usefulness.

Measure reconciliation gaps, not spreadsheet completeness. Useful signals include services with no linked crypto record, crypto uses with no component or configuration source, implementations with no viability source, review dates with no decision, managed services with no responsibility evidence, and change plans with no dependent population. A full row can still describe an unjoined environment.

## QUESTIONS

# QUESTIONS THAT EXPOSE THE WACK

01 Which stable identifiers join the network/service record, secure-configuration record, cryptographic implementation, evidence, owner, and change plan?

02 Where does cryptography actually terminate for every material production path and variant?

03 What is permitted, what is enabled, what is configured, and what has been observed-and where do those states disagree?

- 04 Which cipher suites and protocols are used to meet which PCI DSS requirements, for what purpose, and where?
- 05 Which broader cryptographic uses are included by practitioner method, and are they clearly distinguished from the 12.3.3 normative floor?
- 06 For managed services, which facts come from the provider and which customer choices and observations remain the entity's responsibility?
- 07 Which current sources inform viability, who made the decision, and what response strategy applies if the decision changes?
- 08 Can one deprecation notice produce the complete affected service, component, provider, evidence, test, claim, owner, and change population?
- 09 Which services have no linked cryptographic record, and which cryptographic records have no component, configuration source, observation, or owner?
- 10 Can a reviewer explain the environment from the joined records without asking which of three spreadsheets is supposed to be authoritative?

#### THE CLUSTER OF WACK

#### FIND / MAP / EXPLAIN / REDUCE / PROVE

# ESCAPE THE WACK

## 01 / FIND

Discover the component, service, termination point, provider, configuration source, cryptographic implementation, observed state, and current viability source as separate objects.

## 02 / MAP

Join permitted and enabled service facts to cryptographic purpose, location, PCI DSS use, owner, evidence, monitoring, review, and change dependencies through stable identifiers.

## 03 / EXPLAIN

State what each source proves, keep normative PCI DSS language distinct from the broader practitioner operating model, and reconcile configured, observed, documented, asserted, and validated states.

## 04 / REDUCE

Retire duplicate rows, ambiguous names, orphan records, unsupported provider defaults, dead services, obsolete algorithms, and annual-review theater that cannot identify an impact radius.

## 05 / PROVE

Test the joins with current configuration and observations, run a deprecation impact drill, record the bounded viability decision, and reopen every affected claim when a material dependency changes.

THE FIELD LAW

## THE LINE WORTH REMEMBERING

**A CRYPTOGRAPHIC INVENTORY IS USEFUL WHEN A WEAKENING PRIMITIVE CAN FIND EVERY DEPENDENT SERVICE BEFORE THE ASSESSOR DOES.**

Download the Cryptographic Inventory and Dependency Register. Join one payment service from permitted port through enabled function, termination point, configured and observed cryptography, PCI DSS purpose, viability source, owner, and change plan-then run one deprecation impact drill.

TECHNICAL GROUNDING

## START WITH THE SOURCE MATERIAL

These references ground the practitioner interpretation. They do not turn the metaphor or method into official PCI DSS terminology.

- WACK-SOURCE-0001 / PCI DSS v4.0.1 / PCI Security Standards Council / [Open source](#)

- WACK-SOURCE-0087 / FAQ 1491: Does PCI DSS define which versions of TLS must be used? / PCI Security Standards Council / [Open source](#)

**AUTHORITY CLASSIFICATION / PCI\_SSC\_SOURCE / ASSESSOR\_JUDGMENT / PRACTITIONER\_POSITION / PRACTITIONER\_METHOD / AI\_ASSISTED**

Practitioner reviewed by Scott Norton on 2026-09-12 against canonical digest

e17a6a19f72deac8140fc32748d4a5d299fc1ba34318eebc54f46a6d48d9e701. Educational content, not a compliance determination.

Independent practitioner resource; not affiliated with or endorsed by PCI SSC. Authorship and authority classification do not determine legal intellectual-property ownership.