SvedbergOpen
DISSEMINATIONOFKNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/



Research Paper Open Access

Secrets Lifecycle Management Without Service Interruption: Automated Rotation Across Distributed Services, Credential Propagation Delay, and Failure Recovery in Hybrid Cloud Environments

Sneha Palavi

Independent Researcher, India

Abstract
Organisations know that long-lived credentials are dangerous and rotate them anyway less often than their own policies require. The reason is not ignorance of the risk but a rational fear of the remedy: rotation can cause outages, and an outage is immediate and attributable while a stale credential is a latent risk that has not yet been exploited. The mechanism of that outage is specific. In a distributed environment consuming services do not retrieve or activate a newly issued credential simultaneously, so if the previous credential is disabled before every consumer has adopted the new one, the consumers that lag simply fail to authenticate. This paper sets out a rotation lifecycle designed so that propagation delay is a normal operating condition rather than a failure case. The central mechanism is a controlled overlap window in which both the previous and the new credential remain valid, with the previous credential disabled only after positive verification that consumers have adopted the new one, so the overlap functions as a recovery path rather than merely as a grace period. The paper treats hybrid complexity as one coordinated lifecycle across differing platforms rather than as independent secret updates, and it is explicit about the manual remainder, which is where the greatest operational risk sits. It reports one incident in which a dependent consumer required additional coordination and was caught inside the window. It reports no rotation success rate, no mean propagation time, and no secrets count, because those measurements were not maintained in a form that could be substantiated.

Keywords: secrets management, credential rotation, key lifecycle, propagation delay, overlap window, service availability, hybrid cloud, identity and access management, access keys, certificate rotation, zero downtime, credential revocation, audit.

I. Introduction

1.1 A risk everyone accepts knowingly

Credential rotation is not a contested practice. Every security framework requires it, every audit asks about it, and every engineering organisation agrees with it in principle. It nonetheless gets deferred, and the deferral is usually deliberate rather than negligent.

The reason is an asymmetry in how the two risks present. Rotation can cause service outages, authentication failures, and operational complexity, particularly where credentials are shared across distributed applications, delivery pipelines, databases, and hybrid environments. That risk is immediate, visible, and attributable to whoever performed the rotation. The risk of not rotating is that a forgotten or long-lived key is exposed through source code, logs, configuration files, a former employee, or a compromised system, and if exploited allows an attacker to retain persistent access, impersonate services, reach sensitive data, or move laterally. That risk is latent, diffuse, and attributable to nobody in particular until it materialises.

Teams facing that asymmetry postpone rotation to protect availability. The engineering response is therefore not to argue that they have the risk calculus wrong. It is to remove the availability risk, so that the calculus changes.

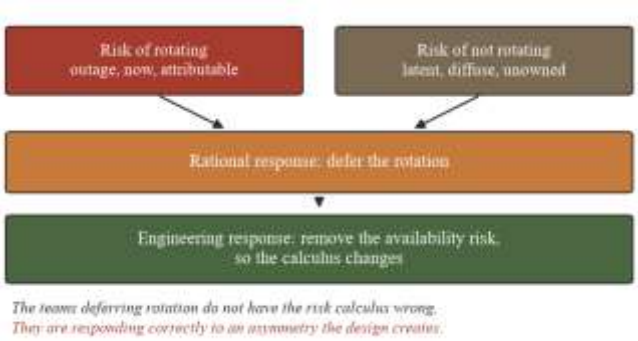


Figure 1. The asymmetry that produces deferral. Both risks are real; only one is immediate and attributable.

1.2 The mechanism that causes the outage

The specific failure is propagation delay. A new credential is issued, but the services consuming it do not all retrieve and activate it at the same moment. Deployment cadence differs, caching differs, restart requirements differ, and some consumers only pick up a credential on their next scheduled run. If the previous credential is disabled before every consumer has adopted the new one, the laggards fail to authenticate.

Every design decision in this paper follows from treating that delay as a normal property of a distributed system rather than as a defect to be engineered away.

1.3 Scope of the work

Automated rotation work began around 2018 to 2020 in a DevOps and cloud engineering capacity supporting distributed applications across a public cloud and hybrid environments, using a managed secrets service for application and service credentials. From 2023 to the present the work has covered automated credential rotation for identity and access management service users, with credentials stored and managed in a centralised secrets platform.

The scope covers approximately forty to sixty secrets and service credentials consumed by fifteen to twenty application services, including containerised workloads, interfaces, databases, delivery pipelines, automation processes, and external integration services. Credential types include database credentials, interface keys, application secrets, certificates, and service-user access credentials. The role covered designing, implementing, and maintaining the credential lifecycle architecture, including automating the lifecycle so that new credentials are generated, securely distributed to consuming applications and automation, validated, and then used to retire older credentials.

1.4 Contribution and scope

1. A rotation lifecycle in which propagation delay is designed for rather than assumed away, with the overlap window as the load-bearing mechanism.
2. An explicit treatment of the manual remainder, which is the part of a credential estate that automation does not reach and where risk concentrates.
3. A reconciliation of hybrid platform differences into one coordinated lifecycle rather than independent per-platform updates.
4. An account of what a missed rotation actually looks like from a consumer's perspective, and how recovery works.

What this paper does not do. It reports no rotation success rate, no mean propagation time, and no count of secrets under management, because those were not maintained as substantiated metrics. Section 14 states the evidence position, and Section 15 records one area where the underlying material was insufficient to support a claim at all.

2. Background and Related Work

Credential lifecycle as a security control. Rotation appears in security frameworks as a periodic control with a stated interval. The frameworks specify the interval and are generally silent on the mechanism, which is where the engineering difficulty lies. A control that says "rotate every ninety days" does not say how to do so without interrupting the consumers, and that omission is precisely why the control gets deferred.

Short-lived credentials as the alternative. An established position holds that the correct answer to credential rotation is to eliminate long-lived credentials entirely, replacing them with short-lived tokens issued on demand against a workload identity. Where this is achievable it is strictly better than rotation, because a credential that

expires in minutes needs no rotation lifecycle at all. It is not universally achievable, and the reason matters for this paper: legacy applications, external vendor integrations, and systems that cannot dynamically retrieve a credential cannot participate. Section 4's manual remainder is largely the population for which the short-lived approach is unavailable, which means rotation remains necessary precisely where it is hardest.

Why the consumer side is the hard side. The asymmetry is worth stating directly. Generating a credential is an operation with a definite completion: it either exists or it does not. Adopting a credential is a distributed state change with no single completion point, because each consumer adopts on its own schedule and the set of consumers is itself frequently uncertain. Every mature tool in this space solves the first problem well. The second is not a tooling gap so much as a consequence of the consumers being systems the secrets platform does not control and often does not know about.

Secret storage and distribution. Centralised secrets platforms provide storage, access control, and audit for credentials, and managed cloud services provide equivalents with native rotation for supported resource types. These solve storage and, for some credential classes, generation. The gap they leave is consumer adoption: a platform can hold the new credential and record who read it, but whether a consuming service has actually begun using it is a property of the consumer, not of the store.

That gap is what this paper addresses. Storage is solved, generation is largely solved for supported types, and the unsolved part is knowing when it is safe to disable what the new credential replaced.

3. What Is Rotated Automatically and What Is Not

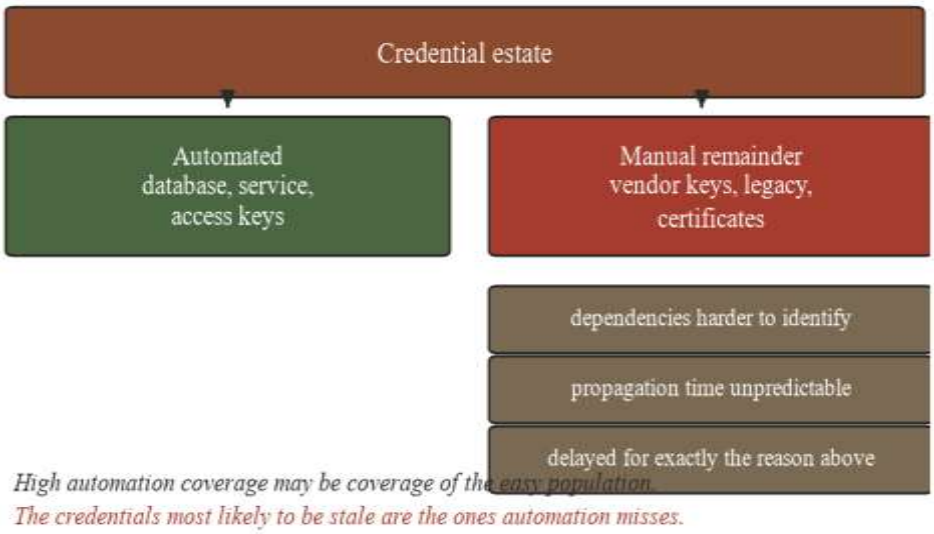


Figure 2. The estate splits into an automated population and a manual remainder, and risk concentrates in the second.

Table 1. Credential classes and rotation status.

Credential class	Rotation	Interval	Why
Database credentials	Automated	30 to 90 days by risk level	Consumer retrieval is programmatic and the platform supports it natively
Application and service credentials in the managed store	Automated	30 to 90 days by risk level	Same
Service-user access keys	Automated	Approximately 90 days	Lifecycle scripted end to end with validation

Credential class	Rotation	Interval	Why
Third-party interface keys	Manual or semi-automated	Varies	Requires coordination with an external vendor
Vendor credentials	Manual or semi-automated	Varies	Outside the organisation's control plane
Legacy application secrets	Manual or semi-automated	Varies	Application cannot retrieve a credential dynamically
Mutual TLS certificates	Manual or semi-automated	Varies	Requires certificate exchange, sometimes a deployment
Credentials for systems that cannot fetch dynamically	Manual	Varies	Configuration change and often a maintenance window

The manual remainder is the interesting part, and it is where risk concentrates. Three properties make it worse than the automated population rather than merely less convenient.

Dependencies are harder to identify. An automated rotation exercises the dependency graph every cycle, so a consumer nobody remembered is discovered routinely. A credential rotated once a year, by hand, has a dependency list that exists in somebody's memory.

Propagation time is unpredictable. Where rotation requires a vendor's action, a deployment, or a maintenance window, the interval between issuing a credential and every consumer adopting it is not bounded by anything the organisation controls.

And teams delay it, for exactly the reason in Section 1.1, with the delay compounding because each deferral makes the next rotation more dangerous as the dependency picture ages further.

The practical consequence is that a rotation programme reporting high automation coverage may be reporting coverage of the easy population. The credentials most likely to be stale are the ones automation does not reach.

4. The Rotation Lifecycle

The lifecycle is a fixed sequence, and the ordering carries the safety property.

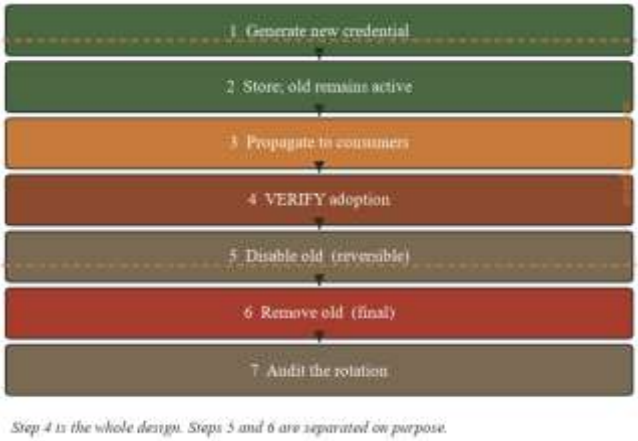


Figure 3. The rotation lifecycle. Step 4 is the verification gate; steps 5 and 6 are separated because only the second is irreversible.

Table 2. The rotation sequence.

Step	Action	Failure behaviour
1	Generate the new credential	Nothing has changed for any consumer

Step	Action	Failure behaviour
2	Store it in the central platform while the existing credential remains active	Both credentials valid; no consumer affected
3	Propagate to consuming applications, automation, and pipelines	Consumers adopt at their own pace
4	Verify adoption by required consumers	The verification gate; nothing is disabled until it passes
5	Disable the previous credential	Reversible: it can be re-enabled
6	Remove the previous credential	The point of no return
7	Audit the rotation	Record retained

Step 4 is the whole design. Every other step is mechanical. The verification gate is what converts rotation from an action that might break things into an action whose safety has been established before the irreversible part.
The overlap window spans steps 2 through 5, during which both the old and the new credential are valid. Its purpose is frequently misdescribed as giving consumers time, which is only half of it. Its more important property is that it is a recovery path: if propagation fails, keeping the previous credential temporarily active means the failure has no service consequence, so the response is to correct the propagation rather than to restore service under pressure.

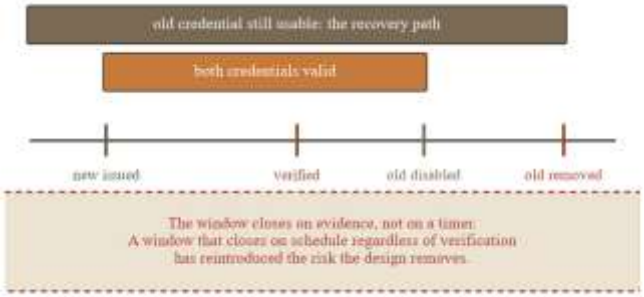


Figure 4. The overlap window as a recovery path rather than a grace period.

The window closes on evidence, not on a timer. The previous credential is disabled once the new one is verified as reaching the required consumers and working. A window that closes on schedule regardless of verification has reintroduced exactly the risk the design removes.
Disable before remove. Steps 5 and 6 are separated deliberately. Disabling is reversible and removal is not, so a consumer discovered after disabling can be recovered by re-enabling, while the same discovery after removal requires a new rotation under incident conditions.

5. When a Consumer Misses a Rotation

A service that misses a rotation continues using the old credential after the new one has been issued. Nothing is visibly wrong until the old credential is disabled, which is the property that makes the verification gate necessary rather than merely prudent.
Detection comes from authentication and authorisation failures, application and infrastructure logs, monitoring alerts, and failed automation or pipeline jobs. For service users managed through the central platform, the failure typically presents as cloud interface authentication errors once the previous key is disabled.
What the user sees depends on which service holds the credential: a failed request, an unavailable application function, a delayed background process, or an integration failure. None of these presents as a credential problem, which is why the diagnostic path usually begins somewhere else.

Recovery involves identifying the affected consumer, retrieving or propagating the current credential from the platform, restarting or redeploying the service where necessary, and validating successful authentication.
The important case is detection inside the window. If the gap is found while the previous credential is still active, it can remain active while propagation is corrected, and there is no service interruption at all. This is the difference between an incident and a task, and it is entirely determined by whether verification happens before disabling or after.

6. Why Detection Is Structurally Late

A missed rotation is not detected when it happens. It is detected when its consequence arrives, and the interval between those two moments is where the operational risk sits.
Table 3 sets out what is observable at each stage. The sequence is invariant. A consumer fails to adopt the new credential, and nothing is observable, because the old credential still works. That state persists for the length of the overlap window, during which the system is in a latent failure the monitoring cannot see, since every authentication is succeeding. The failure becomes observable only when the old credential is disabled, which converts a silent condition into a live one at a moment chosen by the rotation process rather than by the fault.

Table 3. What is observable at each stage.

Stage	System state	Observable to monitoring
New credential issued	Both valid; consumer has not adopted	Nothing. All authentication succeeds
Propagation in progress	Consumer still on the old credential	Nothing, unless adoption is explicitly checked
Verification runs	The gap exists and is detectable	**Only here, and only if verification looks for it**
Old credential disabled	Consumer now fails	Authentication errors, failed jobs, integration failures
Recovery	Consumer repointed and restarted	Restored

The table makes the design argument better than prose does. There is exactly one row where the problem is both present and findable before it has a consequence, and reaching that row requires actively confirming adoption rather than watching for errors. Monitoring designed around error rates cannot detect this class of fault, because until the final row there are no errors to detect.
This also explains why the incident in Section 10 was uneventful. The dependency was identified during the window rather than after it, which is the difference between the third row and the fourth, and nothing about the fault itself changed between those two states.
A secondary consequence concerns what recovery costs. Recovery from the fourth row is straightforward but conducted under live impact, with the associated pressure to restore service before understanding the cause. Recovery from the third row is a scheduling conversation. Same fault, same fix, entirely different circumstances.

7. Hybrid Complexity as One Lifecycle

Hybrid environments complicate rotation because secrets are stored and consumed differently across on-premises systems, cloud platforms, legacy applications, and delivery environments. Each platform has different interfaces, authentication methods, refresh behaviour, and propagation times, so a credential cannot always be rotated everywhere simultaneously.
The response is to treat rotation as one coordinated lifecycle rather than as independent secret updates on each platform. A central platform acts as the controlled source of credentials where possible, while automation integrates with the different consumers and platforms underneath.

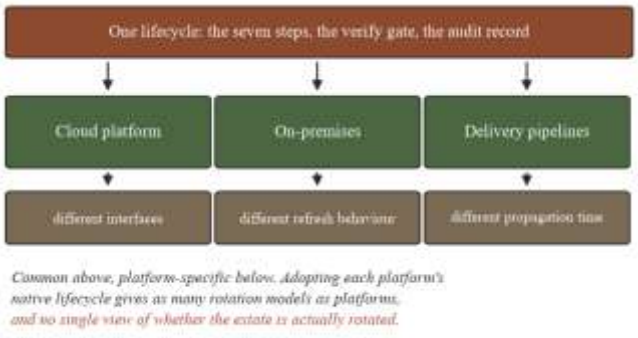


Figure 5. One lifecycle across heterogeneous platforms, with platform differences absorbed underneath rather than replicated as separate rotation models.

Table 4 separates what is common across the estate from what is not.

Table 4. What is common across platforms and what stays platform-specific.

Layer	Common across the estate	Platform-specific
Lifecycle sequence	The seven steps in Table 2	Nothing
Credential source	The central platform	Fallback where a platform cannot integrate
Distribution mechanism	Not common	Interface, refresh behaviour, restart requirement
Propagation time	Not common	Varies widely by consumer and deployment process
Verification method	The requirement to verify	How adoption is confirmed on each platform
Recovery model	Old credential remains valid until verification passes	Re-propagation mechanics
Audit record	One record per rotation	Source of the evidence

This is the design's main structural claim: the lifecycle, the validation requirement, the monitoring strategy, and the recovery model are common, while platform-specific differences remain underneath. A rotation programme that instead adopts each platform's native lifecycle ends up with as many rotation models as platforms, and no single view of whether the estate is actually rotated.

8. What Makes a Consumer Hard

Not all consumers are equally difficult to rotate, and the differences are structural rather than incidental. Knowing which class a consumer belongs to predicts its propagation behaviour better than any general estimate.

Table 5. Consumer classes and their propagation behaviour.

Consumer class	How it obtains the credential	Propagation behaviour
Containerised service reading at startup	Fetches on boot from the platform	Adopts on next deployment or restart; predictable but not immediate
Service reading on each use	Fetches per operation	Adopts almost immediately; the easiest class

Consumer class	How it obtains the credential	Propagation behaviour
Scheduled automation	Fetches at run time	Adopts on next scheduled execution; may be days
Delivery pipeline	Injected at build or deploy	Adopts on next pipeline run; depends on commit activity
Application with cached credential	Reads once, caches in memory	Requires restart; adoption is invisible without one
Configuration-file consumer	Credential written to a file	Requires a configuration change and often a deployment
External vendor integration	Credential supplied to a third party	Requires coordination outside the organisation; unbounded

Three things follow from the table.

The fetch pattern determines everything. A consumer reading per operation is trivially rotatable; one caching at startup is rotatable on a deployment cadence; one holding a credential in a configuration file requires a change someone must make. The rotation difficulty of an estate is therefore largely determined by architectural decisions taken long before rotation was considered, and improving rotatability is mostly a matter of changing how consumers obtain credentials rather than improving the rotation process.

The last two rows are the manual remainder from Section 3, arriving from a different direction. Section 3 classified by credential type; this table classifies by consumer behaviour. They largely agree, which is a useful corroboration: the credentials that resist automation and the consumers that resist propagation are substantially the same population.

Verification cost varies with class. Confirming that a per-operation consumer has adopted is a log query. Confirming that a scheduled job has adopted requires waiting for its next run. Confirming that an external integration has adopted requires asking someone. The verification gate in Section 4 is therefore cheap for some consumers and expensive for others, and an estate weighted toward the lower rows makes verification the dominant cost of rotation rather than a step within it.

9. A Rotation Incident

One rotation involved a service-user credential where a dependent consumer required additional coordination before the previous access key could be safely disabled.

Monitoring and validation identified the dependency during the rotation window. Because the previous credential was still active at that point, it remained active while the new credential was verified and propagated. There was no customer-facing outage and no missed service-level commitment.

What the incident demonstrated is that successful credential creation does not mean every downstream consumer has adopted it. Those are separate facts, and a rotation process that treats creation as the difficult step and disabling as a formality has the difficulty backwards.

What changed as a result. Pre-rotation dependency checks, consumer validation, monitoring, and confirmation before disabling were all strengthened, and the overlap period was made an explicit part of the documented rotation procedure with escalation defined when validation is not completed within the expected window.

The last change is the one worth drawing out. The overlap was already happening; making it an explicit, documented step with an escalation path converted an informal practice into a control that survives the departure of the people who understood why it mattered.

10. Emergency Rotation: A Gap in This Paper

The lifecycle described here is designed for scheduled rotation, where propagation delay can be absorbed because there is no urgency. A suspected compromise inverts the problem: the previous credential must be revoked immediately, which removes the overlap window that every safety property in this design depends on.

This paper does not characterise that case, and the honest reason is that the material behind it does not support a claim. How quickly the estate could be rotated after a suspected compromise, and what would break when it was, was not established, and the analysis was not performed. Rather than reason from the scheduled

case to the emergency one, which would produce a plausible answer with nothing behind it, the paper records this as an open area.

What can be stated follows structurally from the design rather than from measurement. Emergency revocation removes the overlap, so every consumer that has not already adopted the new credential fails. The manual remainder in Section 3 would be the binding constraint, because its propagation time is not bounded by anything the organisation controls. And the verification gate becomes useless, since its function is to delay disabling until adoption is confirmed, which is exactly the delay an emergency cannot afford.

Section 16 identifies characterising this case as the most valuable outstanding work.

11. What Would Make Rotation Structurally Easier

The design in this paper makes rotation safe within an estate as it stands. A separate question is what would make rotation less necessary or less difficult, and the answers are architectural rather than procedural.

Eliminate the credential where possible. A workload identity that obtains a short-lived token on demand has no rotation lifecycle, because nothing persists long enough to require one. Where this is available it dominates every approach in this paper, and the reason it is not universal is the population identified in Section 3: legacy applications, vendor integrations, and systems that cannot dynamically retrieve anything. The strategic move is therefore not to rotate better but to shrink the population that needs rotating, and to treat every new integration's fetch pattern as a decision with a rotation consequence.

Move consumers up the fetch hierarchy. Table 5 orders consumer classes by propagation behaviour, and the ordering is largely a property of how each consumer obtains its credential. A consumer reading per operation is trivially rotatable. Converting a configuration-file consumer into one that fetches at startup, and a startup consumer into one that fetches per operation, is a bounded engineering change that improves rotatability permanently. This is unglamorous work with a compounding return, since every future rotation benefits.

Make dependency discovery automatic. The incident in Section 9 arose because credential creation and consumer adoption are different facts. Deriving the consumer set from access records rather than from documentation removes the failure mode where a consumer nobody remembered is discovered by its failure. This is the single change that would most improve the verification gate, because verification can only confirm adoption by consumers it knows about.

Instrument adoption, not just issuance. Every consumer's first successful use of a new credential is an observable event, and recording it converts the verification gate from a manual confirmation into a query. It also produces the propagation-time distribution that Section 14 identifies as the paper's principal measurement gap, as a byproduct rather than as a separate exercise.

Reduce the blast radius per credential. A credential consumed by fifteen services is fifteen propagation problems bound together. Scoping credentials more narrowly, so that each is consumed by fewer services, makes each rotation smaller and independently recoverable. It increases the number of rotations and decreases the consequence of each, which is the correct direction when the failure mode is a consumer that did not adopt.

These are ordered by leverage rather than by ease. The first is the strongest and applies least widely; the last applies everywhere and helps least. An estate that adopted all five would find that rotation had stopped being a project and become a property of how consumers were built.

12. Evidence and Its Limits

Table 6. Evidence status by claim class.

Claim class	Status
Approximately 40 to 60 secrets across 15 to 20 services	Established from the work. Approximate
Rotation intervals of 30 to 90 days, service-user keys approximately 90	Established as the operating policy
Credential classes automated and manual, Table 1	Established from the estate
Seven-step lifecycle and the overlap window	Design position, implemented in the work described
The rotation incident and the four controls strengthened after it	Observed

Claim class	Status
No production incident or customer-facing outage from a missed scheduled rotation	Stated on the basis of operational experience. Not supported by a retained incident record
Rotation success rate	**Not measured.** Described as high; no figure substantiated
Mean credential propagation time	**Not measured.** No such metric maintained; propagation varies by consumer and deployment process
Current count of credentials under management	**Not supplied.** The figure was left blank in the source material and is not reconstructed here
Earlier managed-service rotation work, 2018 to 2020	Architecture and lessons only. Project environments, logs, dashboards, and historical metrics no longer accessible
Emergency rotation after suspected compromise	**Not characterised.** See Section 10

Two entries deserve emphasis.

The absence of a propagation-time metric is the most significant measurement gap, because propagation delay is the phenomenon the entire design exists to manage. The lifecycle handles it by waiting for evidence rather than by predicting duration, which is the right design, and it also means the organisation never had to measure the thing and therefore did not.

The earlier work cannot be evidenced at all. The architecture, implementation approach, operational challenges, and lessons from that period are described from direct experience, and the underlying environments, logs, and metrics are no longer accessible. Nothing quantitative from that period is claimed.

13. Guidance for a Team Implementing Rotation

Four positions follow from the design and are worth separating from it.

Enumerate consumers before automating anything. The verification gate can only confirm adoption by consumers it knows about, so an incomplete consumer list silently weakens the whole mechanism. Deriving the list from access records rather than from documentation is the difference between verification that means something and verification that means the consumers somebody remembered have adopted.

Instrument first use, not just issuance. A consumer's first successful authentication with a new credential is observable and is the single most useful event in the lifecycle. Recording it converts verification from a manual step into a query and produces propagation timing as a byproduct.

Separate disable from remove in the tooling, not just in the procedure. A procedure that says disable then remove will eventually be executed as one step by someone in a hurry. Making them distinct operations with a mandatory interval between them enforces the recovery path that Section 4 depends on.

Treat the manual remainder as a backlog with an owner. Section 3 argues that risk concentrates in credentials automation does not reach. That population shrinks only if someone is accountable for shrinking it, and in most organisations it is nobody's objective because each individual item is small and awkward.

14. Limitations

The verification gate has an unbounded worst case. Section 4 makes disabling conditional on verified adoption. A consumer that never adopts therefore blocks the rotation indefinitely, and the paper does not state what happens then. In practice this becomes an escalation and eventually a decision to disable anyway, accepting the failure, and that decision has no stated owner or criterion here.

Credential compromise during the overlap is not considered. The overlap window exists so that propagation can complete safely. It also means an exposed old credential remains valid for the duration, which is a deliberate acceptance of risk that the paper does not weigh explicitly against the availability benefit it buys.

No measured rotation outcomes. Rotation completion and operational validation were tracked, but not in a form producing a defensible success rate. The operational measure actually used was completion within the rotation window and verification that consumers were using updated credentials.

Absence of incidents is weaker evidence than it appears. No production incident or customer-facing outage arose from a missed scheduled rotation. That is a genuine and favourable observation, and it is not the same as

a measured reliability rate, because the denominator is not documented. It is also consistent with a process that is well designed and with one that has not yet been stressed.

The manual remainder is described, not sized. Section 3 argues that risk concentrates in the credentials automation does not reach. What proportion of the estate that population represents is not given, so the reader cannot judge how much of the risk the automated lifecycle actually removes.

Emergency rotation is unaddressed. As Section 10 states, the compromise case is not characterised. Since that is the scenario in which rotation matters most acutely, this is a substantive limitation rather than a scoping choice.

Single-estate basis. The design comes from one organisation's estate of moderate size. Whether the verification gate remains practical at an estate of thousands of credentials and hundreds of consumers is untested, and the verification step is the one most likely to become a bottleneck at scale.

Verification is described but not specified. Section 4 makes the verification gate the load-bearing mechanism and does not state how adoption is confirmed for each consumer class. In practice that confirmation ranges from an authentication log entry to a successful scheduled job to a manual check, and those differ substantially in how much assurance they provide. A reader implementing this design has to construct that layer themselves, and it is the part most likely to be done weakly.

Two eras, unevenly evidenced. The paper spans work from roughly 2018 to 2020 and from 2023 onward. Only the later period supports specific description, and the reader should weight the two accordingly.

15. Future Work

Characterise emergency rotation. Running a controlled exercise that revokes and re-issues across the estate under time pressure would establish how long full rotation takes, which consumers break, and whether the manual remainder is the binding constraint the design predicts. This is the largest gap and the most valuable thing to close.

Measure propagation time per consumer class. Instrumenting the interval between credential issue and first successful use by each consumer would convert the overlap window from a judgement into a derived parameter, and would identify which consumers are structurally slow.

Size the manual remainder. Establishing what proportion of the estate cannot be automatically rotated, and what each obstacle is, would show whether the population is shrinking and would prioritise which obstacles to remove.

Automate dependency discovery. Section 3 notes that manually rotated credentials have dependency lists held in memory. Deriving consumers from access records rather than from recollection would remove the failure mode that produced the incident in Section 8.

Extend the verification gate. Verification currently confirms that required consumers have adopted the new credential. Confirming that no consumer is still successfully authenticating with the old one is the stronger assertion, and it is the one that would make disabling provably safe rather than confidently safe.

A note on sequencing. Automated dependency discovery should come first, because the verification gate can only confirm adoption by consumers it knows about, and every other improvement inherits that limitation. Adoption instrumentation follows naturally from it and produces the propagation-time distribution as a byproduct rather than as a separate measurement exercise.

The emergency rotation exercise is listed as the largest gap and should nonetheless not be first, because running it against an estate whose dependency picture is incomplete would measure the incompleteness rather than the rotation capability. Sizing the manual remainder is cheap and can proceed in parallel with any of these.

16. Conclusion

Credential rotation is deferred because the risk of rotating is immediate and attributable while the risk of not rotating is latent and diffuse. The engineering answer is not to correct that judgement but to remove the availability risk that drives it.

The mechanism is a controlled overlap in which both credentials are valid, closed on positive verification that consumers have adopted the new one rather than on a timer. That single ordering decision, verify before disable, is what makes propagation delay a normal operating condition rather than an outage, and it is why the incident in Section 8 was a task rather than an interruption. Separating disable from remove preserves a recovery path for the consumer nobody knew about.

Two things this paper does not establish are worth restating plainly. There is no measured propagation time, which is the quantity the design exists to manage, and there is no characterisation of emergency rotation after a suspected compromise, which is the scenario that removes the overlap window and therefore every safety

property described here. The second is the more consequential omission, and closing it requires an exercise rather than an argument.

References

[1] National Institute of Standards and Technology, *Digital Identity Guidelines: Authentication and Lifecycle Management*, NIST Special Publication 800-63B.

[2] National Institute of Standards and Technology, *Recommendation for Key Management, Part 1: General*, NIST Special Publication 800-57 Part 1.

[3] PCI Security Standards Council, *Payment Card Industry Data Security Standard*, Requirement 8, Identify Users and Authenticate Access to System Components.

[4] Center for Internet Security, *CIS Critical Security Controls, Control 5: Account Management*.

[5] Internet Engineering Task Force, *The OAuth 2.0 Authorization Framework*, RFC 6749, October 2012. doi: 10.17487/RFC6749.