Distributed Privilege

Requiring Coordinated Authority for Consequential System Operations

Dr. Ron Ross
CEO, RONROSSECURE, LLC
ISTS FELLOW, DARTMOUTH COLLEGE

Abstract

Distributed Privilege is a foundational design principle for trustworthy secure systems. NIST SP 800-160, Volume 1, Revision 1 states that multiple authorized entities act in a coordinated manner before an operation on the system is allowed to occur. The principle constrains what one entity can cause, even when that entity holds legitimate privileges. This article connects the allocation and enforcement of distributed privilege to stakeholder needs, loss analysis, architecture, the system life cycle, engineering for compromise, complementary principles, and assurance evidence. Its effectiveness depends on what each entity authorizes, how the system combines their actions, and whether a single entity with common control can fabricate agreement or bypass the requirement. The design must also preserve essential service when agreement is unavailable.

1. The Engineering Problem

A maintenance engineer prepares a change to a control system. An operations supervisor approves the work. The change management system records both actions. Yet the maintenance account can still install the change directly. The organization has two approvals in its process, but the system permits one entity to cause the consequential operation. The distinction matters when that entity makes a mistake, is coerced, or becomes compromised.

Concentrating authority is often convenient. A single deployment service can prepare, authorize, and release an update. A recovery administrator can select a restoration source, change the operating baseline, and reconnect a restored system. An automation agent can interpret a request, choose a tool, and commit the resulting transaction. Each arrangement can make ordinary work faster. The engineering question is whether a single entity should have enough authority to cause the resulting system effect.

> ***"The question is who can cause the effect, even when the approval process says otherwise."***

A policy that assigns two people different duties is a starting point. It establishes an organizational expectation about behavior. The system design determines whether that expectation holds when a privileged entity disregards the process. Review the operation at the point where its effect becomes possible. An approved ticket and an authenticated request may provide useful records, but the action that changes the system must depend on the required combination of authority.

The relevant entities may include people, services, devices, or automated decision functions. They need not all perform the same task. One entity can substantiate the technical basis for a change, another entity can determine whether current operating conditions permit it, and an enforcement mechanism can require both contributions before allowing the change. The value of the arrangement depends on what each contribution establishes and the authority under which it is produced.

Privilege concentration can be difficult to see in architecture diagrams. Two services may have separate names and separate credentials while sharing a deployment service with the privilege to replace both. Two reviewers may use separate accounts while the same administrator can assume either identity. A local

© 2026 RONROSSECURE, LLC | CC BY 4.0.

protective function may enforce a two-party rule in normal operation while a maintenance path allows equivalent changes without it. The significant question is how many independently controlled entities must be defeated to cause the loss.

The issue also applies to AI-enabled systems. Two agents can return agreeing recommendations because they received the same misleading context, relied on the same tool output, or ran under the same controller. Their agreement may help compare analysis, but it cannot carry the weight of independent authorization unless the architecture establishes distinct authority and addresses the shared dependencies that affect the decision.

The security design principle of *Distributed Privilege* makes this design obligation explicit. Identify the operations for which unilateral authority is unacceptable. Allocate the necessary privileges so that multiple authorized entities must contribute. Enforce the coordination requirement where the operation takes effect. Then establish what remains protected when one entity, or a dependency it controls, can no longer be trusted.

2. What the Principle Means

NIST SP 800-160, Volume 1, Revision 1 states the design principle of *Distributed Privilege* as follows [1]:

> ***"Multiple authorized entities act in a coordinated manner before an operation on the system is allowed to occur."***

The principle distributes the privileges necessary for an operation among multiple entities. Their positive actions must satisfy rules, conditions, and constraints before the operation can proceed. Authorization to participate is distinct from authority to complete the operation. Each entity can have all the privilege needed for its assigned action while the system still withholds authority to produce the final effect until the coordination requirement is satisfied [1]. The design principle of *Distributed Privilege* generalizes long-established security concepts. Saltzer and Schroeder described the principle of separation of privilege [4]. Other common security practices, including separation of duties and the two-person rule (dual control), apply it to organizational roles and to critical operations.

NIST describes three forms of coordination. **Simultaneous actions** require different authorized entities to act within a specified time window. **Sequenced actions** enable each successive action only after the preceding action succeeds. **Parallel actions** allow entities to execute concurrently and combine the results by consensus or voting. Designs can combine these forms. For example, technical review can precede a release decision that requires two distinct participants[1] to act within the same operating window [1].

The requirement for positive action matters. Silence, failure to object, and expiration of a review period do not establish that a required entity acted. A timeout may initiate a separately engineered fallback, but it should not be treated as approval of the original operation. This distinction is especially important during disruptions, when the evidence needed for a sound decision may also be unavailable.

NIST supplies the principle and coordination patterns. The following questions translate them into engineering decisions for a particular system:

- **Operation.** Identify the effect that requires distributed authority. Include operations with different names or interfaces that can produce the same material result. Define the target, scope, and relevant operating conditions.

[1] A *participant* can be a human, a service, or an automated decision function authorized to perform a specific action in the coordinated workflow. That action might be proposing a change, reviewing it, or authorizing its release.

© 2026 RONROSSECURE, LLC | CC BY 4.0.

- **Allocation.** Specify the privilege, information, and responsibility of each entity. Determine which entities can act in each role and whether one entity can acquire or impersonate enough roles to satisfy the rule.
- **Coordination.** Define the required actions, ordering, timing, and decision rule. Establish what an entity's action means, which operation it applies to, and when a change in context invalidates it.
- **Enforcement.** Identify the mechanism that determines whether the conditions are satisfied. Follow the operation to its point of effect and examine other paths that can reach the same result.

"Distribute the authority to act. Bind the agreement to the effect."

The applicable unit of agreement is a specific operation or a precisely bounded class of operations. If one entity reviews a proposed configuration and another releases a different configuration under the same approval, the coordination requirement has lost its meaning. Scope the approval to the relevant version, target, parameters, and conditions. When those change materially, require a new decision under the applicable rule.

The principle addresses the ability to act alone. It does not establish that a coordinated decision is technically sound. Authorized entities can be mistaken, influenced by common information, or induced to agree to a harmful operation. Additional engineering constraints and evidence are needed to show that the operation is acceptable under the conditions in which it will be performed.

3. What the Principle Does Not Mean

The following distinctions help identify when an arrangement supplies useful coordination without providing the protection attributed to *Distributed Privilege*:

- **Two steps do not necessarily mean two entities.** One person or service can perform successive workflow steps with different accounts. The relevant test is whether the privileges behind the accounts can be exercised by the same entity under the stated assumptions.
- **Two signatures do not establish two independent entities.** A common signing service or administrator may control both credentials. Distinct signatures can still provide useful provenance, but their value as independent contributions depends on who can produce them.
- **Separation of duties needs enforcement at the operation.** Role assignments and review records can support accountability. A system that accepts an equivalent action through another interface still permits unilateral execution.
- **Multiple approvals do not establish acceptable behavior.** Two reviewers can approve an incorrect parameter or rely on a common invalid assumption. Independent protective constraints may still be needed to keep the resulting action within an approved envelope.
- **Multiple entities need not hold identical privileges.** A proposer, a technical reviewer, and a release approver may provide distinct contributions. Granting each entity complete execution authority defeats the purpose of requiring their combined action.
- **More approvers do not necessarily improve protection.** Additional approvals can increase delay and dependence on shared services. Choose the coordination rule in relation to the loss, the effective independence of the participants, and the time available for a decision.
- **A recorded exception does not preserve the original claim.** An emergency path that permits unilateral action changes the architecture's authority boundary. Evaluate the exception as a distinct operating mode with its own scope and assurance evidence.

© 2026 RONROSSECURE, LLC | CC BY 4.0.

Authentication, least privilege, and distributed privilege answer different questions. Authentication establishes the identity or provenance required for a decision. Least privilege limits what each entity can do. Distributed privilege specifies when several authorized entities must act together. An entity can be correctly authenticated and narrowly privileged while still holding enough authority to cause an operation that should require additional agreement.

A quorum is one possible means of combining decisions. Its protection depends on who is eligible, which participants must be represented, how membership can change, and what shared services remain trusted. A numeric threshold is meaningful only in relation to those assumptions. In some designs a technical concurrence and an operational release provide more relevant protection than two votes drawn from a pool with identical access and information.

The authority model should distinguish administrative separation from organizational separation. A department can assign a reviewer and a releaser from different reporting chains while both rely on the same support team to reset credentials. Conversely, participants in the same organization can act with distinct privileges governed by independently protected enrollment and policy. A credible design describes which relationships matter for the anticipated loss and why the underlying mechanism preserves them.

Consider the ordinary maintenance case alongside deliberate misuse. An engineer may accidentally select the wrong target because two instances look alike. A second participant working from the same ambiguous display can make the same mistake. Providing a stable target identifier and a meaningful description of the expected effect can make the second action informative. The value of coordination depends partly on whether the design lets each participant recognize what has been proposed.

4. Loss-Driven Interpretation

Start with stakeholder protection needs and unacceptable losses. Determine which operations can produce those losses, which entities can currently cause them, and how much delay the mission can tolerate before a legitimate operation must occur. These questions determine where distributed privilege can help. Applying the same coordination rule to every action can add delay without constraining the effects stakeholders care about.

Consider a water treatment facility with two treatment process lines. Routine local control maintains each process line within a validated operating envelope. A proposed maintenance change would alter the approved configuration that governs a process line’s treatment operation. The facility needs a means to make justified changes, but it cannot allow one compromised maintenance entity to introduce a change that can lead to unsafe output. It must also preserve minimum required service if an approving entity or the coordination service becomes unavailable.

In this illustrative allocation, engineering reviews the technical suitability of a specific configuration package. Operations determines whether the affected process line can accept that package in its current state and during the proposed window. The release mechanism requires both positive actions before the package can take effect. Independent protective functions retain their specified authority over the treatment process. Numerical limits, process conditions, allowable transition times, and the behavior on loss of approval must come from facility-specific engineering and validation.

Two unacceptable outcomes guide and inform the design. The first is an unauthorized or improperly scoped configuration change that contributes to unsafe output. The second is loss of required treatment service because the facility cannot perform a necessary protective action or continue local control while the change workflow is unavailable. Requiring more authority for a planned configuration change should not

© 2026 RONROSSECURE, LLC | CC BY 4.0.

make an immediate protective action wait for remote approval. The two actions have different purposes and different timing constraints.

The following steps translate the loss analysis into authority requirements:

- **Identify the loss pathways.** Trace how an unauthorized change, a valid approval applied to the wrong target, or a delayed protective action could affect the mission. Include administrative and restoration paths that can produce an equivalent change.
- **Identify the necessary entities.** Specify what technical or operational judgment each entity contributes. Determine which information each needs to act and whether its view can be controlled by another participant.
- **Define permitted authority.** Allocate proposal, concurrence, release, and execution privileges. State which combinations are required and which authority must remain outside each participant's control.
- **Establish protective independence.** Determine what local control and protective functions must remain effective when a participant or coordination service fails. Identify the trusted dependencies that make this possible.
- **Compare the alternatives.** Evaluate sequenced concurrence, concurrent decisions, constrained automatic enforcement, and independent protective limits. Consider the consequences of both permitting and withholding the operation.

Distributed Privilege is particularly valuable when a system operation combines authority across two different kinds of judgment. A technically sound package can be unsuitable at a particular moment. A convenient maintenance window does not establish that the package is valid. Making both judgments explicit gives the design a basis for requiring contributions that address different causes of loss.

Consider also where a single authorized operation can create a condition for later harm. Changing eligibility rules, enrolling replacement credentials, changing a restoration baseline, or modifying an enforcement policy may confer effective execution authority without immediately affecting the mission. Include such actions in the analysis when they can undo the separation that protects the consequential operation.

5. Engineering the Principle into the Architecture

Define the operation in terms of what changes in the system. In the facility example, the protected effect is making a new treatment configuration active on a specified process line. Submitting a work order, staging a package, and recording review are preparatory actions. The design should make it possible to perform those actions without implicitly gaining the privilege to activate the configuration.

Define what each contribution establishes. The systems engineering concurrence concerns the particular configuration and its supporting technical analysis. The operational release concerns the target, current operating conditions, and authorized time window. Bind both to the same proposed change and ensure that the enforcement mechanism can determine that the binding remains valid at the point of effect. A workflow record that refers only to a ticket number may be insufficient if the contents can change after review.

Architects should examine the authority needed to create, verify, and consume each contribution. Treat a common identity provider, policy administration service, deployment pipeline, or system management plane as a dependency of the claim. The design can rely on a common mechanism if its trustworthiness is established for the intended claim and its compromise is explicitly outside that claim. If compromise of that mechanism must be tolerated, the architecture needs a protective decision or restriction that the same mechanism cannot override.

© 2026 RONROSSECURE, LLC | CC BY 4.0.

"Agreement protects the operation only if one entity cannot fabricate it."

The following design measures help make the coordination requirement enforceable:

- **Separate proposal from execution.** Permit a proposer to assemble and submit a change without permitting that entity to activate it. Give each subsequent participant the minimum privileges needed to contribute its own decision.
- **Bind decisions to an operation.** Associate each contribution with the target, operation, version, applicable parameters, and permitted conditions. Determine which changes invalidate approval and how cancellation takes effect.
- **Verify the coordination rule at the point of effect.** Check eligibility, distinctness, ordering or threshold, freshness, and the required operating conditions before accepting the operation. Cover maintenance, recovery, and local administration interfaces as well as the primary workflow.
- **Protect the enforcement mechanism.** Limit authority over the identities, configuration, code, and records it uses. Determine which entity can modify the rule itself or change what the mechanism considers valid evidence.
- **Bound waiting and partial completion.** Specify what happens when an entity is unavailable, an action expires, or an interrupted operation leaves incomplete state. Preserve a safe and operationally acceptable state through those transitions.
- **Preserve independent protection and recovery.** Ensure that restoring the coordination mechanism does not restore outdated approvals or allow operational credentials to define their own replacement authority.

A sequenced workflow must address the interval between actions. An approval can become outdated because the package changed, the system moved to another mode, the participant lost eligibility, or new information invalidated the decision. Verify the conditions that must still hold at execution. Distinguish conditions that can be bound to the approved request from conditions that require a current observation.

For concurrent decisions, establish what agreement means before selecting a threshold. A rule can require participation by different functional roles, a specified number of independent entities, or evidence of concurrence on a common request. If different entities evaluate different proposed versions, a count of affirmative responses provides no meaningful agreement about the actual operation. The mechanism needs to reject conflicting or ambiguous contributions and retain enough evidence to explain the outcome.

The enforcement mechanism also needs a precise completion model. A distributed operation may take effect in stages. An interruption can leave a configuration partially applied, or a retry can repeat a consequential action. Identify the state in which a decision is consumed, the state in which the operation is committed, and the protection required when those states diverge. Recovery should verify the actual system state before choosing whether to resume, roll back, or initiate a newly authorized action.

Emergency operation requires explicit design. A narrowly scoped local protective action may need to remain available even when the approval service is unreachable. Limit its permitted effects and duration to the mission need, establish how it is invoked, and specify the conditions for leaving the emergency mode. A broad maintenance override can undo the original privilege allocation precisely when the system is under the greatest stress.

Where operators must decide under pressure, give them information that supports a meaningful judgment. A request should disclose the actual target, intended effect, important constraints, and any changes since

© 2026 RONROSSECURE, LLC | CC BY 4.0.

an earlier review. A prompt that reduces an important decision to a generic confirmation invites routine acceptance. Independent contributions require distinct authority and a practical opportunity to exercise judgment.

Automated entities can participate if the architecture defines what they evaluate, the authority they represent, and their failure assumptions. A service can, for example, check that a staged package corresponds to a reviewed baseline. Its automated decision is evidence of that check. Treating it as an independent operational release would require a separate basis for trusting that decision under the conditions that matter.

The *Acquisition* process can determine whether these allocations are feasible. A supplier product may expose detailed workflow approvals but permit an administrator to execute the equivalent action through a management interface. Ask suppliers to identify execution and recovery paths, describe the privilege behind each path, and explain which paths enforce the coordination rule. Where the product cannot enforce it, scope the claim to the architecture that can, or revisit the product and integration trade.

A release decision can be conditioned on information that has a shorter useful life than the reviewed configuration. For example, a review can remain applicable to the same package while the operational conditions supporting release change. Manage these as separate validity conditions. The design should establish who can attest to current conditions, when the attestation expires, and which changes require the entire decision to be reconsidered.

A useful design review distinguishes evidence from authority. A signed analysis report can substantiate a property of a package without authorizing installation. A release approval can authorize installation without proving that the underlying report is correct. Keeping these meanings separate helps prevent one artifact from acquiring more authority than intended. It also supports selective reanalysis when the package, deployment target, or operating conditions change.

There is a cost to enforcing a coordination rule across several systems. Decision records must be available when needed, identity information must be current, and the target must be able to distinguish accepted requests from stale ones. When those services fail, the design needs a bounded disposition for pending requests. Assess that disposition with the same care as the successful path, including how operators learn whether an operation took effect.

6. Engineering for Compromise

Engineering for Compromise starts from the possibility that a system element becomes hostile and asks what must remain true despite that condition [2]. For *Distributed Privilege*, the immediate question is whether a compromised participant can cause the protected effect alone. The broader question is whether that participant can use its existing authority or influence to cause another element to supply the missing contribution. NIST notes that defeating distributed privilege requires collusion to take an unauthorized or improper action and that an adversary must target all of the entities to which privilege is distributed [1].

A compromise containment boundary must account for what the entity legitimately controls. A systems engineering reviewer may be able to submit technically plausible material, define a proposed package, or provide data to another reviewer. A release approver may have a valid decision credential and control over scheduling. The system can constrain each role's direct actions while still allowing one role to shape the information used by another. Analyze both the privilege path and the path through decision inputs.

The following scenarios separate different failures of the coordination arrangement:

© 2026 RONROSSECURE, LLC | CC BY 4.0.

- **Compromised proposer.** Determine whether a proposer can obtain approval for one operation and cause a different one to execute. Examine substitution of target, version, parameters, or timing between review and release.
- **Compromised approving participant.** Determine what one entity can authorize directly and which distinct action still has to occur. Examine whether the compromised entity can influence or impersonate the other participant through authority it already holds.
- **Compromised coordination mechanism.** Determine whether fabricated or incomplete agreement can reach the point of effect. Identify any enforcement that independently checks the required evidence and the trust assumptions on which that check depends.
- **Compromised administrator.** Examine whether one administrative entity can enroll identities, change eligibility, replace enforcement code, or switch the system into a mode that accepts unilateral action.
- **Compromised restoration source.** Determine whether restoration can reintroduce old membership, permissive policy, stale approvals, or an earlier configuration that bypasses the current rule.

The useful measure is the effect that can be produced under the stated compromise assumptions. If two signers rely on the same credential-issuing entity and that entity is compromised, two visible identities can collapse to one effective source of action. If each signer has an independently governed credential but the release mechanism accepts fabricated agreement, the release mechanism becomes the dependency that controls the outcome. Identify both cases in the compromise propagation analysis [2].

Availability is part of the same analysis. A hostile entity may refuse to participate, repeatedly cancel valid work, or cause operators to invoke an exception. The system must have a defined response to missing concurrence. For the facility example, a pending configuration change can be deferred under the stated operating conditions while local control continues on its accepted configuration. The permissibility and duration of that state must be substantiated. An operation that cannot be deferred needs a separately engineered path with constrained authority.

A credible claim states the compromise assumptions it tolerates. The facility might claim that compromise of either the systems engineering reviewer or the operational release approver alone cannot activate a configuration without the other required contribution, assuming the local enforcement mechanism and specified trust anchors remain trustworthy. That claim leaves open other questions, including collusion, common compromise of decision inputs, and whether the approved change is safe. Treat those as separate parts of the system assurance argument.

7. Interaction with Other Principles

Distributed Privilege contributes to trustworthy secure design when composed with other design principles that constrain authority, govern access, and preserve required behavior under adversity. The following relationships are especially relevant [1]:

- **Domain Separation.** Establish boundaries between functions and privileges with different protection needs. *Distributed Privilege* can use those boundaries to keep proposal, review, and release under distinct control. Examine paths through shared administration and recovery before crediting the separation.
- **Mediated Access.** Govern each consequential operation through an appropriate mediation mechanism. *Distributed Privilege* supplies conditions for that decision. A bypass around mediation also bypasses any coordination requirement that depends on it.

© 2026 RONROSSECURE, LLC | CC BY 4.0.

- **Least Privilege.** Limit each entity to the privileges necessary for its assigned action. *Distributed Privilege* specifies which narrowly allocated privileges must be combined. An entity with broader administrative privileges can defeat the allocation even if its application role is narrow.
- **Least Sharing.** Limit common resources and common authority to what the mission requires. A shared approval record can be useful, while a shared entity able to rewrite all participants' decisions can defeat the intended protection.
- **Least Persistence.** Limit the useful life of authorizations, delegated rights, and approval state. Expire decisions that no longer reflect the approved conditions. Remove transitional privileges once the work that justified them is complete.
- **Trustworthy System Control.** Establish that the mechanisms enforcing the coordination rule are non-bypassable, always invoked where required, protected against tampering, and amenable to evaluation. *Distributed Privilege* depends on the trustworthiness of enforcement at the point of effect.
- **Protective Failure and Protective Recovery.** Preserve required protections when a decision is missing, processing fails, or restoration changes system state. Recover authority and policy from a trustworthy basis before treating the workflow as operational.
- **Compositional Trustworthiness.** Analyze the assembled system including the relationships among its system elements. Independently evaluated credential, workflow, and execution services can form a composition with a common management dependency that none of their individual evaluations addresses.
- **Substantiated Trustworthiness.** Support the allocation, enforcement, and independence claims with evidence. Establish what behavior was demonstrated, which dependencies remain trusted, and which circumstances require a narrower claim.

Cyber resiliency engineering provides a complementary perspective through techniques such as *Privilege Restriction* and *Segmentation* and the objective of continuing essential mission functions under adverse conditions [3]. In the water treatment example, the loss of an approval service is a condition the architecture must address without handing unilateral authority to a remote maintenance function. Restricting privilege is useful when the restrictions hold during ordinary operation and through disruption and recovery.

Composition can reveal tensions between design principles. A common enforcement service can simplify analysis and provide consistent decisions while introducing a shared dependency. Separate services can reduce one dependency while adding more interfaces and recovery states. Choose the arrangement in relation to the claimed compromise tolerance and verify the resulting architecture. The number of mechanisms is a poor substitute for the analysis of their authority.

The water treatment example illustrates how responsibilities combine. *Domain Separation* distinguishes local treatment control from remote maintenance. *Least Privilege* limits which role can propose, review, release, or activate a configuration. *Distributed Privilege* requires multiple positive actions for activation. *Trustworthy System Control* supports the claim that enforcement occurs on each path to the protected effect. *Protective Failure* and *Protective Recovery* govern the handling of interrupted changes and restoration. None of these relationships eliminates the need to validate the resulting process behavior.

When an architecture contains both human and automated decisions, consider differences in evidence and failure modes. A human review can notice context omitted from an automated check. An automated check can consistently enforce a precise binding between an approved package and the executed one. A common source of incorrect data can affect both. Identify which complementary functions address the loss and which dependencies limit the independence claim.

© 2026 RONROSSECURE, LLC | CC BY 4.0.

Consider a different system in which two independently administered services must concur before releasing protected information. One evaluates the authorized purpose and scope. The other checks the intended destination and the conditions on release. The independence claim depends on both the allocation of privileges and the integrity of the information used for each decision. A trusted release mechanism must still verify that both decisions apply to the same information and destination.

The architecture should also make it possible to retire an entity without interrupting legitimate work indefinitely or expanding authority by default. Replacement arrangements may require separate enrollment, verification of the replacement's role, and a review of pending decisions. If an entity is removed because its authority may have been compromised, determine which prior decisions remain usable and what evidence supports that determination.

8. System Life Cycle Application

The following activities connect *Distributed Privilege* to the system life cycle processes described in NIST SP 800-160, Volume 1, Revision 1 [1]:

- **Business or Mission Analysis.** Identify the operations whose unilateral execution can contribute to unacceptable loss. Determine which mission functions must continue if required concurrence is unavailable.
- **Stakeholder Needs and Requirements Definition.** Establish protection needs, acceptable decision times, necessary authority distinctions, and the consequences of denying or delaying an operation.
- **System Requirements Definition.** Specify the protected effects, eligible entities, required contributions, coordination constraints, and behavior when agreement is incomplete or invalid.
- **Architecture Definition.** Allocate privileges and decision functions to entities and domains. Identify every path to the protected effect and the common dependencies on which enforcement and independent judgment rely.
- **Design Definition.** Specify request binding, decision freshness, eligibility changes, cancellation, failure handling, and the conditions under which an operation is accepted and committed.
- **System Analysis.** Evaluate compromise propagation, collusion assumptions, common modes, response time, incomplete operations, and the effect of losing the coordination mechanism.
- **Implementation.** Realize the authority allocation, decision binding, enforcement, and state transitions. Trace privileged interfaces to their intended use and protective requirements.
- **Integration.** Examine the assembled system for inherited administration, shared service identities, inconsistent policies, and management paths that create effective unilateral authority.
- **Verification.** Demonstrate that the specified rules are enforced at the point of effect. Exercise incomplete, outdated, mismatched, and improperly combined contributions and inspect the resulting system state.
- **Transition.** Establish authorized membership and trust anchors, remove commissioning exceptions, and confirm that the deployed authority allocation matches the verified configuration.
- **Validation.** Determine whether the realized arrangement supports legitimate work under operating conditions and preserves required protection when a participant cannot act.
- **Operation and Maintenance.** Monitor changes to eligibility, trust anchors, overrides, dependencies, and policy. Revisit the analysis when a new role, interface, or restoration process changes the effective authority.

© 2026 RONROSSECURE, LLC | CC BY 4.0.

- **Disposal.** Remove retired credentials, enrollment paths, and pending approvals. Address residual ability to act on operational or restored systems after the original entity is removed.

The system life cycle matters because authority can migrate without changing the documented workflow. A new support contract may give an external service provider administrative access. A modernization effort may consolidate identity systems. A replacement device may accept an earlier recovery configuration. Treat these changes as potential changes to the assurance boundary and record how the relevant evidence is maintained or renewed.

9. Failure Patterns and Anti-Patterns

The following patterns indicate a difference between the intended privilege allocation and the authority that the realized system permits:

- **Approval is confined to the user interface.** The workflow presents multiple approval steps, while a direct maintenance interface can produce the protected effect. Follow each interface to the state it can change.
- **One administrator controls every approving identity.** The design counts distinct accounts but relies on one entity that can create or assume all of them. Analyze effective control of credentials and eligibility.
- **The approval identifies a ticket but leaves its contents mutable.** An entity can obtain concurrence and then change the package or target. Bind decisions to the reviewed operation and invalidate them when it changes.
- **Approval survives a material state change.** A release remains usable after a maintenance window closes or the system changes modes. Determine which conditions must be rechecked at execution.
- **Timeout becomes permission.** The workflow treats a missing decision as consent. Specify the consequences of incomplete agreement and engineer any fallback as a distinct, bounded action.
- **Retries obscure the point of effect.** A service reports failure after the operation may have committed. The system repeats the action or consumes the wrong decision. Check actual state and define how authorization is tied to completion.
- **Emergency access restores unilateral authority.** A broad override is retained for convenience. Compare the exceptional operation’s authority and scope with the loss analysis that justified the normal rule.
- **Restoration rolls back the authority boundary.** Recovery restores obsolete participants, credentials, or policy along with operational state. Verify current authorization conditions before accepting consequential requests.
- **Agreement is inferred from shared analysis.** Two automated reviewers use the same inputs, tool, and controlling entity. Their outputs may corroborate a computation without providing separate authority to act.
- **Review protects the named operation only.** The system constrains one command but allows another privileged operation to create the same effect. Define coverage in terms of the consequential result.
- **Membership changes escape review.** The execution rule is protected, while one entity can expand the set of eligible actors or lower the effective threshold. Treat changes to the rule and its participants as consequential operations in their own right.

“Two approvals provide little protection when one entity can create both.”

© 2026 RONROSSECURE, LLC | CC BY 4.0.

These patterns can accumulate gradually. A temporary maintenance role becomes permanent, and a workflow service later gains the credentials needed for recovery. Each change can appear reasonable when reviewed alone. Reconstruct the full authority path after integration and significant change. Start with the operation whose effect matters and trace backward to every entity that can authorize, initiate, or enable it.

10. Evidence and Assurance

A distributed-privilege claim needs evidence that the relevant operation is identified, multiple authorized entities must contribute, and no single entity, under the stated compromise assumptions, can cause the protected effect. An approval log can show that two contributions were recorded in one execution. It cannot by itself establish coverage of other execution paths or show that one entity could not have created both contributions.

A top-level claim might be:

> ***Within the defined system scope and under the stated operating, failure, and compromise assumptions, a protected configuration change can take effect only after the required distinct authorized entities act in coordination for that change. Loss of a required contribution preserves the specified protective state and minimum service.***

The argument links the authority allocation, decision binding, and enforcement at the point of effect to the absence of unilateral execution paths under the stated assumptions. It also links the handling of incomplete agreement and interrupted change to the required service and protective behavior. A broader claim about safe treatment must address process conditions, validity of configuration parameters, and the effectiveness of independent protection.

The following supporting subclaims organize the assurance argument and identify the evidence needed to substantiate each part of the top-level claim:

- **Subclaim 1: Complete scope and dependencies.** The analysis identifies each protected operation, equivalent effects, participants, and the dependencies that govern eligibility, administration, execution, and restoration. Evidence includes requirements traceability, interface inventories, authority models, and architecture reviews.
- **Subclaim 2: Justified and distinct authority.** Each required contribution serves a specified mission or protection need, and the assumed compromise of one participant does not provide authority to supply another required contribution. Evidence includes role and credential analysis, membership controls, administrative dependency analysis, and trade studies.
- **Subclaim 3: Binding and enforceable coordination.** The operation can take effect only when current and valid contributions apply to the same authorized request and satisfy the required rule. Evidence includes enforcement specifications, code or configuration review, integration tests, and checks of the actual target state.
- **Subclaim 4: Bounded failure and service impact.** Missing concurrence, interruptions, and restoration delays preserve the specified protective state and minimum service. Evidence includes operating-mode analysis, timing analysis, failure injection, and representative operator exercises.
- **Subclaim 5: Protected enforcement and administration.** Enforcement, eligibility, and decision evidence remain trustworthy under the stated assumptions. Where a particular dependency compromise must be tolerated, independently enforced restrictions continue to operate. Evidence includes platform assurance, privilege analysis, protected configuration, and compromise-oriented verification.
- **Subclaim 6: Preserved protection through change and recovery.** Maintenance, emergency operation, credential changes, and restoration preserve or deliberately reestablish the required authority

© 2026 RONROSSECURE, LLC | CC BY 4.0.

allocation. Evidence includes transition procedures, change impact analyses, recovery demonstrations, and tests for removal of expired or revoked authority.

Verification should distinguish absence of approval from improper agreement. Test a missing contribution, two contributions attributable to the same controlling entity, expired concurrence, approval for another target, and a change in the reviewed package. Demonstrate what happens if the system is interrupted between validation and execution. Observe the protected state, including partial changes and subsequent recovery, rather than relying solely on workflow messages.

Negative tests should follow alternate paths to the protected effect. An authorization rule that works through the primary release workflow can be defeated by an equivalent path through another interface or operating mode. Use the architecture and authority model to establish which paths must be examined. Tests supply observations under exercised conditions. Analysis establishes why the selected paths and states provide coverage appropriate to the claim.

In the water treatment facility, assume the design is intended to tolerate compromise of either one maintenance participant or one approving participant, and loss of the central approval service, under the stated conditions. Local control and specified protective mechanisms remain trusted. Engineering concurrence and operational release are separately allocated. Activation checks both against the exact configuration and the authorized target and conditions. These assumptions define the assurance context and must be revisited if the arrangement changes.

For the unauthorized change scenario, the supporting argument should show that one compromised participant cannot activate an unapproved configuration directly, substitute it after review, or use another management or restoration path to achieve the same effect. Evidence should connect each attempted action to the observed state of the target process line. A rejected request in the central workflow cannot substantiate the claim if the local configuration changes through a separate interface.

For the loss of service scenario, show that the facility can continue required local control for the specified time when a valid approval is unavailable. Demonstrate the transition to the required protective mode when continued operation is no longer acceptable. Assess the burden on operators, the availability of trustworthy local information, and the authority needed to recover normal operation. The permissible duration and process behavior require facility-specific validation.

“Assurance must follow the authority all the way to the operation that changes the system.”

Maintain the assurance argument as the system evolves. A new identity platform, a changed threshold, a new maintenance interface, or a different recovery image can change who can act alone. Keep a trace from each consequential system operation to its allocation, assumptions, and evidence. An earlier successful demonstration provides evidence about the configuration exercised at the time. Sustain the claim by identifying changes that affect it and renewing the analysis and evidence where needed.

A useful verification record includes the initial state, identity and authority of each actor, request version, applicable operating conditions, expected effect, observed effect, and the disposition of pending decisions. This structure helps reviewers distinguish a correctly rejected operation from an operation that was never attempted at the relevant interface. It also makes the evidence easier to reassess when the architecture or the operating assumptions change.

Validation should examine the work people perform around the mechanism. Observe whether reviewers can identify the intended effect, whether they have enough time and information to act, and whether a

© 2026 RONROSSECURE, LLC | CC BY 4.0.

blocked workflow invites an unsafe workaround. Exercises should include routine maintenance and representative disruptions. The result informs both the protection claim and the judgment that the arrangement supports the intended mission.

11. Key Engineering Takeaways

The following takeaways summarize the engineering decisions and assurance obligations needed to apply the design principle of *Distributed Privilege* throughout the system life cycle:

1. Define the operation by its consequential effect. Include equivalent paths through maintenance, administration, emergency operation, and restoration.
2. Derive the coordination requirement from stakeholder protection needs, unacceptable losses, and the time the mission can tolerate waiting for a decision.
3. Allocate privileges so that multiple authorized entities must contribute. Give each entity the minimum authority and information needed for its assigned action.
4. Select simultaneous, sequenced, parallel, or combined actions to match what the system must establish before permitting the operation.
5. Bind each contribution to the authorized operation, target, version, and conditions. Reevaluate the decision when those conditions materially change.
6. Examine effective authority over identities, policy, enforcement, and recovery. Separate accounts or services can depend on the same controlling entity.
7. Engineer behavior for incomplete agreement, interrupted execution, and recovery. Preserve essential service and protection without silently broadening execution authority.
8. Compose *Distributed Privilege* with *Domain Separation*, *Mediated Access*, *Least Privilege*, *Least Sharing*, *Trustworthy System Control*, *Protective Failure*, and *Protective Recovery*.
9. Apply *Engineering for Compromise* to participants, administrators, common mechanisms, and restoration sources. Make the stated compromise assumptions and trusted dependencies explicit.
10. Substantiate the claim with evidence from actual system behavior and analysis of the paths that can produce the protected effect. Revisit the evidence when authority or dependencies change.

12. Conclusion

Distributed Privilege makes the authority for consequential operations an explicit system design decision. When one entity can cause a change with unacceptable consequences, an approval policy can reduce ordinary error while leaving the decisive privilege concentrated. The architecture must determine when a single entity can act, when additional authority is required, and how the requirement will be enforced.

NIST provides the foundation. Multiple authorized entities act in a coordinated manner before the operation is allowed to occur [1]. Applying that principle requires allocating privilege, defining positive actions and their relationship to the intended effect, and maintaining enforcement across normal, exceptional, and recovery paths. The design must also explain how essential functions continue when agreement cannot be obtained.

Engineering for Compromise sharpens the analysis [2]. Assume that one entity becomes hostile and follow the authority it can still exercise. Examine the ability to influence another decision, change the rules that determine agreement, or control shared infrastructure. Where the same entity can produce all of the required contributions, the architecture supports a narrower claim than the workflow suggests.

© 2026 RONROSSECURE, LLC | CC BY 4.0.

Producers should make these design choices and their limits visible. Consumers should ask who can cause a consequential system operation, what distinct entities must agree, which dependencies can collapse that distinction, and what evidence demonstrates the intended behavior. These questions apply to enterprise systems, operational technology, cyber-physical systems, and AI-enabled architectures.

A trustworthy secure system supports legitimate change while constraining unilateral actions that could lead to unacceptable loss. *Distributed Privilege* establishes a design basis for that constraint and an obligation to substantiate it throughout the system life cycle.

"Require distinct authority for consequential action. Preserve protection when agreement is unavailable."

Acknowledgments

The author gratefully acknowledges the support provided by Dartmouth College's Institute for Security, Technology, and Society (ISTS). The resources provided by the ISTS contributed to the research and development of this paper. The views and conclusions expressed herein are those of the author and do not necessarily represent the views of Dartmouth College.

References

[1] R. Ross, M. Winstead, and M. McEvilley, Engineering Trustworthy Secure Systems, NIST Special Publication 800-160, Volume 1, Revision 1, National Institute of Standards and Technology, November 2022. https://doi.org/10.6028/NIST.SP.800-160v1r1

[2] R. Ross, "Engineering for Compromise: Designing Systems That Remain Trustworthy Secure Under Adversity," RONROSSECURE, LLC, August 2026. https://ronrossecure.com/library

[3] R. Ross, V. Pillitteri, R. Graubart, D. Bodeau, and R. McQuaid, Developing Cyber-Resilient Systems: A Systems Security Engineering Approach, NIST Special Publication 800-160, Volume 2, Revision 1, National Institute of Standards and Technology, December 2021. https://doi.org/10.6028/NIST.SP.800-160v2r1

[4] J. H. Saltzer and M. D. Schroeder, "The Protection of Information in Computer Systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, September 1975. https://www.cs.virginia.edu/~evans/cs551/saltzer

© 2026 RONROSSECURE, LLC | CC BY 4.0.