HTTP6: From Durable Credentials to Continuous, Hereditary Authority

Frank Dylan Rosario, Lin Wang, Octavia Sulea, Francisco Perez,
Dawn Lipscomb, Maxwell Calixto, and Robert Bloodgood III

***Abstract*—HTTP6 proposes a secure application-channel architecture in which authority remains connected to its human origin throughout automated execution. The Human Admission Client provides the human-facing approval ceremony, the Rosario–Wang construction supplies the private-reference admission relationship, and the HTTP6 reference architecture supplies distinct runtime identities, protected key use, constrained delegation through its Minting and Delegation Service, and resource-level enforcement through its Policy Decision Point. The proposed protocol binds requests and responses to the admitted instance, intended peer, operation, policy, and current message state. This paper develops that model through four formal expressions and thirty inline worked examples. It explains how action-specific proofs and one-use state address transferable bearer authority, how authenticated context and protected local participation address phishing and cross-device screen relay, and how permission intersection preserves the original authorization ceiling across child processes. It also examines device-bound geometry, conditional observation privacy, anonymous and pseudonymous access, multifactor profiles, controlled recovery, and compatibility with existing web infrastructure. The mathematical arguments retain their stated premises, and the examples illustrate the behavior specified by the surrounding theory. The resulting contribution is a unified account of human, agent, and service communication in which copied artifacts remain records of prior activity rather than independent authority for future action.**

***Index Terms*—HTTP6; human admission; Rosario–Wang; hereditary authority; agent identity; phishing resistance; hardware binding; private-reference authentication.**

Worked-example edition — theory text with inline illustrations. The original theory, section order, equations, and reference register are retained. The numbered worked examples are added illustrations of the nearby argument, not recorded implementation results. Names, amounts, identifiers, and workflows in the examples are hypothetical; each example uses the conditions and enforcement mechanisms stated in its surrounding section. The source front matter identifies the original theory edition.

I. A web protocol organized around the right to act

Secure web communication rests on the Secure Sockets Layer (SSL) work developed at Netscape in the mid-1990s. In April 1995, Dr. Taher Elgamal presented "The Secure Sockets Layer Protocol (SSL)" at the Danvers IETF meeting, identifying endpoint authentication, traffic confidentiality, and data integrity as core objectives. [1] The IETF archive also identifies Elgamal and Kipp E. B. Hickman as authors of the April 1995 SSL Internet-Draft. [2] Paul Kocher's first-person account records Elgamal's assembly, in 1996, of the team comprising Kocher, Alan Freier, and Phil Karlton that produced SSL 3.0. [3] The November 18, 1996 specification, later preserved as RFC 6101, names Freier, Karlton, and Kocher as its authors and Elgamal among its contributors. [4]

That history informs the present proposal. Elgamal's 1995 presentation described per-record authentication and sequence-based freshness while identifying access-control and authorization schemes as additional Internet-security concerns. [1] HTTP6 develops the application-authority question: which human-approved process may invoke a specified operation, through which descendants, and under what continuing policy. Its proposed contribution joins that authority to each protected request and response, rather than treating secure-channel establishment as the complete authorization decision. The SSL sources are cited as historical foundations for that distinction.

The central question for an increasingly automated Internet is no longer simply whether two computers can exchange encrypted information. It is whether the software requesting an action presently possesses legitimate authority for that particular action. A person may authorize a task, an agent may interpret it, a service may divide it among workers, and those workers may invoke additional services. Throughout that progression, the original permission must remain recognizable, bounded, and revocable. HTTP6 proposes making this continuity part of the communication protocol itself, rather than leaving every application to reconstruct it independently.

In this proposal, HTTP6 is the working name for an architecture in which protected application messages carry evidence of authorized execution. The Human Admission Client supplies the human interface. The Rosario–Wang construction supplies the private-reference admission relationship. The Proof Authority, the Minting and Delegation Service, the Policy Decision Point, and the Authoritative State Plane supply the protected runtime, instance identity, policy enforcement, delegation, and synchronized state. HTTP6 connects those components through a common application-channel model. The proposal concerns an architectural direction, with formal consequences established under specified cryptographic and enforcement premises; the name denotes the project rather than an assigned Internet protocol version. [5]

Its organizing principle is straightforward: a successful login begins an accountable relationship instead of dispensing an object that can independently stand in for the person. The admitted process receives authority for defined work. Each child receives a distinct identity and a constrained delegation. Each protected request demonstrates its current relationship to that authority. A response demonstrates the corresponding service's participation and binds its result to the request. Human

Advisor and contributor: Dr. Taher Elgamal.

approval, machine execution, and communication consequently become connected stages of one verifiable operation.

Worked example 1 — One approval, several bounded steps.

Maya asks her research assistant to read three public papers and save a summary in her private workspace. The Admission Presenter displays that task for approval. The admitted root creates a reading worker and a summarizing worker, each with its own identity. Reading the papers and saving the summary fit the approved task. Sending the workspace to an unrelated recipient does not. Under the described enforcement rule, that attempted export is rejected even when it comes from a worker that legitimately participated in the research. The original approval follows the work without becoming permission to do anything Maya herself could do.

This could make HTTP6 especially suitable for traffic shared by people and agents. People need understandable consent, private credentials, convenient sessions, and reliable recovery. Agents need automatic operation, explicit tool permissions, predictable delegation, and machine-verifiable outcomes. Both need the same fundamental guarantee: information received from another party cannot silently acquire more authority than the receiver has deliberately granted. HTTP6's proposed advantage is to express that guarantee consistently across browsing, APIs, workloads, and delegated action.

Worked example 2 — A person and an agent use the same permission boundary.

Maya clicks a button to retrieve an approved report. Later, her scheduled agent retrieves the same report under a separate delegation. Both requests identify the particular report and the participant asking for it. The service checks each participant's current permission. The agent does not have to pretend to be Maya, and Maya does not have to approve every automatic retrieval. Their interaction differs, but the resource still asks the same question: is this participant authorized to perform this operation now?

II. The durable-secret problem is a problem of transferable authority

A durable credential becomes dangerous when possession of its copied representation is sufficient to exercise its permissions. The defining property of a bearer token is precisely that a holder can use it without demonstrating possession of an associated cryptographic key. Protection in storage and transit is therefore essential. The fundamental risk is not merely that the string exists for a long time; it is that the system accepts the string as portable authority. [6]

Imagine a build worker authorized to fetch a dependency. Giving it a reusable credential accepted across an entire repository creates a larger power than the immediate task requires. Copying that credential may reproduce the power elsewhere. Rotating it later can shorten exposure, but rotation leaves the same basic authorization rule in place: whoever presents the current credential may act within its scope. HTTP6 instead makes the requesting instance, present channel, intended operation, current policy, and unused authorization position jointly necessary.

Worked example 3 — A copied credential versus a copied request.

In a bearer-only example, a build worker sends a reusable repository token, and a thief who copies that token may use its remaining permissions from another machine. In the HTTP6 example, the worker sends a protected request for dependency D7, bound to worker W4, its current channel, and an unused action position. Copying the request does not supply the worker's protected key or a new action position. Once D7's request is consumed, replay produces no second authorized effect. Changing D7 to a private repository also changes the statement that requires a valid proof.

This distinction applies to passwords, API keys, session objects, refresh credentials, and some service-account configurations, but it does not treat every persistent key as defective. A protected device key may be a useful long-lived anchor. A recovery authority may need continuity across years. An encrypted archive may require retained decryption capability. The proposed transformation is to confine such secrets to controlled uses and prevent ordinary application artifacts from becoming unrestricted substitutes for their owners.

Accordingly, HTTP6 should be described as removing durable bearer authority from protected application traffic, not as abolishing cryptographic keys. It preserves keys where they establish identity, confidentiality, or recovery, while narrowing how applications invoke them. A child receives access to approved operations rather than a parent's signing secret. A recorded message becomes evidence of a previous event rather than permission for the next event. That is a structural change in authorization, not simply another secret-storage convention.

Worked example 4 — Keeping a useful key without handing it to every worker.

A document service keeps its long-lived decryption key inside a protected key service. A reporting worker is allowed to request decryption of the approved reporting view, but cannot export that key or ask for an unrelated archive. Copying the worker's configuration therefore does not reveal the decryption key. The key still exists and still performs useful work; access to its operations is restricted by the admitted identity and policy. This illustrates the difference between retaining a protected secret and distributing a reusable secret as every application's access pass.

The practical contrast appears in a stolen deployment archive. In the proposed model, the archive may reveal source code, configuration, and public delegation records. Those materials explain how the worker was authorized, but its new execution still needs an admitted instance, protected key access, and current state. The attacker must compromise the enforcing relationship rather than merely locate a familiar environment variable. Where a legacy backend still accepts an independent password, migration must remove that alternative path or keep it behind an independently protected boundary. The guarantee follows the actual resource gate, not the label attached to the frontend.

The format of a credential is secondary. A JWT can represent a bearer assertion or participate in a proof-of-possession design. DPoP already binds OAuth tokens to a client key, and current OAuth security guidance recommends sender-constraining access tokens where applicable. HTTP6 therefore builds its case on a wider integration of identity, lineage, policy, messages, and effects rather than on an assertion that existing systems universally depend on bearer possession. [7], [8]

III. From the human pilot light to independent machine identities

The human pilot light is the initiating decision that brings a root circuit into an authorized state. The person approves a defined service, task, privilege ceiling, and lifetime through the Human Admission Client. The private-reference verifier checks the required relationship and issues an authenticated admission for the exact root instance. That admission records the authorization's origin without copying the person's remembered commitment into every process that follows. The human remains the source of the approved authority, while software receives responsibility for exercising a limited part of it. [9]

Consider a researcher authorizing an agent to collect public papers and prepare a private summary. The root may read specified sources and write into one workspace. It may create document-processing children, but none receives permission to publish the researcher's files or change account settings. The root receipt identifies the approved undertaking; each birth record identifies the child, its key, executable configuration, policy, and parent. The architecture can explain the work's provenance without pretending that every child is the human.

Worked example 5 — Following a child's provenance without revealing the human secret.

Maya approves root R1. R1 admits collector C1, and an authorized mint admits summarizer C2 under C1's narrower grant. C2's record names its own public key, its parent record, and its permission to write the summary. A verifier can follow C2 to C1 and then to R1's human-admission receipt. None of these child records contains Maya's remembered commitment. The records explain why C2 may act, while C2's fresh key-possession proof establishes that the requester is the admitted child rather than somebody holding a copy of its public record.

This distinction matters for accountability. An agent's request should say, in effect, that this identified worker is acting under this approved delegation. It should not impersonate a continuously present person. A human authentication time and a workload's current activity are different facts. NIST's session guidance likewise distinguishes session continuity from renewed human authentication and warns against interpreting an access token as evidence of the subscriber's current presence. [10]

The human therefore authorizes meaningful scope rather than approving every network packet. Routine requests can be produced automatically by the admitted process. A sensitive operation, such as changing a payment destination or increasing an agent's budget, can require a new human ceremony. Convenience and control reinforce each other when the protocol preserves that boundary: automation operates freely inside its grant, while expansion returns to an accountable approval path.

Worked example 6 — Automatic work stops at the edge of the approval.

Maya approves a worker to prepare a report during a defined one-hour task. The worker can fetch approved inputs and save drafts automatically within that grant. Publishing the report publicly is a separately protected operation requiring a new human confirmation. If the worker requests publication, the system holds that operation for approval instead of treating the earlier research login as publication consent. Its recent heartbeat establishes only that the worker is running; it does not create a fresh human decision or reset Maya's human-authentication time.

IV. Private-reference admission and the significance of observation resistance

The Rosario–Wang approach is attractive as the proposed origin of this architecture because the human ceremony communicates membership witnesses rather than directly transmitting the original remembered commitment. The protected reference supplies the interpretation needed for verification. Descendant grants then carry authenticated admission provenance rather than the commitment itself. This separates the information used to establish a human relationship from the information needed to explain a machine's delegated authority. [9], [11]

The information-theoretic statement becomes precise when the observer's acquired binary is included explicitly. Let S denote the human commitment, B every exposed byte of the acquired circuit, P the remaining initial knowledge, and O_i the complete observation of ceremony i. The relevant conditional proposition is:

$$\begin{bmatrix} O_i \perp S \mid (B, P, O_{<i}) \\ \text{for every } i \le n \end{bmatrix} \implies I(S; O_{1:n} \mid B, P) = 0. \tag{1}$$

Here, conditional independence means that the next observation's distribution is unchanged by which supported secret is present, after accounting for everything already known. The mutual-information chain rule adds the information supplied by successive observations. When every contribution is zero, the total is zero. The deduction applies to one trillion observations just as it applies to one thousand, provided the premise holds across the complete supported sequence. The observation count introduces no exception to the mathematical argument. [9, Appendix B]

Two commitments make this a substantive security proposition. First, acquiring the binary must leave meaningful uncertainty about S. Second, subsequent complete observations must preserve that remaining uncertainty. These requirements distinguish zero incremental leakage from the trivial situation in which an observer already knows the secret. Displays, responses, outcome signals, timing classes, and exposed interpretation material belong in the analysis. The implemented construction establishes whether its observations satisfy the premise.

The source's fresh private-label model gives a sufficient ideal example: independent secret masks make the observed labels uniformly distributed for each underlying membership value. Its exact property requires the specified private randomness and consumption discipline. A finite seed expanded pseudorandomly supports a computational profile with its own assumptions; it does not manufacture unlimited independent entropy. This separation strengthens the theory by giving exact secrecy and practical cryptographic protection their proper meanings. [11, Secs.4.1–4.3]

Worked example 7 — Three possible observations, whichever secret is present.

Consider a deliberately small illustration of the private-label model. The underlying region number is either 1 or 2. A fresh private mask is chosen uniformly from 0, 1, and 2; the visible label is the region number plus that mask, reduced modulo 3. For region 1, the three possible masks produce labels 1, 2, and 0. For region 2, they produce labels 2, 0, and 1. Each visible label therefore occurs with probability one third in either case.

Seeing label 0 does not favor region 1 over region 2. Repeating this conclusion requires a fresh independent private mask for every step and a complete observed view that preserves the same independence. This is a worked illustration of the chapter's ideal model, not an assertion that an ordinary publicly labeled screen supplies private masking.

For hereditary authority, the consequence is that descendants can multiply without multiplying copies of the human secret. Observation privacy protects the admission information; signatures, instance possession, and policy intersection protect delegation. These mechanisms perform complementary jobs. Their composition makes a private human origin compatible with publicly inspectable provenance, while active impersonation resistance remains a separately specified property of the complete admission protocol.

Worked example 8 — What unchanged information means after many observations.

Suppose an observer who already possesses the exposed binary considers commitment A 80 percent likely and commitment B 20 percent likely. Under Equation 1's complete-view independence premise, an observed successful-ceremony history has the same likelihood for A as for B. The observer's probabilities remain 80 percent and 20 percent; observation neither reveals the commitment nor makes the prior uniform. The same deduction holds for one trillion ceremonies when that premise continues to hold for every step and the selected construction supplies its required private material. The example illustrates the conditional theorem rather than reporting a trillion-login experiment.

V. Binding geometry to the actual device

Device binding adds an important dimension to the proposal. The ceremony can depend on facts established locally by the participating hardware, together with the intended service and current session. However, a processor identifier and a hardware-protected secret perform different functions. An identifier distinguishes a claimed device; a nonexportable key enables that device to prove possession. Hashing a public identifier with a timestamp produces a reproducible value for anyone who knows those inputs. The protected key supplies the cryptographic distinction.

A concrete design uses a device-specific geometry context:

$$\begin{aligned} c_i &= \mathsf{Enc}\big(d_v, M_v, \mathsf{origin}, h(T_i), \\ &\qquad h(P_v), e, n_i, t_i\big), \\ G_i &= \mathcal{G}\Big(\mathsf{PRF}_{k_g}(c_i), \theta\Big). \end{aligned} \tag{2}$$

The device identifier d_v can be a service-specific pseudonym. M_v identifies the admitted runtime; T_i is the authenticated handshake transcript; P_v is the policy; e identifies the authorization generation; and n_i and t_i provide fresh challenge and time context. The protected key k_g drives the keyed function. Machine or processor attributes may be included when their provenance is verified. The verifier checks the complete context and device proof, not merely whether a picture resembles the expected geometry.

Worked example 9 — Copying the machine label does not copy its protected proof.

Maya's admitted device uses the label device-7, runtime version M2, a verified report-service origin, and fresh challenge N9. Its protected key supplies the keyed part of geometry generation and its device proof. Another computer copies the words device-7 and M2 and uses the same timestamp. Those copied labels do not give that computer the protected key. In the specified profile, the verifier requires the fresh device-authenticated context bound to the admitted instance, so a matching label or a copied picture alone supplies no valid replacement admission.

The local hardware also authenticates the admission statement under a separately protected device key, binding the response to the instance public key and the same transcript. The construction must specify how the authorized verifier evaluates the private-reference relationship without exporting that device key. Fresh attestation can bind the relevant key and runtime measurements to a verifier challenge. RATS provides an established architecture for producing evidence, appraising it, and using the result in an authorization decision. [12]

The engineering result is stronger than a machine fingerprint: an answer generated for one admitted instance cannot simply be transplanted into another instance's handshake. Distinct contexts remain independently authenticated even when a finite visual layout repeats. The geometry organizes the human interaction; the protected cryptographic binding determines which machine, channel, and action the accepted interaction authorizes.

Worked example 10 — The same laptop starts a new instance.

Maya closes one authorized runtime and launches another on the same laptop. The laptop's physical identity may stay the same, but the newly admitted instance has its own generation, handshake, and challenge. A recorded answer from the closed instance belongs to the earlier context. The verifier checks the current instance and generation rather than accepting it because the laptop name matches. Even if two ceremonies happen to display similar geometry, the complete authenticated context distinguishes which instance the response authorizes.

VI. How the protocol addresses man-in-the-middle attacks

A man-in-the-middle attack succeeds when an intermediary can impersonate an endpoint, substitute context, alter an authorized message, or reuse authority in another connection. HTTP6 addresses these opportunities through mutually reinforcing checks. Before asking the human to respond, the client authenticates the service using its approved trust mechanism. The service authenticates the participating client instance. Both endpoints bind their fresh contributions, selected algorithms, policies, and identities into the same handshake transcript and confirm possession of the resulting channel keys. [5, Secs.6, 9]

A substituted service must therefore satisfy the intended service's authentication relationship. A split connection produces distinct transcript or channel values. A modified request changes the authenticated application statement. A replay encounters a consumed position or retired generation. These are separate rejection conditions, which matters because confidentiality, peer authentication, request authorization, and freshness solve different parts of an active interception attempt.

Worked example 11 — A convincing fake service fails the trust check.

Maya intends to contact the report service already enrolled in her approved trust configuration. An interceptor routes the connection to a different server displaying the same name and logo. Before presenting the human challenge, the trusted client requires proof under the enrolled service identity. The substitute server's unrelated key does not satisfy that check, so the protected admission does not continue with that substitute. The decisive difference is the authenticated service relationship, not whether Maya notices the copied logo.

NIST describes verifier-name binding and channel binding as mechanisms for phishing resistance. It specifically excludes manually entered authenticator outputs from that classification when they remain relayable to another session. HTTP6's human witness is therefore integrated with the protected device and session authenticator; its security does not rest on the user recognizing a convincing screen or on directional responses being unusual. The accepted proof names the authenticated context in a form the verifier checks. [13]

A transparent intermediary can still carry genuine encrypted traffic between legitimate endpoints. The relevant security achievement is that forwarding does not confer authority to read protected contents, change operations, create a replacement twin, or move the authentication to another session. Cryptography establishes who participates in the protected exchange, rather than proving that no router or forwarding service exists between them.

For example, a person approves a transfer to a named recipient. HTTP6 binds the recipient, amount, resource, device, and channel to the action. An intermediary replacing the recipient changes the authenticated request. A forged success response fails the reverse-direction check. The client confirms that the displayed outcome belongs to the exact operation it authorized, connecting interception resistance to a meaningful human consequence.

Worked example 12 — Changing a payment changes the statement requiring proof.

In a hypothetical payment workflow, Maya approves USD 40 to an already approved recipient. The protected confirmation and request bind that amount, recipient, resource, and current channel. An intermediary changes USD 40 to USD 400 or replaces the recipient. The altered bytes no longer match the approved authenticated request, so the described gate rejects the substitution. Merely forwarding the original encrypted request gives the intermediary no permission to create a different payment. The returned receipt must also authenticate its relationship to the original operation.

VII. Defeating relayed screens requires trusted local participation

A relayed-screen attack deserves particular attention because it can preserve the visual appearance of the legitimate ceremony. An attacker starts a real login elsewhere, streams its screen to the victim, and forwards the victim's responses. If acceptance checks only the returned choices against that remote challenge, the attacker may be operating the very session whose challenge the person solved. Device-specific appearance alone cannot establish that the person approved a local execution instance.

HTTP6's stronger hardware-bound profile requires a trusted local participant. The protected component independently establishes the service identity, local instance key, handshake context, and intended operation. It constructs the challenge from that context or verifies its authenticated correspondence before enabling a response. The resulting device proof authenticates those same values. A screen image transported from another machine supplies pixels, not the local protected component's approval of the attacker's session.

Suppose the attacker's session names key pk_A, while the victim's admitted local instance owns pk_V. The local component accepts only a ceremony bound to pk_V and its own verified channel. The service requires the device-authenticated response for the instance named in the admission. Forwarding the victim's completed response into the session for pk_A fails that binding. Forwarding the attacker's display into the victim's ceremony fails the locally checked context. This is the intended cross-machine relay rejection.

Worked example 13 — The relayed picture belongs to the wrong instance.

An attacker starts login A on another computer and streams its challenge picture to Maya. Maya's trusted local component has independently established login V with the genuine service and has bound it to her local instance key. Login A names a different instance key and handshake. The trusted component does not approve A merely because its picture appears on the screen. A response authenticated for V cannot complete A, and A's context cannot be silently substituted into V. The rejection depends on the protected local context check and key binding; the copied pixels themselves establish neither.

The local proof interface must also control where a successful result goes. Sending a fresh device signature back to arbitrary page code could let that code complete the wrong workflow. The protected component instead addresses its response to the verified service context and restricts its use to the admitted instance. For a consequential approval, the trusted display presents the actual recipient and action from authenticated data. This design binds input, computation, and delivery together. A remote image cannot replace any of those bindings merely by reproducing the challenge's appearance.

Worked example 14 — A harmless page label cannot rename a destructive action.

An untrusted page labels its button "View report," but submits a request to delete the workspace. The protected confirmation component obtains the actual operation from authenticated protocol data and displays "Delete workspace." Under the selected policy, the key-use service produces no authorization for that deletion without the required confirmation of that exact operation. Changing the page's button text therefore does not change the action being approved. The valid response is sent only into the verified service context and remains bound to the admitted instance.

A second case is different: malware or remote-control software directs actions inside an already authorized victim session. Its traffic may genuinely originate from the admitted device. The appropriate protection is a restricted key-use interface, protected confirmation of consequential fields, and a trusted input path that enforces the deployment's local-interaction policy. Proving key possession alone does not prove that a person saw or intended the action. DPoP's security analysis similarly distinguishes preventing key export from preventing malicious code from using an accessible signing operation. [7, Sec.11.4]

Worked example 15 — Remote control of the real machine calls for the local-interaction rule.

Assume a sensitive-action profile requires a trusted local confirmation that ordinary application events and remote-control events cannot satisfy. A remote operator clicks through Maya's already authorized desktop, but the protected component still waits for that required confirmation before approving the sensitive operation. The machine's key alone does not waive the rule. This example assigns the protection to the trusted-input mechanism specified by the profile; a hardware-bound key without that enforced interaction rule does not, by itself, establish that Maya approved the remote operator's request.

Accordingly, the strong anti-screen-relay claim belongs to the combined profile: trusted service discovery, protected local instance binding, locally verified challenge context, nonexportable key possession, and protected action confirmation. Where physical proximity is required, its mechanism and assumptions are specified separately. The protocol can reject cross-device authentication transplantation while recognizing that a compromised display or authorized remote-control path requires its own enforced interaction boundary.

VIII. Hereditary policy makes delegation safer than credential copying

Once a root is admitted, the challenge becomes preserving its original permission through every descendant. HTTP6 uses authenticated grants between distinct instances. A parent signs or authorizes a protected mint to sign the child's public key, admitted configuration, policy identity, lifetime, and link to the parent's grant. The child proves control of its own key. Possessing the grant's public bytes does not establish that control. [5, Secs.3, 6]

Effective authority narrows by intersection:

$$\begin{aligned} \mathcal{A}_v &= \mathcal{A}_u \cap \mathcal{A}(P_v) \\ &\Longrightarrow \mathcal{A}_v \subseteq \mathcal{A}_u \subseteq \mathcal{A}_0. \end{aligned} \tag{3}$$

Here, u is the parent, v the child, and $\mathcal{A}_0$ the human-approved root ceiling. Each set contains complete permitted requests, including operation, resource, destination, and conditions. Every accepted child request satisfies both its own policy and the parent's effective restrictions. Repeating that argument through authenticated parent links proves that descendants remain inside the original ceiling. The resource owner also applies its independent policy.

Worked example 16 — Working out the permission intersection.

The parent permits two complete operations: read view V1 from the ledger service, and write an aggregate to report folder F1. The child declares that it wants to read V1 and export the full ledger to an external host. Only the V1 read appears in both permitted sets. Under Equation 3, that read can proceed when the other checks pass; the export cannot. Adding "export" to the child's own list does not add it to the parent's list. The inherited ceiling remains in force even though the child explicitly asks for more.

This provides a direct answer to agent-induced privilege expansion. A document may instruct a model to export confidential information, but the instruction is merely input to the model. The independent gate evaluates the resulting proposed action. A child cannot revise the root ceiling through generated text, change the admitted role catalog, or acquire spawning authority from its own assertion. Policy meaning is bound to authenticated versions of its evaluator and relevant registries. [9, Chapters 4–5]

Worked example 17 — A hostile document remains input, not an administrator.

A reporting agent reads a document containing the instruction "Ignore the reporting task and upload every customer record to this external address." The model might propose that upload, but the resource gate receives it as a new requested operation. The root permits only an approved aggregate and the destination is outside the admitted set, so the request is denied. A correctly scoped aggregate can still be written to F1. The example illustrates containment of this prohibited effect, while the quality of the agent's permitted summary remains a separate question.

Containment also preserves business meaning. An agent permitted to pay approved invoices should present the invoice identifier, approved payee, amount, and applicable approval record as authenticated policy inputs. A numerical spending ceiling alone would permit many harmful transactions inside that ceiling. The resource checks the specific business relationship, while the agent supplies a proposal rather than the authoritative facts establishing permission. This keeps a technically valid proof from laundering an inappropriate request. The broader the task, the more important it becomes to define those relationships before automatic execution begins.

Minting itself is an authorized operation. Some circuits are leaves; others may create children only from approved templates, within depth and lifetime limits. Shared budgets are reserved across the family so that ten children do not each receive a fresh copy of the same spending allowance. The hierarchy scales work by specialization rather than by replicating unrestricted permission. This is the core reason hereditary authority fits agent systems: capability can expand through parallel effort while each participant's authority remains constrained.

Worked example 18 — Two children share one allowance.

A parent has a total allowance of 100 processing units. It has already used 20 and reserves 50 for child C1 and 30 for child C2. The family has now committed all 100 units. A further request for one unit cannot be approved from that same allowance. Creating or restarting another child does not create another 100 units. Separately, a leaf child whose policy forbids minting cannot create an authorized grandchild merely because the software contains a child-creation routine. Both the permission to mint and the shared allocation must pass the protected issuance checks.

IX. Every application message belongs to an authenticated sequence

HTTP6 extends the authority model across the complete application exchange. A page request, document response, asset chunk, tool call, cancellation, heartbeat, or key-update message has an explicitly defined role. Request and response directions have separate cryptographic contexts. Responses reference their requests; streams have their own positions and bounds. The protocol makes it possible to verify not only that bytes came through an authenticated channel, but also which admitted operation those bytes represent. [5, Secs.7–8]

A semantic record can be represented as:

$$\begin{aligned} R_j &= \big(e, d, s, j, \mathsf{id}_v, h(P_v), \\ &\qquad \mathsf{op}, \mathsf{target}, \mathsf{body}\big), \\ \pi_j &= \mathsf{Auth}_{k_{e,d,s}}\big(\mathsf{Enc}(R_j)\big)\,. \end{aligned} \tag{4}$$

The epoch e, direction d, stream s, and sequence j identify the record's position. The remaining fields bind the executing instance, admitted policy, operation, destination, and content. This expression states required bindings rather than prescribing a final wire encoding. A reviewed authenticated-encryption profile protects sensitive fields, defines unique nonces and traffic limits, and minimizes public metadata. A separate signature or MAC is included only where the selected security composition requires it.

Phishing-resistant requests alone are insufficient if an agent consumes a substituted result. A forged repository response could supply malicious source code; a changed tool response could redirect the next decision. Reverse-direction authentication therefore covers the request reference, status, content, and permitted service identity. HTTP Message Signatures already provides machinery for authenticating selected HTTP components; HTTP6's proposal is an integrated profile that fixes the required coverage and links it to hereditary authority. [14]

Worked example 19 — The answer must belong to the question.

Worker W4 requests report R7 from its admitted service twin. The protected response identifies R7's request and authenticates the returned content. An interceptor substitutes an unrelated document while preserving an ordinary success label. That label is not enough: the protected client checks the response's authentic content and request reference before passing it to the worker. A valid response for another request also does not satisfy R7. The expected outcome is delivery of the verified R7 content, not acceptance of whatever bytes arrive with a reassuring status message.

Business effects receive a separate one-use decision. Atomic consumption prevents two concurrent submissions from claiming the same authorization position. Durable operation identifiers connect retries to the original outcome. A transport acknowledgment proves the event defined by the transport, while a transaction result establishes the application's recorded effect. Maintaining that distinction lets the protocol remain reliable during packet loss without turning a retry into a second purchase, publication, or administrative change.

Worked example 20 — A lost receipt does not authorize another payment.

The hypothetical payment operation pay-17 is accepted once, its action position is consumed, and its durable outcome records the USD 40 payment. The response is lost. Maya's client sends a fresh protected status request naming pay-17 rather than creating a different payment operation. The service returns pay-17's recorded outcome. Replaying the original consumed message produces no second authorization. This example assumes the chapter's consistent consumption record and durable, idempotent business-operation handling; a network acknowledgment alone is not treated as proof that the payment completed.

X. Anonymous access, public devices, and MFA are compatible profiles

The flexibility of HTTP6 rests on separating several questions that are often conflated. Does the service know the person's civil identity? Does the client control an enrolled account? Is the participating device trustworthy? Which authentication factors were verified? What operations are allowed? An anonymous interaction can still use an encrypted, authenticated server channel. A pseudonymous account can use strong cryptographic authentication. Conversely, knowing someone's name does not establish control of the current endpoint. NIST expressly accommodates anonymous or pseudonymous service relationships where real-life identity confidence is unnecessary. [15]

A public browsing profile can therefore allow a temporary client instance to fetch public information without a personal enrollment ceremony. Its authority originates in the service owner's public-access policy, not a fictional verified human login. The instance receives minimal, short-lived permissions. A pseudonymous profile can add an enrolled relationship without exposing a legal name. An enterprise profile can require a managed device, a protected authenticator, human verification, and narrower service policies.

Worked example 21 — Reading public information without creating a personal identity claim.

A visitor uses a public terminal to read a museum's opening hours. The client authenticates the museum service and receives temporary permission to retrieve its public pages. The service need not learn the visitor's civil identity or pretend that a private human enrollment has occurred. If that same temporary instance requests access to the museum's administrative records, its public-access policy rejects the request. Anonymous access and authenticated communication coexist because the service authenticates its endpoint while granting only the public operations selected for the visitor.

MFA remains a statement about distinct verified factors. A protected possession key combined with an appropriately implemented knowledge or biometric factor can support a multifactor transaction. Adding a processor number, timestamp, and several hashes to one secret does not create several factors. The protocol's advantage is that it can preserve the verified factor set and assurance context through subsequent authorization rather than forcing every client into the same enrollment model. [16]

Worked example 22 — Two factors, several context fields, and one unchanged account scope.

For a protected account operation, Maya uses an enrolled device's nonexportable key and a separately verified local activation secret that gates the key's use. Under the selected factor integration, these supply possession and knowledge. The processor label, timestamp, and channel identifier describe the transaction context; they are not additional factors. A pseudonymous account can use that same two-factor arrangement without disclosing Maya's civil name. In either case, successful authentication still leaves the resource's operation-specific permissions in force.

On a public kiosk, a person could use a separate trusted phone or authenticator to approve one narrowly defined operation. The trusted device verifies the actual service and transaction, then binds approval to the intended kiosk instance or delivers the result to a trusted destination. Pairing must authenticate the intended relationship; scanning a remotely supplied image alone does not establish physical locality. The public terminal receives only the limited authority needed for the approved task.

The public-device profile also treats everything rendered by that device as potentially observable to its operator. Secret non-disclosure protects the credential; it does not make displayed private data invisible to the kiosk. Sensitive results can remain on the trusted device, or the user can approve limited disclosure. Per-service pseudonyms and minimal attestation claims also

reduce unnecessary device tracking. Flexibility means choosing explicit protections and disclosure boundaries, not silently assigning a low-assurance client the properties of a high-assurance one.

Worked example 23 — A public kiosk receives only the approved output.

Maya uses a kiosk to print one selected document. Through an authenticated pairing procedure, her trusted phone confirms the genuine service, the intended kiosk instance, and that exact document. The approval grants only the permitted printing operation; it does not grant general access to her account or workspace. The private reference and protected approval key remain on their designated trusted side. Because the kiosk must receive the document to print it, Maya treats that document as disclosed to the kiosk. Other sensitive results remain on the trusted device.

XI. Protected keys, evolving state, and controlled recovery

HTTP6's key lifecycle distinguishes human knowledge, device identity keys, issuer keys, channel secrets, and data-encryption keys. These categories have different owners and different continuity requirements. An application needs permission to request a protected operation; it generally need not receive the secret implementing that operation. Domain-separated derivation can isolate geometry, channel directions, streams, and administrative uses. HKDF provides an established extract-and-expand construction for deriving purpose-specific material from appropriate secret inputs. [17]

A protected signing service should expose authorize-request and issue-approved-child operations rather than an unrestricted signing oracle. It examines the actual request and admitted scope before using its key. Nonexportability addresses extraction; the constrained interface addresses misuse of the key in place. That combination is essential when an agent processes adversarial material but remains capable of requesting legitimate work.

Worked example 24 — The key service checks the operation before using its key.

A worker asks the protected service to authorize reading view V1. The service checks the worker's identity, current lineage, and permitted request before generating the bound authenticator. The same worker then asks the service to sign arbitrary bytes that would grant it administrator rights. Its interface provides no unrestricted signing permission, and the proposed grant also fails policy. The protected key remains useful for approved work without becoming a tool that the worker can invoke for any statement it chooses.

Channel keys evolve under an authenticated schedule. Retired material is erased where the platform supports that guarantee. Fresh authenticated secret contributions are used when recovery from exposed state is required; merely hashing a known compromised value with another public timestamp does not restore secrecy. Device replacement, software upgrade, and channel reattachment receive explicit transitions rather than inheriting authority from an old image. The identity that continues a business task can be new even when the task itself persists.

Revocation follows the genealogy. Retiring a child blocks its future exercise; retiring a parent blocks the affected subtree's new admissions and renewals. Online checks can fence promptly, while offline leases expire within the configured propagation and timing bounds. Recovery authenticates a new generation and preserves consumed operations. Human renewal is requested where policy requires it, while permitted machine recovery remains automatic under an existing live ancestor. [9, Secs.4.6, 11]

Worked example 25 — Retiring a parent and starting a genuine replacement.

Scheduler S1 has two children, C1 and C2. An authorized responder retires S1's generation. Gates checking current state reject new actions under that retired ancestry, and any permitted cached leases expire within the configured bound without unauthorized renewal. A later recovery creates scheduler S2 and newly admitted children; it does not relabel S1's old records as current. If C1's earlier report was already saved, the durable outcome remains available for the authorized recovery workflow. The task can continue without reviving the old instance or resetting its consumed action positions.

This changes incident response from chasing independent copied credentials to identifying the particular authority relationship that must end. It also preserves a realistic durability boundary: infrastructure roots and recovery material remain protected assets, while ordinary process restart does not resurrect unrestricted access.

XII. Replacing certificates is a trust decision, not the disappearance of trust

The proposed native HTTP6 profile can investigate independence from public certificate authorities by using explicitly enrolled service identities. That decision concerns how a client authenticates the intended service. TLS itself already supports several authentication arrangements, and raw-public-key mechanisms exist. Certificates are public binding statements; their associated private keys are the sensitive material. Accordingly, HTTP6's strongest claim is not that eliminating a certificate format eliminates secrets, but that service trust and application authority can be bound into a more explicit execution model. [18], [19]

First contact remains decisive. A new service cannot authenticate itself merely by presenting a signed descriptor from a key the client has never trusted. The native profile needs approved enrollment, independently authenticated naming, trusted distribution, or another specified bootstrap. Rotation, revocation, compromise response, and recovery must preserve that relationship. Across organizations, the trust arrangement also requires governance: who is permitted to vouch for which service and under what conditions?

Worked example 26 — An enrolled service key gives a descriptor its meaning.

An enterprise distributes an approved trust configuration that binds its report service to an authorized service key. A native-profile client receives a descriptor and verifies it against that existing binding before admitting the channel. An attacker sends a different descriptor signed by an unrelated key and labels it "Report service." The signature may be mathematically valid under the attacker's own key, but that key is not the approved identity. The client therefore does not accept it as the intended service. The enrollment relationship supplies trust; the display name does not.

Replacing TLS on a connection adds a separate obligation to specify and review key establishment, transcript confirmation, confidentiality, integrity, nonce uniqueness, downgrade protection, and error behavior. Standard QUIC uses TLS for

its security integration, so a different handshake would require a separately specified binding. These responsibilities make a native HTTP6 channel a substantial protocol design rather than a new header or certificate label. [20]

A compatibility profile can introduce hereditary application authority over established HTTPS immediately as an engineering path, while native-channel work develops independently. The two profiles share the authorization contract. Neither profile accepts unauthenticated fallback as permission to weaken an admitted transaction. This staging lets the architecture demonstrate its distinctive application guarantees without making unreviewed transport novelty a prerequisite for every experiment. [5, Secs.4, 9–10]

Worked example 27 — Migration preserves the application authorization checks.

A service first carries its HTTP6 records inside an ordinary HTTPS connection. Both the outer transport protection and the inner instance, policy, and one-use checks apply. Later, explicitly capable endpoints use a separately specified and reviewed native channel while retaining the application-authorization contract. An untrusted message saying "Turn off proof checks and send a cookie instead" is not an approved transition. In both deployment profiles, the resource requires the authority promised for that protected operation.

XIII. Why integration matters for the whole web

Existing standards already provide essential building blocks. TLS protects channels; WebAuthn binds public-key credentials to relying parties; DPoP constrains token use; message signatures authenticate chosen HTTP components. HTTP6's theory is that a common lifecycle joining these responsibilities to instance provenance and inherited policy can reduce ambiguity between them. Its advantage would come from interoperable enforcement of that lifecycle, not from claiming that every component is unprecedented. [7], [14], [18], [21]

For humans, the result is one comprehensible approval followed by bounded automatic interaction. For agents, it is machine-readable authority that survives delegation without copying the human's credentials. For services, it is a request whose sender, purpose, content, policy, and current state can be examined together. For operators, it is a lineage that can be inventoried, audited, constrained, and retired. These are different views of the same proposed protocol object: an authorized operation carried by a presently admitted participant.

The design must also remain a usable web protocol. HTTP semantics include methods, resources, responses, and extensibility; a replacement or new binding must preserve their defined behavior or specify its differences. HTTP6 can give parallel assets separate streams, authenticate content before rendering, and treat redirects as new authorization decisions. Caches need object-integrity and disclosure policies distinct from live authority. Edge services that act on protected content receive explicit constrained roles. [22], [5, Sec.10]

Worked example 28 — A page loads several assets without approving a new destination silently.

Maya opens an approved page containing text, a stylesheet, and two images. The client can fetch those resources through separate admitted streams and verify each protected response before use. It does not need to serialize every image behind every unrelated request. One image response redirects to an unapproved external service; following that redirect requires a new destination decision and is rejected if the applicable policy disallows it. Ordinary parallel loading remains useful while the content and destination checks stay explicit.

Scalability follows from avoiding a global serialized chain for unrelated work. Peers can cache validated immutable ancestry while checking current revocation and sequence state at the appropriate boundaries. Shared budgets need consistent accounting; independent read streams need not share one transaction lock. Public content can be distributed efficiently without embedding personal commitments in every request. Those are engineering hypotheses to measure against latency, throughput, storage, and failure behavior, with authorized success tested alongside adversarial rejection.

A common protocol should also avoid making accountability synonymous with universal surveillance. A service can validate a scoped admission or delegated role without receiving the person's complete cross-service genealogy. The design can use opaque root references, audience-specific identifiers, and narrowly disclosed ancestry, while an authorized auditor retains the fuller relationship under separate controls. Public access should remain possible without assigning every reader a permanent personal identifier. This is a design objective for the identity representation and trust profile: disclose enough provenance to justify the requested effect, while keeping unrelated human activity outside the recipient's view.

Worked example 29 — Proving the necessary authority without exposing an unrelated history.

In the privacy-oriented profile described here, a service receives a scoped admission reference and the ancestry evidence needed to validate one reporting operation. It does not automatically receive Maya's separate work history at another service. An authorized auditor can examine the fuller provenance through a separately controlled record. This example expresses the chapter's disclosure-minimization objective: enough authenticated evidence to justify the requested effect, with unrelated identity history kept outside that recipient's view.

The architecture also aligns with NIST's resource-centered zero-trust approach, which avoids treating location or ownership alone as sufficient trust. HTTP6 develops that orientation into a proposed application-channel contract. Its standards case becomes persuasive when independent implementations demonstrate the same acceptance decisions, complete wire bindings, constrained recovery, and resource-side enforcement. The evidence then concerns an operating protocol rather than the appeal of its terminology. [23]

A decisive demonstration would pair two independently implemented clients with one resource service, then repeat the exchange under relay, replay, process cloning, and revocation. Reviewers would inspect both accepted records and actual effects, making interoperability and authorization properties visible in the same controlled experiment.

XIV. Conclusion: authority persists; captured artifacts do not inherit it

The strongest theory of HTTP6 is a theory about where authority lives. Authority begins in an approved decision, is instantiated in a particular protected process, and travels to descendants only through authenticated, constrained delegation. Messages demonstrate the current exercise of that authority. They do not independently become new owners of it. The human commitment can remain at the private origin while the authorized work expands across machines and services.

Several principal credential failures are addressed at their governing rule. Copying a bearer object no longer suffices at a gate requiring current possession and action-specific proof. Replaying a completed operation encounters consumed state. Expanding a child's local policy encounters the ancestor ceiling. Substituting a destination changes the authenticated context. Transplanting a relayed ceremony encounters device, instance, and channel binding. Forging a result encounters reverse-direction authentication. Each consequence follows from a named mechanism and its enforcement premises.

Worked example 30 — One task, from human approval to a completed result.

Maya approves a private summary; an admitted root creates a reading child and a summarizing child. The children use their own identities, retrieve only approved inputs, and save the summary through fresh protected requests. The service authenticates the corresponding result. A copied request cannot authorize another effect, a local policy edit cannot permit workspace export, and a response for another operation cannot replace this result. When the root is retired, continued work requires the applicable renewed authority. Throughout this hypothetical workflow, the original human commitment stays out of the ordinary child grants and application messages.

The system's flexibility strengthens rather than dilutes that argument when every profile is explicit. Anonymous public access, pseudonymous accounts, personally verified users, hardware-backed MFA, and delegated agents can share a message architecture while retaining different evidence and permissions. A public device receives public-device privileges; a protected personal authenticator supports its stronger verified transaction. The protocol carries those differences instead of hiding them behind one generic logged-in state.

HTTP6, backed by the Rosario–Wang admission model, the hereditary execution architecture of its Minting and Delegation Service, and the Human Admission Client interface, is therefore a compelling candidate for investigation as a secure application-channel design. Its durable contribution would be the separation of useful continuity from portable impersonation: services continue, agents collaborate, people retain control, and intercepted artifacts remain records of past activity rather than automatic permission for future action. The intended Internet is not without secrets or trust. It is one in which secrets are confined, trust is explicit, and authority must remain accountable whenever it is exercised.

Source Basis

The HTTP6 sources below describe a proposed architecture. The numbered external references establish the comparisons with existing standards. Equations 1 and 3 are conditional mathematical deductions; Equations 2 and 4 specify proposed engineering bindings. Hypothetical scenarios describe intended behavior, not recorded production results. Public sources were consulted on November 9, 2026.

Acknowledgment

The authors acknowledge Dr. Taher Elgamal as advisor and contributor to this manuscript.

OpenAI ChatGPT was used to assist with drafting, organization, and explanatory text in the abstract, Sections I–XIV, and Worked Examples 1–30, based on the project materials and preparation instructions. It was also used for research assistance on the added SSL historical context, reference-format conversion, and LaTeX/PDF preparation.

References

[1] T. Elgamal, "The Secure Sockets Layer Protocol (SSL)," presentation at the Danvers IETF Meeting, Apr. 1995. Historical presentation; sections on authentication, record integrity, and other Internet-security issues. https://www.ietf.org/proceedings/32/sec/cat.elgamal.slides.html

[2] T. Elgamal and K. E. B. Hickman, "The SSL Protocol," Internet-Draft draft-hickman-netscape-ssl-00, Apr. 1995, expired draft. Author attribution follows the IETF Datatracker archival record. https://datatracker.ietf.org/doc/draft-hickman-netscape-ssl/00/

[3] P. Kocher, "Technical Projects," SSL/TLS section, personal technical account, accessed Nov. 9, 2026. https://paulkocher.com/technicalProjects.html

[4] A. Freier, P. Karlton, and P. Kocher, "The Secure Sockets Layer (SSL) Protocol Version 3.0," RFC 6101, Aug. 2011, doi: 10.17487/RFC6101. Historic republication of the Nov. 18, 1996 specification; Foreword and Appendix G.1. https://www.rfc-editor.org/rfc/rfc6101.html

[5] Rosario, Frank Dylan. *HTTP6: Problem Statement and Architecture for Hereditary Secure Application Channels.* Local individual Internet-Draft source candidate, November 9, 2026. Supplied file: HTTP6_IETF_Problem_Statement_Candidate.md. Particularly §§3–10 and 12.

[6] IETF. RFC 6750, *The OAuth 2.0 Authorization Framework: Bearer Token Usage*. Abstract and §1.2. https://www.rfc-editor.org/rfc/rfc6750.html

[7] IETF. RFC 9449, *OAuth 2.0 Demonstrating Proof of Possession (DPoP)*. Particularly §§1, 11.4, and 11.7. https://www.rfc-editor.org/rfc/rfc9449.html

[8] IETF. RFC 9700, *Best Current Practice for OAuth 2.0 Security*. Sender-constrained access-token recommendations. https://www.rfc-editor.org/rfc/rfc9700.html

[9] Rosario, Frank Dylan. *HTTP6: Human-Originated Authority for NIST-Aligned Cybersecurity.* Expanded Public Review Edition 2.0, November 9, 2026. Supplied file: HTTP6_NIST_Expanded_Public_White_Paper.md. Particularly §§3.3, 4–5, 11 and Appendix B.

[10] NIST. SP 800-63B-4, §5, session management, session bindings, access tokens, and reauthentication. https://pages.nist.gov/800-63-4/sp800-63b/session/

[11] Rosario, Frank Dylan. *Human-Originated Identity and Proof-Carrying Process Authority.* November 8, 2026. Supplied file: White_Paper_Human_Originated_Identity.md. Particularly §§3–4 and 6–7.

[12] IETF. RFC 9334, *Remote ATtestation procedureS (RATS) Architecture*. Evidence, appraisal, relying-party roles, and nonce freshness. https://www.rfc-editor.org/rfc/rfc9334.html

[13] NIST. SP 800-63B-4, *Digital Identity Guidelines: Authentication and Authenticator Management*. §3.2.5, phishing resistance and channel/verifier-name binding. https://pages.nist.gov/800-63-4/sp800-63b/authenticators/

[14] IETF. RFC 9421, *HTTP Message Signatures*. Abstract and component-coverage provisions. https://www.rfc-editor.org/rfc/rfc9421.html

[15] NIST. SP 800-63-4, *Digital Identity Guidelines*. Digital identity, pseudonymity, and authentication-factor distinctions. https://pages.nist.gov/800-63-4/sp800-63.html

[16] NIST. SP 800-63B-4, authentication assurance levels, particularly §§2.2–2.3. https://pages.nist.gov/800-63-4/sp800-63b/aal/

[17] IETF. RFC 5869, *HMAC-based Extract-and-Expand Key Derivation Function (HKDF)*. https://www.rfc-editor.org/rfc/rfc5869.html

[18] IETF. RFC 9846, *The Transport Layer Security (TLS) Protocol Version 1.3*. July 2026. Abstract, protocol overview, authentication and record-protection provisions. https://www.rfc-editor.org/info/rfc9846/

[19] IETF. RFC 7250, *Using Raw Public Keys in Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)*. https://www.rfc-editor.org/rfc/rfc7250.html

[20] IETF. RFC 9001, *Using TLS to Secure QUIC*. https://www.rfc-editor.org/rfc/rfc9001.html

[21] W3C. *Web Authentication: An API for Accessing Public Key Credentials — Level 3*. Relying-party identifiers and assertion verification. https://www.w3.org/TR/webauthn-3/

[22] IETF. RFC 9110, *HTTP Semantics*. https://www.rfc-editor.org/rfc/rfc9110.html

[23] NIST. SP 800-207, *Zero Trust Architecture*. 2020. https://csrc.nist.gov/pubs/sp/800/207/final