Authentication: Credential, Proof, Result, and Time

CoreBond Research Series — CBRS-004
August 2026
Todd Kirton
DOI: 10.5281/zenodo.22029244

What authentication proves now, what it does not prove forever, and when a system should ask again

Abstract

Authentication is often described through the thing being possessed: a password, private key, certificate, token, or authenticator. That shorthand is useful, but it can hide the actual transaction. A private key can remain security-critical for years without constituting a current authentication event. Authentication occurs when evidence is presented under a protocol and a verifier determines that the authentication method's requirements have been satisfied.

This paper separates three related objects: the durable credential or registered relationship, the proof presented for an authentication transaction, and the authentication result produced by verification. Modern protocols already implement this distinction. WebAuthn verifies an assertion in the context of a relying-party challenge. TLS 1.3 binds CertificateVerify to the current handshake transcript. EDHOC binds authentication credentials to transcript state and ephemeral key exchange for constrained environments.[1][2][3] This is not a claim that standards designers overlooked verification. The problem is that system design and trust language can blur what persists, what was proved in a transaction, and what later decisions inherit from that result.

Time exposes the consequence. The historical fact that authentication succeeded does not become false merely because conditions later change. What can age is the system's willingness to let later activity continue to rely on that earlier result. NIST session guidance addresses reauthentication and session monitoring in human digital identity, while the OpenID Continuous Access Evaluation Profile addresses post-authentication security-state changes that can alter continued access.[4][5]

The answer is not to pour every available signal into authentication. Authentication, attestation, authorization, session continuity, and continuous access evaluation answer related but distinct questions. The useful design problem is deciding what evidence must be current enough for authentication, what belongs in separate trust decisions, and what later change is serious enough to make the system ask again.

1. The Key Is Not the Proof

A private key can exist for ten years without authenticating anything by itself.

That does not make the key unimportant. Its custody, secrecy, revocation status, and integrity can determine whether future authentication is trustworthy. But a verifier cannot infer current control merely from the key's existence. The claimant has to produce whatever evidence the authentication method requires, and the verifier has to check it.

WebAuthn makes the sequence visible. A relying party registers a public-key credential, supplies a challenge during authentication, receives an assertion, and performs defined verification steps.[1] TLS 1.3 uses CertificateVerify to demonstrate possession of the private key corresponding to a certificate within the current handshake transcript.[2] EDHOC performs credential-backed authentication within a transcript that also incorporates ephemeral key-exchange state.[3]

None of this is new cryptography. That concession matters. Challenge-response and authenticated key exchange have long separated long-lived key material from protocol messages and verification. The useful distinction is architectural: systems often talk about a credential, an authentication event, and an ongoing authenticated state as if those were one thing.

They are not. A credential or registered relationship can persist. A proof is presented under the rules of an authentication method. The authentication result is the verifier's decision, or the authentication state established by successful verification. A later token, session, or channel may represent, reference, or inherit that result, but it is not automatically identical to the result itself.

2. What Did We Actually Prove?

Once those objects are separated, the next question is narrower and more useful: what claim did the authentication method actually establish?

In WebAuthn, successful verification can establish control of the private key associated with the registered credential under the conditions of the ceremony.[1] It does not automatically establish that the endpoint is uncompromised, that the user should be authorized for every resource, or that the same conditions will remain true later.

TLS 1.3 shows the same discipline in machine-oriented protocol design. CertificateVerify is bound to the handshake transcript rather than functioning as a reusable declaration that the endpoint is authenticated forever.[2] EDHOC similarly binds credential-backed authentication to the protocol run in which it occurs.[3]

The point is not that possession is weak. In many authentication methods, proof of control is exactly the property being tested. The point is that the verifier accepts evidence sufficient to satisfy a defined authentication method. Whether that result is also sufficient for a particular resource, operation, device posture, or later point in time is a different question.

That wording matters. Authentication should not quietly absorb authorization simply because both decisions consume security evidence. A strong proof can satisfy authentication and still be insufficient for access under current policy.

3. Freshness Changes the Proof

Freshness makes the separation between credential and proof difficult to ignore.

A long-lived key can remain acceptable while an old authentication message is useless. The credential did not fail. The old evidence simply does not satisfy the current protocol run. WebAuthn uses relying-party challenges, while TLS and EDHOC bind authentication to the current exchange transcript.[1][2][3] These mechanisms prevent an earlier valid proof from automatically answering a later authentication request.

Not every authentication system implements freshness in the same way, and not every event requires a newly generated nonce. The general requirement is narrower: where an authentication method depends on current or transaction-bound evidence, the verifier must establish that the evidence is acceptable for this protocol context rather than merely valid in the abstract.

RATS provides a parallel lesson in attestation. RFC 9334 separates Evidence, Verifier appraisal, Attestation Results, and Relying Party use, and its architecture treats freshness as relevant to evidence appraisal.[6] That does not make attestation authentication. It shows why cryptographic validity and current decision relevance are separate questions.

Credential validity asks whether the underlying key, registration, certificate, or trust relationship remains acceptable. Proof validity asks whether the evidence satisfies the current authentication procedure. The authentication result is the conclusion produced when those required checks succeed. In challenge- and

transcript-bound protocols, freshness does not replace possession; it makes proof of possession meaningful for this exchange.

4. Attestation Makes the Boundary Visible

Remote attestation is useful here because it names stages that security discussions often blur.

The IETF RATS architecture separates Evidence, Verifier appraisal, an Attestation Result, and the Relying Party's later use of that result.[6] RATS is not evidence for a new definition of authentication, and its model should not be projected wholesale onto authentication. Its value is comparative: it demonstrates how much clarity is gained when evidence, evaluation, result, and relying-party action are named separately.

The same discipline helps authentication. A credential is not the verifier's decision. A signature is not the verifier's decision. The verifier performs the checks required by the authentication method and establishes an authentication result. Other evidence may matter without belonging inside that decision.

Platform measurements, firmware state, provenance, boot integrity, or device compliance may be crucial. They can belong to attestation or authorization. If every useful trust signal is relabeled authentication, the word stops identifying which question the system actually answered.

5. How Long Should the Result Be Inherited?

A successful authentication remains a historical fact. What can expire is the system's willingness to keep relying on that result.

Most systems do not repeat a full authentication ceremony before every request. They establish a session, token, channel, or comparable state that lets later activity inherit the earlier result. That is efficient, but it creates distance between the time of proof and the time of later use.

NIST's current human digital-identity guidance explicitly separates an authentication event from the session that may follow it, defines reauthentication requirements, and permits session monitoring as a risk-reduction measure.[4] It also identifies session hijacking as a post-authentication threat. The earlier authentication does not become false; later reliance on it can become insufficient.

Consider a simple example. A device authenticates successfully at 08:00. At 08:30, the session is stolen or the device's compliance state changes. Nothing about the 08:00 verification has to be rewritten. The question at 08:31 is whether continued access should still inherit the 08:00 result.

NIST SP 800-63-4 explicitly does not directly specify machine-to-machine or IoT authentication requirements.[7] TLS and EDHOC therefore remain the stronger primary evidence for machine authentication in this paper. The cross-domain point is an architectural inference: whenever later activity inherits an earlier authentication result, the system needs some rule for when that inheritance is no longer sufficient.

There is no universal timer. The acceptable interval depends on threat model, consequence, protocol, environment, and the cost of asking again.

6. Asking Again Is Not the Only Option

When reliance on an earlier result is no longer sufficient, reauthentication is one response. It is not the only response.

Reauthentication asks for new authentication evidence and establishes the required assurance again. That is appropriate when current control of the authenticator is the question or when policy requires a new ceremony.

Other changes belong to continued-access decisions. The OpenID Continuous Access Evaluation Profile defines security events such as session revocation, credential change, assurance-level change, and device-compliance

change.[5] The Shared Signals Framework defines how security signals and events are shared between cooperating peers.[8]

CAEP does not prove that the earlier authentication became less true. It shows that later security state can change what access should continue. A receiver may terminate a session, demand reauthentication, reduce access, or take another policy action while leaving the historical authentication event untouched.

That distinction is why 'continuous authentication' can be a misleading umbrella term. NIST discusses session monitoring in human digital identity,[4] but CAEP is better understood as continuous or event-driven access evaluation. Sometimes the right question is 'Who are you again?' Sometimes it is 'Given what changed, should this access continue?'

7. More Evaluation Is Not Automatically Better

Describing authentication as evaluation creates an obvious temptation: evaluate everything available.

Behavior, location, network characteristics, device posture, usage history, firmware state, browser traits, risk scores, and threat intelligence can all provide useful security information. They can also create new failure and governance surfaces.

NIST's session-monitoring guidance identifies usage patterns, behavioral biometrics, device or browser characteristics, geolocation, and IP-address characteristics as possible session signals in human digital identity, and requires the associated collection and processing to be included in a privacy risk assessment.[4] That evidence supports a privacy warning, not a universal claim that contextual systems fail more often.

The broader architectural countercase is straightforward. Additional telemetry can be missing, delayed, manipulated, or poorly calibrated. A new signal can become a new availability dependency. A complex policy can become harder to explain after an incident. Correlated data sources can fail together. Silent monitoring can also collect far more information than a bounded proof-of-control ceremony requires.

Narrow cryptographic authentication has a virtue that should not be dismissed as old-fashioned: the verification question can be explicit. WebAuthn, TLS, and EDHOC define concrete checks that can be tested, implemented across vendors, and audited.[1][2][3] Context can add value, but value has to be weighed against privacy, reliability, governance, and explainability costs.

> **More evidence is not automatically better authentication. Evidence earns its place by being relevant to the decision, sufficiently reliable, and worth the failure surface it adds.**

8. Keep the Decisions Separate

The cleanest defense against conceptual collapse is to make each layer own a question.

Authentication asks whether the claimant or peer satisfied the authentication method. Attestation asks what evidence supports about properties, state, composition, provenance, or trustworthiness. Authorization asks what the recognized actor may do under current policy. Session continuity asks whether an existing session should continue to inherit an earlier authentication result. Continuous access evaluation asks whether later security-state changes require access to change.

These layers can share evidence and influence one another without becoming the same decision. A device can fail attestation while still controlling the expected private key. A role can be removed without making the earlier authentication invalid. A session can be revoked without rewriting the history of how it began. A credential compromise, by contrast, can directly damage the basis for future authentication.

The distinction is operational, not merely semantic. It tells defenders what failed, which state has to change, and whether the proper response is credential revocation, reauthentication, session termination, attestation remediation, or authorization change.

9. Authentication as a Bounded Decision Process

The phrase "authentication is evaluation" is useful only if evaluation remains bounded.

It does not mean authentication should become a universal risk engine. It does not make possession-based cryptography obsolete. It does not merge attestation, authorization, behavior, and session monitoring into one trust score.

It means authentication concludes when a verifier applies the authentication method's rules to the evidence presented and establishes an authentication result. The useful architecture is therefore: durable credential or registered relationship; proof acceptable for the current authentication context; verifier checks; authentication result; then later or separate trust decisions.

Standards already implement these pieces. WebAuthn, TLS, and EDHOC provide transaction-bound verification. RATS makes evidence appraisal and relying-party use explicit in attestation. NIST distinguishes authentication from sessions and reauthentication in human digital identity. CAEP carries post-authentication state changes into continued-access decisions.[1][2][3][4][5][6]

The paper's claim is not that those standards misunderstood authentication. It is that architecture becomes easier to reason about when the persistent credential, the transaction proof, the authentication result, and the later decisions that inherit or consume that result are not allowed to collapse into the single label 'authenticated.'

The useful question is not possession versus evaluation. It is which evidence belongs in authentication, which belongs elsewhere, and when the system should ask again.

10. What Deserves to Be Discovered

Once the boundaries are visible, the unresolved questions matter more than the slogan.

What evidence must be current enough at authentication? Which facts are appropriately represented by durable credentials, and which should be demonstrated again? How should an authentication result express its limits so that later systems do not silently assume more than was proved?

Which post-authentication changes should force a new authentication ceremony? Which should instead trigger attestation, authorization reevaluation, session termination, or another response? How should machine systems answer those questions when human session guidance does not directly apply?

How much context can be added before the decision becomes too opaque to test, audit, or explain? When does background monitoring create more governance and availability risk than the security value it contributes? When is an explicit challenge the cleaner answer?

Authentication is easier to trust when a system can say what claim was proved, under what protocol conditions, which later decisions are inheriting that result, and what event would make the system ask again.

The difficult question is not whether possession matters. It is how long later activity should be allowed to rely on a verified proof, what new evidence should change access, and when the system should stop inheriting yesterday's authentication decision.

Research Agenda

- What evidence belongs inside authentication rather than attestation or authorization?
- How should machine systems define when reliance on an authentication result is no longer sufficient?

- Which state changes should force explicit reauthentication rather than access reevaluation?
- How can continuous or event-driven signals be used without producing opaque or privacy-invasive trust scoring?
- What interfaces best preserve the distinction between credential state, proof validity, authentication result, and continued access?

References

1. World Wide Web Consortium. (2026, May 26). Web Authentication: An API for Accessing Public Key Credentials — Level 3 (Candidate Recommendation Snapshot).

2. Rescorla, E. (2018). The Transport Layer Security (TLS) Protocol Version 1.3 (RFC 8446). Internet Engineering Task Force.

3. Selander, G., Preuß Mattsson, J., & Palombini, F. (2024). Ephemeral Diffie-Hellman Over COSE (EDHOC) (RFC 9528). Internet Engineering Task Force.

4. National Institute of Standards and Technology. (2025). Digital Identity Guidelines: Authentication and Authenticator Management (NIST SP 800-63B-4). U.S. Department of Commerce.

5. Cappalli, T., & Tulshibagwale, A. (2025, August 29). OpenID Continuous Access Evaluation Profile 1.0. OpenID Foundation.

6. Birkholz, H., Thaler, D., Richardson, M., Smith, N., & Pan, W. (2023). Remote ATtestation procedureS (RATS) Architecture (RFC 9334). Internet Engineering Task Force.

7. National Institute of Standards and Technology. (2025). Digital Identity Guidelines (NIST SP 800-63-4). U.S. Department of Commerce.

8. Tulshibagwale, A., Cappalli, T., Scurtescu, M., Backman, A., Bradley, J., & Miel, S. (2025, August 29). OpenID Shared Signals Framework Specification 1.0. OpenID Foundation.