International Journal of Emerging Trends in Computer Science and Information Technology
ISSN: 3050-9246 | https://doi.org/10.63282/3050-9246.IJETCSIT-V7I2P118
Eureka Vision Publication | Volume 7, Issue 2, 134-143, 2026

Original Article

Accountable Disclosure of Sensitive Data in Team Chat: The SealedChat System

Sudheer Avula
Independent Researcher, Provo, Utah, USA.

Received On: 04/03/2026 ***Revised On: 05/04/2026*** ***Accepted On: 12/04/2026*** ***Published On: 20/04/2026***

Abstract - Collaboration platforms such as Slack, Microsoft Teams, and Discord are widely used for operational coordination, where sharing sensitive data such as credentials, tokens, personally identifiable information (PII), and protected health information (PHI) is sometimes unavoidable. In public channels, such data becomes persistently and passively exposed through message history, search, notification previews, and screen sharing. Existing approaches, including data loss prevention (DLP) systems, focus on blocking, detecting, or removing content, but do not provide a mechanism for controlled, accountable disclosure when sharing is operationally necessary. We introduce Accountable Disclosure, a security model that replaces ambient visibility with intentional, auditable access to sensitive data in team chat. In this model, sensitive content is sealed from the channel and revealed only through explicit user action that requires justification and produces a visible accountability event. Disclosure is constrained by time-bounded viewing, shifting the security objective from strict prevention to reduction of passive exposure and deterrence through auditability. We present SealedChat, a system design that realizes this model across heterogeneous collaboration platforms. SealedChat stores sensitive payloads outside platform message histories and enforces disclosure through a secure view mechanism with strict temporal bounds. We analyze how accountable-disclosure semantics can be realized across Slack, Microsoft Teams, and Discord despite differences in platform messaging, interaction, and visibility primitives. Finally, we outline an evaluation methodology for measuring passive exposure reduction, usability costs, and accountability effects in realistic operational workflows, while identifying the platform-specific constraints that affect faithful realization of the model. Our approach demonstrates how accountability-driven disclosure can complement existing preventive controls in collaborative systems.

Keywords - Accountable Disclosure, Team Chat Security, Sensitive Data Sharing, Auditability, Break-Glass Access, Usable Security, Collaboration Platforms.

1. Introduction

Team chat platforms such as Slack, Microsoft Teams, and Discord have become core operational infrastructure for engineering, customer support, incident response, and clinical coordination. In these settings, users sometimes need to share exposure-sensitive content such as credentials, tokens, personally identifiable information (PII), or protected health information (PHI) in order to complete time-critical work. In this paper, we use sensitive data to refer to operationally necessary but exposure-sensitive content such as credentials, tokens, PII, and PHI. When such information is posted in plaintext to a public channel, it becomes broadly and persistently visible to channel members, increasing the risk of passive exposure through message history, search, notification previews, screen sharing, and later rediscovery.

Existing controls do not address this problem well. Enterprise deployments increasingly rely on data loss prevention (DLP), policy enforcement, and retrospective remediation to detect and block sensitive content in collaboration platforms. These approaches are valuable, but they are fundamentally oriented toward prevention, warning, deletion, or cleanup. They do not provide a usable mechanism for cases in which disclosure is operationally necessary but should not be ambiently visible to everyone in the channel. Similarly, moving the content to direct messages or external tools may reduce immediate public exposure, but it removes the interaction from the shared channel context and does not inherently provide justification, time-bounded viewing, or transparent accountability for access.

We argue that public team chat lacks an appropriate disclosure primitive for sensitive data. We call this primitive Accountable Disclosure: a model in which sensitive content is not stored as plaintext in the shared channel, but may be revealed through explicit user action that is justified, time-bounded, and auditable. The goal of Accountable Disclosure is not to prevent a determined insider from exfiltrating data after reveal. Rather, it is to transform disclosure from ambient visibility into intentional access, thereby reducing passive exposure surfaces while increasing deterrence and after-the-fact reviewability.

To realize this model, we present SealedChat, a system design for accountable disclosure of sensitive data in public team chat. In SealedChat, the channel contains only a sealed placeholder, while the sensitive payload is stored externally as an encrypted object. A channel member may request reveal by explicitly invoking access, selecting a justification category, and opening a secure view with strict temporal bounds. Each reveal and re-reveal generates an

accountability event, making access visible and reviewable without exposing the sensitive content itself in the channel. SealedChat is designed to preserve these semantics across heterogeneous collaboration platforms, including Slack ephemeral responses, Teams task modules, and Discord interaction-scoped ephemeral replies.

This paper makes three contributions. First, we define Accountable Disclosure as a security model for handling sensitive data in public team chat, centered on sealed content, explicit reveal, mandatory justification, time-bounded viewing, and auditable access. Second, we present the SealedChat system design, which instantiates this model and identifies how it can be realized across heterogeneous collaboration platforms while keeping sensitive payloads out of platform message histories. Third, we outline an evaluation methodology for measuring passive exposure reduction, usability costs, and accountability effects in realistic operational workflows, while identifying the platform-specific constraints that affect faithful realization of the model.

2. Background and Motivation

2.1. Passive exposure in team chat

Public team chat changes the security properties of sensitive communication. In many operational environments, users occasionally need to share exposure-sensitive content such as credentials, tokens, PII, or PHI in order to complete urgent work. When such content is posted as plaintext in a shared channel, it becomes not only visible to current participants, but also persistently available through the interaction model of chat itself.

This creates a form of passive exposure that differs from traditional unauthorized access. Sensitive content may be observed or rediscovered without any explicit access request through several ordinary platform features: message history and scrollback, in-client search, desktop and mobile notification previews, screen sharing during meetings or troubleshooting, and secondary propagation through quoting, forwarding, or copy-paste. The risk is therefore not limited to malicious intrusion; it also arises from ambient visibility in a durable, shared workspace.

This distinction motivates our focus. The problem is not simply that sensitive data may appear in chat, but that public-channel plaintext transforms disclosure into a persistent, low-friction visibility event. A useful security mechanism for team chat must therefore reduce these passive exposure surfaces while preserving the operational context in which disclosure occurs.

2.2. Accountability and break-glass as a design lineage

Our approach is grounded in a long-standing security principle: in some settings, exceptional access cannot be fully prevented in advance, but it can be constrained through justification, auditability, and after-the-fact review. This logic is most visible in break-glass mechanisms, particularly in healthcare and other regulated environments, where users may override ordinary restrictions in urgent circumstances but must do so in a way that is explicitly logged and reviewable [6]-[10]. Such mechanisms do not eliminate misuse; rather, they seek to balance operational necessity with deterrence and accountability.

SealedChat adopts this lineage but applies it to a different object of control: not exceptional access to a record in a protected database, but exceptional disclosure of sensitive data within a shared conversational medium. This shift matters. In team chat, the main risk is often not that access is technically impossible to control, but that posting plaintext makes disclosure ambient, durable, and socially invisible. By requiring explicit reveal, justification, and visible accountability events, Accountable Disclosure reframes access to sensitive content as an intentional act rather than a background property of channel membership.

This perspective also clarifies the paper's security objective. SealedChat is not designed to guarantee secrecy after intentional reveal by a determined insider. Instead, it aims to reduce passive exposure and strengthen deterrence and reviewability through accountable, auditable disclosure events. In this sense, the paper is best understood as extending accountability-oriented security controls into collaborative communication systems [1]-[5].

2.3. Heterogeneous platform primitives and semantic preservation

A practical accountable-disclosure mechanism must operate within real collaboration platforms, but those platforms offer substantially different interaction and visibility primitives. Slack supports ephemeral user-visible responses in channels, but they are non-persistent and tied to user interaction patterns [13], [14]. Microsoft Teams provides modal interaction surfaces such as task modules that are better suited to controlled reveal workflows [16]. Discord supports ephemeral replies in interaction-scoped contexts, with different constraints on when and how private responses can be issued [18], [19].

These differences are not merely implementation details. They shape whether the same disclosure semantics can be realized faithfully across platforms. If a design relies too heavily on platform-native private messaging behavior, then properties such as justification-gated reveal, time-bounded viewing, and channel-visible accountability may vary across deployments. This motivates a platform-independent design in which the collaboration platform carries the shared placeholder and accountability signals, while a secure view mechanism enforces the sensitive-data disclosure semantics.

Accordingly, SealedChat is designed not as a cross-platform integration layer, but as a common disclosure model realized through platform-specific adapters. The research challenge is to preserve the semantics of Accountable Disclosure across heterogeneous collaboration environments, despite differences in message visibility, interaction flow, and private-response capabilities.

3. Problem Statement and Design Requirements

Public team chat creates a difficult trade-off for sensitive data handling. In operational settings, users may need to share credentials, tokens, PII, or PHI in order to complete urgent work in a shared channel context. Posting such data in plaintext creates persistent passive exposure, while blocking all disclosure can obstruct legitimate workflows. We therefore address the following problem: how can a team chat system support operationally necessary disclosure of sensitive data in public channels while reducing passive exposure and ensuring accountable access?

To address this problem, we derive the following design requirements for Accountable Disclosure in team chat.

- R1. Sender-initiated sealing. A sender must be able to mark sensitive content for sealed handling before it becomes ambiently visible in the channel. The system should treat the sealed content as an opaque payload rather than ordinary channel text.
- R2. Self-service reveal in channel context. Any channel member should be able to initiate reveal through explicit action without requiring the sender to remain continuously involved. Reveal should remain anchored in the shared channel context rather than forcing migration to ad hoc side channels or unrelated tools.
- R3. Mandatory justification. Each reveal request must require an explicit justification selection. This creates structured evidence for review and reinforces that disclosure is an accountable action rather than frictionless reading.
- R4. Visible accountability for access events. Each reveal and re-reveal should generate a channel-visible accountability event identifying who accessed the sealed content, while not exposing the sensitive payload itself. This supports deterrence, transparency, and after-the-fact review.
- R5. Time-bounded viewing. Revealed content should be accessible only through a time-bounded view rather than persistent plaintext presentation in the channel. Once the viewing interval expires, access should terminate and a new reveal request should be required.
- R6. Operational usability. The mechanism must remain usable in realistic workflows such as incident response, support, and coordination under time pressure. It must reduce passive exposure without making legitimate disclosure operationally impractical.
- Non-goals. SealedChat does not aim to prevent exfiltration by a determined insider once content has been intentionally revealed. Screenshots, manual transcription, copy-based reuse, or reposting remain possible after access is granted. The goal is instead to reduce passive exposure, replace ambient visibility with intentional access, and create auditable disclosure events that support deterrence and reviewability.

4. Accountable Disclosure Model

We model Accountable Disclosure as a mechanism for handling sensitive data in shared team chat without making that data ambiently visible in the channel. The central idea is to separate channel visibility from data disclosure. Rather than storing sensitive content as ordinary channel text, the channel contains only a sealed reference, while access to the underlying payload occurs through explicit reveal actions that are justified, time-bounded, and auditable.

This model is intended for settings in which disclosure may be operationally necessary, but should not be frictionless or passively persistent. Accordingly, Accountable Disclosure does not attempt to eliminate all downstream misuse after reveal. Instead, it transforms access to sensitive content from a background property of channel membership into an intentional event subject to accountability and review.

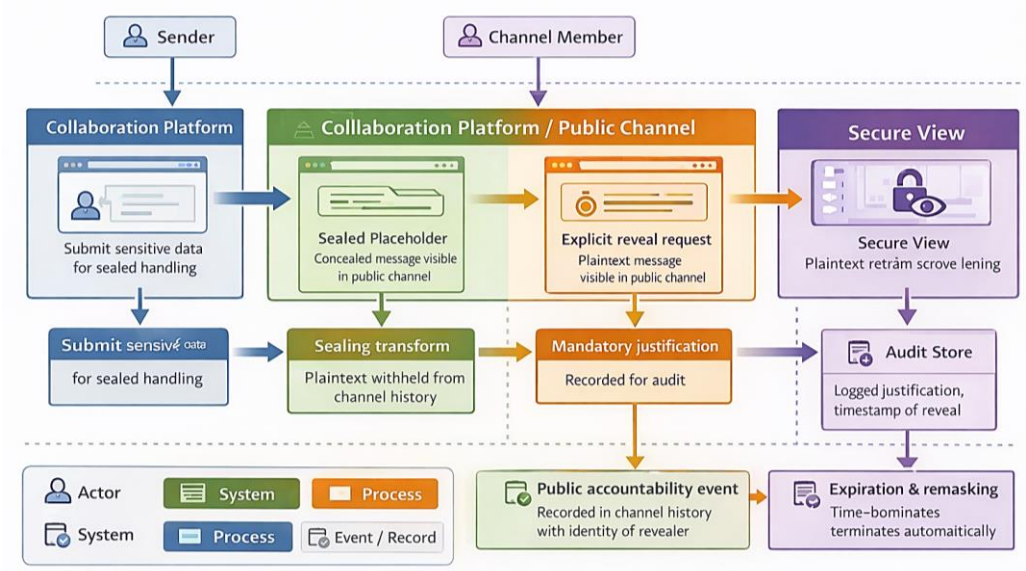

Figure 1. Accountable Disclosure Model: Actors, System Boundaries, Processes, and Governance Outputs

A sender initiates sealed handling of sensitive data before it becomes plaintext in the public channel. Any channel member may later request reveal, but disclosure requires mandatory justification, produces public accountability and private audit records, and occurs only through a secure view with time-bounded viewing.

4.1. Actors and objects

The model includes four actors: a sender who creates a sealed message containing sensitive data; a channel member who may request reveal of sealed content; a disclosure service responsible for storing sealed payloads, issuing reveal views, and recording audit events; and a collaboration platform that carries the sealed placeholder and public accountability signals. The model also includes the following objects: a sealed message, which is a channel-visible placeholder representing sensitive content without exposing the plaintext payload; a sensitive payload, which is the protected content associated with a sealed message; a justification record supplied for a reveal request; a public accountability event indicating that a reveal occurred; and a private audit record containing metadata such as requester identity, timestamp, and justification.

4.2. Disclosure lifecycle

A sealed message progresses through a lifecycle governed by disclosure actions rather than ordinary message reading. Sealed. The sender submits sensitive content for sealed handling. The plaintext payload is stored outside ordinary message history, and the channel displays only a sealed placeholder. Reveal Requested. A channel member explicitly requests access to the sealed content. At this stage, the payload is not yet disclosed. Transition to reveal requires submission of a mandatory justification, which is recorded as part of the private audit trail. Revealed. Following a justified request, the disclosure service grants access to the sensitive payload through a secure view. Access is not persistent in the channel and is limited to a defined viewing interval. Expired. After the viewing interval ends, access terminates and the payload is no longer available through the secure view. Further access requires a new reveal request and a new accountability event. Repeated access after expiration is treated as a new disclosure event rather than continuation of the prior one. The requester must again provide justification, and the system records a new accountability event.

4.3. Observable events and visibility boundaries

The model distinguishes between publicly visible events and privately recorded events. Publicly visible events include creation of the sealed placeholder in the channel, accountability events indicating that a reveal or re-reveal occurred, and the identity of the user who performed the reveal. Privately recorded events include the sensitive payload itself, the requester's justification, and internal audit metadata associated with the reveal. This distinction is important for regulated or privacy-sensitive settings. The model seeks to make access behavior visible without reproducing sensitive content or sensitive contextual detail in the public channel.

4.4. Model properties

- P1. No plaintext persistence in channel history. Sensitive payloads are not stored as ordinary channel messages and therefore are not ambiently recoverable through standard chat history mechanisms.
- P2. Explicit access initiation. Disclosure requires a deliberate reveal action by a channel member rather than passive observation through channel membership alone.
- P3. Justification-bearing access. Each reveal is associated with a mandatory justification record that can support later review.
- P4. Auditable disclosure. Each reveal and re-reveal generates an accountability event and a corresponding audit record.
- P5. Time-bounded viewing. Disclosure is limited to a finite viewing interval rather than becoming persistent plaintext within the channel context.
- P6. Re-access requires renewed accountability. Subsequent access after expiration is treated as a new event, requiring a new justification and generating a new accountability signal.

5. Threat Model

SealedChat is designed to reduce exposure of sensitive data in public team chat by replacing ambient visibility with explicit, accountable reveal. Its threat model therefore focuses on passive exposure and misuse within shared-channel environments, rather than on preventing all forms of post-disclosure leakage.

5.1. Adversaries

We consider four main classes of adversaries. Accidental viewers. These are users who were not the intended audience for the sensitive content but may observe it because it appears in plaintext in a shared channel. Exposure may occur through ordinary platform features such as scrollback history, in-client search, notification previews, screen sharing, or later rediscovery in the channel archive. Curious insiders. These are legitimate channel members who do not have a clear operational need for the sensitive data but may choose to reveal it out of curiosity or convenience. Because SealedChat permits any channel member to request reveal, this class is central to the design. The system does not attempt to prevent reveal by channel members through fine-grained access restrictions; instead, it seeks to deter and surface misuse through mandatory justification, visible accountability events, and private audit records.

Opportunistic misusers. These are channel members who repeatedly reveal sensitive content to harvest information or exploit low-friction access. SealedChat seeks to reduce this behavior by ensuring that each reveal and re-reveal is explicit, justified, time-bounded, and independently logged. Determined exfiltrators. These are users who intentionally extract sensitive data after it has been revealed, for example through screenshots, manual transcription, external reposting, or downstream forwarding. SealedChat

does not prevent this class of attack once plaintext has been intentionally disclosed to the requester.

5.2. Security goals

- G1. Reduce passive exposure. Sensitive payloads should not remain ambiently visible in channel history or ordinary channel rendering.
- G2. Require intentional access. Access to sensitive content should occur only through explicit reveal actions rather than passive reading by virtue of channel membership alone.
- G3. Make access accountable. Each reveal should produce evidence that access occurred, including a visible accountability signal and a private audit record.
- G4. Bound disclosure duration. Access to plaintext should be limited to a time-bounded viewing interval rather than persistent presentation in the shared channel.
- G5. Make repeated access observable. Re-reveal should be treated as a new disclosure event so that repeated access is visible and auditable rather than silently persistent.

5.3. Non-goals

SealedChat does not aim to provide end-to-end confidentiality once a reveal has been intentionally granted. In particular, it does not prevent screenshots, camera capture, manual transcription, reposting, or copy-based reuse after disclosure. It also does not defend against compromise of the requester's endpoint or local operating environment; if the endpoint is compromised, plaintext visible during an authorized reveal may be captured independently of the collaboration platform or disclosure service. It also does not attempt to enforce fine-grained authorization policies among channel members beyond the baseline assumption of channel membership. Its primary purpose is to reduce passive exposure and to govern intentional access through justification, visibility, and auditability.

5.4. Trust assumptions

SealedChat relies on several trust assumptions. The collaboration platform is assumed to authenticate users correctly and to provide a reliable identity for the requesting channel member. The SealedChat service is trusted to store sensitive payloads securely, issue secure views correctly, enforce time-bounded viewing, and maintain audit records with integrity. The system assumes that channel membership as reported by the collaboration platform is the relevant eligibility boundary for reveal requests. Public accountability events and private audit records are assumed to be recorded and retained in a manner suitable for later review. These assumptions reflect the intended role of SealedChat as an accountability-oriented disclosure mechanism layered on top of existing collaboration systems, rather than as a replacement for their identity, endpoint, or device-security controls.

6. Sealedchat System Design

SealedChat is a system design for realizing Accountable Disclosure in public team chat. Its core architectural principle is to separate shared conversational context from sensitive payload disclosure. The collaboration platform carries the sealed placeholder and public accountability signals, while a trusted disclosure service stores the sensitive payload, mediates reveal requests, and enforces time-bounded viewing. This separation allows the system to reduce passive exposure in channel history without removing the disclosure workflow from the shared operational context.

6.1. Design overview

At a high level, SealedChat consists of four logical components: a channel-facing interface that presents sealed placeholders and reveal actions in the collaboration platform; a disclosure service that processes reveal requests and issues secure views; a sealed payload store that keeps sensitive content outside platform message history; and an audit subsystem that records both public accountability events and private reveal metadata. The design is guided by the model properties introduced in Section 4. Sensitive data should never appear as ordinary channel plaintext; reveal should require explicit user action and mandatory justification; access should be time-bounded; and repeated access should generate renewed accountability.

6.2. User interaction flow

The sender initiates disclosure by submitting sensitive content for sealed handling rather than posting it as plaintext in the channel. The platform then displays a sealed placeholder that indicates the presence of protected content without exposing the payload itself. Depending on deployment needs, the placeholder may include minimal descriptive metadata such as a general type label or a non-binding intended-recipient hint. When a channel member chooses to access the content, the member explicitly initiates a reveal request. The system requires a mandatory justification selection before granting access. Once justification is supplied, the disclosure service issues a secure view that presents the sensitive payload for a limited interval. After the interval expires, the view is remasked and further access requires a new reveal request. Each reveal and re-reveal generates a corresponding accountability event.

6.3. Data and storage design

Each sealed item is represented by an opaque identifier and an associated encrypted payload stored outside the collaboration platform's normal message history. In addition to the encrypted payload, SealedChat maintains metadata sufficient to support accountability and rendering, including platform and channel references, sender identity, timestamps, optional descriptive type labels, and audit linkage. A representative sealed record includes the following fields: sealed_id, ciphertext, content_type, channel_ref, platform_ref, sender_ref, timestamps, and audit records containing reveal events, requester identity, justification, and related metadata. The design assumes encrypted storage of the payload using a service-side cryptographic mechanism such as envelope encryption. The paper does not depend on a

specific cloud or key-management provider; what matters is that plaintext is kept out of platform message history and disclosed only through the reveal path.

6.4. Secure view and time-bounded viewing

SealedChat enforces time-bounded viewing through a secure view rather than through message edits or delayed deletion inside the collaboration platform. Native platform features for ephemeral or user-specific visibility differ substantially across collaboration systems and may not provide reliable control over persistence, reload behavior, or visibility lifetime. Instead, after a justified reveal request, the disclosure service grants access to a controlled secure view. The secure view presents the plaintext payload for a limited interval and then remasks the content when the interval expires. Once expired, the session does not silently persist access; a new reveal request is required for renewed access. In the current design, the viewing interval is set to 10 seconds. We treat this value as a design parameter rather than a model requirement: the important property is that viewing is explicitly time-bounded and that re-access requires renewed accountability.

6.5. Accountability and audit design

SealedChat distinguishes between public accountability and private audit evidence. Public accountability is realized through a channel-visible event indicating that a reveal occurred and identifying the user who performed it. This event is intentionally minimal and should avoid exposing the sensitive payload or sensitive contextual detail. Its purpose is to make disclosure behavior socially visible within the shared channel. Private audit evidence is maintained by the disclosure service and records the details necessary for later review, including requester identity, time of reveal, justification, and repeated-access history. This separation supports both transparency and confidentiality: channel participants can see that access occurred, while reviewers or administrators can inspect more detailed evidence when necessary.

6.6. Design parameters and trade-offs

Several aspects of SealedChat are deliberate design choices rather than universal requirements. First, the current design allows copy-to-clipboard during the reveal interval. This improves operational utility, especially for short-lived credentials or tokens, but weakens secrecy guarantees after authorized disclosure. SealedChat therefore derives its value primarily from reducing passive exposure and increasing accountability, not from preventing downstream reuse. Second, the current design uses a mandatory justification taxonomy with a small set of categories appropriate to common operational workflows. The exact taxonomy can vary by deployment. What matters at the model level is that justification is required and recorded. Third, SealedChat permits any channel member to request reveal. This is an intentional governance choice: the system relies on explicit action, mandatory justification, visible accountability, and time-bounded viewing rather than on fine-grained pre-authorization among channel members.

7. Realization Approach across Heterogeneous Collaboration Platforms

SealedChat is intended as a common system design for Accountable Disclosure, but its realization depends on the interaction and visibility primitives provided by each collaboration platform. The purpose of this section is not to claim interoperability or a completed multi-platform implementation. Rather, it is to analyze how the same disclosure semantics can be approximated across heterogeneous collaboration environments while preserving the core model properties introduced earlier: sealed channel presence, explicit reveal, mandatory justification, time-bounded viewing through a secure view, and visible accountability.

7.1. Realization goals

Across platforms, the realization approach seeks to preserve the following semantics: the channel should contain only a sealed placeholder, not the plaintext payload; access should require an explicit reveal request by a channel member; reveal should require mandatory justification before disclosure; plaintext should appear only in a secure view with strict temporal bounds; and each reveal and re-reveal should generate a public accountability event without exposing sensitive content. Because collaboration platforms differ in message visibility, modal interaction support, and private-response behavior, these semantics cannot always be implemented using the same native mechanism. This motivates a layered design in which the collaboration platform carries shared context and accountability signals, while the SealedChat service enforces secure-view disclosure and audit behavior outside ordinary message history.

Table 1. Realization of Accountable Disclosure across Heterogeneous Collaboration Platforms

Semantic responsibility	Slack	Microsoft Teams	Discord	Controlled by SealedChat
Reveal initiation	Message action / button	Card action / button	Interaction trigger / button	No
Justification collection	Modal	Task module / dialog	Interaction flow	Partly
Sealed placeholder in channel	Yes	Yes	Yes	Coordinated
Sensitive payload stored outside message history	Not native	Not native	Not native	Yes
User-specific path to disclosure	Ephemeral response / link handoff	Task module or linked view	Ephemeral interaction reply / link handoff	Coordinated

Secure view for plaintext disclosure	Not native	Partly hostable, but service-controlled	Not native	Yes
Time-bounded viewing enforcement	Not native	Not native as a disclosure guarantee	Not native	Yes
Public accountability event	Yes	Yes	Yes	Coordinated
Private audit record with justification	Limited natively	Limited natively	Limited natively	Yes
Re-reveal as a new accountable event	Not native	Not native	Not native	Yes

7.2. Platform-specific realization considerations

Slack realization approach. Slack provides useful primitives for realizing SealedChat, but with important limitations. Channel-visible placeholders can be represented as normal messages with actions, while mandatory justification collection can be handled through modal interactions. Slack also supports ephemeral responses visible only to a single user, but such responses are not guaranteed to persist and are tied to Slack's interaction model [13], [14].

Teams realization approach. Microsoft Teams offers a more structured modal workflow through task modules and related card-based interaction surfaces. This makes Teams a natural fit for justification-gated access and secure-view presentation [16]. In the Teams realization approach, the public channel carries the sealed placeholder and reveal action, while justification and disclosure occur through a task-module-mediated flow. Discord realization approach. Discord supports interaction-scoped ephemeral responses, which are useful for user-specific reveal flows but are more tightly coupled to bot interaction contexts [18], [19]. As a result, realization on Discord depends more explicitly on interaction-driven workflows than on free-form message operations.

7.3. Common pattern across platforms

Although the specific realization differs by platform, the common pattern is consistent. The collaboration platform presents the sealed placeholder and accepts the reveal action. The platform or associated interaction surface collects mandatory justification. The SealedChat service issues a secure view for time-bounded disclosure. The platform receives or displays a public accountability event indicating who revealed the content. This common structure is important because it shows that Accountable Disclosure is not tied to a single vendor feature. Instead, it is a platform-independent model realized through platform-specific adapters and interaction choices.

7.4. Faithful realization and platform-specific constraints

The main research question across platforms is not whether the user interface looks identical, but whether the semantics of Accountable Disclosure remain faithful. In practice, several platform-specific constraints may affect this fidelity: whether user-specific private responses are available and under what triggers; whether modal or structured justification flows are supported; how reliably user identity and channel context can be bound to a reveal request; whether native ephemeral behavior can be trusted, or whether disclosure must be deferred to an external secure view; and how public accountability events can be rendered without exposing additional sensitive context.

These constraints reinforce the need for SealedChat to treat platform-native messaging features as integration surfaces rather than as the sole enforcement mechanism. The secure view, audit logic, and disclosure-state management are therefore centralized in the SealedChat design, while platform adapters are responsible for preserving the surrounding interaction semantics as faithfully as each environment allows.

7.5. Scope of the realization claim

We emphasize that this section presents a realization approach, not a claim of completed feature parity or deployed interoperability across Slack, Teams, and Discord. The contribution is to show how accountable-disclosure semantics can be grounded in real platform capabilities and where platform-specific constraints are likely to affect faithful realization.

8. Evaluation Methodology

We frame evaluation around the central claim of the paper: Accountable Disclosure reduces passive exposure of sensitive data in team chat while preserving sufficient usability for operational workflows. Because SealedChat is currently presented as a system design and realization approach rather than as a completed deployment, our goal here is to define how this claim should be tested in a rigorous and reproducible way.

8.1. Evaluation questions

We propose four guiding evaluation questions. Q1: Does Accountable Disclosure reduce the passive visibility of sensitive data relative to ordinary plaintext sharing in public channels? Q2: What additional time, effort, or friction does sealing and justified reveal introduce into realistic operational tasks? Q3: How do visible accountability events and mandatory justification affect reveal behavior, repeated access, and user willingness to access sealed content? Q4: To what extent can the intended semantics of Accountable Disclosure be faithfully realized across heterogeneous collaboration platforms with different interaction and visibility primitives?

8.2. Baselines

We recommend evaluating SealedChat against at least three baselines. B1: Plaintext channel sharing, in which sensitive content is posted directly into the public channel as ordinary text. B2: Reference-not-paste workflow, in which

users avoid direct posting and instead share a reference to an external system such as a vault, ticket, or secure document. B3: SealedChat accountable disclosure, in which sensitive content is posted as a sealed placeholder and revealed only through explicit, justified, time-bounded access.

8.3. Measures

Passive exposure measures include history visibility, search discoverability, preview visibility, screen-share exposure, and residual visibility duration. Usability measures include task completion time, number of interaction steps, reveal failure or error rate, re-reveal rate as a proxy for whether the viewing interval is too short for practical use, and user-reported friction, confidence, and perceived workflow burden. Accountability and behavior measures include reveal frequency per task, distribution of reveals across users, re-reveal frequency, justification category distribution, workaround behavior such as reposting or moving content to alternate channels, and user-reported deterrence or chilling effect. Platform-specific realization measures include reveal latency, interaction completion success rate, failures caused by platform-specific behavior, differences in justification-flow support, differences in secure-view launch reliability, and differences in accountability-event rendering.

8.4. Study designs

Controlled user study. A controlled study is the most direct first step. Participants perform representative synthetic tasks under different disclosure conditions corresponding to the baselines above. Suitable tasks include sharing a short-lived token during incident triage, sharing a customer-verification snippet containing synthetic PII, and sharing a synthetic PHI-like coordination note. The study would record exposure measures, timing, interaction errors, re-reveals, and post-task perceptions of safety, accountability, and burden. Field pilot. A smaller field pilot may be appropriate once a prototype exists. In such a pilot, a team would use SealedChat in realistic workflows for a limited time, and the study would collect aggregate operational metrics such as number of sealed items, reveal counts, re-reveal counts, justification distributions, and visible accountability-event frequency. Follow-up interviews or surveys could assess acceptability, perceived deterrence, and whether users developed avoidance or workaround practices.

8.5. Synthetic data and ethical considerations

Evaluation should use synthetic or carefully sanitized sensitive content rather than real credentials, real PII, or real PHI. This is especially important because the mechanism intentionally involves disclosure events and user access behavior. If user studies are conducted in institutional or enterprise settings, the protocol may require ethics or IRB-style review depending on local policy, particularly if participant behavior, audit records, or justification rationales are collected for analysis.

8.6. Interpreting outcomes

The key evaluation question is not whether SealedChat makes sensitive disclosure risk-free. Rather, it is whether SealedChat offers a favorable trade-off: substantially less passive exposure than plaintext channel sharing, more in-context usability than external-reference workflows, and meaningful accountability signals without prohibitive operational burden.

9. Discussion

Accountable Disclosure is designed to reduce passive exposure of sensitive data in public team chat without eliminating the operational advantages of shared-channel coordination. This objective necessarily involves trade-offs. SealedChat does not attempt to make sensitive disclosure risk-free; rather, it seeks to replace ambient visibility with intentional, justified, and auditable access.

9.1. Why accountability is central

A central design decision in SealedChat is to permit any channel member to request reveal, while governing that access through mandatory justification, time-bounded viewing, and visible accountability. This reflects the paper's core premise that, in many shared operational settings, the problem is not simply lack of authorization logic but excessive passive visibility. By making reveal an explicit action that produces both a public accountability event and a private audit record, SealedChat shifts disclosure from a background property of channel membership to a visible and reviewable decision.

9.2. Time-bounded viewing as exposure control

Time-bounded viewing is intended to prevent a reveal from turning into effectively persistent access within the chat environment. Once the secure view expires, continued access requires a new reveal request and a new accountability event. This design makes repeated access explicit and measurable. At the same time, the usefulness of time-bounded viewing depends on the task. If the interval is too short, users may experience unnecessary friction or repeatedly re-reveal content, reducing usability and potentially encouraging workarounds. If the interval is too long, the system may recover some of the persistence and low-friction visibility it was designed to avoid.

9.3. Copying, screenshots, and the limits of disclosure control

SealedChat intentionally permits copying from the secure view in the current design. This choice favors operational utility, especially for tokens, credentials, or short verification strings, but it also weakens confidentiality after intentional reveal. More broadly, screenshots, manual transcription, camera capture, or reposting remain possible once plaintext is displayed. These limitations are not accidental gaps in the design; they follow directly from the paper's security objective.

9.4. Chilling effects and user behavior

Visible accountability may deter unnecessary reveals, but it may also discourage legitimate access in situations where users fear being socially scrutinized or second-guessed. This is particularly relevant in high-pressure settings such as incident response or support escalation,

where users may need fast access but may hesitate if every reveal becomes a public event. A mechanism that produces stronger deterrence may also generate stronger chilling effects. For that reason, one of the paper's key empirical questions is whether visible accountability improves behavior without imposing prohibitive social or operational cost.

9.5. Why not direct messages or external tools?

A natural alternative to SealedChat is to avoid public-channel disclosure entirely by using direct messages, secure vault links, or external case-management tools. These mechanisms can reduce public exposure, but they also move the disclosure interaction outside the shared operational context of the channel. As a result, the team may lose transparency around who accessed the data, when disclosure occurred, and how the interaction related to the shared workflow. SealedChat is designed for cases in which disclosure must remain anchored in the shared channel context, but should not remain ambiently visible as ordinary channel text.

9.6. Compliance and audit alignment

SealedChat's design is naturally relevant to regulated or policy-sensitive environments because it produces explicit access evidence rather than relying solely on content blocking or retrospective cleanup. Public accountability events provide lightweight in-channel transparency, while private audit records preserve structured evidence for later review. This does not make SealedChat, by itself, a compliance solution. However, the model is compatible with broader accountability and audit practices in enterprise and regulated systems, particularly where disclosure decisions may later need to be reviewed, justified, or investigated [11], [12].

10. Related Work

SealedChat sits at the intersection of accountability-oriented security, exceptional-access control, and secure collaboration tooling. This section positions the paper relative to those areas.

10.1. Break-glass and exceptional access control

The closest conceptual lineage for Accountable Disclosure is break-glass access control, especially in healthcare and regulated environments. Break-glass systems permit exceptional access under urgent or unusual circumstances while requiring that the action be logged, justified, and subject to later review [6]-[10]. SealedChat inherits the basic governance intuition of this literature:namely, that some forms of access are operationally necessary but should be explicitly accountable:while applying it to a different object of control: disclosure of sensitive data in shared conversational environments rather than access to records in protected repositories.

10.2. Accountability and audit-based security

A broader line of work studies how logging, evidence, and after-the-fact review can serve as security controls in systems where strict prevention is either impossible or undesirable [1]-[5]. Research on accountability emphasizes that meaningful audit requires more than event collection; it requires logs that capture who acted, under what conditions, and with what justification. SealedChat follows this tradition by treating each reveal as a justification-bearing and reviewable event.

10.3. Collaboration platforms, DLP, and message visibility controls

Modern collaboration platforms increasingly include controls for content moderation, retention, and data loss prevention [15], [17]. These mechanisms are important in practice, but they are typically oriented toward detection, blocking, deletion, or remediation of sensitive content once identified. They do not generally offer a shared-channel primitive for intentional, justified, and time-bounded disclosure when communication of the content is operationally necessary. SealedChat is therefore better understood as complementary to DLP-style controls than as a substitute for them.

10.4. Ephemeral and private response primitives

Slack, Teams, and Discord all expose some form of user-specific or interaction-scoped response mechanism, such as ephemeral replies or modal workflows [13], [14], [16], [18], [19]. These primitives are relevant because they provide building blocks for reducing exposure of sensitive data in shared spaces. However, such features alone do not define the semantics of Accountable Disclosure. SealedChat's contribution is not merely to use ephemeral or private-response features, but to combine sealing, mandatory justification, time-bounded viewing, and visible accountability into a single disclosure model.

10.5. Usable security in operational workflows

SealedChat also relates to usable-security work that studies how security mechanisms interact with real operational behavior, particularly under time pressure. In collaborative environments, controls that are too burdensome are often bypassed, while controls that are too invisible may fail to deter misuse. The design of SealedChat reflects this tension directly: it introduces friction through justification and time-bounded viewing, but seeks to keep that friction low enough for incident response, support, and coordination workflows.

11. Conclusion

This paper introduced Accountable Disclosure as a model for handling sensitive data in public team chat. Rather than allowing sensitive content to become ambiently visible in shared channels, Accountable Disclosure requires explicit reveal, mandatory justification, time-bounded viewing through a secure view, and auditable access events. The goal is not to prevent all misuse after disclosure, but to reduce passive exposure and make access intentional, visible, and reviewable.

We presented SealedChat as a system design that realizes this model while keeping sensitive payloads out of

platform message histories. We further analyzed how the model can be realized across heterogeneous collaboration platforms and outlined an evaluation methodology for assessing exposure reduction, usability costs, accountability effects, and platform-specific realization constraints. More broadly, the paper argues that public team chat lacks an appropriate primitive for operationally necessary disclosure of sensitive data. Existing controls emphasize prevention, blocking, or remediation; SealedChat instead explores a complementary path based on accountable access. We believe this framing opens a useful direction for future work at the intersection of usable security, accountability, and collaborative systems.

References

[1] M. Vaughan, S. Chong, and A. C. Myers, "Evidence-Based Audit," in Proceedings of the 21st IEEE Computer Security Foundations Symposium (CSF), 2008.

[2] D. Butin, F. Kelbert, and M. Tai, "Log Design for Accountability," in Proceedings of the IEEE Security and Privacy Workshops, 2013.

[3] S. Etalle and W. H. Winsborough, "A Posteriori Compliance Control," in Proceedings of the 12th ACM Symposium on Access Control Models and Technologies (SACMAT), 2007.

[4] J. Cederquist, R. Corin, M. Dekker, S. Etalle, and J. den Hartog, "Audit-Based Compliance Control," International Journal of Information Security, vol. 6, no. 2-3, pp. 133-151, 2007.

[5] J. Feigenbaum, A. Jaggard, and R. N. Wright, "Towards a Formal Model of Accountability," in Proceedings of the 2011 New Security Paradigms Workshop (NSPW), 2011.

[6] L. Rostad and O. Edsberg, "A Study of Access Control Requirements for Health Care Systems Based on Audit Trails from Accesses to Electronic Patient Records," in Proceedings of the 22nd Annual Computer Security Applications Conference (ACSAC), 2006.

[7] A. Ferreira et al., "How to Break Access Control in a Controlled Manner," in Proceedings of the 19th IEEE International Symposium on Computer-Based Medical Systems, 2006.

[8] A. D. Brucker and H. Petritsch, "Extending Access Control Models with Break-Glass," in Proceedings of the 14th ACM Symposium on Access Control Models and Technologies (SACMAT), 2009.

[9] C. A. Ardagna et al., "Supporting Emergency Access in Healthcare Systems," in Proceedings of the 2007 ACM Workshop on Computer Security Architecture, 2007.

[10] S. Marinovic, N. Dulay, and M. Sloman, "Rumpole: A Flexible Break-Glass Access Control Model," in Proceedings of the 19th ACM Symposium on Access Control Models and Technologies (SACMAT), 2014.

[11] National Institute of Standards and Technology, Security and Privacy Controls for Information Systems and Organizations, NIST Special Publication 800-53 Rev. 5, 2020.

[12] K. Kent and M. Souppaya, Guide to Computer Security Log Management, NIST Special Publication 800-92, 2006.

[13] Slack Technologies, "chat.postEphemeral," Slack Developer Documentation.

[14] Slack Technologies, "Modals," Slack Developer Documentation.

[15] Slack Technologies, "Slack Data Loss Prevention," Slack Help Center / Documentation.

[16] Microsoft, "Task Modules for Bots and Adaptive Cards," Microsoft Teams Developer Documentation.

[17] Microsoft, "Data Loss Prevention in Microsoft Teams," Microsoft Purview Documentation.

[18] Discord, "Receiving and Responding to Interactions," Discord Developer Documentation.

[19] Discord, "Ephemeral Messages FAQ," Discord Support Documentation.