TrustGraph: Deterministic Infrastructure Protocol for Assertion-Level Content Trust Evaluation

Field of Invention

This invention relates to the field of digital content authenticity verification and network security protocols. In particular, it concerns an infrastructure-level protocol for deterministic evaluation of trust in user-provided assertions about digital content, involving cryptographic certificate issuance and content-level trust scoring. The invention lies at the intersection of content integrity assurance, cryptographic verification (digital signatures, hashing), and computational trust evaluation in distributed computing environments.

Background

Modern digital ecosystems lack a standard way to verify the truthfulness of user claims about content. Users often make declarative assertions about their content (e.g. “I authored this text,” “No part of this image is AI-generated,” or “This document is original”), yet no internet-scale protocol exists to evaluate or communicate the trustworthiness of such assertions . While protocols like HTTPS and DNS secure data transport or identity, none addresses content truthfulness. As a result, institutions, publishers, educators, and other stakeholders have no consistent method to assess whether a user’s statement about a piece of content is likely true or false .

This gap leads to various problems and vulnerabilities in digital content management:

- Fabricated Authorship or Ownership: Individuals may falsely claim credit for work they did not create . Without verification, plagiarized or purchased content can be passed off as original.

- Undetected AI-Generated Content: AI-produced text or images can be misrepresented as human-made , undermining authenticity and violating emerging disclosure norms.

- Tampered References or Data: Users might assert that content is accurate or unaltered even when citations, data, or images have been manipulated .

- Falsified Provenance: False claims about origin or originality are easy to make (e.g. claiming a document is original when it is copied) , with honest users lacking means to

prove integrity and malicious actors facing little risk of detection .

In the absence of a dedicated trust protocol, stakeholders rely on ad-hoc tools and methods, each with significant limitations :

- Plagiarism Detection Software: Tools like Turnitin detect textual overlap with known sources, helping flag copied text. However, they only report similarity scores and do not produce any portable, cryptographically verifiable certificate of originality . They address a fragment of the trust problem and cannot conclusively certify an authorship claim.

- AI Content Detectors: Classifiers or heuristics attempt to guess if content was machine-generated. These provide a probability or score but are often unreliable, with studies showing high false positive and negative rates . Crucially, they do not yield a standard, verifiable proof attached to the content. A user flagged (even incorrectly) has no independent artifact to dispute or confirm the finding.

- Embedded Watermarks/Metadata: Some solutions embed hidden markers or metadata at content creation for later verification (for instance, an AI model watermarking its output, or adding metadata tags about origin). Such markers can often be removed or altered , and not all content formats support persistent metadata. Moreover, watermarking only indicates source or creation method; it does not evaluate the truth of a user's claim.

- Blockchain Timestamps/Hashes: Storing a content hash on a blockchain can prove a particular content version existed at a certain time, but it does not verify any user assertion about the content's authorship or originality. It also doesn't provide reasoning or a trust score – only existence and temporal proof.

Another notable effort in the space is the Content Authenticity Initiative (CAI) and C2PA standard, which attach signed provenance metadata to media. These standards do not allow one to determine if content is "true" or accurate – they only record origin information . For example, C2PA can tell who published a photo and how it was edited, but it cannot tell if the content of the photo is truthful or if a user's claim about it is valid . Additionally, typical signing tools under C2PA don't verify the accuracy of metadata either , leaving a trust gap. Adoption of such provenance standards is still very low (as of 2025, very little internet content carries C2PA metadata) . This underscores that existing approaches, while useful in niches, do not provide an integrated, assertion-level trust verification mechanism with a verifiable output .

Need for an Infrastructure-Level Solution: The shortcomings above reveal the lack of an internet-scale "content truth" layer. The invention "TrustGraph" fulfills this need by introducing a

new deterministic protocol for content assertion verification . TrustGraph's design is analogous to how SSL/TLS certificates introduced a trust layer for website identity – here, the trust layer is for content integrity and claim truthfulness . It provides a structured way to evaluate a user's assertion about content and communicate the result as a cryptographically signed certificate. By operating externally to any single platform and not altering content, it overlays onto existing digital systems to enable trust without requiring those systems to change .

This invention is also timely from a policy perspective. Global digital governance and AI ethics frameworks are increasingly calling for stronger content authenticity and transparency measures. For example, at an international level there are calls for systematic labeling or watermarking of AI-generated content, and even authenticating genuine content to combat misinformation . A 2024 United Nations resolution urges the watermarking of both AI-generated and real content to ensure authenticity . Likewise, emerging regulations (e.g. mandates for AI content disclosure) emphasize the need for reliable verification of content origin and integrity . TrustGraph's protocol directly contributes to these goals by providing a technical infrastructure for verifying and certifying content truthfulness, thereby supporting compliance with evolving standards of digital trust and accountability.

In summary, there is a clear unmet need for an infrastructure-level, content-agnostic trust protocol that can evaluate the truth of user assertions about content and distribute that evaluation in a verifiable manner. TrustGraph addresses this by combining multi-faceted content analysis with cryptographic certification to deliver a new layer of trust for digital content.

Summary of the Invention

TrustGraph is a deterministic protocol and system for evaluating the probabilistic trustworthiness of a user's assertion about digital content, and producing a verifiable result. The invention provides both a numeric TrustScore indicating the likelihood that the assertion is true, and a signed Trust Certificate encapsulating the evaluation outcome. Unlike ad-hoc tools, this is a cohesive infrastructure solution with defined interfaces and data formats, enabling broad adoption across platforms.

At a high level, the system accepts two inputs: (1) a digital content item (or a content identifier such as a cryptographic hash of the content), and (2) a declarative assertion about that content (for example: "This document is entirely my own work," or "No part of this image is AI-generated"). TrustGraph then processes these inputs through a sequence of modules to determine how likely the assertion is to be truthful, as follows:

- Signal Collection: Various observable signals related to the content and its context are gathered . This is done by a Signal Derivation Graph (SDG) module. Signals include content-intrinsic features (e.g. text statistics, metadata, similarity against known sources), user behavior data (editing patterns, timing of content creation), environmental data (submitter's device, location, network), and other context. All signals are normalized

into a structured form for analysis. Example: For a text document with an originality claim, signals might include plagiarism check results, writing style metrics, and whether the text was pasted in all at once or typed gradually.

- Contradiction Analysis: An Assertion Integrity Engine (AIE) checks for inconsistencies between the user's assertion and the collected signals . It applies assertion-specific rules or checks (optionally using external detectors or classifiers for certain patterns) to identify any contradictory evidence. The outcome is an Integrity Score quantifying how consistent the content appears with the claim (high score = content largely supports the assertion; low score = evidence strongly contradicts it) . AIE also compiles a Contradiction Analysis Summary explaining what evidence was found to contradict or support the claim, enhancing transparency.

- Probabilistic Trust Scoring: Using the integrity analysis results, the system computes a probabilistic TrustScore representing the estimated likelihood that the assertion is true . This occurs in the Confidence Computation Engine (CCE). In one embodiment, the TrustScore is calculated via Bayesian inference: starting from a prior probability of the assertion's truth, and then updating that belief based on the evidence (Integrity Score) . Other statistical or machine learning models can be used as long as they produce a calibrated probability. Importantly, CCE also generates a confidence interval or similar uncertainty measure for the TrustScore , reflecting how robust the conclusion is given the available data. For instance, the system might output a TrustScore of 0.85 (85% chance true) with a ±0.10 confidence margin at 95% confidence, indicating some uncertainty. Fewer or weaker signals yield a wider interval (less confidence), whereas abundant strong evidence yields a narrow interval (high confidence) .

- Score Normalization and Labeling: The TrustScore Engine (TSE) takes the raw TrustScore and confidence metrics and normalizes them to a standard format . TSE may apply predefined thresholds or policies to map the numeric score to qualitative categories or labels for easier interpretation. For example, TSE might label a score as "High Trust", "Moderate – Review Advised", or "Low Trust" based on where it falls, and attach any policy-driven flags (e.g. do not issue a certificate if score below a certain value) . The output of TSE is a standardized trust report (including the score, confidence, label, timestamp, etc.) ready for certification.

- External Binding (Zero-Fabrication): To avoid altering the original content, TrustGraph uses a Zero-Fabrication Protocol (ZFP) to bind the trust evaluation result to the content without modifying the content file . Essentially, ZFP generates a unique binding token by hashing together the content's cryptographic hash, the TrustScore (or certificate ID), and a random nonce . This token serves as a secure link between the content and the trust score. The original content remains completely unchanged – nothing is embedded in it (hence "zero-fabrication") . Any attempt to apply the resulting certificate to a different content item would fail verification because the content's hash would not match the

certificate's recorded hash .

- Certificate Generation: Finally, a Certificate Generator (CG) module compiles the results into a portable Trust Certificate . The certificate encapsulates key information: the content's identifier (hash), the user's assertion, the TrustScore (with confidence and any labels), the contradiction summary or reason codes, a timestamp, and the issuer's identity. The CG then digitally signs this certificate using a private cryptographic key . The signature makes the certificate tamper-evident and verifiable by anyone with the corresponding public key. The certificate is preferably in a machine-readable structured format (e.g. JSON or XML) so that it can be stored, transmitted, or inspected easily.

The outputs of the TrustGraph process include: (a) the numeric TrustScore (with uncertainty measure), (b) a Contradiction Analysis Summary explaining the evidence, and (c) a signed Trust Certificate as a proof object . The Trust Certificate can be independently verified by third parties without needing to re-run the analysis or trust the issuer's infrastructure, similar to how an SSL certificate can be checked by any web browser . By design, TrustGraph's logic is deterministic and transparent: given the same inputs, it will produce the same TrustScore and certificate, and it discloses the reasons behind the scoring.

Important Note on AI/ML: The core TrustGraph scoring logic is not based on any opaque AI or large language model. The trust evaluation pipeline uses deterministic rules and statistical inference that are fully explainable and reproducible. While the system can incorporate outputs from external AI tools as signals (for example, using an AI text detector's result as one input feature), such tools operate outside the core logic and serve only as optional aids. No machine learning model or LLM is integral to the TrustGraph scoring algorithm – this ensures the process remains auditable, consistent, and avoids the unpredictability of black-box AI in the trust determination. The invention's focus is on transparent, rule-based and statistical methods for trust evaluation, supplemented by external analyses when appropriate but not dependent on them. This distinction is made to emphasize the protocol's deterministic nature and to facilitate compliance with patent eligibility criteria that exclude mere "AI/algorithm abstraction" in favor of concrete technical processes.

Overall, the invention provides a novel content trust protocol that yields a quantifiable measure of trust in user assertions, along with a cryptographic certificate to convey that measure. It offers an infrastructure-level solution to a growing problem of digital content integrity, with built-in technical measures (hashing, signing, external binding) to ensure it operates robustly and securely across various platforms. The ensuing detailed description illustrates the system's architecture, components, and operation, followed by exemplary use cases and claim definitions.

Detailed Description

The invention will now be described in detail with reference to its modular components, data flow, and example use cases. This description outlines one embodiment of TrustGraph to enable those skilled in the art to implement it, but variations in implementation (different algorithms for scoring, different data structures for certificates, etc.) are possible without departing from the core inventive concept.

System Overview

At its core, TrustGraph consists of a pipeline of modules that transform the inputs (content + assertion) into outputs (trust score + certificate). The process is external to the content's native platform – it can be run as a service or integrated layer that processes content on demand. The system architecture is divided into stages, each with a defined role:

1. Signal Derivation Graph (SDG) – Signal collection and preprocessing
2. Assertion Integrity Engine (AIE) – Contradiction analysis against the assertion
3. Confidence Computation Engine (CCE) – Probabilistic trust scoring
4. TrustScore Engine (TSE) – Score normalization, labeling, and policy enforcement
5. Zero-Fabrication Protocol (ZFP) – External binding of trust data to content (no content modification)
6. Certificate Generator (CG) – Trust certificate assembly and digital signing

Supporting components and standards complement these core modules, including a public certificate registry for publishing results, a compliance header format for integrating trust data with content, and a consent & traceability module to handle user permissions and audit logs. Each component is described below in detail.

Signal Derivation Graph (SDG)

Purpose: The SDG is responsible for collecting and normalizing observable signals about the content and its context. It establishes the raw evidence base used to evaluate the user's assertion . The content item is treated as essentially an opaque object; SDG gathers what can be observed from the content itself and related metadata or environment, without requiring any alteration of the content.

Inputs to SDG: At minimum, SDG requires two inputs: (1) the digital content artifact (or a stable reference to it), and (2) the declared assertion about that content (with an indication of what type of claim it is) . It may also take in contextual metadata. Specifically:

- Content Item: This could be the actual data (e.g. the text of a document, image file bytes) or simply a content identifier (like a cryptographic hash or database record ID) that allows retrieval of the content . In many cases the system will need access to the content data (to compute features like word frequencies or image patterns), so the content or its hash must be accessible.

- User's Assertion: The text of the user's claim about the content, along with an assertion type label . Knowing the type (e.g. "original content" claim, "human-authored" claim, "no AI content" claim, etc.) helps tailor which signals are relevant. For example, an authorship claim may prompt analysis of writing style and consistency with the claimed author's known work, whereas a "no AI" claim focuses on detecting AI-generated patterns.

- Contextual Metadata (optional): Any available environmental information about the content's creation or submission . This can include timestamps, the user's environment (IP address, geolocation, device or OS, browser editor used) , system-side metadata (file name, revision history, whether the content was edited extensively or mostly pasted in one go) , and user-declared auxiliary data (e.g. citations provided by the user, declared author name). These contextual pieces can later be cross-checked for consistency with the assertion (for instance, if the user claims "I wrote this on my own," but the metadata shows a different author or unusual editing pattern, that's a signal).

Signal Collection & Extraction: Using the above inputs, SDG computes and gathers a variety of signals. Here, a signal means any piece of information objectively derivable from the content or its context that might support or undermine the user's claim . SDG is essentially feature extraction at scale. Examples of signal categories include:

- Content-Intrinsic Signals: Features derived directly from the content data . For a text document, this could include textual statistics (word count, average sentence length, vocabulary complexity, stylometric markers, readability scores, entropy of the text, etc.), structural features (document formatting patterns, presence of a bibliography or references section, coding style if it's source code), and similarity measures such as plagiarism detection results (e.g. percentage of overlap with known sources) . For images, intrinsic signals might include metadata (EXIF data like camera model or editing software used), detection of known manipulation traces or GAN (AI-generated) artifacts, or reverse image search matches to see if the image has appeared elsewhere . Additionally, outputs of specialized detectors can be included – e.g. running an AI content detector on a text and taking its "likelihood of AI-generated text" score as a signal . Essentially, any analytic tool that produces insight about the content can feed into SDG as long as it's relevant to the assertion.

- Behavioral Signals: Data about how the content was created or submitted, if such data is available . Many creation platforms record user behavior that TrustGraph can leverage. For example, an editing revision history from a document editor could show the number of edits, frequency and timing of those edits, or whether large blocks of text were pasted in at once versus typed gradually . Such data is very telling: e.g. an essay pasted in one go with minimal edits might be suspicious if the claim is original authorship. Other behavioral signals include submission patterns (e.g. a user submitting multiple different assignments in a very short time might indicate use of generative tools) , or timing cues (if a student always submits assignments at 3:00 AM, perhaps they are using automated help). These signals help infer if the creation process aligns with the assertion's implication (genuine independent creation vs. copy-paste, AI generation, collusion, etc.).

- Environmental Signals: The context of the content's environment and submission . For instance, the user's IP address or geolocation at submission time can indicate whether the content came from an expected location . If someone claims personal authorship but the content was uploaded from an IP address in a region where the user doesn't reside (or from a known VPN/cloud service), that discrepancy could be noted. Device or environment fingerprints (OS type, browser version) might be relevant too: if the user usually writes on a personal laptop but this content was created on an unknown device or OS, it's a potential red flag . Also, any known anomalies like content submitted from a data center (possibly indicating use of a cloud AI service) would count as an environmental signal . These signals serve as auxiliary context to either corroborate the user's story or cast doubt on it.

All collected signals are then normalized into a structured format (such as a vector of feature values, or a key–value map of signal types) . Normalization ensures the signals can be uniformly processed by subsequent modules. It may include scaling numeric values, encoding categorical flags, and filtering out signals that are irrelevant to the specific assertion type (for example, plagiarism-related signals are central for an originality claim but might be irrelevant if the assertion is about "no malware in file"). The SDG essentially prepares a comprehensive, structured evidence profile for the content+assertion pair.

Extensibility: The SDG is designed to be extensible. New types of signals can be integrated via a defined Signal Provider Protocol (SPP) . This means third-party tools or data sources can plug into TrustGraph to provide additional evidence as needed. For instance, if a new advanced AI-text detector is developed, a provider adhering to SPP could feed its detection score into SDG without requiring changes to the rest of the pipeline. This modularity ensures TrustGraph can evolve with new content analysis techniques.

Assertion Integrity Engine (AIE)

The Assertion Integrity Engine (AIE) takes the normalized signals from SDG along with the user's assertion and performs a contradiction analysis. Its goal is to identify any evidence

among those signals that contradicts or supports the user's claim, and synthesize an overall integrity metric.

Core Function: AIE applies a set of assertion-specific rules and algorithms to evaluate consistency between the claim and the evidence . In essence, it asks: "If the user claims X about this content, what evidence do we have to the contrary (or in support)?" For each relevant signal, AIE determines whether that signal supports the assertion, contradicts it, or is neutral. It then aggregates these findings into an Integrity Score (or equivalently, a contradiction score) .

The aggregation logic can be as simple or complex as needed:

- If multiple strong contradictions are found, the Integrity Score will be low (indicating the content appears inconsistent with the claim) .
- If no contradictions (and possibly some supporting evidence) are found, the Integrity Score will be high (indicating the content is largely consistent with the claim) .

AIE can implement this analysis using a combination of deterministic rules and, where applicable, trained classifiers for specific tasks . For well-defined checks, straightforward thresholds can be used. Example: for an originality claim, AIE might use a rule like "if >20% of the content matches known sources without citation, flag a contradiction" . For more complex patterns (like detecting AI-generated text when human authorship is claimed), AIE might invoke a machine learning classifier specialized in that detection . Each identified contradiction or supporting cue can be assigned a weight based on severity or confidence . For instance, finding an uncited 30% plagiarism overlap might be a high-severity contradiction, whereas a minor stylistic anomaly might be a low-severity hint.

AIE compiles a Contradiction Analysis Summary listing each notable contradiction or supporting signal and its influence on the outcome . This summary is essentially an explanation of why the score is what it is – providing transparency for end-users or auditors. The summary will be included in the final certificate (or at least referenced by it) so that the trust decision is explainable.

Output: The primary output of AIE is the Integrity Score, a numerical value (e.g. 0 to 1 or a percentage) representing the overall consistency of the content with the assertion . A high Integrity Score means little or no evidence against the claim; a low score means strong evidence of falsehood. Often we can conceptualize it as Integrity = 1 – (weighted contradiction) for simplicity . This Integrity Score, along with the detailed contradiction summary, is passed to the next stage (the probabilistic scoring engine).

Illustrative Example: Suppose a student asserts, "This essay is entirely my own work" about a submitted document. SDG gathered signals including a plagiarism check which found that 30% of the essay's text matches external sources without proper quotation or citation. AIE would flag

this as a major contradiction to the originality claim. It might assign a relatively low integrity score, say 0.4, indicating significant doubt about the claim's truth . The contradiction summary would note something like: "High overlap (30%) with sources X and Y detected, indicating possible plagiarism." If other signals (writing style, metadata) were consistent with the student (e.g. writing style matches the student's past work), those might provide some support, but likely the plagiarism evidence dominates and keeps the integrity score low. Conversely, if the essay had 0% overlap with any source and all other signals (style, timing, metadata) showed nothing unusual, AIE might output a very high integrity score (near 1.0) and note "no contradictions detected" . In either case, AIE has taken raw signals and translated them into a judgment about the truthfulness of the claim.

This step – explicitly evaluating an assertion against evidence – is a key inventive element of TrustGraph. Prior art tools might compute metrics like "text overlap percentage" or "AI likelihood" in isolation, but they do not combine and interpret them in the context of a specific user assertion with an outcome that feeds into a trust credential. AIE provides the logical glue between raw evidence and a trust decision, essentially performing a targeted "fact-check" on the user's claim. The combination of AIE's contradiction analysis with the subsequent probabilistic scoring (CCE) yields a novel way to quantify trust in content assertions .

Confidence Computation Engine (CCE)

The Confidence Computation Engine (CCE) takes the Integrity Score (and optionally the detailed evidence profile) from AIE and converts it into a probabilistic trust score known as the TrustScore . This module performs the statistical inference that estimates how likely the assertion is true given the observed evidence.

Probabilistic Scoring: In one embodiment, CCE uses a Bayesian inference approach . For example, the system can start with a prior probability for the assertion's truth (this prior might be domain-specific or a default like 0.5 if no prior knowledge). Then, using Bayes' theorem, it updates that probability based on the Integrity Score and related evidence:

- If AIE found strong contradictions (low integrity), the posterior probability (the TrustScore) will drop significantly relative to the prior.
- If AIE found no contradictions or supportive evidence (high integrity), the TrustScore will increase relative to the prior (approaching 1, or 100%).

  This yields a principled way to integrate evidence strength into a final probability.

Alternatively, CCE could employ other statistical or machine learning models, such as a logistic regression or a small neural network that takes various features (Integrity Score, number of contradictions, severity levels, etc.) and outputs a probability . The invention is not limited to one

algorithm – the key point is that CCE produces a calibrated probability (or analogous confidence metric) of the assertion's truth rather than a binary verdict . This probabilistic approach provides nuance; it reflects degrees of belief and allows the system to indicate uncertainty.

Confidence Interval: Alongside the TrustScore, CCE computes a measure of uncertainty in that score . This can be expressed as a confidence interval (e.g., "we are 95% confident the true probability lies between X and Y") or as a confidence level. For instance, if evidence is sparse or contradictory signals have variability, CCE might output: TrustScore = 0.70, with 95% confidence interval [0.50, 0.85] . A wide interval like that means the system isn't very certain – the truth could plausibly lie quite lower or higher. Conversely, if evidence is plentiful and consistent, it might output TrustScore = 0.95 with interval [0.90, 0.98], indicating a high-confidence result . The confidence interval can be derived from the statistical properties of the model – e.g., in a Bayesian model, the posterior distribution's variance gives it ; in a frequentist sense, multiple independent signals reinforcing each other yield lower variance.

By producing both a TrustScore and a confidence measure, the system provides a richer output than a simple pass/fail or single score. It answers two questions: "How likely is the assertion true?" and "How sure are we about that likelihood?" . This is an important differentiator over simplistic prior tools (like a plagiarism percentage or an AI detector flag), which do not typically convey uncertainty. Decision-makers (humans or automated systems) can use the TrustScore with appropriate caution. For example, a university might treat a TrustScore of 0.95 ± 0.02 as strong evidence a submission is original, whereas a score of 0.70 ± 0.20 would be deemed inconclusive and trigger further review . Including uncertainty prevents false confidence and encourages trust in the system by acknowledging when evidence is limited.

TrustScore Engine (TSE)

The TrustScore Engine (TSE) receives the raw TrustScore and confidence info from CCE and prepares it for output and certification . TSE ensures consistency and human interpretability of results across different uses. Its functions include normalization, labeling, and policy enforcement:

- Normalization: If different assertion types or models could yield scores on different scales initially, TSE maps the TrustScore onto a standard scale (e.g. 0.0–1.0 or 0–100%) . In practice, the scores from CCE will already be a probability (0 to 1), but TSE makes sure to format it uniformly (for instance, rounding to a certain number of decimal places). It also normalizes the confidence representation (e.g., always express intervals at a fixed confidence level like 95%) for consistency.

- Categorization and Labels: TSE can assign qualitative labels or categories to the score to make it easier for end-users to interpret . For example, thresholds might be defined such that:

    - TrustScore > 0.90 → label as “High Confidence True” (or “High Trust”),
    - 0.70–0.90 → “Likely True”,
    - 0.40–0.70 → “Uncertain”,
    - below 0.40 → “Likely False” (or “Low Trust”).

      These labels are illustrative; actual calibration would be based on testing. For instance, an originality claim with TrustScore 0.95 might be labeled “High Trust – Original” . The use of labels provides a quick, user-friendly indication of the result, which is useful when presenting to non-experts. The system may allow policy makers (like an institution using TrustGraph) to adjust these threshold values based on their risk tolerance.

- Policy Enforcement: Some deployments may have policies about what to do with certain scores. TSE can enforce such policies as part of result packaging . For example, a platform might decide not to issue a Trust Certificate at all if the TrustScore is below a certain cutoff (essentially, if the content likely fails verification). Or it might attach a “Verification Failed” flag in the output for low scores. TSE can implement these rules. E.g., if TrustScore < 0.3, TSE might mark the evaluation as “rejected” or signal downstream systems to take a specific action .

- Packaging Output: TSE then formats all the relevant data into a final trust score record. This includes the numeric score, the confidence interval, the qualitative label, the evaluation timestamp, a unique evaluation ID, and any policy flags or notes . The result is a standardized data structure (essentially a summary of the evaluation) which will be used by the Certificate Generator next.

The separation of CCE and TSE modules is deliberate: CCE focuses on analytical computation of the probability, whereas TSE focuses on presentation and integration. This modular approach allows the underlying scoring methods to change or improve over time without affecting how results are communicated externally . The TSE ensures that no matter how the TrustScore was computed internally, it is always reported in a uniform, interpretable manner externally. This consistency is critical for adoption, as various institutions and systems can rely on the meaning of a “TrustScore” being standardized.

Zero-Fabrication Protocol (ZFP)

The Zero-Fabrication Protocol (ZFP) module binds the trust evaluation result to the content in a tamper-evident way without altering the content . This is a crucial design choice to keep TrustGraph content-agnostic and non-intrusive. “Zero-fabrication” (or zero-footprint) means the

content file remains exactly as it was (no watermarks, no injected metadata), yet we establish a strong external linkage between the content and its TrustScore/certificate.

Process: ZFP takes as input the content item (or at least a stable identifier for it) and the finalized TrustScore (or an identifier for the evaluation result) . It performs the following steps:

1. Compute Content Hash: Calculate a secure cryptographic hash of the content item . The hash (e.g., SHA-256) serves as a unique fingerprint of the content data – even a one-bit change in the content yields a completely different hash. This hash will represent the content in all subsequent linking and verification steps.

2. Generate Binding Token: Combine the content's hash with the TrustScore (or the evaluation's unique ID or certificate ID) and possibly some other fields (like a timestamp or a random nonce) to produce a binding token . For example, one scheme is to concatenate Hash(content) || Evaluation_ID || Nonce and then hash that concatenation to get a fixed-size token . The nonce is included to ensure uniqueness (so that if the same content and same assertion are evaluated twice, we still get distinct tokens). This binding token essentially seals together the content identity and the result identity.

3. Embed Binding in Certificate: The binding token (or its components, such as the raw content hash and eval ID) is then included in the data that will go into the Trust Certificate . Typically, the certificate will explicitly include the content's hash (so verifiers know which content it's tied to) and maybe the binding token or certificate ID.

Crucially, nothing is added to or changed in the content file itself at any point . The binding exists solely in the certificate's data. Because of the cryptographic properties, this is sufficient: anyone later can recompute the hash of a given content file and check if it matches the hash recorded in a certificate.

Security: The approach is tamper-evident in two ways:

- If someone tries to use the Trust Certificate with a different content (even a slightly modified version of the original), the content's hash will not match the certificate's content hash, and verification will fail . The binding token would also fail to validate if recomputed.

- If someone alters the certificate (e.g., changing the score or assertion in it), the digital signature (applied by the Certificate Generator) will no longer be valid. This will also cause verification failure. (Certificate signing is described in the next section.)

By not embedding any data in-content, ZFP avoids the pitfalls of watermark-based approaches. It means TrustGraph can work with any file type (even ones that don't support metadata or would visibly show watermarks) . It also means the presence of a trust certificate doesn't alter the user's experience of the content – an image or text can be shared freely without carrying any baggage, and the trust proof travels separately. Additionally, keeping the trust data separate allows control: for sensitive scenarios, the certificate can be withheld or stored in a controlled registry rather than attached to the content itself.

Example: Suppose we have content with hash H = SHA-256(content) and the TrustScore evaluation has an ID E = 12345. ZFP might generate T = SHA-256(H || E || nonce) as a binding token . The Trust Certificate will then include the content_hash = H (and perhaps the binding_token = T, or it might rely on H and the certificate's own ID to implicitly cover T). To verify later, a third party recomputes the SHA-256 of the content file and checks that it matches the content_hash in the certificate; if it doesn't, that certificate is not for this content. The binding token is mainly an internal consistency check to prevent substitution and to feed into the signature (ensuring uniqueness).

ZFP thus achieves a "zero-footprint" binding: anyone with the content and its certificate can confirm the two belong together, yet the content file has zero new bytes added. This is a novel improvement over prior art that required inserting metadata or watermarks into files (which, as noted, can be removed or might not be supported by some formats) . Here we leverage standard cryptographic primitives (secure hashes) in the new context of content truthfulness verification to get a robust binding .

Certificate Generator (CG)

The Certificate Generator (CG) compiles all the results and produces the final Trust Certificate, then digitally signs it to complete the process . This is the stage that turns the TrustGraph output into a portable credential that can exist independently of the service and be verified by others.

Certificate Composition: CG gathers the following data for inclusion in the certificate :

- The Content Identifier: typically the cryptographic hash of the content (from ZFP). The certificate may also include content metadata like content type (e.g. "text/plain", "image/png") or size, to help identify the artifact .

- The User's Assertion: the actual text of the assertion and possibly a category or type label for it (e.g. "originality_claim") .

- The TrustScore and Confidence: the numeric trust score (post-normalization) and its confidence interval, as well as any qualitative label assigned .

- The Contradiction Summary: either the full summary of contradictions/supporting evidence or a digest of it . This provides context for the score, helping others understand

the result.

- The Timestamp of evaluation: when the trust evaluation was performed, and a unique certificate ID or serial number for reference .

- Issuer Information: details about the TrustGraph service or authority that issued the certificate (for example, the organization name, or a service URL, along with a version number of the protocol/software) .

- Optionally, expiry or revocation data: e.g., an expiration date for the certificate or a flag indicating if it's subject to revocation or periodic re-validation .

CG then formats this collection of information into a structured document according to the TrustGraph Markup Format (TGMF) . The format could be JSON, XML, or another machine-readable schema that defines fields like "content_hash", "assertion", "trust_score", etc. All necessary data to later verify and interpret the trust evaluation is encapsulated.

For illustration, an example Trust Certificate (in JSON-like pseudo-format) might look like :


```
{
  "content_hash": "<hexadecimal hash of content>",
  "content_type": "text/plain",
  "assertion": "This document is original",
  "assertion_type": "originality_claim",
  "trust_score": 0.95,
  "confidence_interval": [0.90, 0.99],
  "label": "High Confidence – Original",
  "timestamp": "2025-07-27T10:00:00Z",
  "certificate_id": "ABC12345...",
  "contradictions": [
    {"signal": "plagiarism_check", "detail": "5% overlap, properly cited", "weight": 0.0},
    {"signal": "ai_content_check", "detail": "AI-detector score 0.10", "weight": 0.0}
```

```
  ],
  "issuer": "TrustGraph Service v1.0",
  "signature": "<digital signature over all the above fields>"
}
```

(This is an illustrative example; actual format and fields may vary, but the principle is to include content identity, assertion details, trust result, and proof.)

Digital Signature: Once the certificate data is assembled, CG digitally signs it using the issuer's private key . The signature covers the entire certificate content (every field except the signature itself). Using a standard asymmetric cryptography scheme (such as RSA or ECDSA), the CG produces a signature string which is then inserted into the certificate (as shown above in the "signature" field). The corresponding public key (or certificate chain for the TrustGraph authority) is made available so that anyone can verify this signature .

The digital signature is what gives the Trust Certificate authority – it's akin to a notary seal. It assures any recipient that "this certificate was issued by a legitimate source and has not been altered." Verification simply involves checking the signature with the public key.

Output Delivery: The final output of CG is the signed Trust Certificate. This certificate can be delivered back to the user or content platform that requested the trust evaluation, and/or published to a TrustGraph Public Registry (TPR) for discoverability (discussed further below) . Because the certificate is a self-contained document with all necessary info and a signature, it can travel separately from the content. A user might attach it when sharing the content, or a third-party can retrieve it from the registry when needed.

Verification Process: Later, any third-party verifier with access to the content and the certificate can:

1. Verify the certificate's digital signature using TrustGraph's public key (to ensure authenticity and integrity of the certificate) .

2. Compute the hash of the given content and compare it to the content_hash in the certificate .

   If both checks pass, the verifier is assured that the certificate was issued by a trusted authority and that it corresponds to that exact content item (and thus the TrustScore inside applies to this content). This verification is analogous to how web browsers verify SSL/TLS certificates (signature check and domain match), except here it's verifying a content assertion .

The generation of a tangible, cryptographically signed Trust Certificate is the cornerstone of this invention's novelty. It provides a detached yet verifiable proof of content integrity and assertion truthfulness – something no prior solution offers in this integrated way . The certificate can be stored, transferred, and audited independently of the content, enabling new workflows. For example, publishers could require authors to submit a TrustGraph certificate with their manuscripts, or social media platforms could allow users to share certificates for their posted content as a mark of authenticity . Because it is separate from the content, it does not interfere with normal content distribution, but can be linked whenever trust verification is required.

Additional Components and Integration

Beyond the core pipeline, TrustGraph includes optional but important components to facilitate usage at scale, compliance, and interoperability:

- TrustGraph Public Registry (TPR): This is an optional network-accessible repository (which could be a decentralized ledger or a centralized database) for storing and indexing Trust Certificates. The Certificate Generator can publish each signed certificate to the TPR, allowing third parties to query for certificates by content hash or certificate ID . The TPR ensures that certificates have persistence and can be found even if not directly provided by the user. It can also record revocation or expiration status. In one embodiment, the TPR could be implemented on a blockchain to provide tamper-resistance and transparency. For example, a verifier who obtains a content hash could query the TPR to retrieve the corresponding certificate and verify if it's still valid . The TPR thus acts as a distributed trust reference, ensuring certificates live beyond their initial issuance and can be audited or cross-checked (e.g., to prevent someone from altering a certificate unnoticed).

- Compliance Header Standard (CHS): This is an ancillary standard for representing trust info in content metadata or headers. The idea is to allow content that has an associated TrustGraph certificate to carry a reference or indicator of that fact, in-band with the content's transport. For instance, a content platform or an email system might include a special header (like X-TrustGraph-Cert-ID: <ID>) when transmitting content, so that receiving systems know a trust certificate exists and can fetch it (e.g. from TPR) . CHS could be applied to HTTP headers, email MIME headers, or file metadata fields. It's essentially a way to integrate with existing protocols to automate trust verification downstream. This is useful in enterprise or government settings where compliance policies might require checking content for trust certificates upon ingestion.

- Consent & Traceability Module (CTM): The CTM addresses ethical, privacy, and audit considerations. Before analyzing content, it ensures user consent is obtained for the evaluation (especially important if user data or behavior is being collected as signals) . The CTM would manage user opt-in/opt-out preferences and also maintain logs of the evaluation process. It creates a traceable record of who evaluated what and when, which is vital for accountability and compliance with data protection regulations. For

example, each trust evaluation event can be logged with a unique ID, timestamp, content hash, assertion, and result – these logs can be made immutable (using append-only ledgers) for audit. CTM thereby ensures the TrustGraph system adheres to privacy guidelines and AI ethics principles by being transparent about its operations and allowing oversight. In many jurisdictions, analyzing user content may require either consent or specific legal bases, and CTM is designed to fulfill those requirements and provide evidence of compliance.

- Public Key Infrastructure (PKI) Integration: TrustGraph leverages PKI for its certificate signing and verification. It can operate either with its own dedicated root key or within an existing PKI hierarchy. For instance, the TrustGraph service could have its public key certified by a reputable Certificate Authority, enabling a chain of trust that verifiers might already trust . The specifics of PKI usage can be implementation-dependent, but the existence of a robust PKI framework assures that trust certificates can be widely verified in a secure manner.

- TrustGraph Markup Format (TGMF): A standardized schema for all TrustGraph data structures, especially the certificate format . By defining this format, the invention ensures that different implementations of TrustGraph (or different instances) produce interoperable outputs that any verifier can parse. TGMF covers field names, data types, and encoding rules for content hashes, scores, evidence codes, etc.

- Signal Provider Protocol (SPP): As noted, SPP defines how external signal sources integrate into SDG . It could be an API or plugin interface where third-party analytics (like a new plagiarism API or an AI detector service) can provide their results in a predefined JSON structure for inclusion as signals. This fosters an ecosystem of extensibility around TrustGraph.

- Advanced Inference (AIE-2) and Reason Codes: In some embodiments, a secondary engine (sometimes referred to as AIE-2) might analyze multiple certificates or perform cross-document inference . For example, if two different users both claim originality for two documents that turn out to be identical, AIE-2 could flag this conflict (collusion or duplication) by scanning the certificate registry for contradictory claims . Such an advanced feature can strengthen the trust network by linking related content assertions and detecting systemic issues beyond single items. Relatedly, Trust Certificate Reason Codes (TCRC) are short codes embedded in certificates that categorize the types of checks or evidence that influenced the score (e.g., “plagiarism_detected”, “metadata_mismatch”) . These codes make the certificate self-explanatory to some degree and facilitate automated handling by downstream systems (for instance, a system could auto-reject content if it sees a “plagiarism_detected” reason code in the certificate above a threshold).

Each of these additional components contributes to making TrustGraph a comprehensive infrastructure. TPR ensures longevity and transparency of trust data; CHS and SPP ensure interoperability and integration; CTM ensures ethical compliance; and PKI/TGMF provide the technical backbone for secure, standardized operation. Not every deployment of TrustGraph must use all of these components (for example, a closed system might skip CHS and just attach certificates manually), but they are included in the invention to cover a robust, scalable implementation that can adapt to various industry requirements.

Technical Effect

The TrustGraph protocol achieves a significant technical effect in the realm of computerized content management by providing a novel mechanism to evaluate and communicate trust in user-generated content assertions. Unlike a mere abstract idea or algorithm, the invention is realized as a concrete integration of specific modules and cryptographic processes that solve a particular technological problem: the lack of a reliable way to verify content-related claims on existing computer networks . Key technical advantages and effects of the invention include:

- Tangible Output – Digital Trust Certificate: TrustGraph produces a cryptographically signed digital certificate as a tangible outcome of the trust evaluation . This certificate is a new type of machine-processable data object in the network ecosystem, representing verifiable trust information attached to content. It introduces an infrastructural mechanism for handling content trust analogous to how SSL/TLS certificates introduced a mechanism for handling website identity trust . The ability to generate and validate this certificate using public-key cryptography demonstrates that the invention is not an abstract idea but a practical system operating within computing environments . The trust certificate can be stored, transmitted, and automatically checked by software – enabling entirely new automated workflows for content verification.

- Enhanced Data Security and Integrity: By utilizing cryptographic hashing and digital signatures, TrustGraph ensures tamper-evident binding of claims to content . The Zero-Fabrication Protocol yields a secure linkage (via content hashes) without requiring any modification to the content itself . The technical effect is that content integrity is preserved (no need to embed watermarks or alter files) while still enabling a strong external association to trust metadata . This is a technical advancement over prior methods that embedded data into content (potentially degrading or being stripped) or had no means to cryptographically bind context to content. The invention leverages known cryptographic primitives in a novel way (for assertion truthfulness verification), thereby enhancing the security of content authentication processes.

- Automated, Scalable Trust Analysis: The invention enables automated analysis at scale of a problem that previously required manual judgment or isolated tools . The modular pipeline (SDG, AIE, CCE, etc.) can programmatically extract and evaluate hundreds of features within seconds for each content item, allowing large volumes of content to be

processed uniformly. This yields technical benefits in processing speed and consistency – something critical for internet-scale deployment . The architecture is suitable for distributed or cloud implementation, supporting parallel processing and high throughput. It addresses practical computing challenges of scale and reliability by its design (e.g., each module can run as a microservice, evaluations can be parallelized, a registry can distribute load). In short, TrustGraph provides a scalable computerized solution to content claim verification, which is a technical effect not achieved by prior art that was either manual or not integrative.

- Content-Agnostic Trust Layer (Interoperability): TrustGraph is designed to be content-type agnostic and platform-independent, which is a technical advantage for broad interoperability . It defines generic interfaces: content input can be any digital media (text, image, audio, etc., referenced by hash or data), assertion input is a text label, and output is a standard certificate. Achieving this required technical structuring – e.g., abstracting the signal extraction so it can handle different media types, and using a unified certificate format that can represent results across domains . The effect is the creation of a general-purpose trust layer for the internet – a new piece of network architecture that did not previously exist. It is analogous to common protocols like DNS or HTTPS but for a different function (verifying content truth), thereby filling a previously unaddressed niche in how networked systems communicate trust . This universality and standardization mean the invention can be adopted across disparate systems, enhancing the overall infrastructure.

- Transparency and Explainability: The system improves the transparency of automated content verification through its design. It outputs explicit explanations (contradiction analysis summaries, reason codes) alongside the trust score . This is a technical design choice aimed at usability and auditability in computer systems – unlike many AI-based tools which are “black boxes,” TrustGraph provides machine-readable and human-readable explanations of how it reached a conclusion. This explainability is a beneficial technical feature, allowing other software or administrators to audit decisions, integrate the output into reports, or comply with governance requirements that demand rationale for automated decisions . Prior algorithmic detectors for plagiarism or AI content typically did not provide such structured explanations; TrustGraph’s inclusion of reason codes and summaries is a novel technical feature that enhances trust in the system’s operation (by enabling debugging, appeals, or oversight when needed).

- Solving a Technical Problem Unaddressed by Prior Systems: The invention clearly addresses a specific technical problem arising in computer networks: evaluating the truthfulness of user assertions about content . Traditional computer security dealt with identity (PKI certificates), data transport (encryption), or file integrity (checksums), but no existing standard verified user-declared content attributes . TrustGraph fills this gap with a particular combination of technical means: multi-source signal analysis, statistical inference, cryptographic binding, and certification. The overall effect is a more trustworthy digital environment wherein content can carry an attached “truth score” and

certificate that any system can independently verify . This new capability enables workflows that were not possible before, such as automated content vetting and cross-platform trust propagation. It essentially introduces an infrastructure-level improvement to content management systems, adding a missing trust verification layer that complements existing layers of identity and security.

- Patent Eligibility (Technical Character): The invention's specific technical contributions ensure it is not merely an abstract algorithm or a "computer program per se," but rather a new computer-implemented process and system architecture that yields a verifiable improvement in digital content handling . It employs concrete technical means – signal processing modules interfacing with hardware (e.g., to capture data, possibly using sensors or system logs), cryptographic computations, and potentially distributed registries – to achieve results, thereby producing a technical effect and solving a technical problem as required for patentability in many jurisdictions . For example, under Indian patent law (Section 3(k)), the invention demonstrates a specific and credible technical effect beyond a mere software algorithm: it improves the security and reliability of content verification processes, analogous to how an improved network protocol or encryption method is recognized as having technical character . Likewise, under U.S. patent eligibility principles (35 U.S.C. §101), TrustGraph is rooted in computer technology (network security and data integrity), is not a mental process one could perform without a computer, and provides a concrete solution to a technological challenge; thus it is far from an abstract idea. The system's novel combination of steps results in a practical application in the realm of computer networks, reinforcing its patent-eligible status.

In sum, TrustGraph's architecture and operations produce multiple beneficial technical effects: improved security (via cryptographic verification and tamper-evident content binding), improved data processing capability (through automated multi-signal analysis and scalable evaluation), improved interoperability (through standardization and content-agnostic design), and entirely new functionality (assertion trust scoring and certification) not seen in prior computer-implemented solutions . These effects demonstrate the invention's technical contribution to the fields of computer science and networked content management. By addressing a crucial gap with a concrete, multi-faceted technical solution, TrustGraph advances the state of the art and provides a foundation for greater trust and integrity in digital information ecosystems.

Comparison with Prior Art

The following is a comparison between the TrustGraph invention and relevant prior art or existing approaches, highlighting how TrustGraph differs and overcomes their limitations:

- Plagiarism Detection Tools: Conventional plagiarism checkers (e.g., Turnitin) scan text against databases to find copied content. They output similarity reports (percentages of overlap, etc.) but do not produce any cryptographically verifiable record or certificate of authenticity . They also focus narrowly on text overlap and cannot integrate other signals (like behavior or metadata). TrustGraph, by contrast, not only detects textual overlap as one signal but also combines it with other evidence, generates a probabilistic trust score, and crucially, issues a signed Trust Certificate that can serve as long-term proof of originality. No plagiarism tool provides a portable trust metric that third parties can later verify independent of the tool itself.

- AI-Generated Content Detectors: Emerging AI text or image detectors use algorithms (often machine learning) to guess if content was produced by AI. These might return a score (e.g., “90% probability of AI-generated”). However, studies have shown such detectors are often inaccurate, yielding high false positives and false negatives . Moreover, like plagiarism tools, their outputs are ephemeral and not standardized – a user cannot easily attach an “AI authenticity certificate” to content from these tools. TrustGraph improves on this by treating AI-detection as just one input signal (with proper weighting and uncertainty), and by providing a structured, signed outcome that factors in AI signals among others. It explicitly quantifies confidence and explains why a piece of content is deemed human or AI, rather than a raw score that could be misinterpreted. Additionally, because TrustGraph’s core logic is deterministic and explainable, it avoids blindly trusting an AI detector’s guess; any AI-based signal is contextualized with other evidence.

- Embedded Watermarks and Metadata Tags: Some content authentication methods embed information into the content itself – for example, invisible watermarks in images or metadata fields in documents indicating origin. While useful for tracking provenance, these methods can be circumvented. Attackers can often remove or alter watermarks/metadata, or simply convert content to a format that strips them . Also, embedding data in content may not be feasible for all content types and can raise privacy issues (embedding creator identity, etc.). TrustGraph deliberately avoids in-content markers (the “zero-fabrication” approach) – the content remains pristine. The cryptographic binding and certificate ensure authenticity, but if someone tries to alter content or certificate, the mismatch is detectable. Unlike static watermarks, TrustGraph’s analysis also evaluates truthfulness (not just origin) and results in a dynamic score and explanation, which watermarking doesn’t provide.

- Blockchain Timestamping/Content Hashing: Some prior art involves recording a hash of content on a blockchain (or other timestamped ledger) to prove existence or timestamp of a particular version (e.g., “proof-of-existence” services). This can ensure a file wasn’t modified since a certain time, but does not verify any assertion about the content’s creation or truth. For example, putting a document’s hash on a blockchain proves the document existed at time X, but not whether “Alice wrote this document” is true or false. It also doesn’t flag plagiarism or AI involvement. TrustGraph incorporates cryptographic

hashing for integrity but goes much further by actually analyzing content and claims, and then packaging the hash within a certificate that also carries a trust evaluation. In essence, TrustGraph addresses the semantic truthfulness question which pure blockchain timestamping does not.

- Content Provenance Standards (CAI/C2PA): The Content Authenticity Initiative (CAI) and C2PA standard focus on capturing provenance metadata for media (who created it, editing history) and securing that with hashes and signatures. While robust for origin tracking, C2PA explicitly does not assert whether the content is “true” or accurate . It leaves the decision of trust to the user’s judgment of the source. In contrast, TrustGraph provides an evaluation of content truthfulness or claim validity. For instance, C2PA can tell you a photo came from XYZ camera and was signed by a news outlet, but it cannot tell if the photo actually depicts what it claims to. TrustGraph can evaluate a user’s claim about the photo (e.g., “This photo is unedited”) by analyzing the photo’s data for editing signs. Another difference: typical C2PA workflows do not verify the accuracy of metadata – they assume the signer (publisher) is honest . TrustGraph, however, is itself a verification engine that checks evidence objectively. Also notable, adoption of C2PA is currently minimal , whereas TrustGraph is designed to be easily integrable as an external service or API without requiring deep changes in content creation tools (making adoption potentially easier by tapping into existing content submission flows). In summary, TrustGraph complements provenance standards by focusing on assertion truth scoring, not just logging history.

- Academic and Forensic Analysis Methods: In academia and digital forensics, often a combination of manual checks and specialized software is used to assess content (e.g., instructors manually check citations, or forensic analysts use tools to check image metadata). These are labor-intensive and not standardized. TrustGraph offers an automated, standardized protocol that could replicate many of these checks deterministically and output a clear result. It reduces reliance on subjective human judgment by providing a numeric score and certificate that encapsulate the findings in a consistent manner.

- Reputation or Social Trust Systems: Some platforms (like Q&A sites or wikis) have user reputation scores or content trust metrics based on community feedback or edit histories. Those are domain-specific and not portable; they also don’t cryptographically assure anything and can be gamed or biased. TrustGraph is different in that it uses direct evidence analysis on content rather than communal voting or reputation. It produces a trust measure anchored in observable data, not opinions, and it is content-specific (assertion-level) rather than a generalized reputation. Additionally, TrustGraph’s cryptographic certification means its trust scores can be carried across platforms – a novel feature compared to localized trust systems.

In summary, no prior art combines: multi-faceted automated evidence analysis, probabilistic scoring with uncertainty, and generation of a verifiable cryptographic certificate for content trust. Plagiarism detectors, AI detectors, watermarking, provenance standards, etc., each address a part of the puzzle, but none provides an end-to-end trust protocol. TrustGraph's unique contributions include its integrated pipeline (signals → contradictions → probability → certificate) and its insistence on deterministic, explainable logic yielding a secure, shareable output. By addressing the weaknesses of prior approaches (lack of portability, lack of holistic analysis, susceptibility to tampering, absence of truth evaluation), TrustGraph establishes a new paradigm in content authenticity verification.

System Architecture and Data Flow (with Figure Descriptions)

To illustrate the system architecture and data flow, consider the conceptual diagrams represented by Figure 1 and Figure 2:

- Figure 1: High-Level System Architecture. This figure depicts the overall interaction in the TrustGraph system. On the left, a user or content platform submits a content item along with the user's assertion to the TrustGraph service. The TrustGraph service (middle) processes the input through its pipeline of modules (SDG, AIE, CCE, TSE, ZFP, CG) and returns a signed Trust Certificate as output . The Trust Certificate is stored externally – for example, it may be returned to the user and/or published to a public registry or ledger (TPR). On the right, third-party verifiers (anyone who later receives the content and certificate) can independently verify the trust certificate. Verification is done by using TrustGraph's public key to check the certificate's digital signature and recomputing the content's hash to ensure the certificate corresponds to the exact original content . The content itself is never altered and does not need to be trusted; trust is conveyed by the certificate. The figure underlines that TrustGraph operates as an external trust layer: it doesn't require integration into the content's native platform beyond providing the content and receiving the certificate, making it broadly applicable without platform modifications .

- Figure 2: TrustGraph Evaluation Pipeline Flowchart. This figure details the internal sequence of processing steps within TrustGraph. The content and assertion enter at the top of the pipeline and pass through six core modules in order: SDG → AIE → CCE → TSE → ZFP → CG . Each module's function is summarized alongside its block in the flowchart. SDG derives signals from the content/context; AIE evaluates contradictions; CCE computes the trust probability; TSE normalizes and labels the score; ZFP binds the result to the content via hashing; and CG signs and outputs the Trust Certificate. The data flow shows how the user's assertion and content are transformed step-by-step into a verifiable certificate . For instance, after AIE, the content+assertion becomes an integrity report; after CCE/TSE, it becomes a trust score with confidence; after ZFP/CG,

it becomes a signed certificate linked to the content. This modular flow highlights how each stage contributes to the final outcome in a logical progression, and how the design localizes different tasks into different components (improving maintainability and clarity).

(No actual drawings are included here; the descriptions above serve as explanatory reference for the conceptual figures.)

The above architecture demonstrates both the high-level usage (Figure 1) and the detailed internal workflow (Figure 2) of TrustGraph. In practice, these figures reflect that TrustGraph can be implemented as a cloud service (accessible via API to various platforms) or as on-premise software for an organization. The trust evaluation request/response pattern (content + assertion in, certificate out) means it can integrate into existing content submission systems with minimal friction. Data flow is secure: content can be hashed client-side if privacy is needed (i.e., only sending hash and signals to the service), and certificates can be verified offline without continuous connectivity to the service (except for retrieving public keys or checking a certificate's revocation status, if applicable).

Additionally, the architecture allows scaling: multiple instances of SDG or AIE can run in parallel for different content items; the TPR registry can be distributed for reliability; and standard cryptographic infrastructure (PKI) is leveraged for key management. The flowchart (Fig. 2) emphasizes the ordered, modular nature of processing – each step feeds the next, and clear interfaces exist between them, which also means parts of the system can be updated (e.g., a new AI detection algorithm in SDG) without overhauling the entire pipeline. This makes TrustGraph a flexible yet deterministic protocol suitable as foundational infrastructure for digital trust.

Real-World Use Cases

The TrustGraph protocol is broadly applicable across many domains where content authenticity and integrity are crucial. Below are several representative real-world use cases, demonstrating how the invention could be utilized:

- Educational Integrity (Academic Submissions): Universities and schools can integrate TrustGraph into their Learning Management Systems (LMS) to verify student work. For example, when a student submits an essay or assignment claiming originality, the system can automatically generate a TrustScore and certificate for that assertion. An academic integrity officer reviewing the work might see a TrustScore of, say, 0.96 with a narrow confidence interval labeled "High Trust – Original," indicating strong evidence the work is the student's own. If instead the score was 0.65 with a wide uncertainty labeled "Uncertain – Review Advised," the instructor knows to scrutinize the submission more closely or investigate further. TrustGraph certificates attached to assignments could serve as a credential of originality that students provide along with their work, deterring

plagiarism and cheating. This use case supports academic honesty policies by providing a consistent, technical measure of content integrity, reducing reliance on patchwork plagiarism checks and subjective judgment.

- Government and Public Sector Auditing: Government agencies often need to authenticate documents, images, or data that are submitted for official purposes. For instance, consider a scenario in which a contractor submits a report claiming “All data in this report is original and unaltered.” A government auditor can use TrustGraph to evaluate that claim. If a Trust Certificate is provided with the report, the auditor can verify it to see if the claim holds (e.g., TrustScore indicating no signs of tampering or fabrication). In internal use, agencies could run TrustGraph on documents to ensure they haven’t been doctored – important for preventing fraud in procurement, scientific research submissions, or legal evidence. Similarly, a government archive might require that any digital photograph or video submitted as evidence comes with a trust certificate verifying it hasn’t been manipulated (especially relevant as deepfakes become a concern). Governments can also use TrustGraph to support compliance with emerging regulations that mandate disclosure of AI-generated content. For example, if new laws require that public communications flag AI involvement, an agency can verify a certificate that explicitly states “No AI content” with high confidence before publishing an official statement.

- Publishing and Media Compliance: In journalism, academic publishing, and media, ensuring content provenance and integrity is paramount. TrustGraph can assist publishers in vetting submissions. For a news outlet, an freelance contributor might be required to submit a TrustGraph certificate with any contributed photo claiming “This photo is unedited” or any article claiming “Original content.” The editors can quickly check the certificate: a high trust score would expedite acceptance, while a low score or evidence of contradictions (like plagiarism or AI text) would trigger investigation or rejection. Scientific journals could similarly use TrustGraph for papers where authors assert “All results are original and no part of the text is plagiarized.” This can streamline the editorial review process by highlighting potential issues (e.g., unacknowledged overlaps with other papers, or AI-generated text in an era where AI writing assistance is common). Compliance regimes: With the rise of global frameworks emphasizing trustworthy AI and content (such as guidelines for ethical AI usage or the EU’s proposed AI Act requiring transparency for AI-generated content), TrustGraph provides a tool for compliance. Publishers can demonstrate due diligence by using an independent protocol to verify claims like “no AI used in writing this article,” aligning with industry best practices for transparency. Moreover, TrustGraph’s detailed outputs (reason codes and summaries) give compliance officers concrete evidence (for example, a certificate might include a reason code indicating some AI usage was detected, prompting either a correction or an explicit disclosure to meet guidelines).

- Enterprise Document Authentication: Companies producing critical documents (financial reports, technical documentation, software code) can use TrustGraph internally to

ensure authenticity and accountability. For instance, consider an engineering firm where employees submit design files or code with assertions like “This code contains no open-source components without approval.” TrustGraph could analyze the code (signals might include checking against open-source repositories) and issue a certificate. During audits or handover to clients, the firm can present these certificates to demonstrate compliance with IP and licensing policies. Another scenario is corporate data integrity: if a company issues a press release claiming “No information in this release is fabricated,” they can attach a certificate that vouches for the authenticity of included data (perhaps signals cross-check data against source databases). Enterprises can also integrate trust checks in workflows—imagine a content management system where any file uploaded with certain claims is automatically processed by TrustGraph, and files failing trust thresholds are flagged for manager review. This fosters a culture of integrity and can catch issues (like an employee trying to pass AI-generated content as their own work product) early.

- User-Generated Content Platforms: (While not explicitly requested, it’s a natural extension) Social media and online communities could empower users to verify their own content. For example, an artist uploading a digital artwork could include an assertion “This art is my original creation, not AI-generated.” The platform might offer a “Verify with TrustGraph” feature that generates a trust certificate for that post. Viewers of the art could then see a badge or link to the certificate, lending credibility in an era where AI-generated art is common and authenticity is questioned. Similarly, a blogger could use a plugin to get TrustGraph certificates for their articles, which readers or advertisers can verify to ensure content originality. This democratizes trust and could reduce misinformation by making it easier for content creators to prove when they are truthful.

Each of these use cases demonstrates the versatility of TrustGraph as infrastructure. It is not a consumer-facing product by itself; rather, it would typically operate under the hood or as a background service to enhance existing systems with a layer of trust verification. In all cases, the invention’s key strengths are evident: it provides consistent, objective, and verifiable trust evaluations that can save time (automating what were manual checks), improve integrity (catching deception or errors early), and facilitate compliance (generating evidence that due process was followed). By integrating into real-world content workflows, TrustGraph can improve confidence among stakeholders – be it teachers trusting student work, editors trusting submissions, or the public trusting official content – ultimately contributing to a more trustworthy digital environment.

Claims (System and Method)

Claim 1. A system for evaluating trust in a user-declared assertion about a digital content item, the system comprising:

- a Signal Derivation Graph (SDG) module configured to collect and normalize a plurality of observable signals from the content item and its context, the signals including content-intrinsic features, user behavior data, and environmental metadata;

- an Assertion Integrity Engine (AIE) configured to analyze the normalized signals relative to the user's assertion to identify any contradictions between the signals and the assertion, and to generate an integrity score quantifying consistency of the assertion with the observed signals;

- a Confidence Computation Engine (CCE) configured to compute a probabilistic trust score representing a likelihood that the user's assertion is truthful, based on the integrity score and predefined statistical or machine learning models associated with the type of the assertion, the trust score including a confidence interval indicating uncertainty of the computation;

- a TrustScore Engine (TSE) configured to normalize and format the trust score and associated confidence data into a standardized output, and to apply one or more predefined policies or thresholds to categorize the trust score with qualitative labels for interpretability;

- a Zero-Fabrication Protocol (ZFP) module configured to generate a cryptographic binding between the content item and the trust score without modifying the content item, by computing a content identifier comprising a cryptographic hash of the content item and combining the content identifier with the trust score or an identifier thereof to produce a binding token; and

- a Certificate Generator (CG) configured to create a digital trust certificate containing at least the content identifier, the user's assertion, and the formatted trust score, and to digitally sign the trust certificate with a private cryptographic key to produce a verifiable, tamper-evident certificate.

Claim 2. The system of claim 1, wherein the SDG module collects signals including at least one of: content-intrinsic textual features, similarity matches against known content sources, user editing or revision history data, and environmental context metadata related to the content's creation.

Claim 3. The system of claim 1, wherein the Assertion Integrity Engine applies a set of assertion-specific rules or machine learning models to determine whether each collected signal supports or contradicts the assertion, and assigns a weight to each identified contradiction based on its severity, thereby producing a contradiction analysis comprising a list of contradictions with associated weights.

Claim 4. The system of claim 1, wherein the Confidence Computation Engine employs a Bayesian inference model to update a prior probability of truthfulness of the assertion in view of the integrity score, such that the trust score is higher when fewer or no contradictions are present and lower when multiple strong contradictions are present.

Claim 5. The system of claim 1, wherein the Certificate Generator is further configured to include in the trust certificate a timestamp of evaluation and a digital signature generated using an asymmetric private key, such that authenticity of the certificate is verifiable via a corresponding public key and any post-issuance alteration of the certificate's content is detectable.

Claim 6. The system of claim 1, wherein the trust certificate comprises a structured data format that includes: the cryptographic hash of the content item, an identifier of the assertion or assertion type, the numeric trust score with its confidence interval, a summary of any detected contradictions, and the digital signature of an issuing authority.

Claim 7. The system of claim 1, further comprising a certificate registry accessible via a network, wherein the Certificate Generator submits each signed trust certificate to the registry for storage and later retrieval or verification by third parties.

Claim 8. A computer-implemented method for verifying a user's assertion about a digital content item, the method comprising:

- receiving, by a trust evaluation server, the digital content item and a declarative assertion made by a user regarding the content item;
- collecting a plurality of signals related to the content item and its context, including features extracted from the content item and metadata or environmental information about its creation and submission;
- analyzing the collected signals to identify any contradictions with the user's assertion by applying assertion-specific rules or models to the signals, and computing an integrity score that represents the degree of consistency between the observed signals and the assertion;
- computing, based on the identified contradictions, a probabilistic trust score indicating a likelihood that the assertion is true, the computing including applying a statistical inference model that adjusts a prior probability of truthfulness according to a magnitude of a contradiction score, and determining a confidence interval reflecting uncertainty in the trust score;

- normalizing and formatting the trust score and associated metadata into a standardized output, including assigning at least one qualitative label to the trust score based on predefined threshold criteria;

- generating a cryptographic binding between the trust score and the content item without altering the content item, by computing a cryptographic hash of the content item and combining the hash with data representing the trust score to produce a binding identifier;

- creating a digital trust certificate comprising the content item's cryptographic hash, the user's assertion, and the normalized trust score, and digitally signing the trust certificate using a private key associated with a trust authority to yield a signed trust certificate; and

- providing the signed trust certificate to one or more relying entities, thereby enabling independent verification of the assertion's evaluated trustworthiness by any third party in possession of the content item and a corresponding public key for signature verification.

Claim 9. The method of claim 8, wherein collecting the plurality of signals includes extracting content-intrinsic textual features and checking the content item against external reference databases for plagiarism or prior publication, monitoring user editing behavior and revision history during content creation, and capturing environmental data such as the user's IP address or device type at time of submission.

Claim 10. The method of claim 8, wherein analyzing the collected signals comprises using a machine learning classifier or a rule-based engine tailored to the assertion type to flag discrepancies, including detecting copied content when originality is claimed or detecting linguistic patterns of AI-generated text when human authorship is claimed, each flagged discrepancy contributing to a contradiction score inversely correlated with the trust score.

Claim 11. The method of claim 8, wherein computing the probabilistic trust score includes applying Bayesian updating to adjust a base probability of truthfulness based on the integrity score, such that the trust score remains high if the integrity score indicates no substantial contradictions and is significantly reduced if the integrity score indicates multiple strong contradictions.

Claim 12. The method of claim 8, wherein generating the cryptographic binding comprises concatenating the cryptographic hash of the content item with a unique evaluation identifier and a nonce, and hashing the combination to produce the binding identifier that uniquely links the trust certificate to that specific content item and evaluation instance.

Claim 13. The method of claim 8, wherein the digital trust certificate is formatted as a machine-readable data structure containing at least the content item's cryptographic hash, the user's assertion, the trust score with its confidence interval, and a digital signature, and wherein verifying the trust certificate by a third party comprises: validating the digital signature using a

corresponding public key to confirm the certificate's authenticity, and recomputing a hash of the digital content item to confirm it matches the content hash recorded in the certificate.

Claim 14. The method of claim 8, further comprising storing the signed trust certificate in a network-accessible registry, and upon receiving a verification query from a remote verifier including a content identifier, retrieving the corresponding trust certificate and enabling the verifier to validate the certificate's signature and compare the content's hash to the certificate's content identifier without requiring access to the original issuing server.

Claim 15. The method of claim 8, wherein the trust certificate remains separate from the content item such that the content item can be distributed or transmitted without modification, and the trust certificate can be presented or transferred alongside the content item as cryptographic proof of the content's authenticity and the validity of the user's assertion.

Claim 16. The method of claim 8, further comprising performing a secondary inference analysis using an Assertion Inference Engine (AIE-2) that correlates the evaluated content item and its trust certificate with external data or other content items to refine or contextualize the trust score, wherein said inference identifies cross-document inconsistencies or corroborating evidence beyond the original content item's signals, and updates the trust certificate with any additional reason codes or metadata reflecting said inference.

Claim 17. The method of claim 8, wherein the digital trust certificate further includes one or more Trust Certificate Reason Codes (TCRC) corresponding to categories of evidence or checks performed, each reason code indicating a factor that influenced the trust score, thereby providing an explainable context for the trust determination to relying parties.

Claim 18. The method of claim 13, wherein the machine-readable data structure conforms to a predefined TrustGraph Markup Format (TGMF) such that the certificate can be embedded, exchanged, or parsed uniformly across different platforms, the format specifying standard fields for content identity, assertion details, trust score, reason codes, and signature.

Claim 19. The method of claim 8, further comprising generating a compliance header or metadata tag associated with the content item, the header including a reference to the trust certificate or trust score in accordance with a Compliance Header Standard (CHS), thereby enabling external systems or protocols to automatically identify that the content item has an associated trust verification and to process it according to compliance policies.

Claim 20. The method of claim 8, further comprising obtaining explicit user consent for performing the trust evaluation on the content item and recording traceability information for the evaluation process, including logging of evaluation events with timestamps and identifiers, such that the trust evaluation is auditable and complies with data governance requirements via a Consent & Traceability Module (CTM).

Glossary and Definitions

- Assertion: A user-provided declarative statement about a digital content item whose truthfulness is to be evaluated. It is the claim made by the user that TrustGraph will verify. Examples: “This essay is entirely my own work,” “No part of this image is AI-generated.” The assertion is always associated with a specific content item.

- TrustGraph: The name of the proposed protocol and system for evaluating trust in user assertions about content. It encompasses the entire pipeline of modules (SDG, AIE, CCE, TSE, ZFP, CG) that collectively produce a TrustScore and Trust Certificate. It also refers to the broader ecosystem introduced by this invention, including data formats (TGMF), public registries (TPR), and compliance modules (CHS, CTM).

- SDG (Signal Derivation Graph): The component responsible for signal collection and normalization. SDG gathers various observable signals from the content and its creation context, such as content features, metadata, user behavior logs, and environmental information. Note: In earlier drafts this was also called “Signal Data Grid,” but Signal Derivation Graph is used herein to emphasize the structured relationship of signals. SDG provides the raw evidence for the trust evaluation.

- AIE (Assertion Integrity Engine): The component that performs contradiction analysis. AIE takes the signals from SDG and the user’s assertion, and checks for contradictions or consistencies between them. It outputs an Integrity Score (or equivalently a contradiction score) and details of any contradictions found (the Contradiction Analysis Summary).

- CCE (Confidence Computation Engine): The component that computes the TrustScore. Using the integrity score and possibly detailed evidence from AIE, CCE applies statistical inference (e.g., Bayesian updating or other probabilistic models) to estimate the probability that the assertion is true, and also generates a confidence interval for that estimate.

- TSE (TrustScore Engine): The component that normalizes and formats the trust score for output. It scales the score to a standard range (e.g., 0 to 1 or percentage), attaches qualitative labels (like “High Trust” or “Low Trust”) based on defined thresholds, and enforces any output policies. Essentially, TSE prepares the human-readable and standardized result that will be certified.

- ZFP (Zero-Fabrication Protocol): The component/method that binds the trust result to the content externally (without altering the content). It computes a cryptographic hash of the content and combines it with the TrustScore (or its ID) and a nonce to create a binding token. “Zero-fabrication” (or zero-footprint) means nothing is embedded into the content file; the binding is completely external via cryptographic references. This ensures tamper-evident linkage between content and its trust data.

- CG (Certificate Generator): The component that generates and digitally signs the Trust Certificate. It compiles the assertion, TrustScore (with confidence and label), content hash, timestamp, etc., into a structured certificate, then uses a private key to sign it. This signature enables others to verify the certificate's authenticity (using the corresponding public key).

- TrustScore: The probability (e.g., 0 to 1, or 0% to 100%) that the user's assertion is true, as determined by the system. It reflects the system's confidence in the assertion's truthfulness. For example, a TrustScore of 0.90 indicates a 90% estimated likelihood that the claim is true (usually accompanied by a confidence interval).

- Confidence Interval: A statistical range indicating the uncertainty around the TrustScore. For instance, a TrustScore might be reported as 0.8 ± 0.1, meaning the system is (say) 95% confident that the true probability lies between 0.7 and 0.9. A narrow interval implies high certainty; a wide interval implies more uncertainty due to limited or noisy evidence.

- Integrity Score (or Contradiction Score): A numerical value output by AIE representing how consistent the evidence was with the assertion. A high integrity score means evidence largely supports the assertion; a high contradiction score means evidence strongly conflicts with it. This feeds into the TrustScore calculation in CCE.

- Content Artifact: The digital content item under evaluation (document, image, video, code, etc.). Often identified by a cryptographic hash for purposes of the certificate and binding, so that the content can be referred to without embedding it.

- Cryptographic Hash: A one-way function that takes an input (like a file) and produces a fixed-size output string (the hash) such that even a slight change in input yields a completely different hash. Hashes (e.g., SHA-256) are used in TrustGraph to uniquely identify content without needing the actual content in the certificate.

- Digital Signature: A cryptographic scheme where a private key is used to sign data and a corresponding public key is used to verify the signature. In TrustGraph, the certificate is digitally signed by the issuer's private key. Anyone with the corresponding public key can verify that signature to confirm the certificate's integrity and origin.

- Trust Certificate: The output data package (formatted per TGMF, e.g., JSON or XML) that contains the details of the trust evaluation (content hash, assertion, TrustScore, timestamp, etc.) and is digitally signed. It serves as a portable proof that the evaluation was performed and what the results were. Third parties can use this certificate to verify content assertions without re-running the analysis, by checking the signature and content hash.

- Certificate Registry (TrustGraph Public Registry – TPR): An optional database or ledger where Trust Certificates are stored or indexed for lookup. This can be centralized or decentralized (e.g., blockchain-based) and allows querying for certificates by content hash or certificate ID. It also assists in certificate revocation or audit processes by providing a public record of issued certificates.

- PKI (Public Key Infrastructure): The framework for issuing and managing digital certificates and keys. TrustGraph can use a PKI model where the TrustGraph service has a public/private key pair (and possibly the public key is itself certified by a higher authority). Verifiers need the public key (or a chain of trust from a known Certificate Authority) to verify TrustGraph certificates.

- TGMF (TrustGraph Markup Format): The standardized format for TrustGraph data, particularly certificates. It defines how trust information is structured (fields like content_hash, assertion, trust_score, etc.) and ensures interoperability between different systems reading or writing TrustGraph certificates.

- SPP (Signal Provider Protocol): The interface/protocol by which external signal providers can feed signals into the TrustGraph system. SPP defines APIs or data formats for third-party modules (e.g., plagiarism databases, AI detectors) to submit analysis results to SDG/AIE in a consistent manner.

- TPR (TrustGraph Public Registry): The public repository or ledger where TrustGraph certificates can be published and queried. It's the implementation of the certificate registry concept, possibly using blockchain or a distributed database, to allow independent lookup of certificate records and ensure longevity of trust data.

- AIE-2 (Assertion Inference Engine, second-tier): An advanced engine that performs inference across multiple assertions or content items, going beyond single-item analysis. AIE-2 might identify patterns, conflicts, or corroborations in a broader context (e.g., cross-document consistency or detecting if two users claimed originality on the same content). Its outputs can refine trust assessments or provide additional reason codes.

- TCRC (Trust Certificate Reason Codes): Standardized codes included in certificates that denote reasons or factors affecting the TrustScore. They map to categories of evidence (like plagiarism_detected, ai_generated_likely, metadata_mismatch, etc.), providing a quick explanation of why the score is what it is. TCRCs make the certificate more self-explanatory and facilitate automated handling by other systems.

- CHS (Compliance Header Standard): A defined standard for representing trust info in content headers or metadata (such as HTTP headers, email headers, or file metadata). CHS allows content that has a TrustGraph certificate to carry an indicator or reference within the content's transport or storage format, enabling automatic detection and

retrieval of trust data by compatible systems.

- CTM (Consent & Traceability Module): A component/module focusing on obtaining user consent before analysis and logging all actions for traceability. It ensures the TrustGraph process is used ethically and that all trust evaluations are auditable. CTM might handle user opt-in management and maintain an immutable log of evaluations (recording who, when, what was claimed, and the result).

- C2PA: Stands for Coalition for Content Provenance and Authenticity. A standard focused on embedding provenance info (origin, edit history) in content. It is mentioned in prior art for comparison purposes and is not part of TrustGraph's system, but is relevant to the general landscape of content authenticity solutions.

- Plagiarism Detection: Software tools or algorithms that check text against sources to find copied content. In TrustGraph, such a tool can provide signals to SDG/AIE for originality claims (e.g., indicating what portion of text matches existing sources).

- AI Detector: A tool or model that attempts to determine if content (usually text, sometimes images) was generated by AI. In TrustGraph, the output of such detectors can be used as one signal within SDG, particularly for assertions of the type "not AI-generated."

Each of these terms is defined and used consistently throughout this document to describe the invention. This glossary is intended to clarify the specific meaning of each term within the context of TrustGraph.

Appendix

No formal drawings are included in this provisional specification. Figures 1 and 2 are described conceptually in Section 8 to illustrate the architecture and data flow. In a full non-provisional filing, schematic diagrams corresponding to those figures may be provided. Additionally, while example data structures (such as a JSON certificate example) are given in the Detailed Description, no separate code listings or flowcharts are attached in this appendix. The described protocol and components above suffice to enable implementation by one skilled in the art. Any further examples or implementation notes can be appended in a subsequent filing if necessary.