

GovBIND: From Authenticated Fetch to Zero-Knowledge Predicate over Government Documents

Yunus Gürlek¹[0009–0001–1867–8851]* and Ahmet Ramazan Ağirtaş²[0000–0002–4574–0067]

¹ Sorbonne University

² AYWARE, Ankara, Türkiye

yunus@etu.sorbonne-universite.fr, a.r.agirtas@gmail.com

Abstract. Government portals often issue PDF certificates whose provenance can be verified only through a live HTTPS verification service. Proving a limited fact from such a document therefore commonly requires disclosing the complete document or giving the verifier access to the portal. We present GovBIND, a protocol that combines zkTLS provenance with a zero-knowledge content proof, without requiring the issuer to modify its service or embed an independently verifiable signature. A baseline construction binds both stages through a commitment to the complete response body. To make client-side proving feasible for compressed PDFs, an optimized construction instead commits separately to private ranges, discloses the remaining bytes, requires their exact partition, and opens the claim-bearing compressed stream in a document profile-specific proof that constrains DEFLATE decompression. We formalize GovBIND with game-based definitions of correctness, soundness, and privacy, where soundness jointly captures authenticity, content validity, and binding, and we prove both constructions secure relative to the explicit leakage and trust assumptions of Proxy-mode TLSNotary. Our prototype supports proofs of no overdue tax debt, bounded traffic penalties, and residence city over documents returned by the Turkish government verification service, e-Devlet. Across 72 generated proofs in 24 sessions, end-to-end generation took 54-119 seconds, verification took 44-127 milliseconds, and proofs were 11-12 KiB.

Keywords: zkTLS · TLSNotary · zero-knowledge proofs · selective disclosure · government documents · PDF

1 Introduction

Online government services are increasingly used by citizens to obtain digital certificates useful in many contexts, from proving the residency address to the lack of a tax debt. These certificates can often be verified using signatures, barcodes, or similar online verification mechanisms. However, proving a single

* Corresponding author.

fact from such a document, e.g., that an address is in a particular city or that public tax debt is less than a threshold, still typically requires disclosing the entire document or granting access to the government portal. Both options reveal more information than the claim requires and create unnecessary data liability.

There are some partial solutions to this problem, such as the EU’s eIDAS 2.0 European Digital Identity Wallet, which addresses it by asking a government to issue a selectively disclosable credential [11]. However, this solution requires the government to adapt its systems to support additional cryptographic mechanisms, which is not the same as a citizen extracting a claim from a document already issued.

Another approach verifies an embedded issuer signature of a PDF document inside a zero-knowledge circuit and then proves claims about the signed content [26,1]. This approach requires no interaction with the issuer, but it applies only to documents that carry an independently verifiable signature, which is not always the case. In particular, the Turkish government documents considered in this work rely on online barcode verification and do not expose such a signature in the PDF structure.

A third approach to this problem is zkTLS, which allows a party to authenticate data as coming from a TLS server while selectively disclosing its content [35,36,14]. On its own, zkTLS cannot prove arbitrary claims about the content of a document, but some solutions propose combining zkTLS attestations with additional proofs for such claims. In particular, DECO proposes a combined solution by authenticating a TLS response, committing to its plaintext, and proving a predicate on the same commitment [36].

In this work, we introduce GOVBIND, a protocol that applies this commit-and-prove pattern to government-issued PDF documents to selectively disclose and prove various claims over the content. Unlike existing solutions, GOVBIND can be used in government documents that do not have a self-authenticating signature, without requiring any issuer-side modification. We call the complete response construction of GOVBIND the *Baseline Protocol*, where a zkTLS presentation authenticates a commitment to the entire response body, and the content proof opens that commitment as a public input. We further propose an optimized construction based on selective-range commitments, named the *Optimized Protocol*. This optimization separately commits to sensitive PDF ranges, reveals every remaining body byte, and requires the verifier to check an exact partition. The Optimized Protocol additionally constrains DEFLATE decompression and a versioned document profile in the content proof. Both constructions require an authenticated government session at proof-generation time, unlike approaches based on a self-authenticating downloaded document. We implement the approach for Türkiye’s e-Devlet platform.³

Related work. Town Crier [35] authenticates web data using trusted hardware, while DECO [36] introduces a three-party TLS protocol and the commit-and-prove construction that GOVBIND adopts. DECO also addresses *context in-*

³ <https://www.turkiye.gov.tr>

egrity, i.e., ensuring that a revealed or proved value occupies the intended position in a structured response, through selective opening and two-stage parsing of text-based formats. Garble-then-Prove [33] reduces the cost of such TLS provenance proofs. Subsequent TLS-oracle systems explore different security and performance points: Janus [16] optimizes privacy-preserving TLS provenance, ORIGO [9] targets constant online communication, and DiStefano [5] provides maliciously secure private commitments for TLS 1.3. In [18], Luo et al. formalize proxy-only TLS oracles and show that their integrity depends on application semantics or AEAD context unforgeability. TLSNotary [31,14] provides both MPC and Proxy modes for zkTLS, while Primus [23], Reclaim [24], zkPass [37], and Opacity [21] provide related designs, several of which combine attestations with predicates over the response.

On the other hand, various works focus on proving issued documents. Cinderella proves policies over X.509 certificate chains [8], zk-creds converts existing identity documents into anonymous credentials [25], and Crescent adds selective disclosure to signed JWT and mDL credentials without issuer changes [22]. ZK Email similarly proves predicates over DKIM-authenticated messages [17]. For government documents, zkPDF [26] parses PDFs, including their compressed content streams, inside a zkVM and proves predicates over CMS-signed PDFs [13], and Anon Aadhaar [1] verifies signed QR data in zero knowledge. These systems motivate reuse of existing infrastructure, but require a signature or authenticated object that can be verified independently of a live government response. The EUDI Wallet [11,10] takes an issuer-coordinated route based on standardized credential formats.

Our contribution. We apply DECO’s commit-and-prove pattern to compressed binary documents, i.e., PDF files, returned by government verification services that do not have independently verifiable signature. Existing context-integrity techniques target text-based responses instead of PDFs, and existing PDF proofs require an embedded issuer signature. To bridge this gap, we propose an Optimized Protocol that commits separately to private ranges, reveals the remaining bytes, and requires the Verifier to check an exact partition of the response against a versioned document profile. As a reference construction that isolates the binding argument, we also give a Baseline Protocol that commits to the complete response. We give a formal model of GOVBIND, including an abstraction of TLSNotary as an attested-transcript scheme and document profiles that capture both constructions, and prove correctness, soundness, and privacy through game-based arguments. We implement both constructions⁴ for Türkiye’s governmental issuer platform, e-Devlet. We further demonstrate and benchmark the applicability of the Optimized Protocol across three examples, i.e., tax debt, traffic record, and residency.

Organization of the paper. Section 2 reviews the cryptographic building blocks and zkTLS. Section 3 defines the parties, trust assumptions, and adversary

⁴ Available under <https://github.com/FAME-CRYPT/GovBind>

model, and gives the formal model of GOVBIND. Section 4 describes the Baseline Protocol and its binding mechanism. Section 5 proves the correctness, soundness, and privacy of that protocol with respect to the formal model. Section 6 describes the implementation, the Optimized Protocol, and its security properties. Section 7 discusses limitations and future work, and Section 8 concludes.

2 Preliminaries

We write λ for the security parameter, $\text{negl}(\lambda)$ for a negligible function, and $\parallel$ for byte-string concatenation. All adversaries are probabilistic polynomial-time algorithms. We use standard cryptographic definitions from [15].

Hash functions and commitments. We require a collision-resistant hash function H . We use the blinded hash commitment

$$\text{Com}(m; r) = H(m \parallel r),$$

where $r \xleftarrow{\$} \{0, 1\}^{\lambda_r}$ is a fixed length secret random blinder. Collision resistance provides binding, while the secrecy and entropy of r prevent direct guessing of the committed message. We assume that this construction is computationally hiding for a uniformly sampled secret blinder, as it is in the random-oracle model. Our construction instantiates H with SHA-256 [19]. This instantiation is inherited from TLSNotary. Since λ_r is fixed and the committed length is public, binding reduces to the collision resistance of SHA-256. Hiding is argued in the random-oracle model, which is a heuristic for SHA-256 because of its Merkle-Damgård structure.

TLS (Transport Layer Security). We abstract TLS as a server-authenticated secure channel between a client and a server with service identity d . After successful validation of the certificate path and the service identity, the client obtains integrity-protected and confidential byte string m from the endpoint identified by d [7,27]. We denote an encrypted TLS transcript by $\mathbf{E}$ and the secret required to recover its plaintext by k , writing $\text{TLS.Dec}_k(\mathbf{E}) = m$. We assume that, except with negligible probability, an adversary cannot cause an honest participant to accept modified or attacker-generated bytes as having been received from d , or learn secret plaintext without having k . These guarantees are local to the TLS participants, i.e., an ordinary TLS session does not by itself provide a transferable proof about the authenticity or integrity of the server response to a third party. The TLSNotary protocol described below includes this additional provenance mechanism.

Zero-Knowledge Proofs. Let $\mathcal{R}$ be an efficiently decidable relation over a public statement $\mathfrak{x}$ and private witness $\mathfrak{w}$. We use a non-interactive zero-knowledge argument of knowledge for $\mathcal{R}$ with proving key $pk_{\mathcal{R}}$ and verification key $vk_{\mathcal{R}}$ [12,3]. We require:

- *Completeness*: if $\mathcal{R}(x; w) = 1$, an honestly generated proof is accepted.
- *Knowledge soundness*: except with negligible probability, a prover cannot produce an accepting proof without knowing a witness w such that $\mathcal{R}(x; w) = 1$.
- *Zero knowledge*: the proof reveals no information about w beyond the public statement and the fact that the relation is satisfied.
- *Succinctness*: the proof size and verification cost are small relative to the computation represented by $\mathcal{R}$.

We use the conventional term *proof* throughout the paper, although the computational soundness guarantee formally makes the employed system an argument of knowledge.

zkTLS. zkTLS augments the session-local guarantees of TLS with selectively disclosable provenance evidence. In general, zkTLS systems allow a client (the *prover*) to authenticate data received from a server during a TLS session without revealing the complete plaintext transcript. In this paper, we use and extend TLSNotary [31], a specific implementation of such a protocol.

In TLSNotary, the prover and a distinguished *notary* first jointly execute the TLS protocol (detailed below). Then, the notary issues a signed attestation binding the authenticated server identity to commitments over the TLS transcript. Finally, the prover derives a presentation that selectively reveals transcript ranges while leaving the remaining ranges committed. By verifying the signed attestation, the presentation, and the server identity, any third-party, i.e., *verifier* may conclude that the disclosed bytes are authenticated by the respective server.

TLSNotary supports two modes for jointly executing the TLS protocol: MPC and Proxy. In MPC mode, the prover and notary jointly perform the cryptographic operations required for the TLS session using multi-party computation. In Proxy mode, the notary instead forwards the encrypted traffic between the prover and server, reducing the cost while introducing a different trust model. In both modes, the notary does not learn the undisclosed plaintext [14,30]. Our implementation uses Proxy mode.

It is important to note that TLSNotary treats the authenticated transcript as raw bytes and does not interpret its application-level structure. This is especially useful in our setting because the transcript can contain a binary PDF rather than a structured API response. GOVBIND interprets the authenticated bytes separately and uses their commitments to bind the TLS provenance evidence to the content proof described in Section 4.

3 System and Threat Model

3.1 Parties

In the protocol, we assume the following parties, whose interactions are depicted in Fig. 1:

1. Prover is the party that holds a government document or the information required to access its public verification service. Depending on the government system, this verification information may include a barcode, an identification number, or similar values. The Prover retrieves the authenticated document and generates a proof over its content.
2. Government Server is the designated HTTPS endpoint that returns the document to be authenticated. It does not need to be modified to support GOVBIND.
3. Notary is the party that participates in the TLSNotary protocol with the Prover and issues a signed attestation binding the Government Server’s identity to commitments over the TLS transcript. The Notary has a key pair $\langle sk_{notary}, pk_{notary} \rangle$. Its trust assumptions are described in more detail later.
4. Verifier is any party who independently verifies the proving protocol. It does not have prior knowledge of the government document, but it knows the approved Government Server, the Notary’s public key pk_{notary} , and the public parameters described below.

We take the Government Server as the publicly known authoritative source for the statement being proved. The protocol does not prove the internal execution of the governmental system or the factual correctness of its statement. Instead, GOVBIND proves that the designated Government Server served particular content and that this content satisfies the proved predicate.

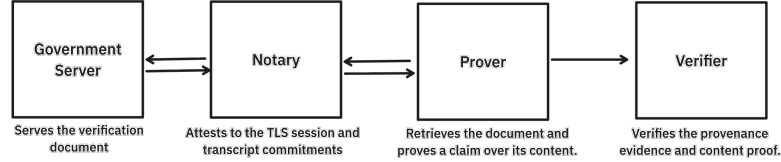


Fig. 1. Parties and their interactions in GOVBIND. The diagram assumes Proxy mode and abstracts most protocol steps.

3.2 Trust Assumptions

In order to verify a claim made by the Prover, the Verifier is required to:

1. know the approved Government Server’s DNS name and accept it as authoritative for the relevant statement;
2. rely on correct TLS certificate validation and trusted public-key infrastructure;
3. trust the Notary under the assumptions of TLSNotary’s Proxy mode and know its public key pk_{notary} ;
4. know the approved ZK relation, verification key or program identity, document profile, disclosure policy, and freshness policy;

5. rely on the cryptographic assumptions stated in Section 2; and
6. assume that the connection between the Notary and the Government Server is network-authentic, as discussed below.

Knowing the Government Server’s domain can be considered a lightweight assumption because governmental domains are generally public and stable. DNS is required to locate this server, but the authenticity of the server is established through TLS service-identity and certificate-path validation rather than DNS alone [27,7]. In ordinary TLS, a malicious DNS response may prevent or redirect the connection, but it cannot impersonate the approved domain without also defeating TLS authentication. This argument does not carry over to Proxy mode. There, the Prover holds the TLS session keys, including the server write key, and can therefore compute valid server records itself. The authenticity of the response thus relies on the Notary observing the traffic of the genuine Government Server. A Prover that controls the Notary’s name resolution or network route could relay the genuine handshake, so that certificate validation succeeds, and then inject records of its choice. We therefore assume that the connection between the Notary and the Government Server is network-authentic, i.e., the Prover can neither influence the Notary’s resolution of d nor inject traffic into the Notary-server connection, in line with the analysis of proxy-based TLS oracles in [18]. Introducing a separate signing key for the government service could provide another form of authentication but would require changing the government’s systems, contrary to one of the main goals of GOVBIND.

We do not assume that a statement published by the government system is factually correct. The Verifier only accepts the Government Server as authoritative for making that statement. GOVBIND proves what the Government Server published, not whether the statement accurately represents facts outside that system.

On the other hand, trusting the Notary introduces a greater risk, as GOVBIND relies on the Notary to honestly perform its role and attest only to sessions it observed with the authenticated Government Server. A malicious Notary could forge provenance evidence as if it came from the Government Server.

Here, it is important to emphasize that the honesty assumption over Notary is a distinct problem from the resistance of the Notary to a malicious Prover who lies about the plaintext of an observed ciphertext. This resistance is directly connected to the TLSNotary execution mode that we adapt, which is the Proxy mode [30]. The integrity of the proxy-only TLS-oracle can depend on semantics of the application and the negotiated AEAD. In particular, Luo et al. show that ChaCha20-Poly1305 satisfies their notion of context-unforgeability, while AES-GCM does not [18].

Moreover, use of the TLSNotary Proxy implementation does not ask the Notary to sign an arbitrary plaintext commitment supplied by the Prover. The joint zkTLS proof computation commits to the Prover’s private TLS secret, derives the record keys, verifies the authentication tags of received records, and checks the disclosed and committed plaintext against the recorded ciphertext. Then, the range commitments are computed on those same authenticated plaintext

references, using fresh secret blinders [31]. Since the record keys are derived inside the proof from the committed TLS secret and the handshake observed by the Notary, the Prover cannot open the recorded ciphertext under a key other than the session key. The integrity of the received records therefore does not rely on the context unforgeability of the AEAD, and the negative result of [18] for AES-GCM does not apply to this setting. We nevertheless treat the soundness of TLSNotary as an explicit imported assumption in Definition 3.

Nevertheless, these observations only apply to the resistance of Notary to a malicious Prover but do not handle the problem of a malicious Notary. We consider four possible approaches to reducing or relocating trust in the Notary.

The first solution is to have the Notary authorized, for example, through a certificate by the relevant government. This simplifies the Verifier’s decision about which Notary key to accept, although the Notary must still operate honestly. However, government authorization requires government cooperation and weakens the goal of deployment without issuer-side modification. Thus, we do not rely on this option.

A more cryptography-native solution is to run the Notary within a trusted execution environment (TEE), as demonstrated by Town Crier to authenticate data obtained from public sources [35]. This replaces some trust in the Notary operator with trust in the hardware and manufacturer. However, we consider that trust in hardware weakens the benefit of a publicly verifiable proving system. We therefore do not rely on a TEE in the protocol described here.

In a blockchain-oriented deployment, the Notary may instead be represented by a committee using a BFT protocol and quorum attestations. This is similar to distributed oracle networks such as Chainlink’s Offchain Reporting protocol [4]. Such a construction requires extending TLSNotary to define how the committee participates in or verifies a session, together with an explicit corruption threshold. We leave this option for future work.

The final solution is to place the Verifier, or a party already trusted by the Verifier, in the Notary role. This makes the protocol interactive in many scenarios, but removes the need for that Verifier to trust an additional attestation party. In later sections, we discuss examples where the Verifier can run a Notary service without preventing the intended use of GOVBIND.

In short, the Notary is the largest additional trust assumption in the GOVBIND protocol, and the choice of Notary deployment may greatly affect the adoption and security of the system.

Finally, it is important to note that the Notary participates in the live TLSNotary session and therefore has a different view from the offline Verifier. Although Proxy mode is designed not to reveal undisclosed plaintext to the Notary, the Notary may observe protocol and connection metadata that is absent from the final presentation. For a conservative privacy model, we allow the privacy adversary to know the combined views of the Notary and Verifier.

In particular, the signed TLSNotary attestation includes the Notary-reported start time of the authenticated session. Under the assumption that the trusted Notary uses an honest clock, the signature authenticates when the Government

Server’s content was observed. To guarantee freshness of a claim over GOVBIND, the Verifier must compare the signed session time and any claim-specific public date with its own clock and freshness policy.

3.3 Adversary Model

For the GOVBIND protocol, we consider two adversaries with different goals:

1. An honest-but-curious privacy adversary follows the protocol but attempts to learn private information about the Prover or document from the combined views of the Notary and Verifier.
2. A malicious Prover attempts to make the Verifier accept a forged claim.

In the first case, the adversary receives everything learned by both the Notary and Verifier during an honest execution. The adversary may arbitrarily analyze this information, including the secret key of the Notary, but does not deviate from the protocol. The privacy goal requires that it learns no private document content beyond the approved public statement, intentional disclosures, and information explicitly present in the combined protocol view.

In the second case, the adversary controls the Prover and may choose its input and protocol messages arbitrarily. Except with negligible probability, it should not make the Verifier accept unless there exists content authenticated from the approved Government Server that satisfies the approved relation, and the provenance evidence and content proof refer to the same content. This is the soundness and binding goal of GOVBIND.

Both adversaries are limited to probabilistic polynomial time. Soundness assumes an uncompromised TLS identity for the Government Server, an honest Proxy-mode Notary with a trusted public key, a network-authentic connection between the Notary and the Government Server, TLSNotary presentation soundness against a malicious Prover, correct verifier-approved public parameters, and uncompromised cryptographic primitives. Availability, compromise of these trusted components, and factual errors in the government system are outside the adversary model.

3.4 Formal Model

In this subsection, we formalize the syntax of GOVBIND and its security properties. We first abstract the zkTLS layer, i.e., TLSNotary in Proxy mode, as an attested-transcript scheme whose security is imported as an assumption. We then define document profiles in a general form, so that the Baseline and the Optimized Protocols become two instantiations of a single syntax. Finally, we define correctness, soundness, and privacy, which make the informal goals of Section 3.3 precise relative to the trust assumptions of Section 3.2.

Notation. We write $\mathbb{B} = \{0, 1\}^8$ for the set of bytes. For a byte string $m \in \mathbb{B}^*$ and an interval $I = [i, j)$ with $0 \leq i \leq j \leq |m|$, we write $m[I] = m[i : j]$ and $|I| = j - i$. A *layout* is an ordered list of intervals $\Lambda = (I_1, \dots, I_k)$, each labeled as public or private, and we denote by Λ^{pub} and Λ^{priv} its public and private intervals. We say that Λ is an *exact partition* of m if its intervals are pairwise disjoint, appear in increasing order, and their union is $[0, |m|)$. The commitment $\text{Com}(m; r) = H(m \parallel r)$ with $r \in \{0, 1\}^{\lambda_r}$ is the one of Section 2.

The Government Server with service identity d is modeled as an honest stateful oracle $\mathcal{S}_d$ that returns the response body $m = \mathcal{S}_d(q)$ on an HTTPS request q . We require that the attested transcript contain exactly one HTTP request and one response. $\text{Check}_{\mathcal{P}}$ derives m from the disclosed response head and rejects any status other than 200. For an approved endpoint e , the predicate $\text{End}_e(q) \in \{0, 1\}$ decides whether q targets e . We assume that End_e depends only on the request method, the request path, and the Host header of q , all of which every presentation discloses. The session secret s contained in q is never disclosed. We write x for the public claim to be proved, e.g., $x = (\text{id}, \mathbf{v})$ in the Baseline Protocol.

Attested Transcripts

Definition 1 (Attested-Transcript Scheme). *An attested-transcript scheme AT consists of three algorithms, i.e., $\text{AT} = (\text{NotaryKeyGen}, \text{Attest}, \text{VerifyPres})$, described below.*

- $\text{NotaryKeyGen}(1^\lambda) \rightarrow \langle sk_{\text{notary}}, pk_{\text{notary}} \rangle$ takes a security parameter λ as input, and outputs the key pair of the Notary.
- $\text{Attest}(\mathcal{P}(q, \text{Lay}), \mathcal{N}(sk_{\text{notary}}), \mathcal{S}_d) \rightarrow ((m, \rho, \tau_{\text{TLS}}); \text{view}_{\mathcal{N}})$ is an interactive protocol in which the Prover $\mathcal{P}$ sends the request q to $\mathcal{S}_d$ through the Notary $\mathcal{N}$. The Prover takes a layout function Lay as input, and outputs the response body m , the openings $\rho = (r_I)_{I \in \Lambda^{\text{priv}}}$ for the layout $\Lambda = \text{Lay}(m)$, and a presentation τ_{TLS} . The Notary outputs its view $\text{view}_{\mathcal{N}}$ of the session.
- $\text{VerifyPres}(pk_{\text{notary}}, \tau_{\text{TLS}}) \rightarrow \mathcal{V}/\perp$ takes the Notary public key and a presentation as inputs, and outputs either $\perp$ or an attested view

$$\mathcal{V} = (d, t_s, \hat{q}, \Lambda, (m_I)_{I \in \Lambda^{\text{pub}}}, (\mathbf{C}_I)_{I \in \Lambda^{\text{priv}}}),$$

where d is the authenticated server identity, t_s is the Notary-reported session time, $\hat{q}$ contains the disclosed request bytes, m_I are the disclosed response bytes, and $\mathbf{C}_I$ are the commitments to the private response ranges.

In an honest execution, $m_I = m[I]$ for every $I \in \Lambda^{\text{pub}}$, and $\mathbf{C}_I = \text{Com}(m[I]; r_I)$ with $r_I \xleftarrow{\$} \{0, 1\}^{\lambda_r}$ for every $I \in \Lambda^{\text{priv}}$.

Definition 2 (Consistency). *Let q be a request, m a response body, $\rho = (r_I)_I$ a list of openings, t a session time, and $\mathcal{V} = (d, t_s, \hat{q}, \Lambda, (m_I)_{I \in \Lambda^{\text{pub}}}, (\mathbf{C}_I)_{I \in \Lambda^{\text{priv}}})$ an attested view. The consistency predicate $\text{Cons}(q, m, \rho, t, \mathcal{V})$ outputs 1 if and only if all of the following hold:*

- (i) $\hat{q}$ agrees with q on every disclosed request position;
- (ii) Λ is an exact partition of m ;
- (iii) $m[I] = m_I$ for every $I \in \Lambda^{\text{pub}}$;
- (iv) for every $I \in \Lambda^{\text{priv}}$, ρ contains a blinder r_I such that $\mathbf{C}_I = \text{Com}(m[I]; r_I)$;
and
- (v) $t_s = t$.

Intuitively, $\text{Cons}(q, m, \rho, t, \mathcal{V}) = 1$ means that the attested view could have been produced from the exchange (q, m) with the openings ρ at time t : every disclosed byte, every commitment, and the session time in $\mathcal{V}$ agree with this specific request and response.

The attestation oracle $\mathcal{O}_{\text{Att}}(d', q)$ runs **Attest** with the adversary in the role of the Prover and the honest Notary $\mathbf{N}(sk_{\text{notary}})$. If $d' = d$, the server is the honest oracle $\mathcal{S}_d$, and the tuple $(q, \mathcal{S}_d(q), \rho, t)$ is appended to a list $\mathcal{M}$, where ρ are the openings of the commitments computed in that session and t is its start time according to the honest clock of the Notary (Section 3.2). Otherwise, the adversary also plays the server with identity d' , and nothing is recorded. TLSNotary computes the transcript commitments inside a VOLE-based zero-knowledge proof verified by the Notary [34]. We assume that the challenger, which plays the Notary and the underlying VOLE functionality, learns ρ even from a malicious Prover, as in the straight-line extraction of VOLE-based proofs in the VOLE-hybrid model, and we make this explicit by recording ρ . The server oracle $\mathcal{O}_{\text{Srv}}(q)$ returns $\mathcal{S}_d(q)$ over an ordinary HTTPS connection and records nothing. These oracles capture the requirement of Section 3.3 that the Notary and the Government Server are honest while the Prover is malicious. Since the adversary cannot interfere with the connection between the Notary and $\mathcal{S}_d$ in $\mathcal{O}_{\text{Att}}$, the oracles also capture the network-authenticity assumption of Section 3.2.

Definition 3 (Presentation Soundness).

Let $\text{Exp}_{\text{AT}}^{\text{pres}}(\mathcal{A})$ be the following game. The challenger samples $\langle sk_{\text{notary}}, pk_{\text{notary}} \rangle \leftarrow \text{NotaryKeyGen}(1^\lambda)$, sets $\mathcal{M} \leftarrow \emptyset$, and runs $\tau_{\text{TLS}}^* \leftarrow \mathcal{A}^{\mathcal{O}_{\text{Att}}, \mathcal{O}_{\text{Srv}}}(pk_{\text{notary}})$. The game outputs 1 if $\mathcal{V}^* \leftarrow \text{VerifyPres}(pk_{\text{notary}}, \tau_{\text{TLS}}^*)$ satisfies $\mathcal{V}^* \neq \perp$ and $\mathcal{V}^*.d = d$, and there is no $(q, m, \rho, t) \in \mathcal{M}$ with $\text{Cons}(q, m, \rho, t, \mathcal{V}^*) = 1$. We say that AT is presentation-sound if, for all PPT adversaries $\mathcal{A}$, the value

$$\text{Adv}_{\text{AT}}^{\text{pres}}(\mathcal{A}) = \Pr[\text{Exp}_{\text{AT}}^{\text{pres}}(\mathcal{A}) \Rightarrow 1]$$

is negligible.

Definition 3 is the TLSNotary presentation soundness assumption used throughout the paper. It covers the checks relating the observed ciphertext, the Prover's TLS secret, the authenticated server identity, and the committed plaintext, including the in-proof key derivation that makes the integrity of Proxy mode independent of AEAD context unforgeability [18], as discussed in Section 3.2. We treat it as an imported assumption on TLSNotary and do not prove it.

Definition 4 (Presentation Privacy). Let $\mathcal{L}_{\text{TLS}}$ be a leakage function. We say that AT is $\mathcal{L}_{\text{TLS}}$ -private if there exists a PPT simulator Sim_{TLS} such that, for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the following holds. Let $(q, m, \text{Lay}, st) \leftarrow \mathcal{A}_1(sk_{\text{notary}})$, program $\mathcal{S}_d(q) = m$, and run **Attest** honestly to obtain view_{N} , τ_{TLS} , and the commitments $(\mathbf{C}_I)_{I \in \Lambda^{\text{priv}}}$. Then $\mathcal{A}_2(st, \cdot)$ distinguishes the pair $(\text{view}_{\text{N}}, \tau_{\text{TLS}})$ from $\text{Sim}_{\text{TLS}}(sk_{\text{notary}}, \mathcal{L}_{\text{TLS}}(q, m, \Lambda), (\mathbf{C}_I)_{I \in \Lambda^{\text{priv}}})$ with at most negligible advantage.

The simulator receives the honestly computed commitments, so Definition 4 states only that the Notary and the presentation reveal nothing about undisclosed plaintext beyond $\mathcal{L}_{\text{TLS}}$ and the commitments themselves. The hiding of the commitments is used separately in the privacy proofs.

Document Profiles

Definition 5 (Document Profile). A document profile is a tuple

$$\mathcal{P} = (d, e, \text{Lay}_{\mathcal{P}}, \text{Check}_{\mathcal{P}}, J_{\mathcal{P}}, \mathcal{R}_{\mathcal{P}}^{\text{cnt}}),$$

described below.

- d is the approved server identity and e is the approved endpoint.
- $\text{Lay}_{\mathcal{P}}(m) \rightarrow \Lambda$ is the layout function used by an honest Prover to partition a response body m into public and private ranges.
- $\text{Check}_{\mathcal{P}}(\Lambda, (m_I)_{I \in \Lambda^{\text{pub}}}, x) \rightarrow 0/1$ is a deterministic public algorithm that parses the disclosed bytes, re-derives the expected layout from them, and accepts only if it equals Λ and all public structural checks of the profile pass for the claim x . It rejects every unsupported structure.
- $J_{\mathcal{P}}(\Lambda) \subseteq \Lambda^{\text{priv}}$ selects the claim-bearing private ranges.
- $\mathcal{R}_{\mathcal{P}}^{\text{cnt}}(x, (m_I)_{I \in J}) \rightarrow 0/1$ is the content relation evaluated on the plaintext of the claim-bearing ranges.

We write $\text{Accept}_{\mathcal{P}}(m, x) = 1$ if there exists an exact partition Λ of m such that $\text{Check}_{\mathcal{P}}(\Lambda, (m[I])_{I \in \Lambda^{\text{pub}}}, x) = 1$ and $\mathcal{R}_{\mathcal{P}}^{\text{cnt}}(x, (m[I])_{I \in J_{\mathcal{P}}(\Lambda)}) = 1$. The profile induces the relation

$$\mathcal{R}_{\mathcal{P}} = \left\{ (x, (\mathbf{C}_I, |I|)_{I \in J}; (m_I, r_I)_{I \in J}) \mid \begin{array}{l} \mathbf{C}_I = \text{Com}(m_I; r_I), |m_I| = |I| \quad \forall I \in J, \\ \mathcal{R}_{\mathcal{P}}^{\text{cnt}}(x, (m_I)_{I \in J}) = 1 \end{array} \right\}.$$

Section 4.1 instantiates Definition 5 with the Baseline profile $\mathcal{P}_{\text{base}}$, and Section 6.2 with the profiles of the Optimized Protocol, which implement $\text{Check}_{\mathcal{P}}$ through the native PDF verifier and $\mathcal{R}_{\mathcal{P}}^{\text{cnt}}$ through the DEFLATE, template, and predicate constraints of the Noir circuit.

$\text{Accept}_{\mathcal{P}}$ is a syntactic condition on the bytes served by the Government Server. Its meaning is fixed by the following property, which is an assumption about the issuer format and the reviewed parser rather than a consequence of the cryptographic proof system.

Definition 6 (Profile Soundness).

Let $\text{Sem}_{\mathcal{P}}$ map a response body to its claim-relevant interpretation, and let $\text{Pred}_{\mathcal{P}}(\cdot, x)$ be the intended claim predicate. We say that $\mathcal{P}$ is sound if $\text{Sem}_{\mathcal{P}}(m)$ is uniquely defined and $\text{Pred}_{\mathcal{P}}(\text{Sem}_{\mathcal{P}}(m), x) = 1$ for every (m, x) with $\text{Accept}_{\mathcal{P}}(m, x) = 1$.

Syntax of GovBIND**Definition 7 (GovBIND).**

Let $\mathcal{P}$ be a document profile, AT an attested-transcript scheme, and $\text{ZK} = (\text{ZK.Setup}, \text{ZK.Prove}, \text{ZK.Verify})$ a non-interactive zero-knowledge argument of knowledge for $\mathcal{R}_{\mathcal{P}}$. A GovBIND scheme Π for $\mathcal{P}$ consists of five algorithms, i.e., $\Pi = (\text{Setup}, \text{Prepare}, \text{Attest}, \text{Prove}, \text{Verify})$, described below.

- $\text{Setup}(1^\lambda, \mathcal{P}, pk_{\text{notary}}, \varphi) \rightarrow pp$ takes a security parameter λ , a profile $\mathcal{P}$, the Notary public key pk_{notary} , and a freshness policy φ as inputs, runs $(pk_{\mathcal{P}}, vk_{\mathcal{P}}) \leftarrow \text{ZK.Setup}(1^\lambda, \mathcal{R}_{\mathcal{P}})$, and outputs the public parameters $pp = (\mathcal{P}, pk_{\text{notary}}, pk_{\mathcal{P}}, vk_{\mathcal{P}}, \varphi)$.
- $\text{Prepare}(\mathcal{P}(aux), \mathcal{S}_d) \rightarrow (q, s)$ takes the verification information aux , e.g., a barcode and an ID number, and interacts with $\mathcal{S}_d$ over ordinary HTTPS. The Prover outputs the final request q together with the session secret s contained in it.
- $\text{Attest}(\mathcal{P}(pp, q), \mathcal{N}(sk_{\text{notary}}), \mathcal{S}_d) \rightarrow ((m, \rho, \tau_{\text{TLS}}); \text{view}_{\mathcal{N}})$ runs AT.Attest with the layout function $\text{Lay}_{\mathcal{P}}$.
- $\text{Prove}(pp, x, \tau_{\text{TLS}}, m, \rho) \rightarrow \pi$ takes the public parameters, a claim x , a presentation, the response body, and its openings as inputs. It reads Λ and $(\mathbf{C}_I)_{I \in \Lambda^{\text{priv}}}$ from τ_{TLS} , sets $J = J_{\mathcal{P}}(\Lambda)$, and outputs

$$\pi \leftarrow \text{ZK.Prove}(pk_{\mathcal{P}}, \mathbb{X} = (x, (\mathbf{C}_I, |I|)_{I \in J}); \mathbb{W} = (m[I], r_I)_{I \in J}).$$

- $\text{Verify}(pp, x, \tau_{\text{TLS}}, \pi) \rightarrow 0/1$ takes the public parameters, a claim, a presentation, and a proof as inputs, and outputs 1 if and only if all of the following checks pass:
 1. $\mathcal{V} \leftarrow \text{VerifyPres}(pk_{\text{notary}}, \tau_{\text{TLS}})$ and $\mathcal{V} \neq \perp$;
 2. $\mathcal{V}.d = d$ and $\text{End}_e(\hat{q}) = 1$;
 3. $\text{Check}_{\mathcal{P}}(\Lambda, (m_I)_{I \in \Lambda^{\text{pub}}}, x) = 1$;
 4. $\varphi(t_s, x) = 1$; and
 5. $\text{ZK.Verify}(vk_{\mathcal{P}}, (x, (\mathbf{C}_I, |I|)_{I \in J_{\mathcal{P}}(\Lambda)}), \pi) = 1$.

The Prover sends the proof record $\Sigma = (x, \tau_{\text{TLS}}, \pi)$ to the Verifier.

The Prepare step establishes only application-level session state and carries no cryptographic guarantee. In the security games below, the adversary may therefore issue arbitrary requests to $\mathcal{S}_d$.

Security Properties A GOVBIND scheme should at least satisfy correctness and two security properties, namely soundness and privacy. Correctness requires that an honest execution over a document satisfying the claim is accepted. Soundness captures jointly the informal goals of authenticity, i.e., the Verifier accepts only content served by the approved Government Server, content validity, i.e., this content satisfies the approved relation, and binding, i.e., the provenance evidence and the content proof refer to the same content. Privacy is defined relative to an explicit leakage function.

Definition 8 (Correctness).

Π is correct if, for every λ , every $\langle sk_{notary}, pk_{notary} \rangle \leftarrow \text{NotaryKeyGen}(1^\lambda)$, every $pp \leftarrow \text{Setup}(1^\lambda, \mathcal{P}, pk_{notary}, \varphi)$, and every (aux, x) , the following holds. Let (q, s) be the output of an honest Prepare with $\text{End}_e(q) = 1$, and let $m = \mathcal{S}_d(q)$. If $\Lambda = \text{Lay}_{\mathcal{P}}(m)$ witnesses $\text{Accept}_{\mathcal{P}}(m, x) = 1$ and the honest Notary reports a session time t_s with $\varphi(t_s, x) = 1$, then an honest execution of Attest and Prove yields (τ_{TLS}, π) such that

$$\Pr[\text{Verify}(pp, x, \tau_{\text{TLS}}, \pi) = 1] \geq 1 - \text{negl}(\lambda).$$

The proving oracle $\mathcal{O}_{\text{Prove}}(q, x)$ runs an honest Prover on the request q and the claim x . It runs Attest with the honest Notary and $\mathcal{S}_d$, obtaining $(m, \rho, \tau_{\text{TLS}})$ and the session time t , and appends (q, m, ρ, t) to $\mathcal{M}$. If $\text{End}_e(q) = 1$ and $\text{Lay}_{\mathcal{P}}(m)$ witnesses $\text{Accept}_{\mathcal{P}}(m, x) = 1$, it returns the proof record $\Sigma = (x, \tau_{\text{TLS}}, \pi)$ with $\pi \leftarrow \text{Prove}(pp, x, \tau_{\text{TLS}}, m, \rho)$; otherwise, it returns $\perp$. This oracle models honest proof records that the adversary may observe, e.g., on a public ledger.

Definition 9 (Soundness). Let $\text{Exp}_{\Pi, \mathcal{P}}^{\text{snd}}(\mathcal{A})$ be the game of Fig. 2. We say that Π is sound with respect to $\mathcal{P}$ if, for all PPT adversaries $\mathcal{A}$, the value

$$\text{Adv}_{\Pi, \mathcal{P}}^{\text{snd}}(\mathcal{A}) = \Pr[\text{Exp}_{\Pi, \mathcal{P}}^{\text{snd}}(\mathcal{A}) \Rightarrow 1]$$

is negligible.

The winning condition requires a *single* recorded exchange (q, m, ρ, t) that was served by the approved endpoint of the Government Server, is consistent with the attested view of τ_{TLS}^* , and is accepted by the profile for x^* . Hence, Definition 9 captures authenticity, i.e., the content was served by $\mathcal{S}_d$, content validity, i.e., the content satisfies $\text{Accept}_{\mathcal{P}}$, and binding, i.e., the provenance evidence and the content proof refer to the same response, at once. In particular, an adversary that combines the authenticated bytes of one response with a content proof over another wins the game. Since $\mathcal{O}_{\text{Prove}}$ records every honest exchange, replaying an honest proof record is not a win, but transforming it into a proof record for a claim that the recorded response does not satisfy is. If $\mathcal{P}$ is additionally sound in the sense of Definition 6, an accepted proof record implies $\text{Pred}_{\mathcal{P}}(\text{Sem}_{\mathcal{P}}(m), x^*) = 1$ for the served response m . Moreover, condition (v) of Definition 2 ties the attested session time t_s to the time at which the exchange actually took place, so the freshness check $\varphi(t_s, x^*)$ of Verify refers to the served response.

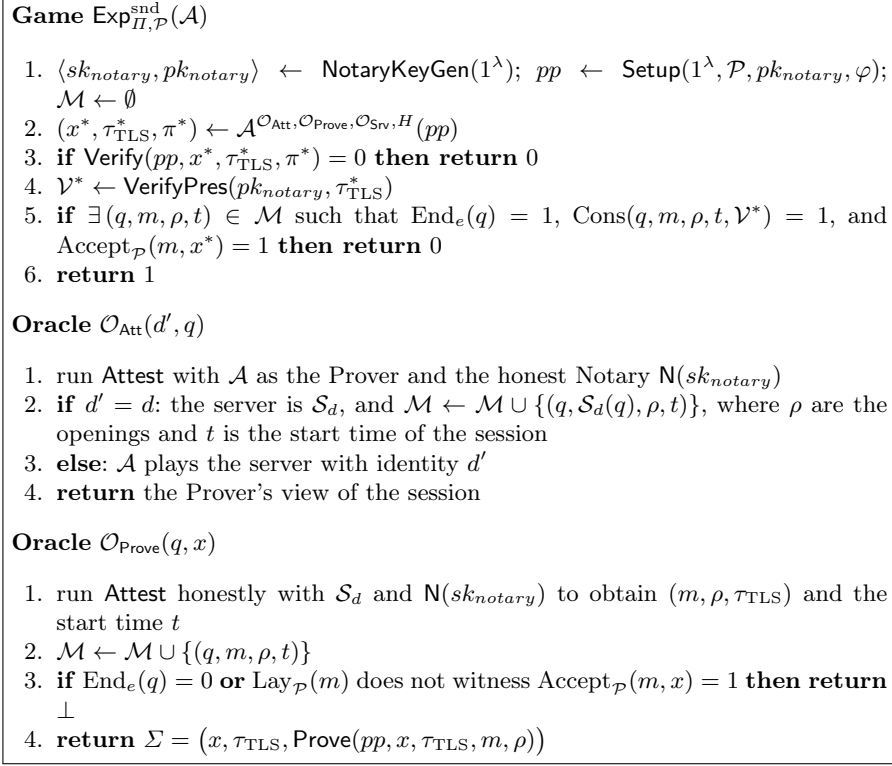


Fig. 2. The soundness game of GovBIND. $\mathcal{O}_{\text{Prove}}$ is described in the text, $\mathcal{O}_{\text{Srv}}(q)$ returns $\mathcal{S}_d(q)$ without recording it, and H is the random oracle.

Definition 10 (Privacy). Let $\mathcal{L}$ be a leakage function, and let $\text{Real}_{\Pi, \mathcal{P}}$ and $\text{Ideal}_{\Pi, \mathcal{P}, \mathcal{L}, \text{Sim}}$ be the games of Fig. 3. We say that Π is $\mathcal{L}$ -private with respect to $\mathcal{P}$ if there exists a PPT simulator Sim such that, for all PPT adversaries $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$, the value

$$\text{Adv}_{\Pi, \mathcal{P}, \mathcal{L}}^{\text{priv}}(\mathcal{A}, \text{Sim}) = |\Pr[\text{Real}_{\Pi, \mathcal{P}}(\mathcal{A}) \Rightarrow 1] - \Pr[\text{Ideal}_{\Pi, \mathcal{P}, \mathcal{L}, \text{Sim}}(\mathcal{A}) \Rightarrow 1]|$$

is negligible.

In both games, $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ is a two-stage adversary. The first stage $\mathcal{A}_1$ receives pp and the secret key sk_{notary} of the Notary, whose view is part of the combined view, and outputs a request q , the response body m that the Government Server returns on q , a claim x , and a state st that is passed to $\mathcal{A}_2$. Step 3 restricts both games to inputs on which an honest Prover can run the protocol, i.e., q targets the approved endpoint and the honest layout $\text{Lay}_{\mathcal{P}}(m)$ witnesses $\text{Accept}_{\mathcal{P}}(m, x) = 1$, as in Definition 8. Since this check is identical in both games, it gives $\mathcal{A}$ no advantage. Definition 10 models the honest-but-curious privacy adversary of Section 3.3: the Notary and the Prover execute the

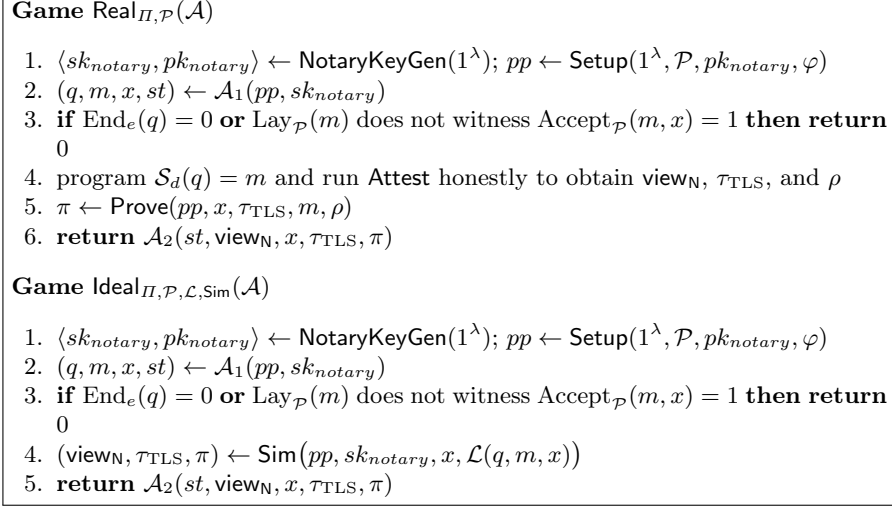


Fig. 3. The privacy games of GOVBIND. $\mathcal{A}_2$ receives the combined view of the Notary and the Verifier. In the random-oracle model, Sim may program H .

protocol honestly, and $\mathcal{A}_2$ receives both the view of the Notary, including its secret key, and the proof record received by the Verifier. Since $\mathcal{A}_1$ chooses the response body, the definition holds for every served document rather than for a random one. The connection metadata observed by the Notary is part of $\mathcal{L}$, so we let $\mathcal{A}_1$ fix it together with (q, m, x) . The leakage functions $\mathcal{L}_{\text{base}}$ and $\mathcal{L}_{\text{opt}, \mathcal{P}}$ instantiate $\mathcal{L}$ for the Baseline and the Optimized Protocols, respectively, where we suppress the dependence on q and write $\mathcal{L}_{\text{base}}(m, x)$ and $\mathcal{L}_{\text{opt}, \mathcal{P}}(m, x)$.

Remark 1 (Scope of the Definitions). The definitions above deliberately leave out the following properties.

- *Factual correctness.* Soundness concerns what $\mathcal{S}_d$ served, not whether it is true outside the issuing system.
- *Freshness.* Under an honest Notary with an honest clock, soundness implies that the recorded exchange took place at the attested time t_s , so the policy φ enforces freshness. Whether φ is appropriate for a given claim, and a Notary with an incorrect clock, are outside the model.
- *Subject binding.* $\mathcal{A}$ may query $\mathcal{O}_{\text{Att}}$ on any request, including one that retrieves another person’s document. Soundness therefore does not imply that the Prover is the subject of the document.
- *Replay and transferability.* Neither game binds Σ to a Verifier or to a verification session, so an accepted proof record, including one returned by $\mathcal{O}_{\text{Prove}}$, can be reused within its freshness window.
- *Privacy against malicious parties.* Since the Verifier only receives Σ and does not interact with any party, an honest-but-curious Verifier is as powerful as a malicious one, so Definition 10 also covers a malicious Verifier. A Notary that

- deviates from the protocol, e.g., by tampering with the forwarded traffic, is not covered, although in Proxy mode it never holds the TLS session secrets.
- *Network adversaries.* An adversary that controls the network path between the Notary and $\mathcal{S}_d$ can break soundness in Proxy mode, since the Prover knows the session keys. This is excluded by the network-authenticity assumption of Section 3.2.
- *Malicious Notary and availability.* The Notary and $\mathcal{S}_d$ are honest in both games, and their availability is not modeled.

Section 7 discusses mechanisms that address these properties.

4 GOVBIND Protocol

4.1 Protocol Setting and Relations

Properties of a Compatible Government Verification Service The Government Server must provide an HTTPS endpoint that returns a verification document when given the required verification information. Depending on the service, this information may include a barcode, an ID number, or similar values. The service may be public or authenticated, but an authenticated service can be supported only when its authentication process produces session information, such as a cookie or token, that can be used in the final zkTLS request.

The Turkish government document verification service, e-Devlet, is an example of such a system. A document issued by e-Devlet includes a verification barcode and an ID number. The barcode is unique and generated during the creation of the PDF document, while the ID number identifies the citizen, similarly to a social security number in some countries. By using these values, the document can be verified from a public address under the e-Devlet domain through the following process:

1. The Prover submits the verification barcode in the document to the public verification address.
2. If the barcode exists, the e-Devlet server returns a one-time token and asks for the ID number on the document.
3. The Prover submits the ID number along with the first authentication token returned by e-Devlet.
4. If the ID is valid and matches the previously given barcode, the e-Devlet server issues a final authentication token.
5. By sending this final authentication token, the Prover retrieves the verification document.

In the notation of Section 3.4, the barcode and the ID number form the verification information aux , and Steps 1–4 above constitute $\text{Prepare}\langle P(aux), \mathcal{S}_d \rangle \rightarrow (q, s)$, where the final authentication token is the session secret s . The request of Step 5 is the request q with $\text{End}_e(q) = 1$, which the Prover later issues inside Attest .

We call the document returned by the Government Server the verification document and denote its byte representation by m , i.e., $m = \mathcal{S}_d(q)$. However, m may not be byte-to-byte identical to the original document and may differ in metadata, e.g., creation date. The verification service regenerates the document from the current state of the issuing system, so the session time t_s reflects the time of the claim. In other words, this verification service authenticates only the document that it returns, not the original document. In GOVBIND, the original document is therefore used only to obtain the information required by the verification service, and all claims are proved directly over m .

Note that a conventional verification on e-Devlet requires accessing the service each time and comparing the returned m with the original document. This comparison may be unsafe if a Verifier checks only whether the documents appear similar. For example, an attacker may change a field in the original document without changing its barcode or ID, and such a small change may go unnoticed by the Verifier. A Verifier can avoid this scenario by ignoring the original document after reading the barcode and ID from it. In GOVBIND, we similarly prove the claim directly over m , without relying on the integrity of the original document. Another possible solution would prove that the original and verification documents have identical extracted content, but that work is unnecessary for the binding mechanism in Section 4.4.

Relations Used in the GOVBIND Protocol The Baseline Protocol uses two proof stages. First, the attested-transcript scheme AT of Definition 1, instantiated with TLSNotary, generates a presentation τ_{TLS} whose attested view authenticates the Government Server identity d and a commitment to the verification document, $\mathbf{C}$. Then, GOVBIND generates one or more ZK proofs over the same commitment. We treat the provenance property of TLSNotary as an imported protocol assumption rather than representing the complete interactive TLSNotary protocol as an NP relation.

TLSNotary presentation soundness. For the Baseline layout, where the complete response body is a single private range, the presentation soundness of Definition 3 states that if a presentation τ_{TLS} is accepted with server identity d and commitment $\mathbf{C}$, then, except with negligible probability, there is an exchange $(q, m_{\text{TLS}}, r_{\text{TLS}}, t_s)$ recorded from the approved Government Server such that $\mathbf{C} = \text{Com}(m_{\text{TLS}}; r_{\text{TLS}})$. This assumption covers the TLSNotary checks relating the observed ciphertext, private TLS state, authenticated server identity, and plaintext commitment. It is separate from the binding property used below to rule out a second opening, and from the honesty of the Notary [31]. In our implementation, τ_{TLS} binds the authenticated Government Server, session information, metadata, and commitment $\mathbf{C}$ under the Notary’s signature.

The second relation is $\mathcal{R}_{\text{include}}$, which proves that the message committed to by $\mathbf{C}$ contains a specified substring:

$$\mathcal{R}_{\text{include}} = \left\{ (\mathbf{x}; \mathbf{w}) = (\mathbf{C}, \mathbf{v}; m, r, i) \left| \begin{array}{l} \mathbf{C} = \text{Com}(m; r), \\ r \in \{0, 1\}^{\lambda_r}, \\ m[i : i + |\mathbf{v}|] = \mathbf{v} \end{array} \right. \right\}.$$

Here, $(\mathbf{C}, \mathbf{v})$ is the public statement and (m, r, i) is the witness. The value $\mathbf{v}$ is the public substring, while its position in m , i , remains private. This relation describes byte inclusion of $\mathbf{v}$ in m , but concrete implementations may use stronger relations to interpret encoded or compressed document content, as discussed later.

The Baseline profile. Using $\mathcal{R}_{\text{include}}$, we instantiate Definition 5 with the Baseline profile

$$\mathcal{P}_{\text{base}} = (d, e, \text{Lay}_{\text{base}}, \text{Check}_{\text{base}}, J_{\text{base}}, \mathcal{R}_{\text{base}}^{\text{cnt}}),$$

where $\text{Lay}_{\text{base}}(m) = ([0, |m|])$ labels the complete response body as a single private range, $\text{Check}_{\text{base}}(\Lambda, \emptyset, x) = 1$ if and only if Λ consists of a single private interval $[0, \ell]$, where ℓ is the authenticated length of the response body, $J_{\text{base}}(\Lambda) = \Lambda^{\text{priv}}$, and $\mathcal{R}_{\text{base}}^{\text{cnt}}((\mathbf{id}, \mathbf{v}), m) = 1$ if and only if there exist positions i_{id} and $i_{\mathbf{v}}$ such that $m[i_{\text{id}} : i_{\text{id}} + |\mathbf{id}|] = \mathbf{id}$ and $m[i_{\mathbf{v}} : i_{\mathbf{v}} + |\mathbf{v}|] = \mathbf{v}$. The claim is therefore $x = (\mathbf{id}, \mathbf{v})$. Instead of a single proof for $\mathcal{R}_{\mathcal{P}_{\text{base}}}$, the Baseline Protocol uses the pair $\pi = (\pi_{\text{id}}, \pi_{\mathbf{v}})$ of proofs for $\mathcal{R}_{\text{include}}$ over the same commitment $\mathbf{C}$, with keys $(pk_{\text{include}}, vk_{\text{include}})$. Although Definition 7 assumes a single argument for $\mathcal{R}_{\mathcal{P}}$, the proof of Theorem 2 shows that this pair suffices, because the binding of Com forces the two extracted openings to coincide. The resulting scheme Π_{base} is given in Fig. 4.

4.2 Honest Protocol Execution

With these parties and preliminary information, we may now describe an honest execution of the GOVBIND Baseline Protocol. We assume that the Prover has the information required by a compatible verification service and wants to prove that the returned verification document includes a public ID $\mathbf{id}$ and a public substring $\mathbf{v}$. For example, $\mathbf{v}$ may show that the person identified by $\mathbf{id}$ has a tax debt less than a given threshold without revealing other private information.

We assume that the Notary has run $\text{NotaryKeyGen}(1^\lambda)$ and that the public parameters $pp \leftarrow \text{Setup}(1^\lambda, \mathcal{P}_{\text{base}}, pk_{\text{notary}}, \varphi)$ of Fig. 4 are fixed and known to the Verifier. The claim is $x = (\mathbf{id}, \mathbf{v})$.

With these definitions, the protocol is executed as follows:

1. The Prover obtains the information required by the verification service, described in Section 4.1. This information may be extracted from an original document or obtained through another method, which is irrelevant to the rest of the protocol.

```

Setup( $1^\lambda, \mathcal{P}_{\text{base}}, pk_{\text{notary}}, \varphi$ )
1.  $(pk_{\text{include}}, vk_{\text{include}}) \leftarrow \text{ZK.Setup}(1^\lambda, \mathcal{R}_{\text{include}})$ 
2. return  $pp = (\mathcal{P}_{\text{base}}, pk_{\text{notary}}, pk_{\text{include}}, vk_{\text{include}}, \varphi)$ 

Prepare( $P(aux), \mathcal{S}_d$ )
1. run Steps 1–4 of the verification flow of Section 4.1 with  $aux$  over ordinary HTTPS
2. return the final request  $q$ , which contains the final authentication token  $s$ 

Attest( $P(pp, q), N(sk_{\text{notary}}), \mathcal{S}_d$ )
1. run AT.Attest in Proxy mode with the layout function  $\text{Lay}_{\text{base}}$ 
2. the Prover obtains  $m = \mathcal{S}_d(q)$ ,  $\rho = (r)$ , and  $\tau_{\text{TLS}}$ , where  $\mathbf{C} = \text{Com}(m; r)$ 

Prove( $pp, x = (\mathbf{id}, \mathbf{v}), \tau_{\text{TLS}}, m, \rho$ )
1. read  $\mathbf{C}$  from  $\tau_{\text{TLS}}$ , and find the positions  $i_{\text{id}}$  and  $i_{\mathbf{v}}$  of  $\mathbf{id}$  and  $\mathbf{v}$  in  $m$ 
2.  $\pi_{\text{id}} \leftarrow \text{ZK.Prove}(pk_{\text{include}}, (\mathbf{C}, \mathbf{id}); (m, r, i_{\text{id}}))$ 
3.  $\pi_{\mathbf{v}} \leftarrow \text{ZK.Prove}(pk_{\text{include}}, (\mathbf{C}, \mathbf{v}); (m, r, i_{\mathbf{v}}))$ 
4. return  $\pi = (\pi_{\text{id}}, \pi_{\mathbf{v}})$ 

Verify( $pp, x = (\mathbf{id}, \mathbf{v}), \tau_{\text{TLS}}, \pi$ )
1.  $\mathcal{V} \leftarrow \text{VerifyPres}(pk_{\text{notary}}, \tau_{\text{TLS}})$ ; if  $\mathcal{V} = \perp$  then return 0
2. if  $\mathcal{V}.d \neq d$  or  $\text{End}_e(\hat{q}) = 0$  then return 0
3. if  $\text{Check}_{\text{base}}(\Lambda, \emptyset, x) = 0$  then return 0; else let  $\mathbf{C}$  be the commitment of the single private range
4. if  $\varphi(t_s, x) = 0$  then return 0
5. return  $\text{ZK.Verify}(vk_{\text{include}}, (\mathbf{C}, \mathbf{id}), \pi_{\text{id}}) \wedge \text{ZK.Verify}(vk_{\text{include}}, (\mathbf{C}, \mathbf{v}), \pi_{\mathbf{v}})$ 

```

Fig. 4. The Baseline Protocol Π_{base} in the syntax of Definition 7.

2. The Prover interacts with the Government Server through ordinary HTTPS to prepare the verification request. These requests do not need to use the same TLS connection or secrets, but application-level session state, including cookies and tokens, must be maintained.
3. The Prover receives the session secret s required to authorize the final document request, described in Section 4.1. Steps 1–3 constitute $\text{Prepare}(P(aux), \mathcal{S}_d) \rightarrow (q, s)$.
4. The Prover connects to the Notary and starts a TLSNotary session with the Government Server in Proxy mode. The present construction and implementation are analyzed only in Proxy mode.
5. Within this session, the Prover sends the final request q , which contains s , and the Notary forwards the encrypted traffic. The Notary does not learn undisclosed plaintext, as it does not have the TLS secret, but it may observe protocol and connection metadata.
6. The Government Server returns the verification document $m = \mathcal{S}_d(q)$. For the Baseline Protocol, the layout is $\text{Lay}_{\text{base}}(m) = ([0, |m|])$, so TLSNotary

creates the commitment $\mathbf{C} = \text{Com}(m; r)$ to the complete response body, with the opening $\rho = (r)$. The Notary issues an attestation that binds the authenticated Government Server and TLS metadata to $\mathbf{C}$.

7. The Prover derives the TLSNotary presentation τ_{TLS} from this attestation. Its attested view is $\mathcal{V} = (d, t_s, \hat{q}, ([0, |m|]), \emptyset, \mathbf{C})$, i.e., the presentation discloses the information needed to identify the Government Server and validate the relevant HTTP request and response, while the verification document remains committed. Steps 4–7 constitute $\text{Attest}(\mathbf{P}(pp, q), \mathbf{N}(sk_{\text{notary}}), \mathcal{S}_d) \rightarrow ((m, \rho, \tau_{\text{TLS}}); \text{view}_{\mathbf{N}})$. Fig. 5 visualizes Steps 1–7.

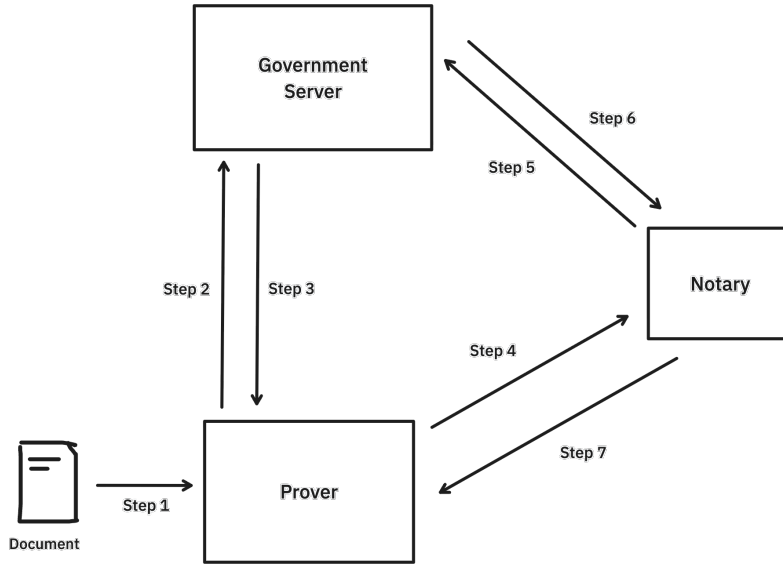


Fig. 5. Visualization of the first 7 steps of the protocol.

8. The Prover generates a proof that $\mathbf{id}$ is included in m :

$$\pi_{\mathbf{id}} \leftarrow \text{ZK.Prove}(pk_{\text{include}}, \mathbb{X} = (\mathbf{C}, \mathbf{id}); \mathbb{W} = (m, r, i_{\mathbf{id}})).$$

Here, $i_{\mathbf{id}}$ is the position of $\mathbf{id}$ in m .

9. The Prover generates a proof that $\mathbf{v}$ is included in m :

$$\pi_{\mathbf{v}} \leftarrow \text{ZK.Prove}(pk_{\text{include}}, \mathbb{X} = (\mathbf{C}, \mathbf{v}); \mathbb{W} = (m, r, i_{\mathbf{v}})).$$

Here, $i_{\mathbf{v}}$ is the position of $\mathbf{v}$ in m . Both proofs open the same commitment $\mathbf{C}$ authenticated by τ_{TLS} . Steps 8–9 constitute $\text{Prove}(pp, x, \tau_{\text{TLS}}, m, \rho)$, whose output is $\pi = (\pi_{\mathbf{id}}, \pi_{\mathbf{v}})$.

10. The Prover sends the proof record $\Sigma = (x, \tau_{\text{TLS}}, \pi)$, with $x = (\mathbf{id}, \mathbf{v})$ and $\pi = (\pi_{\mathbf{id}}, \pi_{\mathbf{v}})$, to the Verifier. Since $\mathbf{C}$ is contained in the attested view of τ_{TLS} , Σ carries the same information as the tuple $\langle \tau_{\text{TLS}}, \mathbf{C}, \mathbf{id}, \mathbf{v}, \pi_{\mathbf{id}}, \pi_{\mathbf{v}} \rangle$.

Note that this execution proves a claim over the verification document m , not over the original document. The original document is only one possible source of the verification information and is not part of the final proof.

Moreover, notice that proving that **id** and **v** occur in the same document does not always prove that they are semantically related. For simple documents, the relation may instead check a longer statement that contains both values. More complex documents may require a new relation that both checks the document structure and the positions of the relevant fields.

Finally, recall that PDF documents are not linear text messages and that visible text may be encoded or compressed. The inclusion relation is therefore only a simple protocol abstraction. A concrete implementation may need a document-specific relation that decompresses the relevant content and checks an approved document profile. We discuss this challenge in the implementation section. In particular, e-Devlet documents store their page content in FlateDecode-compressed streams, so the claim-bearing text does not occur in m as a plain substring, and $\mathcal{R}_{\text{include}}$ cannot be satisfied over these documents. The Baseline Protocol therefore serves as a reference construction that isolates the binding argument, while the Optimized Protocol of Section 6.2 is the construction we implement and evaluate.

Remarks on the Protocol Execution There are two important remarks to be made in this description of the protocol.

First, the execution above asserts only that a particular substring occurs in the authenticated document. The relation can instead hide the substring and prove a predicate over a value within it. For example, it can prove that a hidden statement has the form “has **X** dollars of tax debt” and that $\mathbf{X} < 1000$. We implement such predicates for specific use cases described in Section 6.3.

Second, the protocol may be extended with a recursive wrapper proof that verifies its cryptographic objects. We discuss this possibility in Section 7.

4.3 Honest Protocol Verification

Receiving $\Sigma = (x, \tau_{\text{TLS}}, \pi)$, an honest Verifier runs $\text{Verify}(pp, x, \tau_{\text{TLS}}, \pi)$ of Fig. 4 on the claim $x = (\mathbf{id}, \mathbf{v})$ it expects. Grouped by the checks of Definition 7, it verifies the following:

1. τ_{TLS} is cryptographically valid, including its signed attestation and presentation proofs, under the trusted Notary public key pk_{notary} , i.e., $\mathcal{V} \leftarrow \text{VerifyPres}(pk_{\text{notary}}, \tau_{\text{TLS}})$ and $\mathcal{V} \neq \perp$.
2. The Government Server identity authenticated by τ_{TLS} is the approved server identity, e.g., e-Devlet verification domain, and the authenticated HTTPS request and response match the expected endpoint and document response, i.e., $\mathcal{V}.d = d$ and $\text{End}_e(\hat{q}) = 1$.
3. The commitment authenticated by τ_{TLS} covers the complete response body, i.e., $\text{Check}_{\text{base}}(\Lambda, \emptyset, x) = 1$. The commitment **C** is then read from $\mathcal{V}$.

4. The Notary-reported session time t_s in τ_{TLS} satisfies the Verifier's freshness policy, i.e., $\varphi(t_s, x) = 1$.
5. $\text{ZK.Verify}(vk_{\text{include}}, (\mathbf{C}, \mathbf{id}), \pi_{\text{id}}) = 1$ and $\text{ZK.Verify}(vk_{\text{include}}, (\mathbf{C}, \mathbf{v}), \pi_{\mathbf{v}}) = 1$, where vk_{include} is the approved verification key.

Since `Verify` is run on the claim that the Verifier expects, the checks that $\mathbf{v}$ is the expected claim substring and that $\mathbf{id}$ is the expected ID number are enforced by the choice of x . Similarly, the commitment authenticated by τ_{TLS} equals $\mathbf{C}$ by construction, since $\mathbf{C}$ is read from $\mathcal{V}$.

Here, we assume that the Verifier knows the expected ID $\mathbf{id}$. Alternatively, the public input could be a commitment to a hidden ID. That commitment could later be linked to a public set, another identity proof, or an earlier proof, although these extensions are not implemented here.

4.4 Binding Mechanism

The binding mechanism ensures that τ_{TLS} and the content proofs refer to m . It does not bind the original document, which is not an input to these proofs, as noted before.

Intuitively, presentation soundness ties $\mathbf{C}$ to the response body m_{TLS} of a single exchange recorded from the approved Government Server, with an opening r_{TLS} . Knowledge soundness yields openings of $\mathbf{C}$ from both content proofs, and the binding property of the commitment scheme forces these openings to coincide with $(m_{\text{TLS}}, r_{\text{TLS}})$, except with negligible probability. Therefore, both content proofs apply to the same verification document that was returned by the approved Government Server, and an adversary cannot combine the provenance evidence of one exchange with content proofs over another. Theorem 2 makes this argument precise.

Note that the meaning of the proved claim still depends on the approved content relation.

5 Security Analysis

In this section, we prove that the Baseline Protocol Π_{base} of Fig. 4 satisfies correctness, soundness, and privacy in the sense of Definitions 8, 9, and 10. Our proofs are game-based sketches that build on the assumed security of `TLSNotary` (Definitions 3 and 4), the commitment, and the ZK argument. Section 6.2 gives the corresponding results for the Optimized Protocol.

For a primitive X and a property y , we write $\text{Adv}_X^y(\mathcal{B})$ for the advantage of an adversary $\mathcal{B}$ against y in the standard sense [15], e.g., $\text{Adv}_{\text{Com}}^{\text{bind}}$, $\text{Adv}_{\text{Com}}^{\text{hide}}$, $\text{Adv}_{\text{ZK}}^{\text{ks}}$, and $\text{Adv}_{\text{ZK}}^{\text{zk}}$. We write $\text{Adv}_{\text{AT}}^{\text{priv}}$ for the distinguishing advantage in Definition 4.

5.1 Correctness of GovBIND Baseline Protocol

Theorem 1 (Baseline Protocol Correctness). *Assume that every honest execution of `AT` is accepted by `VerifyPres` with the attested view of Definition 1, and that ZK is complete. Then Π_{base} is correct in the sense of Definition 8.*

Proof sketch. Fix pp and (aux, x) with $x = (\mathbf{id}, \mathbf{v})$ as in Definition 8, let q be the output of an honest **Prepare** with $\text{End}_e(q) = 1$, and let $m = \mathcal{S}_d(q)$. Suppose that $\text{Lay}_{\text{base}}(m)$ witnesses $\text{Accept}_{\mathcal{P}_{\text{base}}}(m, x) = 1$, i.e., $\mathbf{id}$ and $\mathbf{v}$ occur in m at some positions i_{id} and $i_{\mathbf{v}}$. An honest **Attest** yields τ_{TLS} whose attested view satisfies $\mathcal{V} \neq \perp$ and $\mathcal{V}.d = d$, and $\hat{q}$ discloses the request fields of q on which End_e depends, so $\text{End}_e(\hat{q}) = 1$. Since $\text{Lay}_{\text{base}}(m) = ([0, |m|))$, the check $\text{Check}_{\text{base}}$ accepts, and $\mathbf{C} = \text{Com}(m; r)$. The freshness check holds by assumption. Finally, (m, r, i_{id}) and $(m, r, i_{\mathbf{v}})$ are witnesses for the statements $(\mathbf{C}, \mathbf{id})$ and $(\mathbf{C}, \mathbf{v})$ of $\mathcal{R}_{\text{include}}$, so both proofs verify by the completeness of ZK. $\square$

5.2 Soundness of GovBIND Baseline Protocol

Theorem 2 (Baseline Protocol Soundness).

Assume that **AT** is presentation-sound (Definition 3), that **Com** is binding, and that **ZK** is knowledge-sound. Then Π_{base} is sound with respect to $\mathcal{P}_{\text{base}}$ in the sense of Definition 9. Concretely, for every PPT adversary $\mathcal{A}$, there exist PPT adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$ such that

$$\text{Adv}_{\Pi_{\text{base}}, \mathcal{P}_{\text{base}}}^{\text{snd}}(\mathcal{A}) \leq \text{Adv}_{\text{AT}}^{\text{pres}}(\mathcal{B}_1) + \text{Adv}_{\text{ZK}}^{\text{ks}}(\mathcal{B}_2) + \text{Adv}_{\text{ZK}}^{\text{ks}}(\mathcal{B}_3) + \text{Adv}_{\text{Com}}^{\text{bind}}(\mathcal{B}_4).$$

Proof sketch. We proceed through a sequence of games, where G_0 is $\text{Exp}_{\Pi_{\text{base}}, \mathcal{P}_{\text{base}}}^{\text{snd}}(\mathcal{A})$ of Fig. 2. Let $(x^*, \tau_{\text{TLS}}^*, \pi^*)$ with $x^* = (\mathbf{id}, \mathbf{v})$ and $\pi^* = (\pi_{\text{id}}, \pi_{\mathbf{v}})$ be the output of $\mathcal{A}$, and let $\mathcal{V}^* \leftarrow \text{VerifyPres}(pk_{\text{notary}}, \tau_{\text{TLS}}^*)$.

Game G_1 . G_1 is identical to G_0 , except that it returns 0 if there is no $(q, m, \rho, t) \in \mathcal{M}$ with $\text{Cons}(q, m, \rho, t, \mathcal{V}^*) = 1$. Whenever G_0 reaches its final checks, **Verify** has accepted, so $\mathcal{V}^* \neq \perp$ and $\mathcal{V}^*.d = d$. The two games therefore differ only if $\mathcal{A}$ produces an accepted presentation for d that is consistent with no recorded exchange. The adversary $\mathcal{B}_1$ runs **ZK.Setup** itself, forwards the oracle queries of $\mathcal{A}$ to its own oracles $\mathcal{O}_{\text{Att}}$ and $\mathcal{O}_{\text{Srv}}$, answers each query to $\mathcal{O}_{\text{Prove}}$ by running the honest Prover through its own $\mathcal{O}_{\text{Att}}$, which records the same exchange, simulates H , and outputs τ_{TLS}^* , so that $|\Pr[G_0 \Rightarrow 1] - \Pr[G_1 \Rightarrow 1]| \leq \text{Adv}_{\text{AT}}^{\text{pres}}(\mathcal{B}_1)$.

In G_1 , fix a consistent exchange (q, m, ρ, t) with $\rho = (r)$. The check $\text{Check}_{\text{base}}$ of **Verify** forces $A^* = ([0, |m|))$ to be a single private range with commitment $\mathbf{C}$, and the consistency conditions (ii) and (iv) of Definition 2 give $\mathbf{C} = \text{Com}(m; r)$. Moreover, the condition (i) together with $\text{End}_e(\hat{q}) = 1$ gives $\text{End}_e(q) = 1$, since End_e depends only on disclosed request fields.

Game G_2 . G_2 additionally runs the knowledge extractors of **ZK** on π_{id} and $\pi_{\mathbf{v}}$, obtaining $(m_{\text{id}}, r_{\text{id}}, i_{\text{id}})$ and $(m_{\mathbf{v}}, r_{\mathbf{v}}, i_{\mathbf{v}})$, and returns 0 if either is not a valid witness of $\mathcal{R}_{\text{include}}$ for $(\mathbf{C}, \mathbf{id})$ or $(\mathbf{C}, \mathbf{v})$, respectively. Since **Verify** has accepted both proofs, the adversaries $\mathcal{B}_2$ and $\mathcal{B}_3$, which run G_1 internally, answer $\mathcal{O}_{\text{Prove}}$ with the real witnesses, and output the respective statement and proof, give $|\Pr[G_1 \Rightarrow 1] - \Pr[G_2 \Rightarrow 1]| \leq \text{Adv}_{\text{ZK}}^{\text{ks}}(\mathcal{B}_2) + \text{Adv}_{\text{ZK}}^{\text{ks}}(\mathcal{B}_3)$. We instantiate **ZK** with UltraHonk as implemented in Barretenberg [2], a sumcheck-based variant of PLONK whose underlying argument follows the line of work analyzed in [6], and we assume its knowledge soundness in the algebraic group and random-oracle models. Extraction is applied to a constant number of proofs, so running the extractors sequentially keeps the reductions polynomial-time.

Game G_3 . G_3 additionally returns 0 if $m_{\text{id}} \neq m$ or $m_{\mathbf{v}} \neq m$. In that case, (m, r) and $(m_{\text{id}}, r_{\text{id}})$, or (m, r) and $(m_{\mathbf{v}}, r_{\mathbf{v}})$, are two different openings of $\mathbf{C}$. The adversary $\mathcal{B}_4$ runs G_2 internally, including the challenger's record of ρ , and outputs these openings, so that $|\Pr[G_2 \Rightarrow 1] - \Pr[G_3 \Rightarrow 1]| \leq \text{Adv}_{\text{Com}}^{\text{bind}}(\mathcal{B}_4)$.

In G_3 , the response m contains id at position i_{id} and $\mathbf{v}$ at position $i_{\mathbf{v}}$, so $\mathcal{R}_{\text{base}}^{\text{cnt}}(x^*, m) = 1$. Since Λ^* is an exact partition of m accepted by $\text{Check}_{\text{base}}$, we obtain $\text{Accept}_{\mathcal{P}_{\text{base}}}(m, x^*) = 1$. Together with $\text{End}_e(q) = 1$ and $\text{Cons}(q, m, \rho, t, \mathcal{V}^*) = 1$, the game returns 0 in Step 5 of Fig. 2, i.e., $\Pr[G_3 \Rightarrow 1] = 0$. Summing the differences gives the stated bound. $\square$

The recorded openings ρ are used only by $\mathcal{B}_4$. They are the reason why the oracle $\mathcal{O}_{\text{Att}}$ records the openings of each exchange: without them, the second opening of $\mathbf{C}$ in G_3 would exist, but it would not be known to the reduction.

Through $\mathcal{O}_{\text{Prove}}$, the adversary observes honest proof records and may try to transform one of them into a proof record for a claim that the recorded response does not satisfy. Theorem 2 covers this attack without assuming simulation-extractability. The reductions $\mathcal{B}_2$ and $\mathcal{B}_3$ answer $\mathcal{O}_{\text{Prove}}$ with real witnesses, so plain knowledge soundness applies to their outputs, and a transformed proof over a recorded commitment $\mathbf{C}$ has a valid witness only if $\mathbf{C}$ has a second opening, which G_3 rules out. Simulation-extractability would be needed only if a reduction had to simulate honest proofs without their witnesses.

Note that this result does not show the factual correctness or semantic validity of the proved claim. It only shows that the content authenticated as coming from the approved Government Server satisfies the relations checked by the honest Verifier, which shows the soundness of GOVBIND in the sense of Definition 9.

5.3 Privacy of GOVBIND Baseline Protocol

Definition 11 (Baseline Protocol Leakage). *For the Baseline Protocol, let the leakage $\mathcal{L}_{\text{base}}(m, x)$ contain the approved server identity and endpoint, the trusted Notary and content-program identities, the Notary-reported session time, the public claim, all intentional request and response disclosures, the lengths and boundaries of the TLS transcript and application messages, and connection timing and network-layer identifiers observed by the Notary. This leakage is explicit and is not claimed to be hidden.*

We assume that the leakage of AT for the Baseline layout is contained in the Baseline leakage, i.e., $\mathcal{L}_{\text{TLS}}(q, m, ([0, |m|]))$ can be computed from $\mathcal{L}_{\text{base}}(m, x)$. This holds for TLSNotary, since its presentation and the view of the Notary expose only the server identity, the session time, the disclosed request bytes, the transcript lengths, and the connection metadata.

Theorem 3 (Baseline Protocol Privacy). *Assume that AT is $\mathcal{L}_{\text{TLS}}$ -private (Definition 4), that Com is computationally hiding, and that ZK is zero-knowledge. Then Π_{base} is $\mathcal{L}_{\text{base}}$ -private with respect to $\mathcal{P}_{\text{base}}$ in the sense of Definition 10. Concretely, there exists a PPT simulator Sim such that, for every PPT adversary $\mathcal{A}$, there exist PPT adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4$ with*

$$\text{Adv}_{\Pi_{\text{base}}, \mathcal{P}_{\text{base}}, \mathcal{L}_{\text{base}}}^{\text{priv}}(\mathcal{A}, \text{Sim}) \leq \text{Adv}_{\text{ZK}}^{\text{zk}}(\mathcal{B}_1) + \text{Adv}_{\text{ZK}}^{\text{zk}}(\mathcal{B}_2) + \text{Adv}_{\text{AT}}^{\text{priv}}(\mathcal{B}_3) + \text{Adv}_{\text{Com}}^{\text{hide}}(\mathcal{B}_4).$$

Proof sketch. We proceed through a sequence of hybrids, where H_0 is $\text{Real}_{\Pi_{\text{base}}, \mathcal{P}_{\text{base}}}(\mathcal{A})$ of Fig. 3. In H_0 , the combined view of the Notary and the Verifier is $(\text{view}_{\text{N}}, x, \tau_{\text{TLS}}, \pi)$ with $x = (\text{id}, \mathbf{v})$ and $\pi = (\pi_{\text{id}}, \pi_{\mathbf{v}})$.

Hybrids H_1 and H_2 . In H_1 , we replace π_{id} with a proof simulated by the zero-knowledge simulator of ZK from its public statement $(\mathbf{C}, \text{id})$, where $\mathbf{C}$ is read from τ_{TLS} . In H_2 , we also replace $\pi_{\mathbf{v}}$ with a proof simulated from $(\mathbf{C}, \mathbf{v})$. The adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ run $\mathcal{A}_1$ and an honest **Attest** themselves, so they know the witnesses (m, r, i_{id}) and $(m, r, i_{\mathbf{v}})$ and can embed the zero-knowledge challenge. Although the two proofs share m, r , and $\mathbf{C}$, the zero-knowledge property allows them to be replaced one at a time. In H_2 , the proofs depend only on $(\mathbf{C}, x)$.

Hybrid H_3 . In H_3 , the game computes $\mathbf{C} = \text{Com}(m; r)$ itself and replaces $(\text{view}_{\text{N}}, \tau_{\text{TLS}})$ with $\text{Sim}_{\text{TLS}}(sk_{\text{notary}}, \mathcal{L}_{\text{TLS}}(q, m, ([0, |m|])), \mathbf{C})$. The adversary $\mathcal{B}_3$ plays the choice of $\mathcal{A}_1$ in the game of Definition 4, receives either the real or the simulated pair, reads $\mathbf{C}$ from τ_{TLS} , and simulates the two proofs as in H_2 .

Hybrid H_4 . In H_4 , we replace $\mathbf{C}$ with a commitment $\text{Com}(0^{|m|}; r)$ to an all-zero string of the same length. The adversary $\mathcal{B}_4$ embeds its hiding challenge as $\mathbf{C}$.

In H_4 , the combined view is computed from $pp, sk_{\text{notary}}, x, \mathcal{L}_{\text{TLS}}(q, m, ([0, |m|])),$ and $|m|$, all of which can be computed from $\mathcal{L}_{\text{base}}(m, x)$. We define Sim as this procedure, so that H_4 is $\text{Ideal}_{\Pi_{\text{base}}, \mathcal{P}_{\text{base}}, \mathcal{L}_{\text{base}}, \text{Sim}}(\mathcal{A})$. Summing the differences gives the stated bound. $\square$

6 Implementation and Evaluation

6.1 Implementation

The current implementation of the GOVBIND protocol covers the Turkish governmental document certification system, e-Devlet, and implements three different explanatory use cases. Even though the design is agnostic to the type of verification mechanism, different types of documents, even within e-Devlet, may require document-specific modifications in the ZK proof circuit implementations.

The implementation may be summarized in 4 stages:

1. Preparation of the Verification Session
2. zkTLS Connection and Authorization
3. zk-SNARK Proof Generation
4. Protocol Verification

Preparation of the Verification Session Before zkTLS proving, the Prover performs ordinary HTTPS requests to the Government Server to prepare the verification session, as described in Section 4. Our implementation targets Türkiye’s e-Devlet platform.

Performing this preparation before the zkTLS session is an implementation choice. Multiple requests could instead pass through the Notary, authenticating the entire verification flow rather than only the final download. However, for the

claim proved here, the Verifier needs to authenticate only the final response from the approved Government Server, and any earlier navigation steps only establish the session state needed to request m .

The verification script is implemented in TypeScript. It extracts the information needed for verification from the original document uploaded by the Prover and obtains the cookies and session state required for the final request. The basic idea of the preparation script is generic except that different governmental systems may use different TLS versions, cookie types, etc. Note that the verification URL on the e-Devlet server is public. However, this protocol can also be extended to authenticated verification services as long as they use a client-side authentication mechanism. In this case, the authentication would again happen inside the preparation script.

Finally, note that this preparation may not be necessary in some other systems, i.e., where it is possible to download the verification PDF with a single request sending the document identifiers. In this case, the preparation script is completely unnecessary, which can be understood as a trivial case of the preparation script.

zkTLS Connection and Authorization For the zkTLS part, our implementation extends the existing TLSNotary package with minor changes [14,28].

First, the Notary service is implemented as a minimal Rust server, since TLSNotary and its packages are written in Rust. We use TLSNotary in Proxy mode, where the Notary forwards the encrypted traffic between the Prover and the Government Server. The Notary issues a signed attestation that binds the authenticated server identity and session information to the generated commitment. As a flexibility of TLSNotary, multiple signature schemes may be chosen for the Notary, but we use an ECDSA key pair over secp256k1 to prioritize compatibility with widely used blockchains, such as Ethereum. Finally, the Prover derives a selectively disclosed presentation from the attestation of Notary and commitment.

Our first modification in TLSNotary increases the maximum number of multiplexed protocol streams from 128 to 4096, as the default value is not enough for our PDF workload, and this parameter is not exposed to be modified by TLSNotary. This allows the TLSNotary session to last longer, without affecting how the protocol actually works.

TLSNotary uses SHA-256 for transcript commitments proved with QuickSilver’s binary computation model [34,28]. We retain the same hash in the content proof so that the TLSNotary commitment can be used directly as a public input. Although a field-oriented hash may be cheaper in an arithmetic circuit, using one here would require an additional binding step between the two proof stages.

Finally, we perform a very important performance optimization, i.e. the Optimized Protocol, which is the main reason our implementation becomes feasible for real-life use. In our first implementation of the Baseline Protocol, partial benchmarking revealed that complete hashing of the document is not feasible to be performed inside a zk-SNARK. As a result, we designed the Optimized

Protocol which instead commits separately to sensitive ranges and reveals the remaining bytes, decreasing the amount of hashing significantly in the content proof as a result. Moreover, we show in Section 6.2 that this change preserves binding to one authenticated response as long as the Verifier checks exact coverage, and that the approved document profile fixes the meaning of the accepted claim when it is sound.

However, in order to achieve this optimization we needed to perform a second modification over the TLSNotary: To allow the Verifier to inspect the commitments covering the PDF body, we have modified TLSNotary to expose the transcript commitments. In TLSNotary, it is already possible to commit on different parts of a response without hiding the rest, but the commitments are not normally exposed inside the final presentation. We modify this property to be able to implement our proposed improvement.

To summarize, the Optimized Protocol creates separate SHA-256 commitments for private ranges and reveals every safe response-body byte. This structure allows the Verifier to derive the expected ranges from the public PDF structure and check that the safe and private ranges form an exact, non-overlapping partition of the PDF document.

zk-SNARK Proof Generation As explained before, our first approach was to open commitments to complete PDF documents inside the proof system, which was not possible to be completed in a feasible time over zk-SNARKs in a client-side device. As a result, we introduced selective-range commitments in the Optimized Protocol, which decreased the private hashing workload so that we were able to use a client-side zk-SNARK system, i.e., UltraHonk, for local proof generation. On the other hand, partial commitments introduced a new requirement: In order to preserve the soundness, we needed the circuit to prove that the committed private range is consistent with an approved PDF document profile.

Unfortunately, PDF documents are not simple text, and their content may be influenced by parts of the file that do not follow each other sequentially. We therefore divide this validation between two components. The native verifier parses the selectively disclosed PDF structure and verifies the exact public and private range partition. The Noir circuit opens the TLSNotary commitment to the private compressed page-content stream, constrains its DEFLATE decompression, and checks the exact static template and the claim-specific predicate. It does not reconstruct or interpret the entire PDF.

Although this optimization is important for performance, it decreases the adaptability of GOVBIND: each supported document type or static template may require a separate circuit. We discuss this limitation in Section 7. Nevertheless, by implementing these profile-specific modifications, we extend our circuit for three different use cases, which are discussed in Section 6.3.

Protocol Verification The final proof record contains the TLSNotary presentation, the public range commitments, the UltraHonk proof and public inputs,

and the identity of the Noir program and verification key. It does not contain the PDF, private content stream, commitment openings, cookies, or other private witnesses.

To show and benchmark protocol verification, we implement a verifier script that checks the trusted Notary key and signature, the e-Devlet server identity, the authenticated HTTP request and response, and the exact PDF range partition. The script asserts that the content-stream commitment and length authenticated by TLSNotary equal the corresponding Noir public inputs. Finally, it verifies the public claim parameters, pinned program identity, verification key, and UltraHonk proof. Additionally, the Verifier must separately decide whether the signed session time and public claim parameters satisfy its freshness policy as successful cryptographic verification alone does not establish present-day freshness.

6.2 Security of the Optimized Protocol

The Optimized Protocol replaces the complete-body commitment with commitments to private ranges and reveals the remaining body bytes. It instantiates Definition 5 with a document profile $\mathcal{P}$ whose layout function $\text{Lay}_{\mathcal{P}}$ and public check $\text{Check}_{\mathcal{P}}$ encode the approved server and response shape and the public PDF parser and range-derivation rules, whose selector $J_{\mathcal{P}}$ picks the claim-bearing content stream, and whose content relation $\mathcal{R}_{\mathcal{P}}^{\text{cnt}}$ captures the accepted decompressed templates and the claim predicate for a public claim x . We denote the resulting scheme by Π_{opt} , given in Fig. 6. The meaning of an accepted claim is fixed by profile soundness (Definition 6), which is an assumption about the issuer format and the approved parser and template, not a consequence of the cryptographic proof system. A changed issuer format therefore requires a new reviewed profile.

The implementation interprets a response through this profile’s byte-level rules, rather than claiming equivalence for every possible PDF renderer. The native verifier requires an unambiguous profile-approved PDF structure, derives the private ranges semantically from the page and its content references rather than fixed object numbers, and rejects unsupported additional or incremental structures. It also checks the public font or character encodings needed by the profile, where applicable. The circuit opens only the claim-bearing content commitment, constrains its DEFLATE decompression and accepted template, and checks the profile-specific field relation. Other opaque authenticated ranges that do not affect the claim need not be opened by the circuit.

The Optimized Protocol commits only to private document ranges, so its soundness depends on the profile correctly validating both the disclosed PDF structure and the committed content. Exact partition verification prevents the Prover from omitting or overlapping authenticated bytes, but does not by itself rule out parser ambiguity, unusual cross-reference structures, font differences, or divergent rendering behavior. The implementation is deliberately fail-closed for the reviewed formats. Although the profile checks a designated field rather than

```

Setup( $1^\lambda, \mathcal{P}, pk_{notary}, \varphi$ )
1.  $(pk_{\mathcal{P}}, vk_{\mathcal{P}}) \leftarrow \text{ZK.Setup}(1^\lambda, \mathcal{R}_{\mathcal{P}})$ 
2. return  $pp = (\mathcal{P}, pk_{notary}, pk_{\mathcal{P}}, vk_{\mathcal{P}}, \varphi)$ 

Prepare( $\mathcal{P}(aux), \mathcal{S}_d$ ): as in Fig. 4.

Attest( $\mathcal{P}(pp, q), \mathcal{N}(sk_{notary}), \mathcal{S}_d$ )
1. run AT.Attest in Proxy mode with the layout function  $\text{Lay}_{\mathcal{P}}$ 
2. the Prover obtains  $m = \mathcal{S}_d(q)$ ,  $\rho = (r_I)_{I \in \Lambda^{\text{priv}}}$  for  $\Lambda = \text{Lay}_{\mathcal{P}}(m)$ , and  $\tau_{\text{TLS}}$ ,
   where  $\mathbf{C}_I = \text{Com}(m[I]; r_I)$ 

Prove( $pp, x, \tau_{\text{TLS}}, m, \rho$ )
1. read  $\Lambda$  and  $(\mathbf{C}_I)_{I \in \Lambda^{\text{priv}}}$  from  $\tau_{\text{TLS}}$ , and set  $J \leftarrow J_{\mathcal{P}}(\Lambda)$ 
2.  $\pi \leftarrow \text{ZK.Prove}(pk_{\mathcal{P}}, (x, (\mathbf{C}_I, |I|)_{I \in J}); (m[I], r_I)_{I \in J})$ 
3. return  $\pi$ 

Verify( $pp, x, \tau_{\text{TLS}}, \pi$ )
1.  $\mathcal{V} \leftarrow \text{VerifyPres}(pk_{notary}, \tau_{\text{TLS}})$ ; if  $\mathcal{V} = \perp$  then return 0
2. if  $\mathcal{V}.d \neq d$  or  $\text{End}_e(\hat{q}) = 0$  then return 0
3. if  $\text{Check}_{\mathcal{P}}(\Lambda, (m_I)_{I \in \Lambda^{\text{pub}}}, x) = 0$  then return 0 (re-derives  $\Lambda$  and checks
   the exact partition)
4. if  $\varphi(t_s, x) = 0$  then return 0
5. return  $\text{ZK.Verify}(vk_{\mathcal{P}}, (x, (\mathbf{C}_I, |I|)_{I \in J_{\mathcal{P}}(\Lambda)}), \pi)$ 

```

Fig. 6. The Optimized Protocol Π_{opt} in the syntax of Definition 7.

arbitrary substring inclusion, each new profile still requires a separate semantic review.

For each private range $I \in \Lambda^{\text{priv}}$, TLSNotary computes

$$\mathbf{C}_I = \text{SHA256}(m[I] \parallel r_I), \quad r_I \in \{0, 1\}^{128},$$

i.e., the commitment Com of Section 2 with $\lambda_r = 128$. The 16-byte blinder is appended directly, without a length prefix or domain separator. For the claim-bearing content-stream range, the content circuit hashes the identical byte sequence. Its 32-byte digest is encoded as four consecutive big-endian 64-bit limbs in the public inputs, and the separately encoded compressed length must equal the authenticated content-range length. The implementation checks this encoding on both sides of the composition.

TLSNotary authenticates the disclosed ranges and private commitments under one attestation [29,32]. The Verifier re-derives the expected ranges from the disclosed PDF structure and accepts only if the disclosed and private ranges give exact, ordered, non-overlapping coverage of the response body.

Theorem 4 (Optimized Protocol Correctness). *Assume that every honest execution of AT is accepted by VerifyPres with the attested view of Definition 1,*

that $\text{Check}_{\mathcal{P}}$ re-derives $\text{Lay}_{\mathcal{P}}(m)$ from the disclosed bytes of m , and that ZK is complete. Then Π_{opt} is correct in the sense of Definition 8.

Proof sketch. As in Theorem 1, the checks on $\mathcal{V}$, End_e , and φ pass for an honest execution. Since $\text{Lay}_{\mathcal{P}}(m)$ witnesses $\text{Accept}_{\mathcal{P}}(m, x) = 1$, the check $\text{Check}_{\mathcal{P}}$ accepts, and the openings $(m[I], r_I)_{I \in J_{\mathcal{P}}(\Lambda)}$ of the claim-bearing ranges form a witness for $\mathcal{R}_{\mathcal{P}}$, so the proof verifies by the completeness of ZK . $\square$

Theorem 5 (Optimized Protocol Soundness).

Assume that AT is presentation-sound (Definition 3), that Com is binding, and that ZK is knowledge-sound. Then Π_{opt} is sound with respect to $\mathcal{P}$ in the sense of Definition 9. Concretely, for every PPT adversary $\mathcal{A}$, there exist PPT adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ such that

$$\text{Adv}_{\Pi_{\text{opt}}, \mathcal{P}}^{\text{snd}}(\mathcal{A}) \leq \text{Adv}_{\text{AT}}^{\text{pres}}(\mathcal{B}_1) + \text{Adv}_{\text{ZK}}^{\text{ks}}(\mathcal{B}_2) + \text{Adv}_{\text{Com}}^{\text{bind}}(\mathcal{B}_3).$$

If $\mathcal{P}$ is additionally sound (Definition 6), then, except with the same probability, every accepted proof record $(x, \tau_{\text{TLS}}, \pi)$ implies $\text{Pred}_{\mathcal{P}}(\text{Sem}_{\mathcal{P}}(m), x) = 1$ for the recorded response m , i.e., the profile's claim predicate holds for x .

Proof sketch. The games follow those of Theorem 2, with $\mathcal{O}_{\text{Prove}}$ answered in the same way. In G_1 , presentation soundness gives a recorded exchange (q, m, ρ, t) with $\text{Cons}(q, m, \rho, t, \mathcal{V}^*) = 1$, which binds every disclosed byte and each private-range commitment to this single authenticated response, except with probability $\text{Adv}_{\text{AT}}^{\text{pres}}(\mathcal{B}_1)$. The consistency conditions (ii) and (iii) state that Λ^* is an exact, ordered decomposition of m with no omitted or overlapping body byte, and that $m_I = m[I]$ for every $I \in \Lambda^{*\text{pub}}$. Hence, the check of Verify gives $\text{Check}_{\mathcal{P}}(\Lambda^*, (m[I])_{I \in \Lambda^{*\text{pub}}}, x^*) = 1$. In G_2 , the knowledge extractor of ZK outputs openings $(m'_I, r'_I)_{I \in J}$ for the claim-bearing ranges $J = J_{\mathcal{P}}(\Lambda^*)$ that satisfy $\mathcal{R}_{\mathcal{P}}$, except with probability $\text{Adv}_{\text{ZK}}^{\text{ks}}(\mathcal{B}_2)$. In G_3 , if $m'_I \neq m[I]$ for some $I \in J$, then $(m[I], r_I)$ with $r_I \in \rho$ and (m'_I, r'_I) are two different openings of the same commitment, which $\mathcal{B}_3$ outputs. Thus, in G_3 , the extracted claim-bearing content and the disclosed bytes belong to the same authenticated response m , and the remaining private ranges stay opaque and need not be opened by the content circuit. It follows that $\mathcal{R}_{\mathcal{P}}^{\text{cnt}}(x^*, (m[I])_{I \in J}) = 1$, so Λ^* witnesses $\text{Accept}_{\mathcal{P}}(m, x^*) = 1$, and G_3 returns 0. If $\mathcal{P}$ is sound, $\text{Accept}_{\mathcal{P}}(m, x^*) = 1$ further implies the stated claim. $\square$

Definition 12 (Optimized Protocol Leakage).

For privacy, define $\mathcal{L}_{\text{opt}, \mathcal{P}}(m, x)$ to contain $\mathcal{L}_{\text{base}}(m, x)$, the profile identity and verification parameters, public claim x , all disclosed PDF bytes, and the ordered offsets and lengths of every private range, including the compressed claim-bearing content length. These values are intentional public outputs or metadata and are not claimed to be hidden.

Theorem 6 (Optimized Protocol Privacy). Assume that AT is $\mathcal{L}_{\text{TLS}}$ -private (Definition 4) with $\mathcal{L}_{\text{TLS}}(q, m, \text{Lay}_{\mathcal{P}}(m))$ computable from $\mathcal{L}_{\text{opt}, \mathcal{P}}(m, x)$, that Com is computationally hiding, and that ZK is zero-knowledge. Then Π_{opt} is

$\mathcal{L}_{\text{opt}, \mathcal{P}}$ -private with respect to $\mathcal{P}$ in the sense of Definition 10. Concretely, there exists a PPT simulator Sim such that, for every PPT adversary $\mathcal{A}$, there exist PPT adversaries $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ with

$$\text{Adv}_{\Pi_{\text{opt}, \mathcal{P}}, \mathcal{L}_{\text{opt}, \mathcal{P}}}^{\text{priv}}(\mathcal{A}, \text{Sim}) \leq \text{Adv}_{\text{ZK}}^{\text{zk}}(\mathcal{B}_1) + \text{Adv}_{\text{AT}}^{\text{priv}}(\mathcal{B}_2) + k \cdot \text{Adv}_{\text{Com}}^{\text{hide}}(\mathcal{B}_3),$$

where k is the number of private ranges.

Proof sketch. The hybrids follow those of Theorem 3. Fix the declared leakage and public claim. In H_1 , zero knowledge allows π to be replaced by a proof simulated from its public statement $(x, (\mathbf{C}_I, |I|)_{I \in J})$, which uses no private opening or document field. In H_2 , TLSNotary privacy allows $(\text{view}_N, \tau_{\text{TLS}})$ to be replaced by the output of Sim_{TLS} on $\mathcal{L}_{\text{TLS}}$ and the honestly computed commitments, which hides the undisclosed transcript contents from the combined view. In the hybrids $H_{2+1}, \dots, H_{2+k}$, we replace the private-range commitments one at a time with commitments to all-zero strings of the same length. Commitments to any two equal-length private ranges are computationally indistinguishable under the hiding assumption, and a standard hybrid argument yields the factor k . The final hybrid is determined only by $\mathcal{L}_{\text{opt}, \mathcal{P}}(m, x)$, and we define Sim as this procedure. A distinguisher for any additional private content would therefore break TLSNotary privacy, commitment hiding, or zero knowledge of the ZK argument. $\square$

This theorem is relative to explicit leakage. The fresh 16-byte blinders computationally hide the contents of the range commitments, but they do not hide the disclosed PDF structure, range offsets, or compressed lengths. These lengths may distinguish hidden values when an issuer uses a small number of predictable document variants. The result therefore guarantees no additional leakage beyond $\mathcal{L}_{\text{opt}, \mathcal{P}}$, but it does not claim that the disclosed metadata is semantically independent of the hidden contents.

6.3 Case Studies

In this section, we discuss three potential applications of the GOVBIND protocol. In all cases, we assume that the Verifier has authenticated the applicant by other means, since GOVBIND alone does not bind the document to the Prover (Section 7). All use cases here are implemented for e-Devlet, the Turkish governmental document-certification system. We further discuss potential parties that can run the Notary without increasing trust assumptions or overcomplicating the application.

Tax Debt The first use case is proving that a person has no overdue tax debt according to a document issued by the Turkish Revenue Administration. This claim can support financial decisions such as loan or lease applications, credit assessment, and onboarding for other financial services. Ordinarily, an applicant may need to disclose the complete document even though the institution only needs the debt-status claim.

Our implementation proves the document’s no-debt statement and binds it to the public identity number and issuance date. Other fields in the document, including the person’s name, tax number, petition number, and additional dates, remain private. Thus, a financial institution can verify the relevant tax-debt status without collecting unrelated personal information.

This use case is relevant for two reasons. First, it supports common financial decisions, and our benchmarks (Section 6.4) indicate that its costs are compatible with such workflows. Secondly, the document is actually verified on a different URL that again belongs to the Turkish government, which follows a slightly simpler verification path. This indicates that supporting additional endpoints of the same issuer requires only a new preparation step and document profile.

Driver Penalties The second use case concerns the number of traffic penalties and the penalty points associated with a driver’s license. These values can be relevant to motor insurance, vehicle financing or leasing, car rental, and commercial fleet eligibility. In these settings, the Verifier may only need to know whether the applicant falls below an accepted risk threshold, rather than learn the exact values or receive the complete driving record.

Our implementation proves that each of three private values, i.e., the traffic-ticket count, total penalty points, and active penalty points during the last year, is strictly below a corresponding public threshold chosen by the Verifier. The proof hides the actual values as well as the driver’s identity details and license information. This gives financial and mobility-service providers a limited risk signal while reducing the personal data they must collect and retain. As with the tax-debt case, the surrounding service must authenticate the applicant and enforce an appropriate freshness policy.

Residential Address City The city that someone lives in can be useful information in various use cases without needing the rest of the address details. Some examples include geographic access restrictions, local civic participation, local benefits or services, or some event or program eligibility. In all these examples, the residential city may be proved to the respective Verifier, while none of the other address details are relevant. Here, GOVBIND’s example implementation solves this issue by proving the city on a given residential address document.

Disclosing a complete address exposes its holder to risks such as stalking and targeted fraud. On the other hand, the city reveals substantially less information about the person than the complete address. GOVBIND therefore provides a substantial privacy improvement in this use case.

Possible Notary Identities Finally, let us discuss the trust assumptions over a possible Notary. Assume that a civil organization or community is trying to verify the tax debt, driver penalties, or the addresses of applicants for some eligibility criterion, which can be anything from local services to community participation. In this scenario, having the organization run the Notary service

itself is a trust-minimized and accessible option, since the organization does not need to trust an additional third-party Notary. If the organization already has an online service allowing its users to connect and claim eligibility, it can integrate the additional Notary step, whose performance requirements are discussed more in Section 6.4.

As a result, we can observe that as long as the Verifier can run an online service, such as a registration page, it may also host a Notary service. Of course, using Proxy mode in such situations introduces the additional limitations discussed in Section 7. Adopting MPC mode can provide stronger security guarantees and allows the Prover to connect directly to the Government Server, but requires greater protocol bandwidth and computation.

A general conclusion is that, for many real-world applications, a Verifier running the Notary itself is an acceptable trust model, and thus we define our privacy adversary in Section 3.3 with the combined view of Notary and Verifier. Our security results hold for any honest Notary. When the Verifier operates the Notary, the resulting proof is interactive in effect and non-transferable, which suits the scenarios above. By contrast, publicly verifiable use, e.g., on a public ledger (Section 7), requires a Notary trusted by all verifiers, such as a committee.

6.4 Evaluation

We evaluate the Optimized Protocol for the tax-debt, driver-penalty, and residence-city profiles. The benchmark machine was an Apple M2 with eight logical cores and 16 GiB of memory, running Darwin 23.4.0. We used Noir 1.0.0-beta.22 [20] and Barretenberg 5.0.0-nightly.20260522 [2] with UltraHonk and the EVM verifier target.

For each profile, we used one document from each of two consenting researchers and ran four sessions per document. Every session generated three proofs and verified each proof ten times, giving 24 sessions, 72 proof generations, and 720 verification measurements in total. The session order was rotated across profiles and documents. Each session first retrieved one response from the e-Devlet server through the Notary. To avoid rate limiting, every measured generation then re-executed the TLSNotary protocol against a local replay of that captured TLS exchange. The Notary ran on the same machine, so the zk-TLS timings exclude wide-area round trips to the Government Server. Session preparation and capture time are excluded from the results.

Unless stated otherwise, the experimental unit is a session: each plotted point is the mean of its three proof runs, and error bars show a 95% Student- t confidence interval over the eight session means for that profile. Circles and triangles distinguish the two documents without identifying their owners.

Fig. 7 shows mean end-to-end generation times of 53.57 s for tax debt, 111.50 s for driver penalties, and 118.94 s for residence. The respective zk-TLS/Noir stage means were 13.58/40.01 s, 47.03/64.49 s, and 12.29/106.67 s. Thus, Noir dominates tax-debt and residence generation, while zkTLS is substantial for the larger driver response. Mean Noir peak RSS was 9.09, 11.03, and 11.57 GiB, respectively, and the largest individual observation was 11.95 GiB.



Fig. 7. Prover cost for the three evaluated profiles. Left: mean generation time split between fresh zkTLS generation and the Noir/UltraHonk stage, including its self-verification where points are session means and the whisker covers total-generation uncertainty. Right: mean peak RSS measured separately for the two sequential stages

Table 1. Mean artifact and response sizes over 24 generated proofs per profile. “Private ranges” includes every opaque PDF range committed by TLSNotary. “Noir stream” is the claim-bearing compressed stream opened by the content proof. Parentheses give percentages of the response body.

Profile	Body (KiB)	Private ranges (KiB, %)	Noir stream (KiB, %)	Presentation (KiB)	Proof (KiB)	Public inputs (B)
Tax debt	89.4	2.5 (2.7%)	1.8 (2.1%)	93.3	11.2	224
Traffic penalties	185.8	6.4 (3.4%)	2.9 (1.6%)	186.1	11.2	256
Residence city	83.6	5.2 (6.2%)	5.1 (6.1%)	84.7	11.6	1248

Therefore, even though local proving is completed in under two minutes on this machine, its memory demand was significant.

As shown in Fig. 8, mean cold/warm verification times were 46.35/43.68 ms for tax debt, 89.92/70.14 ms for driver penalties, and 127.39/89.31 ms for residence. We keep the first verification separate rather than treating all 720 repeated measurements as independent. All generation and verification runs completed successfully.

Table 1 shows that the content circuit opens only 1.6–6.1% of the response body, while all private ranges account for 2.7–6.2%. UltraHonk proofs are 11.2–11.6 KiB, and most transmitted bytes belong to the TLSNotary presentation, which authenticates the disclosed PDF structure and private-range commitments.

Note that this benchmark does not include the complete-body Baseline Protocol and therefore does not establish a numerical speedup over it.

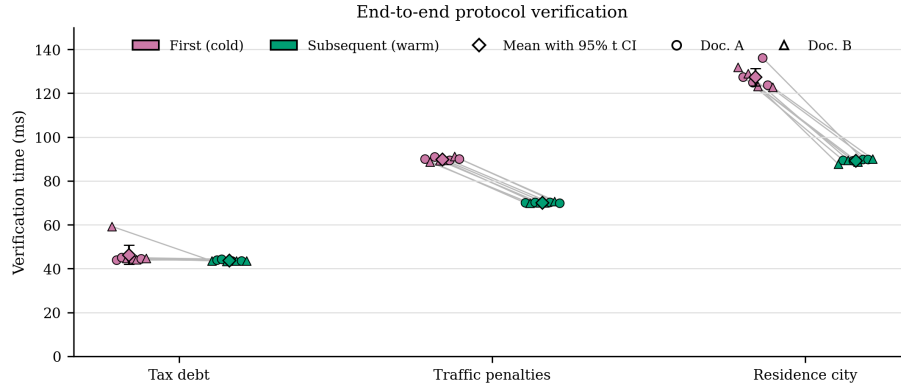


Fig. 8. End-to-end protocol verification. For each generated proof, “cold” is its first verification and “warm” is the mean of the next nine, and then values are averaged within each session. Lines pair cold and warm measurements from the same session, and diamonds show the profile mean with a 95% Student-*t* confidence interval.

6.5 Ethical Considerations

Our evaluation has used synthetic inputs and, where it is necessary to exercise the live e-Devlet verification flow (since proper testing material is not available), only government documents belonging to researchers were used, who voluntarily authorized their use. No third-party documents were collected or processed. No real document, personal identifier, authentication credential, or similar is included in the paper or released artifacts, and sensitive test data was not retained beyond the experiments.

Our interaction with e-Devlet was limited to ordinary verification requests initiated using documents legitimately held by the participating researchers. We performed no bulk collection, identifier enumeration, or rate-limit evasion. The implementation is intended solely as a research prototype and has not undergone a production security or privacy audit, so it should not be used to process real personal data in operational settings. Deployment would additionally require an assessment of applicable law, platform terms, consent requirements, and organizational data-protection obligations. The protocol must not be used to access or prove claims about another person’s records without their authorization.

7 Limitations and Future Work

7.1 Limitations of the Current Protocol

This section focuses primarily on the implemented Optimized Protocol rather than the Baseline Protocol in Section 4.

Adaptability of the optimized circuit. Due to the customized nature of each circuit, existing proofs cannot be used for a different document type or profile version. The ZK argument constrains the private compressed content stream, so a changed format may require a different circuit and verification key. Maintaining and auditing these circuits creates an engineering burden and increases the attack surface. More general circuits may become practical with future proof-system improvements, but the present implementation deliberately uses versioned, document-specific profiles.

Subject binding and third-party possession. Authenticating a document containing a specific ID number does not bind that identity to the Prover. A party possessing another person’s document and its verification information may therefore be able to generate a proof about that person since GOVBIND does not establish the subject’s participation or consent. A lightweight mitigation lets the Verifier choose a random salt σ and adds $h = H(\mathbf{id} \parallel \sigma)$ to the public inputs, where the circuit computes $\mathbf{id}$ from the claim-bearing range. A Verifier that has already authenticated the applicant, e.g., during onboarding, can compare h with the identifier it holds, without the proof revealing $\mathbf{id}$ to other parties. Binding the proof to the holder without such a prior step requires an additional identity mechanism, which we leave for future work.

Scalability. The system scales with the authenticated transcript size and proof configuration. Larger responses increase the zkTLS workload, while larger private content streams increase the circuit workload. The present design therefore targets bounded document profiles and does not establish efficient proving for arbitrary or multi-page documents.

The private content-stream length is separate from the total PDF size. Selective commitments reduce the private witness, but some settings may require the complete document to remain hidden. Such documents would lose this optimization and may not be suitable for local proving with the current construction.

Replay and transferability. The current protocol does not bind a proof to a particular Verifier or verification session, so an accepted proof can be transferred or reused during its accepted freshness window. Deployments that require one-time use need an additional replay-prevention mechanism, such as a verifier challenge or a claim-specific nullifier. A simple instantiation may add a Verifier-chosen nonce as a disclosed request header of q . The Government Server ignores it, but the attestation binds it to the session, so the Verifier accepts a proof record only for its own fresh nonce. This requires an interactive Verifier and is not implemented in our prototype.

Trust assumption over the Notary. As discussed in Section 3.2, a deployment must select an appropriate Notary and trust policy. The current implementation uses Proxy mode, and its interaction with provenance of the MPC-mode has not been established in this work.

Liveness assumptions. The Notary and Government Server must be available during proof generation. Either party may refuse service, and the Government Server may rate-limit verification requests. A Verifier-operated Notary reduces this to the Verifier’s ability to refuse an interaction, but does not remove the Government Server’s availability requirement.

7.2 Future Work

Generalization of the Protocol. An important direction for real-world adoption is increasing the scope of the implementation across governmental systems and document types. Combining multiple documents may also support more expressive claims. Future work should harden the PDF and template validation logic while reducing the need for separate circuits.

Improving Notary’s Trust Assumptions. As described in Section 3.2, mechanisms such as TEEs or decentralized quorums may redistribute trust in the Notary. None of these options is implemented here. MPC mode is another possible direction, but its interaction with the current protocol modifications remains to be implemented and evaluated.

Recursive Proof Composition for Verification. Right now the Verifier is required to perform multiple steps to verify the entire protocol. This process can be improved by generating a single proof verifying all related information and sending this proof to the Verifier. This includes a recursive proof verification, which is already supported in the chosen proof system for our content-proof implementation. However, being able to verify TLSNotary presentation inside the circuit remains future work. Moreover, this process increases proof generation costs significantly, while only introducing a constant improvement in the communication costs as the Prover already needs to communicate a constant number of proofs to the Verifier. However, in cases where the protocol is used multiple times or the verification happens on a very restrictive environment, e.g., blockchain, the recursive proof composition can be useful.

Potential Financial Implications through Public Ledgers. Being able to prove governmental claims privately is interesting on its own, but decentralized financial systems and blockchains offer one of the most prominent future directions. By using the GOVBIND protocol, for instance to prove a credit score, one may be able to get a decentralized credit over blockchains. Such use cases of course require a subject or holder binding, freshness enforcement, and replay/nullifier protection, and we consider them a natural direction for future work.

8 Conclusion

In this paper, we have proposed a new protocol, GOVBIND, to prove confidential claims over private governmental documents without necessarily requiring

modifications by government-side issuers. The protocol combines existing work on zkTLS and zk-SNARKs and can be adapted to governmental systems that provide an HTTPS endpoint for downloading private documents, although each new system and document profile may require implementation-specific changes. We formalized GOVBIND through an abstraction of TLSNotary and document profiles, and proved the correctness, soundness, and privacy of both constructions through game-based arguments.

We have implemented the design for the Turkish governmental certification system. In particular, the selective-range commitment optimization reduces the private content handled by the proof system at the cost of document-specific validation logic. The main future directions are to generalize the implementation across governmental systems and document types and to improve the trust assumptions of the Notary system.

Overall, GOVBIND demonstrates that zkTLS provenance and zero-knowledge content proofs can be combined to prove claims about existing government-issued documents without requiring issuer-side modifications. At the same time, its practical deployment depends on addressing the tradeoffs in document-specific validation, Notary trust, and proof performance identified in this work.

Acknowledgments. This research was conducted during the first author’s internship at FAME Crypt.⁵

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Anon Aadhaar: Anon aadhaar: Private identity. Project documentation (2024), <https://documentation.anon-aadhaar.pse.dev/>
2. Aztec Protocol: Barretenberg: An optimized elliptic curve library and PLONK SNARK prover. GitHub repository (2025), <https://github.com/AztecProtocol/aztec-packages/tree/next/barretenberg>, version 5.0.0-nightly.20260522. Accessed 23 September 2026
3. Bitansky, N., Canetti, R., Chiesa, A., Tromer, E.: From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again. In: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. pp. 326–349. ACM (2012)
4. Breidenbach, L., Cachin, C., Coventry, A., Ji, Y., Karantias, K., Schindler, P., Stathakopoulou, C., Topliceanu, A.: Chainlink offchain reporting protocol 3.0. Technical report (2025), <https://research.chain.link/ocr3.pdf>
5. Celi, S., Davidson, A., Haddadi, H., Pestana, G., Rowell, J.: DiStefano: Decentralized infrastructure for sharing trusted encrypted facts and nothing more. In: Network and Distributed System Security Symposium (2025). <https://doi.org/10.14722/ndss.2025.241367>

⁵ <https://famecrypt.com.tr/>

6. Chen, B., Bünz, B., Boneh, D., Zhang, Z.: HyperPlonk: Plonk with linear-time prover and high-degree custom gates. In: *Advances in Cryptology – EUROCRYPT 2023*. Lecture Notes in Computer Science, vol. 14005, pp. 499–530. Springer (2023)
7. Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., Polk, W.T.: Internet X.509 public key infrastructure certificate and certificate revocation list (CRL) profile. Tech. Rep. RFC 5280, RFC Editor (2008). <https://doi.org/10.17487/RFC5280>, <https://www.rfc-editor.org/info/rfc5280>
8. Delignat-Lavaud, A., Fournet, C., Kohlweiss, M., Parno, B.: Cinderella: Turning shabby X.509 certificates into elegant anonymous credentials with the magic of verifiable computation. In: *2016 IEEE Symposium on Security and Privacy*. pp. 235–254. IEEE (2016). <https://doi.org/10.1109/SP.2016.22>
9. Ernstberger, J., Lauinger, J., Wu, Y., Gervais, A., Steinhorst, S.: ORIGO: Proving provenance of sensitive data with constant communication. *Proceedings on Privacy Enhancing Technologies* **2025**(2), 416–433 (2025). <https://doi.org/10.56553/popets-2025-0069>
10. European Commission: Commission implementing regulation (EU) 2026/1731 as regards applicable standards and specifications. Official Journal of the European Union (2026), https://eur-lex.europa.eu/eli/reg_impl/2026/1731/oj
11. European Parliament and Council of the European Union: Regulation (EU) 2024/1183 establishing the european digital identity framework. Official Journal of the European Union (2024), <https://eur-lex.europa.eu/eli/reg/2024/1183/oj>
12. Goldwasser, S., Micali, S., Rackoff, C.: The knowledge complexity of interactive proof systems. *SIAM Journal on Computing* **18**(1), 186–208 (1989). <https://doi.org/10.1137/0218012>
13. Housley, R.: Cryptographic Message Syntax (CMS). RFC 5652 (Sep 2009), <https://www.rfc-editor.org/info/rfc5652/>
14. Kalka, M., Kirejczyk, M.: A comprehensive review of TLSNotary protocol. arXiv preprint arXiv:2409.17670 (2024), <https://arxiv.org/abs/2409.17670>
15. Katz, J., Lindell, Y.: *Introduction to Modern Cryptography*. CRC Press, 3 edn. (2020)
16. Lauinger, J., Ernstberger, J., Finkenzeller, A., Steinhorst, S.: Janus: Fast privacy-preserving data provenance for TLS. *Proceedings on Privacy Enhancing Technologies* **2025**(1), 511–530 (2025). <https://doi.org/10.56553/popets-2025-0028>
17. Londhe, S., Gupta, A., Suegami, S., Shahi, Y., Figueiredo, R., Hassanizadeh, P., Ebrahimi, S.: Zero-knowledge proofs of generalized regular expression matching for anonymized email verification. *Proceedings on Privacy Enhancing Technologies* **2026**(4), 740–771 (2026). <https://doi.org/10.56553/popets-2026-0143>
18. Luo, Z., Jia, Y., Shen, Y., Kate, A.: Proxying is enough: Security of proxying in TLS oracles and AEAD context unforgeability. In: *7th Conference on Advances in Financial Technologies*. Leibniz International Proceedings in Informatics, vol. 354, pp. 4:1–4:24. Schloss Dagstuhl–Leibniz-Zentrum für Informatik (2025). <https://doi.org/10.4230/LIPIcs.AFT.2025.4>
19. National Institute of Standards and Technology: Secure hash standard (SHS). Tech. Rep. FIPS PUB 180-4, National Institute of Standards and Technology (2015). <https://doi.org/10.6028/NIST.FIPS.180-4>
20. Noir Team: Noir: A domain-specific language for zero-knowledge proofs. GitHub repository (2026), <https://github.com/noir-lang/noir>, version 1.0.0-beta.22. Accessed 23 September 2026
21. Opacity Network: What is opacity? Technical documentation, <https://docs.opacity.network/>, accessed 26 August 2026

22. Paquin, C., Policharla, G.V., Zaverucha, G.: Crescent: Stronger privacy for existing credentials. Cryptology ePrint Archive, Paper 2024/2013 (2024), <https://eprint.iacr.org/2024/2013>
23. Primus Labs: Overview of zkTLS. Technical documentation, <https://docs.primuslabs.xyz/primus-network/tech-intro/>, accessed 26 August 2026
24. Reclaim Protocol: Reclaim protocol whitepaper. Technical whitepaper, <https://trust.reclaimprotocol.org/section/whitepaper>, accessed 26 August 2026
25. Rosenberg, M., White, J., Garman, C., Miers, I.: zk-creds: Flexible anonymous credentials from zkSNARKs and existing identity infrastructure. In: 2023 IEEE Symposium on Security and Privacy. pp. 790–808. IEEE (2023). <https://doi.org/10.1109/SP46215.2023.10179430>
26. Rushi, V.: zkPDF: Unlocking verifiable data in the world’s most popular document format. Privacy and Scaling Explorations blog (2025), <https://pse.dev/blog/zkpdf-unlocking-verifiable-data>
27. Saint-Andre, P., Salz, R.: Service identity in TLS. Tech. Rep. RFC 9525, RFC Editor (2023). <https://doi.org/10.17487/RFC9525>, <https://www.rfc-editor.org/info/rfc9525>
28. TLSNotary: TLSNotary performance benchmarks (2025), <https://tlsnotary.org/blog/2025/08/31/benchmarks/>, accessed 31 August 2026
29. TLSNotary: Notarization. TLSNotary Protocol Documentation (2026), <https://tlsnotary.org/docs/protocol/notarization>, accessed 6 September 2026
30. TLSNotary: Proxy mode. TLSNotary Protocol Documentation (2026), <https://tlsnotary.org/docs/protocol/proxy-mode>, accessed 6 September 2026
31. TLSNotary: TLSNotary source code, version 0.1.0-alpha.15. GitHub repository (2026), <https://github.com/tlsnotary/tlsn/tree/47aee45b53e06648c1b2ad3689b367b8c923fdec>, commit 47aee45b53e06648c1b2ad3689b367b8c923fdec
32. TLSNotary: Verification. TLSNotary Protocol Documentation (2026), <https://tlsnotary.org/docs/protocol/verification>, accessed 6 September 2026
33. Xie, X., Yang, K., Wang, X., Yu, Y.: Lightweight authentication of web data via garble-then-prove. In: Proceedings of the 33rd USENIX Conference on Security Symposium. SEC ’24, USENIX Association, USA (2024)
34. Yang, K., Sarkar, P., Weng, C., Wang, X.: QuickSilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In: Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security. pp. 2986–3001. ACM (2021). <https://doi.org/10.1145/3460120.3484556>
35. Zhang, F., Cecchetti, E., Croman, K., Juels, A., Shi, E.: Town crier: An authenticated data feed for smart contracts. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (2016)
36. Zhang, F., Maram, D., Malvai, H., Goldfeder, S., Juels, A.: DECO: Liberating web data using decentralized oracles for TLS. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security (2020)
37. zkPass: zkPass technical whitepaper. Technical whitepaper (2025), https://paper.zkpass.org/zkPass_WP2025.pdf