Age-based Eligibility meets User Agency:
Client-side proof, send nothing, achieve compliance

Thibault Meunier Marwan Fayed
Cloudflare, Inc.

Enforcing age-based restrictions at the server creates a tension between enabling compliance and preserving clients' privacy. We argue that both are achievable by expanding the problem statement to include *unsolicited content* alongside restricted material. Doing so means we can prevent accidental exposure to ineligible material for those who should not have it; we can also include adults who do not want to be exposed to certain materials that may appear without warning or consent. Both are achievable even in the presence of VPNs.

In this document, we argue that a client-side proof that sends nothing to the server is sufficient for compliance, and has stronger privacy properties, with minimal server changes which facilitate wider adoption. In addition this model of delivery can cater to an important demographic: Eligible users who do not wish to be accidentally exposed to *unsolicited content*, and should not have to express that preference to servers.

Given reasonable operating system and software supports, eligible and ineligible content preferences can be expressed and provably enforced by the client --- without returning data to the server or involving third parties while browning the web and accessing the Internet.

Prove on the client and send nothing

The key that unlocks our arguments begins with a question:

Why should the client send a proof for something it knows will be refused?

In daily life we expect to present ID for access to alcohol or cigarettes, or gain admission to some films at the cinema, but also to prove senior status for discounts and services. In all cases, no person would present ID knowing it does not meet the requirements. On the Internet, forcing the server to always validate eligibility means sending data that would not ordinarily be shared in real life circumstances.

A secure client-side implementation is sufficient, with two fundamental properties. First, it securely stores a credential or identification that is one-time verified by the issuer, and is tied to the user's biometrics or equivalent protection. Second, it *transmits nothing to servers or third parties*. Instead it takes a signal from the server about the nature of content. If the local proof fails, then the client is prevented from accessing the website. This is a model exchange that most closely matches real-life expectations: Verifiable credentials are used to access restricted content, and give eligible users a choice to prevent accidental exposure.

The send-nothing model encapsulates three important properties:

1. Inclusivity — a single mechanism for both restricted and unwanted content;
2. Privacy — no data is transmitted to the server or third parties;
3. Precision and Compliance at the client — **resilient to VPN and DNS circumvention**, and easier integration with app stores and online services that already label content.

We find these properties are both possible and desirable by evaluating the technical constructions, below, and showing the strawperson is ideal and sufficient.

Evaluating technical proposals leads to the send-nothing model

We assume the client securely stores an identification that is verified by the issuer and unlocked by a user secret such as a pin, biometrics, or a password. We disregard systems that inject third parties into the 'hot path' because they are prone to reveal users' activity (and potentially other information).

We begin by describing client enforcement architecture that sends nothing. We then walk through the alternatives and their merits, only to show those merits are equally met, and potentially exceeded, by sending nothing.

1. 'Naively' send nothing.

In this model the user intentionally accesses a restricted website, or the client software may receive a restricted object in a non-restricted website (i.e. 'accidental' exposure):

1.1. The server sends a signal to the client, e.g. "alcohol", "adult", or "sensitive"[1].
1.2. The client receives the signal, validates that the criteria are met locally.
1.3. If any of the criteria fail, then the client sends nothing to the server.
1.4. If the criteria are met and the signal is for the requested website, then the client transmits the request.
1.5. Alternatively if the criteria are met but the signal is for a subresource, then the client prompts the user to accept or reject the content.
1.6. The server accepts the client's request as normal, since the request must have passed local checks.

The server trusts it can only see valid requests that have succeeded securely at the client. Other than eligibility, no information is transmitted by the client. The relative simplicity of sending a signal means a server is more likely to send the signal, always, even over VPNs.

2. Selective Disclosure model

The selective disclosure model is a modification on the standard OAuth flow.

2.1. Client contacts the server
2.2. Server replies, "please confirm age"
2.3. Client returns full or partial credential, for example a birthdate; all other fields are blinded.
2.4. Server validates.

This model needs substantial modifications to every client and server, offers weak compliance, and is an immense privacy violation. Every server would need to have a list of credential issuers and means to communicate with them. Also, birthdates are personal

[1] At least some vocabulary exists in PEGI, and similar classifications.

information that is not needed to verify a user's eligibility. Finally, compliance is not guaranteed since most servers for most users lie outside of users' jurisdictions.

3. Zero-knowledge Proofs (ZKP) to the server

ZKP is attractive because it proves a statement without revealing the statement's details.

3.1. Client contacts the server
3.2. Server replies, "please confirm age"
3.3. Client generates ZKP that the age exceeds a value, and their credentials are not expired, then leverages a Privacy Pass system to transmit to the server.
3.4. Server validates the proof, and learns nothing about the client.

Every client and server will need an implementation, and to support multiple ZKP systems. Privacy Pass enables the interaction to be agnostic, and independent from the underlying cryptographic constructions. The computational complexity and cost of ZKP may be a barrier, but is likely to diminish as ZKP finds greater use and adoption[2].

Irrespective, this cost and complexity is added to every access, which affects users' experience and servers' physical requirements. Servers are likely to trigger the flow according to IP geolocation, which is imprecise and can be circumvented by VPNs.

Discussion and Final Remarks

We close where we began: **Why should the client send a proof for something it knows will be refused or for something it does not want?**

Our walkthrough should make clear that no information needs to be transmitted to the server. A send-nothing model shares many properties with the ZKP and other strictly client-side proof models. In comparison to server-enforced restrictions, ZKP offers greater compliance since devices are in-region and in the best position to understand local requirements.

Sending nothing can be coupled with ZKP but differs in subtle and important ways. In ZKP, servers out of region may or may not choose to comply. Servers that choose to comply will need either to implement complex features, or outsource to third parties. In either case servers incur costs, and inject performance penalties that negatively impact user experience. For proofs that fail, the costs are wasted. In addition, the server has to protect the client's credentials in any form---implying similar protections must already exist on the client.

Sending nothing gives the strongest privacy guarantees. Support at servers can be rapidly deployed, for instance, with a header-level change or new HTML tag. Minimal burden on the server increases likelihood of adoption, which improves compliance: A server is more likely to send the signal in the context of VPNs, and irrespective of IP geolocation or DNS behaviours.

Sending nothing is minimally invasive, easier to support, and offers greatest compliance with greater controls. If sending nothing is insufficient, then we should understand and ask why.

[2] Ladd, Watson. "Introducing Zero-Knowledge Proofs for Private Web Attestation with Cross/Multi-Vendor Hardware." Cloudflare blog, published 2021-08-12.