Privacy Pass: Bypassing Internet Challenges Anonymously

(COM-506 Report)

Majdouline Ait Yahia, Lúcás C. Meier

2022-04-04

Abstract

The internet is full of malicious and spam traffic. Some websites try to mitigate this through mechanisms like CAPTCHAs, requiring users to solve puzzles before accessing their site. These websites use fingerprinting in order to keep track of vetted users. This avoids a constant need to re-solve puzzles to access websites. On the other hand, this harms user privacy, through excessive tracking. If users try and circumvent this tracking through privacy tools like Tor [DMS04], then they're faced with a large amount of puzzles to solve once again.

This paper [DGS+18] addresses the issue by designing a protocol in which a challenge server can issue tokens to users, which can later be redeemed to access a site. By issuing multiple tokens for each challenge solved, the effort of legitimate users is reduced. To preserve their privacy, the protocol is designed so that issued tokens can't be linked with redeemed tokens, so that the activity of users can't be easily tracked.

1 Background

2 Implementation

2.1 Core Idea

The core idea of the scheme is that if you have a PRF

$$F : K \times X \to Y$$

Then evaluating $F_k(x)$ should be difficult without knowing k. Thus, we can issue a token by choosing a random t, calculating $F_k(t)$, and then sending that to the user. The user will then present (x, y) to us later, and we check that $F_k(x) = y$. It should be difficult for them to forge such a pair without knowing k.

One problem is that the issuer decides what the token is, so they can trivially link the issuance and the usage later on, allowing them to track users.

To get around this, we need to design a protocol for the user and the issuer to work together, in order for the user to learn $F_k(x)$, without learning k, and without the issuer learning x or $F_k(x)$ either. This idea is known as an *Oblivious PRF* (OPRF).

2.2 More Concretely

Now, let's go over their implementation of an OPRF.

A long time ago, in a galaxy far far away, there was a cyclic group $\mathbb{G}$ of prime order q, and with generator $G \in \mathbb{G}$, suitable for use in Cryptography.

2.2.1 A first go

The PRF we use is defined as:

$$F : \mathbb{F}_q \times \mathbb{G} \to \mathbb{G}$$
$$F(k, P) := k \cdot P$$

In English, the issuer take a point P, and multiplies it by its secret scalar k.

The protocol for turning this into an OPRF uses the same principle behind a Diffie-Hellman key-exchange: multiplying by scalars commutes:

1. The user generates $T \xleftarrow{R} \mathbb{G}$, $r \xleftarrow{R} \mathbb{F}_q$, sets $P \leftarrow r \cdot T$, and sends P to the issuer.
2. The issuer computes $Q \leftarrow k \cdot P$, and sends Q to the user.
3. The user computes $W \leftarrow \frac{1}{r} \cdot Q$. The pair (T, W) is their token.

Note that $Q = k \cdot P = kr \cdot T$, so $\frac{1}{r} \cdot Q = k \cdot T$, so our protocol is at least correct.

2.2.2 Malleability

One little issue with the scheme as described is that using a pair $(T, W = k \cdot T)$, a user can calculate another valid pair, as $(\alpha \cdot T, \alpha \cdot W)$. In other words, the tokens are malleable.

To get rid of this malleability, we use the oldest trick in the book: a hash function. We define a hash function $H_1 : \{0,1\}^\lambda \to \mathbb{G}$. Instead of picking a random point T, we pick a random *seed* $t \in \{0,1\}^\lambda$, defining $T := H_1(t)$. Our token then becomes the pair (t, W), which isn't malleable.

2.2.3 Using diferent values for k

Another issue with the scheme so far is that nothing stops the issuer from using different values of k for different users. This would allow them to link tokens together, because they would be able to tell which k was used, by virtue of the token being valid for that k.

To avoid this, we can require the issuer to commit to a public key $Y = k \cdot G$ in advance, via some kind of Public-Key-Infrastructure (PKI).

During step 2, the issuer then proves that they used the same k, in zero-knowledge:

$$\Pi(Y, Q, P; k) := Y = k \cdot G \wedge Q = k \cdot P$$

The paper cites a protocol for this ZK proof by Chaum and Pedersen [CP92]. Maurer generalized this type of proof to arbitrary one-way group homomorphisms, in [Mau09]. In this case, the homomorphism is $\varphi(x) := (x \cdot G, x \cdot P)$, and we prove that $\varphi(k) = (Y, Q)$ in zero-knowledge. This is the same type of proof as in the ubiquitous Schnorr signature.

2.2.4 Batching

It would be convenient to issue multiple tokens at the same time, which would allow us to reduce the number of puzzles users have to solve. We can do this straightforwardly by having the user generate multiple seeds t_i, send multiple points P_i, and receive multiple points Q_i in return. The zero-knowledge proof that the same k was used is still necessary, and can also be repeated multiple times. This proofs can also be combined into a more efficient batch proof, as described in [HG13], and explained in the paper. This reduces communication cost.

2.3 Redemption

We've described how to issue tokens, but not how to use those tokens to request content. The idea is that given a token (t, W), hashing this token with a hash function H_2 allows us to derive a shared key $K := H_2(t, W)$ with the issuer. The issuer can derive this key from just t, since W satisfies $W = k \cdot H_1(t)$, and the issuer knows k.

We can then redeem our token (t, W) to access some content R by sending $(t, \text{MAC}_K(R))$ to the issuer. They then recalculate K from t, and check that $\text{MAC}_K(R')$ is the same, using the content R' they're responding with. Binding the request to the content R in a way the server can recompute avoids potential issues with someone intercepting a token redemption protocol, and using that to access content.

3 Security Notions

4 Potential Problems

5 Conclusion

References

[CP92] David Chaum and Torben Pryds Pedersen'. Wallet Databases with Observers. *CRYPTO '92*, 1992.

[DGS+18] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy Pass: Bypassing Internet Challenges Anonymously. *Proceedings on Privacy Enhancing Technologies*, 2018, 2018.

[DMS04] Roger Dingledine, Nick Mathewson, and Paul Syverson. Tor: The Second-Generation Onion Router:. Technical report, 2004.

[HG13] Ryan Henry and Ian Goldberg. Batch Proofs of Partial Knowledge. In *ACNS*, volume 7954, pages 502–517. 2013.

[Mau09] Ueli Maurer. Unifying Zero-Knowledge Proofs of Knowledge. *Progress in Cryptology - AFRICACRYPT 2009*, 2009.