Comments on SWAN and Unified ID 2.0

Martin Thomson
Mozilla
mt@mozilla.com

Eric Rescorla
Mozilla
ekr@mozilla.com

August 13, 2021

Executive Summary

SWAN[1] and Unified ID 2.0[2] (UID) describe a new approach to Web tracking. Each provides a service which assigns a pseudonymous identifier to each user that can then be used for tracking or ad targeting. While these proposals do not depend on third-party cookies, they rely on other technical mechanisms to bypass browser anti-tracking mechanisms.

These proposals depend heavily on policy controls: asking the user for consent before tracking them and then restricting the use of tracking data. However, it is not possible for the user to verify that these policies are being followed and it is unclear whether it will be practical to effectively enforce them. Even if these policies were to be stronger and clearly enforceable, the end result would be a large number of entities possessing user browsing history, which is precisely the situation that browsers are currently trying to address.

1 Introduction

The basic design of the current Web advertising ecosystem involves advertisers and ad servers gathering large amounts of user browsing history information, which is then used to target advertisements. Anti-tracking technologies such as Firefox Enhanced Tracking Protection are designed to prevent that data collection, for instance by restricting the ability of third parties to store cookies on the user's browser. Most of the existing work in Privacy Preserving Ads (PPA), such as FLoC[3] or Private Click Measurement[4] is designed to allow for advertising without requiring that data collection.

SWAN and Unified ID 2.0 reflect a different approach: they take large-scale collection of user browsing history data as a given and superimpose governance controls that are intended to limit the use of user data. The general idea behind both proposals is that the user's pseudonymous identity will be gathered only by a small number of entities, who will then make it available to other entities (principally advertisers) under specific terms and conditions. For instance, here's how SWAN describes it:

> When the data leaves the SWAN network all entities that receive the data are subject to strict contractual rules that require them to meet clearly defined requirements and transparency in

[1] `https://github.com/SWAN-community/swan`
[2] `https://github.com/UnifiedID2/uid2docs`
[3] `https://wicg.github.io/floc/`
[4] `https://privacycg.github.io/private-click-measurement/`

data processing. This includes signing an audit trail to indicate receipt of the data, or if no receipt was provided the audit trail is added to by the sender signing to indicate no response was received.

These proposals should not be thought of as alternative designs for PPA. Rather, they are designs for new tracking systems that are not blocked by existing anti-tracking mechanisms, combined with advertising industry self-regulation for the handling of the resulting data. SWAN and UID 2.0 fundamentally rely on the user trusting the data collector as well as everyone they provide the data to. (This is already the case now, although not explicitly so.) By contrast, a strong PPA system would have technical controls that limit the scope of compromise even if some entities behave incorrectly, thus providing stronger and more verifiable protection to the user.

2 Technical Comparison

At a very high level, the SWAN and UID 2.0 proposals are substantially similar. They assign a pseudonymous identifier to every user which can then be used to track users and target ads to them. Data consumers (e.g., advertisers) who want to make use of user profile data are required to conform to some terms of service and in return are able to access those pseudonymous identifiers. The exact technical mechanisms for this differ to some extent between SWAN and UID 2.0, as described below.

3 How SWAN Works

The basic idea behind SWAN is that there will be a set of independent SWAN *Operators*, each of which is responsible for maintaining a stable mapping between users and a pseudonymous per-user identifier. Publisher sites are free to contract with one or more Operators, each of which has a more or less complete picture of user behavior (limited by the number of publishers they work with). When a user first visits a site that works with a given Operator, they are presented with a consent dialog that asks if they are willing to share their data.

Assuming an affirmative response from the user, the Operator generates a pseudonymous identifier and stores that value on the user's browser (e.g., as a cookie). If the user subsequently visits a site affiliated with that Operator, the publisher of that site can query the Operator for the user's pseudonymous ID. This works even for publishers that are not affiliated with the original publisher.

All of this data sharing is governed by a set of "Model Terms"[5]. These terms are transitive: if a permitted entity ("Contracting Party") receives SWAN data then it is required to establish that any other entity to which it passes the data is also a Contracting Party (Section 5.1). If all Contracting Parties behave perfectly, then in principle user data should not be passed to other than Contracting Parties.

3.1 SWAN Data Gathering

The authors of SWAN envision that it would be built on an as-yet-unspecified new browser feature that allows for cross-domain storage. However, the current specification is designed to operate within existing browser storage mechanisms.

Because SWAN is intended to track the user across sites, it is in direct conflict with many of the anti-tracking mechanisms built into most browsers. The ideal that browsers are working toward is a model where

[5] `https://github.com/SWAN-community/swan/blob/main/model-terms.md`

each top-level site has completely separate storage, including for all of the content that it embeds from other sites; this kind of storage partioning would obviously prevent SWAN from functioning as intended.

In order to work with these browsers, SWAN uses a technique called redirect tracking[6]. The basic idea is that when a user visits site A, A briefly *redirects* the user through site B. B records the visit in a cookie stored on the user's browser and then redirects the user back to A. Because the user's browser treats this as a visit to B this is a same site (first-party) cookie and therefore anti-tracking mechanisms which block or partition cross-site (third-party) cookies do not prevent it. Some browsers, such as Firefox, include defenses against redirect tracking, but they are not perfectly effective.

The process[7] for distributing tracking data to SWAN storage is lengthy. The complete process for a single storage node that includes consent management has tens of discrete steps; additional SWAN storage nodes each add multiple steps. This process covers only loading the HTML for the site. Additional steps are needed to load subresources and advertisements. The browser is redirected many times in this process, bouncing between a SWAN home node, SWAN storage nodes, SWAN consent management, and the site that the user visited. For simplicity, we will refer to these collectively as SWAN nodes.

Though there are many steps involved, the building blocks are relatively simple. The SWAN operator nodes cooperate as follows:

- Redirects are used to pass the browser between different nodes in sequence.
- Redirects include URI parameters that hold information about the other nodes and about the user.
- Information that is included in redirects is encrypted, which ensures that nodes can control who can access the information.
- SWAN nodes access persistent stores to add information to the encrypted blob.
- SWAN nodes access those parts of the encrypted blob that is encrypted for them and store any information that is of interest.

Conceptually, there are two main scenarios: an initial visit from a new browser that has not stored any SWAN cookies and subsequent visits (we omit the consent flow).

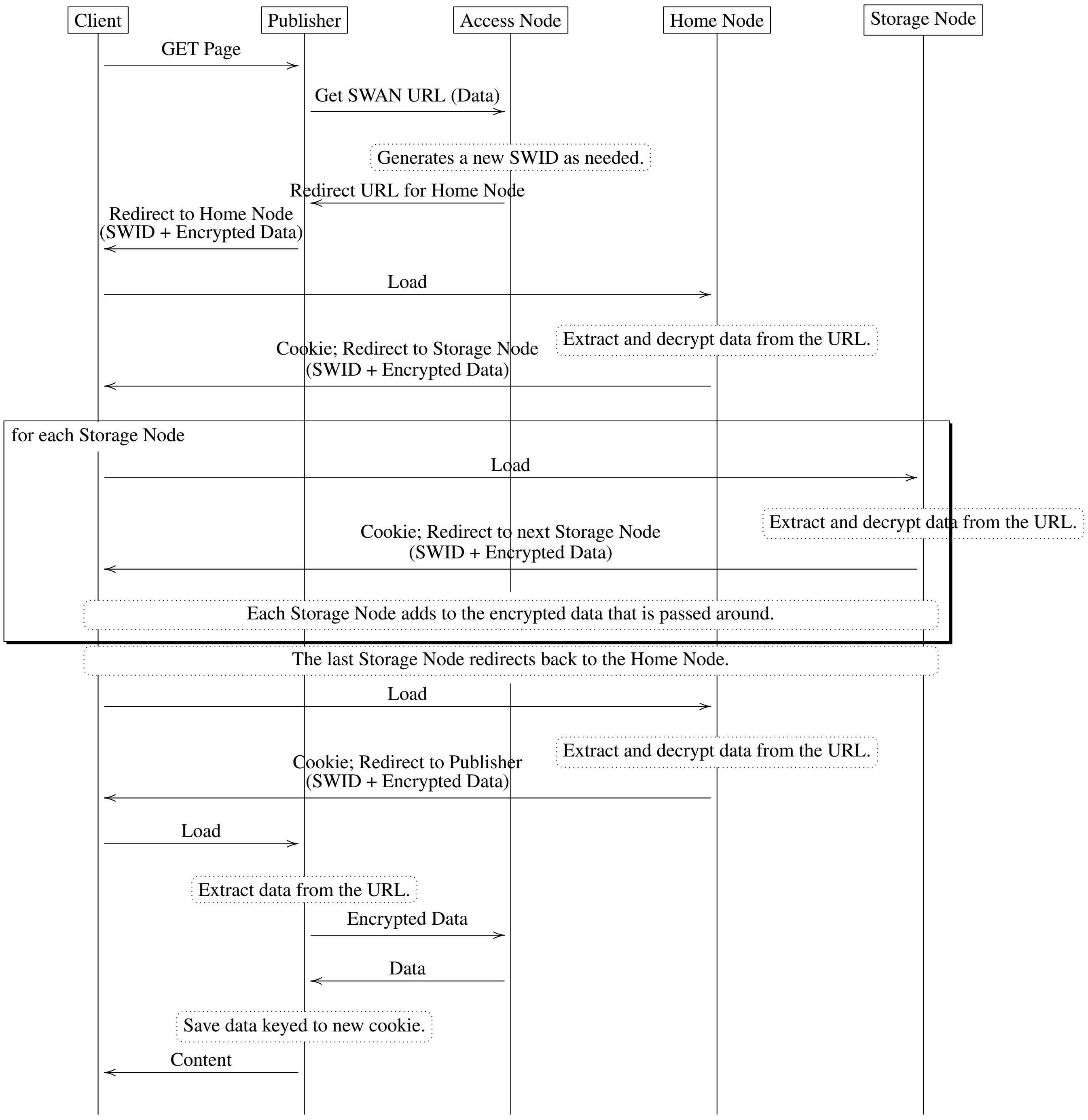
3.2 Initial Visit

In the initial visit, the publisher consults a SWAN access node which is run by the SWAN operator. The access node verifies that the publisher is part of the SWAN system and then provides a URL for a SWAN "home node" to redirect the browser to. Home nodes are designed to be for the most part consistently chosen for a given Web user (e.g., based on the browser's IP address) and so ideally a browser will see the same home node on multiple visits no matter which publisher it visits.

The home node stores a binding to the SWAN identifier (SWID) in a cookie in the user's browser and then redirects the browser to another storage node. Each storage node stores the SWID in its own cookie and then redirects to another storage node and so on. Eventually, the browser is redirected back to the home node, which decrypts the original URL and redirects the data back to the original publisher page.

[6]`https://blog.mozilla.org/security/2020/08/04/firefox-79-includes-protections-against-redirect-tracking/`

[7]`https://github.com/SWAN-community/swan/blob/main/data-flows.md`



Client
Publisher
Access Node
Home Node
Storage Node
GET Page
Get SWAN URL (Data)
Generates a new SWID as needed.
Redirect URL for Home Node
Redirect to Home Node (SWID + Encrypted Data)
Load
Extract and decrypt data from the URL.
Cookie; Redirect to Storage Node (SWID + Encrypted Data)
for each Storage Node
Load
Extract and decrypt data from the URL.
Cookie; Redirect to next Storage Node (SWID + Encrypted Data)
Each Storage Node adds to the encrypted data that is passed around.
The last Storage Node redirects back to the Home Node.
Load
Extract and decrypt data from the URL.
Cookie; Redirect to Publisher (SWID + Encrypted Data)
Load
Extract data from the URL.
Encrypted Data
Data
Save data keyed to new cookie.
Content

Figure 1: SWAN Overview

The SWAN demo page[8] demonstrates this process and its user experience. As expected, the first load is quite slow. Once the pseudonymous identifier (or tracking key) is established, the inefficiencies are much better hidden.

The net impact of this process is:

- Each storage node has an independent cookie which binds to the SWID.
- The publisher gets the SWID (from the access node) and can store it with its own cookie, so it doesn't need to consult SWAN frequently.

A given SWAN operator might have a large number of storage nodes and only redirect a given browser through a subset of the storage nodes. In addition, a publisher might work with more than one SWAN operator. The result is a given storage node might have bindings for only some users, as discussed below.

3.3 Subsequent Visits

The current SWAN documents do not provide a complete description of the flow for subsequent visits, but based on the available documents it appears that the flow is conceptually similar to the initial visit. In the simplest case, shown in Figure 2 below, the home node already has a cookie for the browser (the reason for consistently choosing the same home node is to maximize this probability). In this case, it can just look up the SWID from the cookie and redirect back to the publisher.

If the home node does not already have a cookie for the browser, e.g., because a different home node was chosen, it will then redirect the user through a series of storage nodes. The idea seems to be that as soon as the browser is redirected to a storage node which has stored a cookie, then it can shortcut the process and redirect the browser back to the home node. In either case, this process should be far faster than for the initial visit. In general, SWAN will work best if a critical mass of storage nodes have stored cookies in the browser, so it seems likely that under some conditions there will be additional redirects to new storage nodes in order to maintain the number of total stored mappings.

Note that it is possible to have a situation in which a visit will appear to be from a new browser, but in fact only a disjoint set of storage nodes were queried. In this case, SWAN might attempt to store new user records on nodes which already know about the user, in which case the conflict will need to be removed using SWAN's built-in conflict resolution mechanisms.

In addition to the automatically assigned identifier, SWAN can also gather the user's contact informtion (e.g., e-mail address) and use this to link up multiple visits from different devices. These visits could otherwise be disjoint.

3.4 SWAN Bidding

In either case, once the home node has redirected back to the publisher, the publisher knows the user's SWID. It can then provide it to advertisers as part of the bid request, where it can be used in a similar fashion to a cookie to build a user profile and to target advertisements. Publishers are required to ensure that they provide the SWID only to advertisers who have agreed to the Model Terms.

[8] `https://new-pork-limes.uk/`

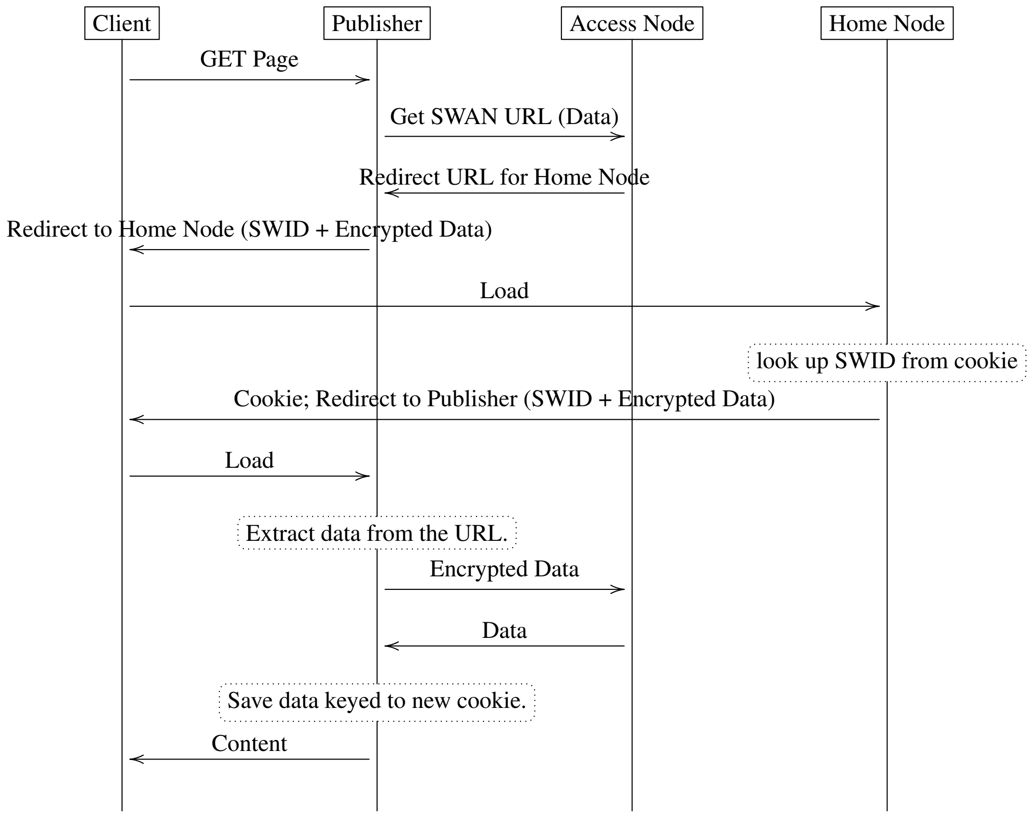

Client
Publisher
Access Node
Home Node
GET Page
Get SWAN URL (Data)
Redirect URL for Home Node
Redirect to Home Node (SWID + Encrypted Data)
Load
look up SWID from cookie
Cookie; Redirect to Publisher (SWID + Encrypted Data)
Load
Extract data from the URL.
Encrypted Data
Data
Save data keyed to new cookie.
Content

Figure 2: SWAN Subsequent Visits

3.5 Privacy Implications of Identifier Management

SWAN allocates a single pseudonymous identifier to a user and distributes that identifier between SWAN storage nodes, with each node providing its own binding between a first-party cookie and that pseudonymous identifier. A consequence of this design is that whenever that mapping is updated or queried the store has an opportunity to learn about the user's browsing history. The "home node" always learns this information because it has to see both the user's identity and the final URL to redirect to. The storage nodes do not need this information because they redirect back to the home node and so data can be carried in an encrypted blob, but whether it is present depends on the behavior of the home node.

This design *allows* the SWAN nodes to be stateless (because all necessary data is stored in a cookie) but does not *require* it. In particular, nothing technically stops any node that sees the user's browsing history and SWID from storing that information locally. This is invisible to the browser and therefore cannot be enforced except by policy. Specifically, browsers cannot detect or prevent policy violations. Even though data is passed via the browser, all of the data that storage nodes pass to each other is encrypted, preventing the browser from determining what information is being passed to the storage nodes.

3.6 Return Receipts

As described above, SWAN requires any entity which receives the user's SWAN data to only pass that data on to other entities which have agreed to the Model Terms. The intended result is to transitively enforce that data is only released to entities which have agreed to those terms. In order to help ensure that entities actually conform to those terms, receivers of information provide a digitally signed receipt[9] for that information in order to allow for audit. These receipts flow in the opposite direction from the information, so that users eventually receive receipts from every receiver of their data. (If entity A passes data to entity B, and B does not provide a receipt, then A is obligated to report this.)

It's somewhat unclear how useful this information will be in practice: given the scale of existing user tracking, any given piece of user data is likely to be shared with many entities, so the user will receive a large number of receipts. Even if it is discovered that a piece of data is misused or leaked, it seems quite difficult to track down the precise entity responsible. Moreover, in many cases that misuse will not be apparent to the user.

4 How Unified ID 2.0 Works

Unified ID 2.0 is a far simpler proposal than SWAN. Rather than creating a new identifier, Unified ID 2.0 has participating sites collect primary identifiers, specifically email addresses. These email addresses are then transformed into a unique long-term identifier, the UID2, via salting and hashing (more on this below). As shown in the diagram below, this tracking is mediated by the UID2 service.

When a user first visits a participating site, that site prompts the user to provide their email address. This information can be stored in first-party a cookie in the user's browser in the usual fashion, as shown below. From then on, whenever the user visits the site, the site has their email address.

When the site wants to render an ad, it uses the UID2 service to map that email address to a one-time encrypted token which contains the UID2. This token is then provided to potential advertisers as part of the bid request, as shown below. Advertisers who are participating in the UID 2.0 system — and specifically have agreed to the privacy terms — can get a decryption key from the UID 2.0 service, which will permit

[9]The OWID document says that this signature is RSA-SSA-PKCS#1v1.5, but then it says that each signature is 64 bytes. That would mean using 512-bit RSA, which is clearly insecure.

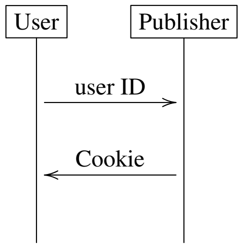


Figure 3: UID2 Login

them to decrypt the UID2. That identifier can then be used for tracking and ad targeting in the usual fashion. Because a new UID2 is provided for each visit, the tokens themselves cannot be used for tracking.

4.1 Consent Management

UID 2.0 integrates a consent management solution for users who wish to request that data platforms stop assembling profiles. The centralized system provides a web page[10] into which users are able to enter their email address. Following the opt-out process results in the UID 2.0 service storing a record for the (hashed) email address and returning no raw UID2 when one is requested.

Data platforms that use the UID 2.0 service can periodically check the consent status of each raw UID2 for which they have data. They are expected to remove stores in a timely fashion. Sites then engage any "opt-out" logic for users that have opted out of tracking.

4.2 Salted Primary Identifiers

While UID 2.0 is based on email addresses, it is designed to be pseudonymous. In order to ensure this, the UID 2.0 service receives email addresses (or perhaps hashed email addresses), which it then salts and hashes to form a "raw UID2". The UID 2.0 service uses a secret high entropy salt, chosen out of one of a set of "salt buckets", thus preventing brute force attack on the UID2 to recover the email address. The result is that the raw UID2 is unique for a given email address and "salt bucket". No rationale is given for this partitioning, though it might be to improve scalability.

The specification also states that the salt for each bucket changes every 12 months, but provides no rationale for doing so. We might theorize that this is to limit the harm caused by losing the associated keys. While it might seem that this rotation could prevent the accumulation of long-lived profiles of users, linking users across any rotation of keys is trivial. As data platforms routinely link profiles that use different email addresses, this linkage might even occur accidentally.

4.3 Scrambled, Salted Primary Identifiers

As described above, the UID 2.0 provider supplies a fresh UID2 token to participating sites to use in the bidding process. This token is created by encrypting the raw UID2 along with a random nonce. The token is intended to be useless to anyone without the encryption key, which is supposed to be available only to authorized participants.

[10] `https://transparentadvertising.org/`

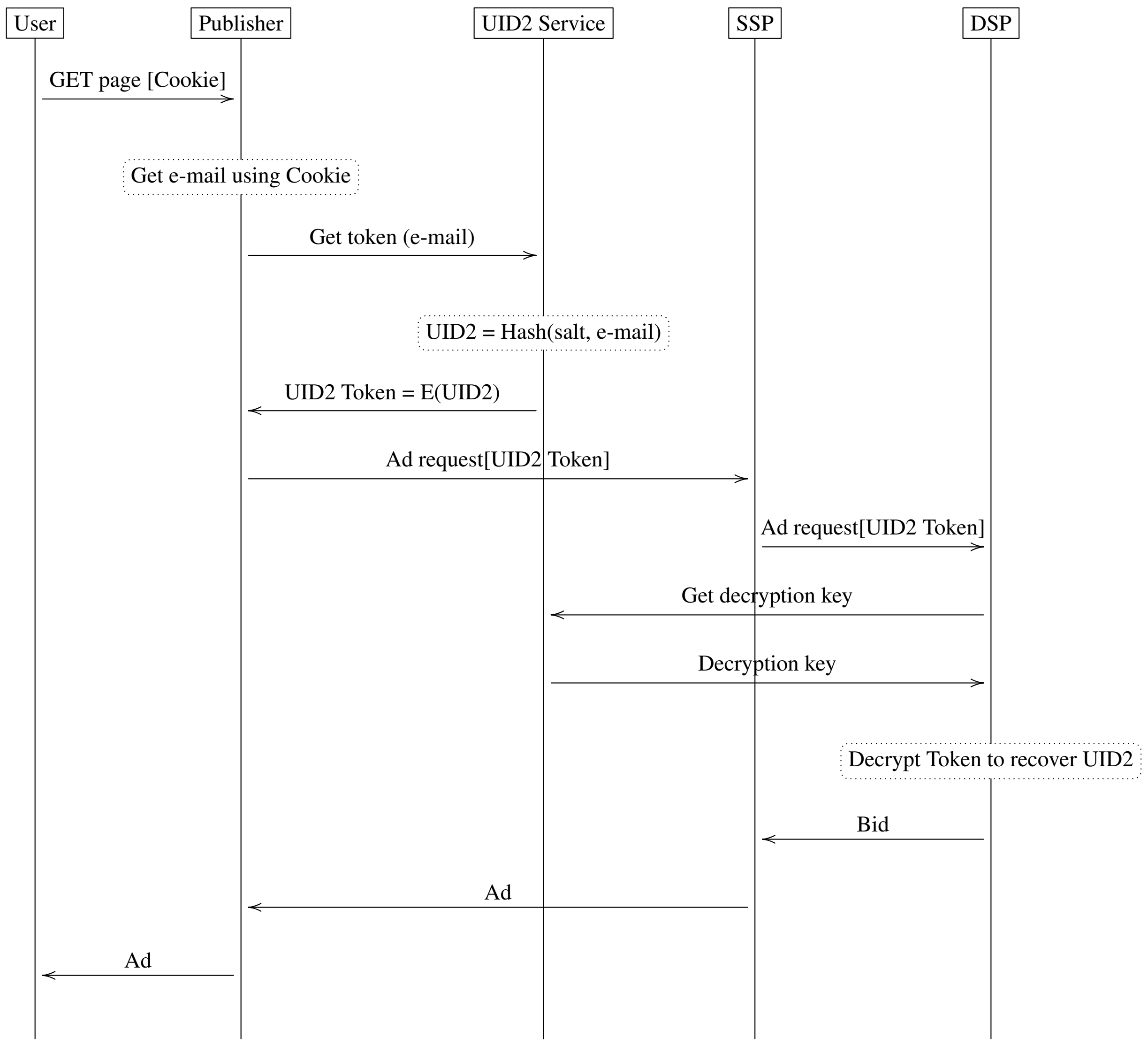


Figure 4: UID2 Ad Display

These keys need to be provided to every Unified ID 2.0 data consumer, which has two important security implications. First, a single malicious data consumer can share the decryption keys with a non-participant, enabling them to decrypt UID 2.0 tokens that they receive. Because sites are not expected to restrict the dissemination of these tokens, this presents a major privacy risk which seems hard to ameliorate, as there is no straightforward mechanism for tracking down the leaking consumer.

Second, because the only authentication for the tokens is the keys, which are widely distributed, any consumer can create valid appearing tokens which map to any UID2 of their choice. For instance, they could use this to create false browsing history information for a given user. It might be possible to repair this by having the tokens be digitally signed by the UID2 service, thus preventing an attacker from minting their own tokens.

5 Governance

Both SWAN and UID2 depend heavily on policy-based mechanisms for controlling information. While both systems incorporate technical mechanisms for managing access (encryption in the case of UID2 and signed receipts in the case of SWAN), these mechanisms are primarily intended to limit receipt of information to entities which are part of the system and expected to conform to specific policies. The policy structures of SWAN and UID 2.0 are quite different, as detailed below.

5.1 SWAN Policies

SWAN participants are required to conform to the Model Terms[11] for SWAN, which govern how they are to handle the data. As noted above, these terms are transitive: no entity may send user data governed under the Model Terms to an entity which has not accepted them. However, conversely, data may be shared with any entity which has accepted the Model Terms, which is only safe if every entity behaves correctly all the time.

As a practical matter, the SWAN Model Terms provide somewhat unclear protections for user: They are designed to be incorporated by reference into other agreements between participants in the SWAN Network. Companies collecting user data promise that they have implemented technical and organizational measures to safeguard data (§ 5.12). Participants only have to verify that the entity they are sharing data with is a member of the SWAN Network (§ 5.2), but they do not have to verify that the entity actually has the expertise required to protect user data. Furthermore, the agreement does not require SWAN participants to delete user data when they leave the network (§ 8.4.i).

From a users perspective, this type of data sharing between an unlimited number of unvetted companies should be concerning. Further, its unclear how users would know a violation has occurred, who to complain to, and what recourse they would have.

5.2 Unified ID 2.0 Policies

Unified ID 2.0 requires participants to comply with a “code of conduct”, but that CoC does not seem to be available, so we are unable to evaluate it.

[11] `https://github.com/SWAN-community/swan/blob/main/model-terms.md`

5.3 Technical versus Policy Controls

The Internet community has generally shown a preference for technical mechanisms to protect information, where it makes sense to do so. There are several reasons for this. First, technical controls are more likely to be user-verifiable in the sense that the user does not need to trust that other entities follow the appropriate policies. Second, they are generally easier to analyze to ensure that they are providing the expected guarantees[12]). Thus it is concerning that SWAN and UID 2.0 depend so heavily on policy controls.

Moreover, in both cases, there are a large number of entities involved and no technical controls ensure that they in fact conform to those policies. Misbehavior by any entity threatens the entire system and neither SWAN nor Unified ID 2.0 seem to have a clear plan about how such misbehavior would be detected and handled.

Under both proposals, any data relating to a user is likely to be available to a large number of players, so attributing leaks will be difficult. Note that this is especially true in the case of UID 2.0, where the UID2 token is widely distributed and access control is limited by distribution of the keying material.

Second, even if it is clear that a single entity has been misbehaving, it is not clear how enforcement will really work. This is especially problematic in SWAN, where there is no central entity which controls access and each individual participant seems to be expected to blocklist bad actors.

6 Conclusion

The basic design concept of both SWAN and Unified ID 2.0 is to provide a service that assigns a pseudonymous identifier to each user. This identifier can then be used for tracking and ad targeting. These systems differ from the current cookie-based tracking system in a number of major respects:

- They require an explicit user consent step where the user agrees to tracking.
- Recipients/consumers of the identifier are required to conform to policies regarding the use of that identifier.
- They explicitly make use of technical mechanisms designed to bypass browser anti-tracking protection (redirect tracking for SWAN, identifier-based tracking for Unified ID 2.0).

From a purely technical standpoint, these proposals are a regression in privacy in that they allow tracking of users who are presently protected against tracking. Moreover, the techniques used here — especially redirect tracking but also identifier-based tracking — are already ones which most major browsers attempt to prevent[13].

These proposals assert two major reasons why this form of tracking is acceptable: that users must first consent to tracking and that the use of this data is subject to policy controls. Neither of these is persuasive. Existing experience with GDPR-type cookie banners clearly indicates that users do not understand what is being asked of them and will frequently click through just to make the dialog go away. The proposed policies themselves are either weak (as in SWAN) or underspecified (as in Unified ID 2.0). Moreover, there is no way for users to actually determine if policies are being followed and the enforcement story is unclear.

Finally, even if the relevant policies were strong and clearly enforceable, both of these systems would ultimately result in a large number of entities — both the system operators and various ad-tech players —

[12]`https://blog.mozilla.org/en/mozilla/more-on-covid-surveillance-mobile-phone-location/`

[13]`https://blog.mozilla.org/security/2020/08/04/firefox-79-includes-protections-against-redirect-tracking/`

possessing user browsing history, even if their actual use of that history is intended to be restricted. This is clearly a risk for user privacy and something that any privacy-preserving advertising system should avoid.

Version History

2021-08-04	Initial version
2021-08-13	Corrected description of the interaction with the Access Node.