PARAKEET test setup

PARAKEET – Testing Plan

- Two Stage Approach
  - Stage1 [Q3 2021]: Functional Testing to validate PARAKEET
  - Stage2 [Q4 2021]: Set level of anonymization needed to balance privacy vs. monetization

- **Goals for stage 1**
  - Validate PARAKEET APIs can be used to replace existing 3p cookie-based functionality
  - Confirm if PARAKEET model that retains auction in SSP/DSP is relatively easy to enable

- **Goals for stage 2**
  - Evaluate Privacy and Monetization criteria for PARAKEET
  - Influence industry standards by meeting privacy, monetization, and adoption needs

- Ask: Ad-Tech partners (SSP/DSP/Publisher/Advertisers) to run functional test during in Q3 2021.
  - Sign up via Edge Origin Trials Page

PARAKEET Functional test setup

Polyfill library for ad interests, ad request and rendering

Request:

- Participants for alpha trials

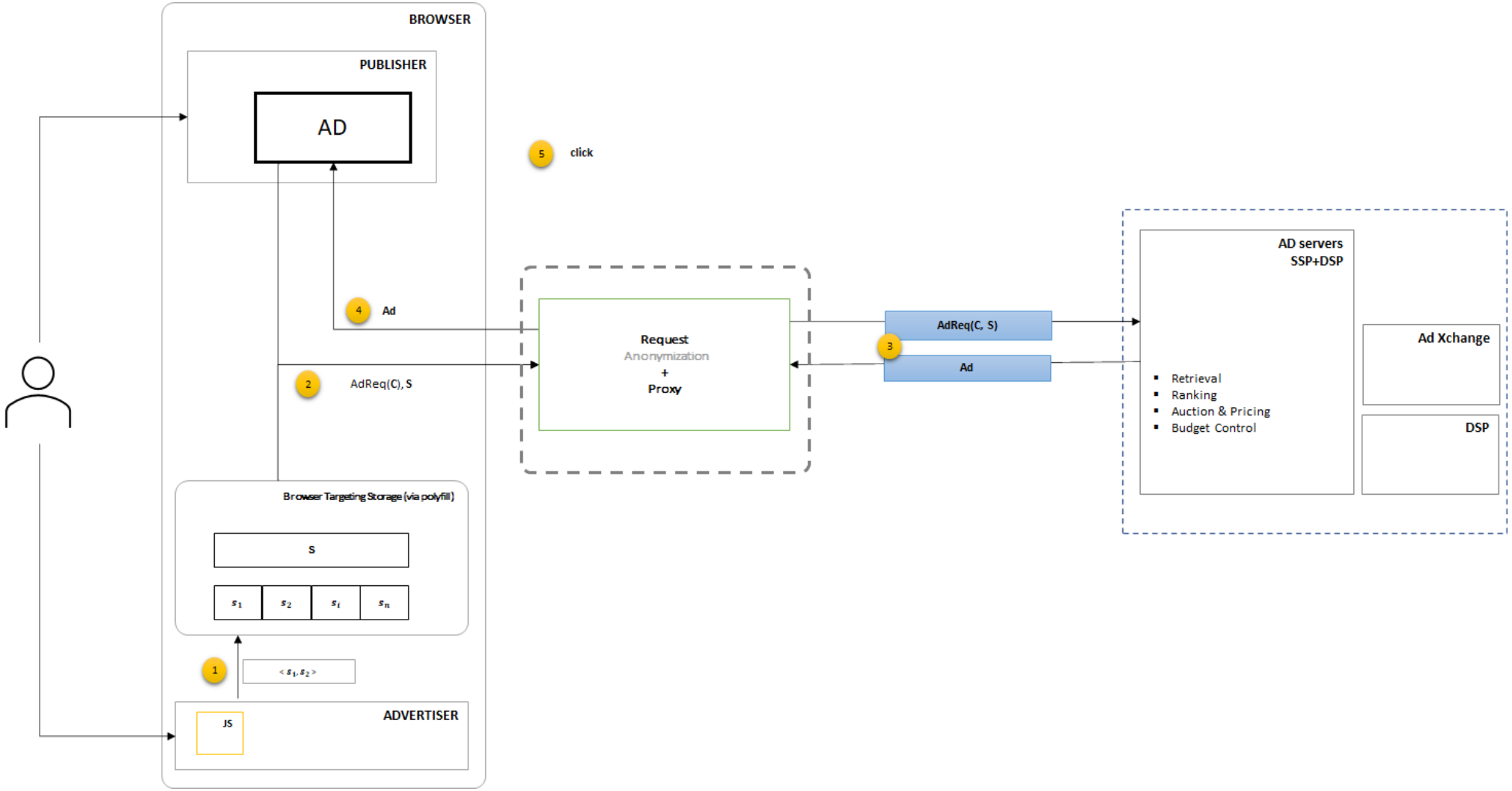
Dev setup

- Advertiser.com JS changes to leverage polyfill library to handle user features
- Ad Request changes to access user features
- Ad server changes to accept user features in request (instead of cookie), and enable ad serving

Goal:

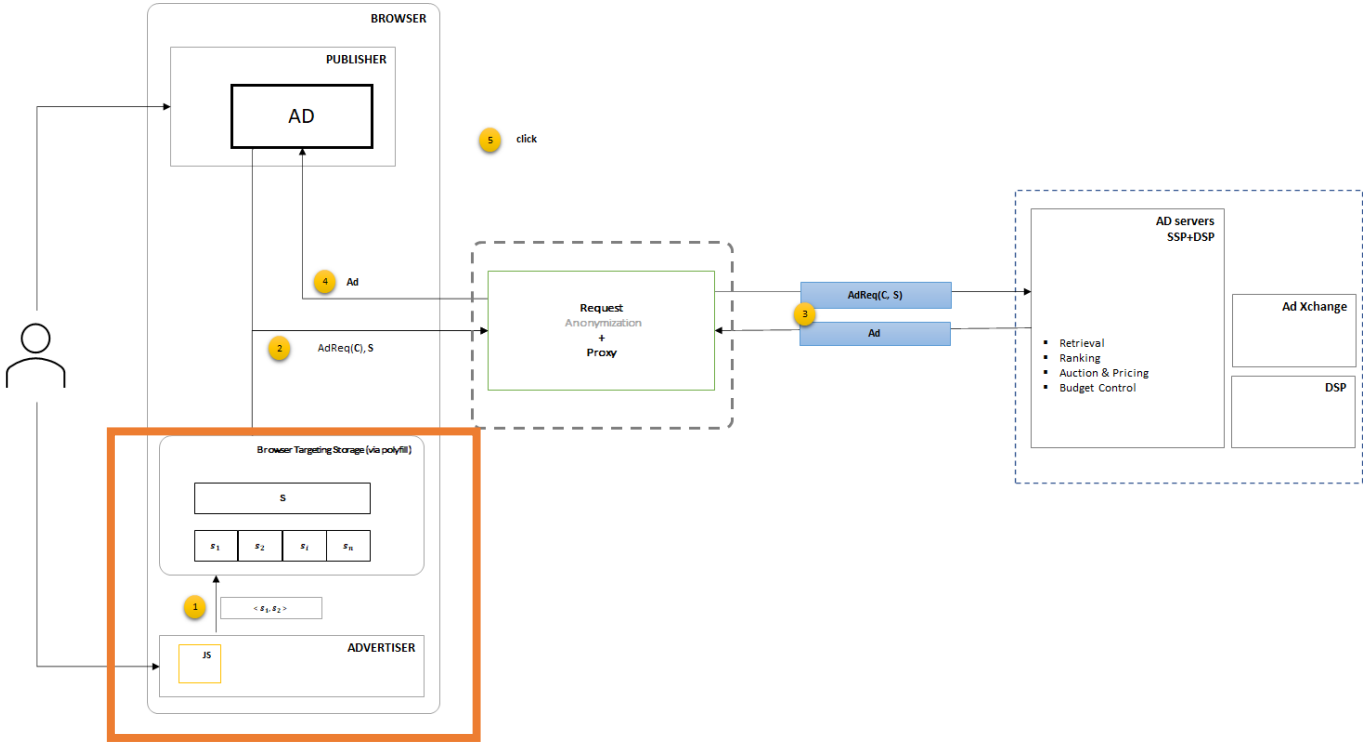
- Implement ad serving flows – user features in browser and runtime read access
- Provide robust ad parameter setup – placement size, location, device etc.
- Data flows and key management for user feature encryption/decryption

Parakeet v0 – First step



PARAKEET/Polyfill

Advertiser Page changes



1. Include the PARAKEET polyfill either directly on a page or with other scripts/ad-tech (DSP)

2. Use existing 1st party knowledge to select relevant interest groups or identify user features.

3. Execute new Javascript APIs to join the relevant interest groups with an appropriate duration.
   - Set *readers* to list DSP and SSP partners you want to be able to read your interest groups.
   - Set a *duration* that is appropriate. Interests will automatically expire and be excluded after this.

As an alternative, DSP can enable PARAKEET polyfill and ad interest flow on behalf of the advertiser. DSP(s) needs to work with the advertiser to setup correct readers.

Include JS on a webpage

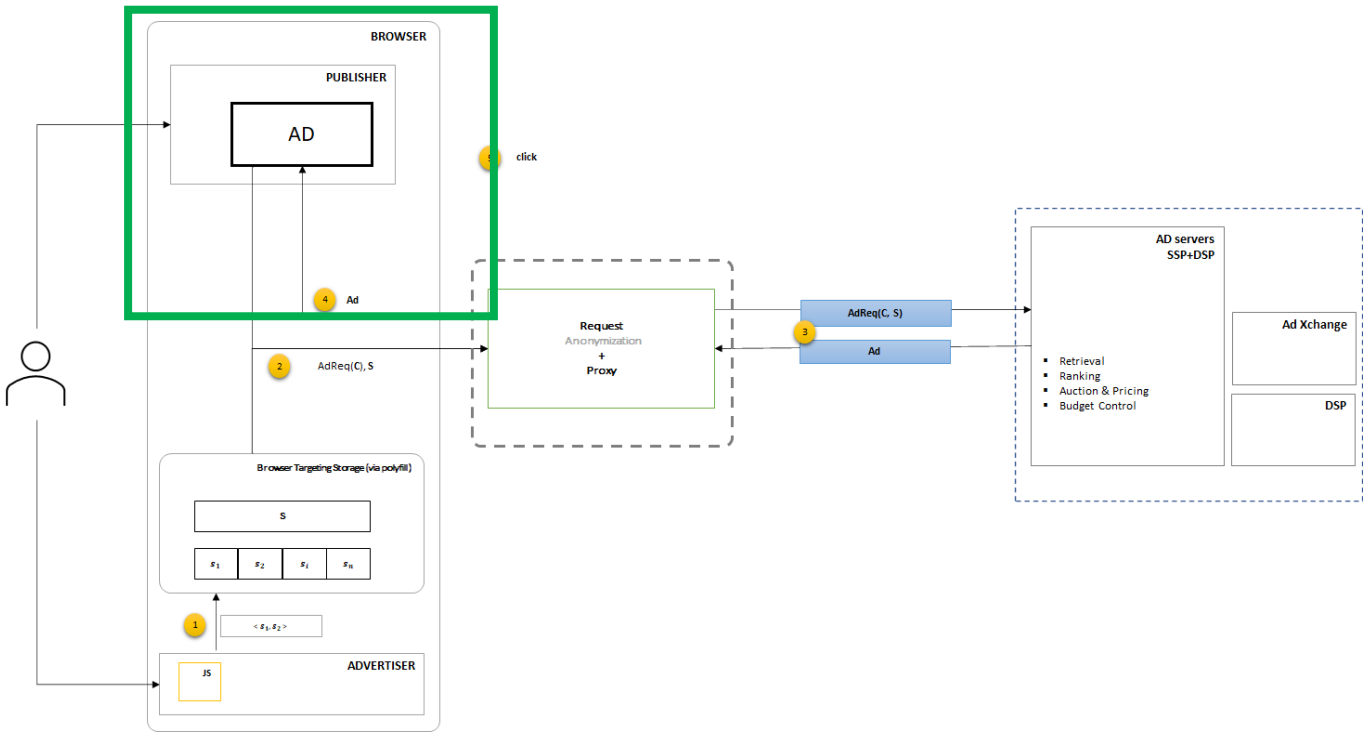
```
<script src='https://somecontentsite.example/libs/parakeet.min.js' type='text/script'/> <script>Parakeet.SetStorageOrigin(new URL("https://edge.microsoft.com/parakeet/frame.html"));</script> <!-- You're now ready to use PARAKEET! -->
```

Add JS through DSP

```
let scriptTag = document.createElement('script'); scriptTag.src = 'https://somecontentsite.example/libs/parakeet.min.js';
document.body.appendChild(scriptTag); // You're now ready to use Parakeet!
```

PARAKEET/Polyfill

Publisher Page changes



1. Include the PARAKEET polyfill either directly on a page or with other scripts/ad-tech (SSP)

2. Execute new Javascript API to request an ad with any contextual information or interests available. See: Create an AdRequest and serve an ad
   I. **On success,** create an iFrame, set the src to the provided URL and insert into the Document to display the ad.
   II. **On failure**, be sure to use sendBeacon or other mechanisms to log and track failures.

As an alternative, programmatic SSP can enable PARAKEET ad request on behalf of publisher. SSP needs to create and support service at endpoint specified in proxiedAnonymizingOrigin

Include JS on a webpage

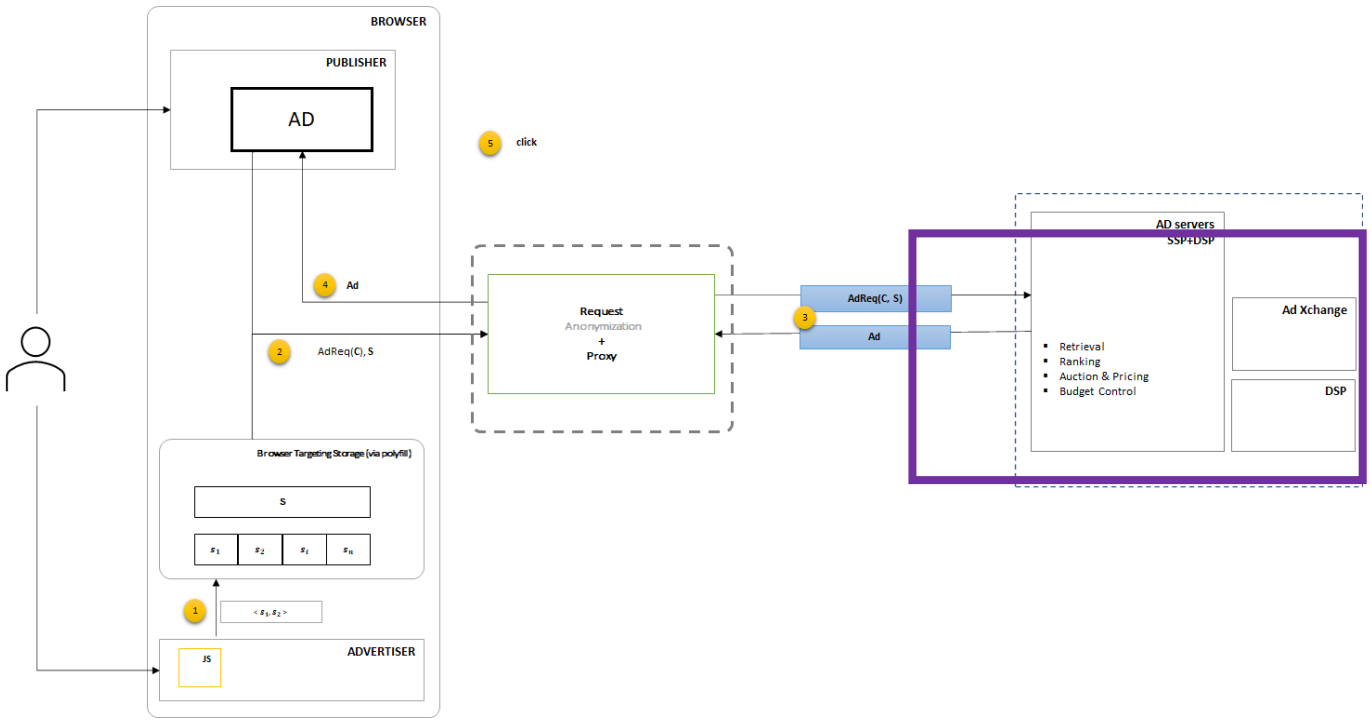
```
<script src='https://somecontentsite.example/libs/parakeet.min.js' type='text/script'/> <script>Parakeet.SetStorageOrigin(new URL("https://edge.microsoft.com/parakeet/frame.html"));</script> <!-- You're now ready to use PARAKEET! -->
```

Add JS through DSP

```
let scriptTag = document.createElement('script'); scriptTag.src = 'https://somecontentsite.example/libs/parakeet.min.js';
document.body.appendChild(scriptTag); // You're now ready to use Parakeet!
```

PARAKEET/Polyfill

DSP Server changes



1. Read user interest groups ***s*** (based on namespace) and publisher context ***c***.
2. Retrieve the best **ad** for *context c and s*.
3. Respond with **optimal bid and creative** to SSP/programmatic exchange.
4. To support **encryption for s**: Publish your public keys at well known PARAKEET location. **See .well-known/parakeet.jwk**
5. COMING SOON: Decrypt the interest groups that are encrypted with your public key for step 1-3.

PARAKEET/Polyfill

Next Steps

Please express interest & signing up in GitHub: Polyfill Sign-up

Encryption steps will be coming soon; please prepare to share your domain well-known URL for validation of data encryption

Additional development for polyfill capabilities – participate in development

- Support **s in** ad request with standard RTB format **($user.data** or **$user.ext)**
- Ad Response validation and register flows
- Capabilities for Trust Token and additional fraud detection
- Publisher JS for direct-sold decision

Ask questions and provide feedback by opening Polyfill Github Issues