👋 Hello, here is a guide I made to help simplify the setup of your App's security **(original video).**

Table of Contents *(Estimated Time: 30 Minutes)*

1) Enable Anonymous Authentication

- In the Firebase Console ► Authentication ► Sign-in method, enable **Anonymous**.
- This gives every browser session a Firebase ID token without asking the user to sign in.

2) Register & Activate App Check (reCAPTCHA v3)

- In the Firebase Console ► App Check, register your Web app.
- Choose **reCAPTCHA v3**, enter your site key, and finish setup.
- Your front end will now obtain App Check tokens silently.

3) Allow Unauthenticated Invocations in GCP

- In the Google Cloud Console ► Cloud Functions ► select your `generate_timestamps` function ► Security, choose **Allow unauthenticated invocations**.
- Security is enforced inside the function by Auth + App Check, avoiding CORS preflight failures.

4) Write & Deploy Your Callable Function

- In `functions/requirements.txt`, list the Firebase Functions Python SDK and any other packages.
- In `functions/main.py`, implement your callable function with Auth and App Check checks, and the external API call.
- From your project root, install dependencies and deploy with `firebase deploy --only functions:generate_timestamps`.
- **See relevant code here** (GitHub Repo).

5) Initialize Firebase + App Check + Auth in Your App

- In your front-end code (e.g. `src/firebase.js`), initialize the Firebase app, then set up App Check with ReCAPTCHA v3, sign in anonymously, and get the Functions instance.

- **See relevant code here** (GitHub Repo).

6) Call the Function from React

- In the component where you need the timestamps, import `httpsCallable` from Firebase Functions, call your `generate_timestamps` function, and handle any errors.

- **See relevant code here** (GitHub Repo).

7) (Optional) Client-Side Rate-Limit UX

- Store each call's timestamp in `localStorage` and on each render count calls in the past hour.

- If the user has reached your chosen limit (e.g. 10 calls/hour), replace the "Generate" button with a link to upgrade or retry.

- **See relevant code here** (GitHub Repo)..

8) Why This Is Secure & CORS-Free

- App Check ensures only your genuine front-end can call the function.

- Anonymous Auth provides a UID that your function verifies.

- Callable Functions automatically handle CORS and inject both Auth and App Check tokens.

- Together, these measures block bots, unauthorized clients, and raw HTTP requests—only valid users from your app can succeed.

Follow more cool stuff

https://x.com/corbin_braun