Third Party Secure Key Communication

SAKUA2Random, produces around some 150 Million 108 character long random strings. This is just a test program, not perfect in anyway.

There are three parties involved.

- SAKUA2Random String generated site (The third party)
- The End-User
- The Website or Service Provider the End-User wishes to work with.

The End-User installs an app from the random strings generator site, the app downloads 150Million+ 108 character long random strings file per device.

The SAKUA2Random Strings Site (keyserver) will create one 150Million+ 108 character long random strings file, which is then shuffled around keeping the strings intact, but the order changed to generate around 8 Billion+ combinations, which are used by devices belonging to other users. Each user registers the (n)sites and usernames with the random strings file.

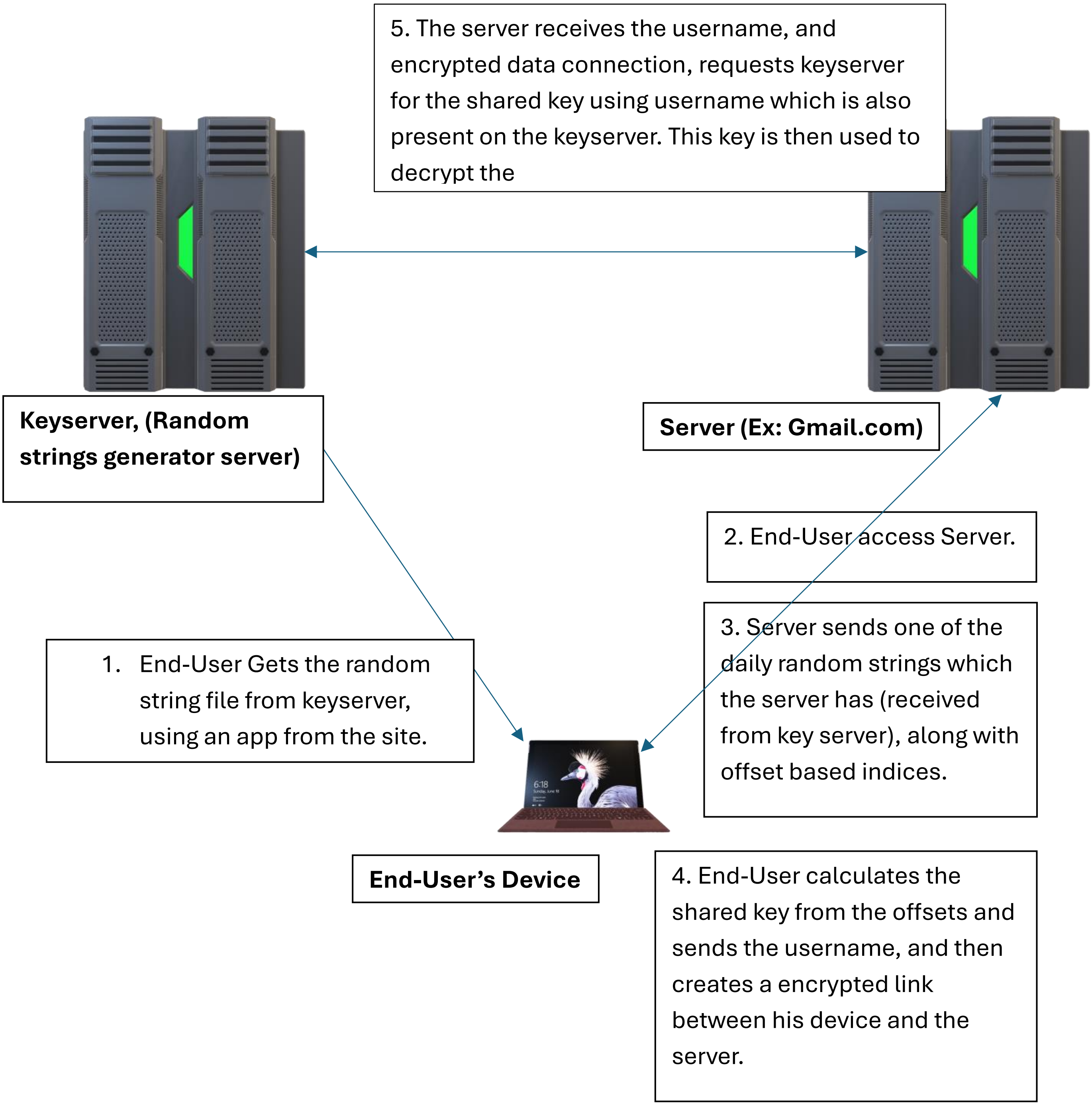
The End-User accesses the site, Let's say Gmail.com (server). Gmail.com uses one of the daily strings (random search string) and sends the string along with a list of indices. These indices represent the offset from the end location of the random search string. The values together in these indices represent the shared key. Which could optionally be mixed in with the password to create an even more secure shared key.

The End-User's app then uses these indices which it receives over TCP to calculate the shared key, from its version of the random strings file, the indices are at an offset from the end of the search string which is present in all 8 billion+ versions of the file, including the one with Gmail.

The End-User then sends his username and uses the encrypted key to create a secure connection.

At the server-end which is Gmail in this case, it contacts SAKUA2Random Strings Site (keyserver) sends the username and retrieves the shared key. Which is then used to decrypt the data from the End-User and vice versa.

The random strings file can be regenerated every week or so to improve the security of the End-User / Server.



Keyserver, (Random strings generator server)

Server (Ex: Gmail.com)

End-User's Device

1. End-User Gets the random string file from keyserver, using an app from the site.

2. End-User access Server.

3. Server sends one of the daily random strings which the server has (received from key server), along with offset based indices.

4. End-User calculates the shared key from the offsets and sends the username, and then creates a encrypted link between his device and the server.

5. The server receives the username, and encrypted data connection, requests keyserver for the shared key using username which is also present on the keyserver. This key is then used to decrypt the