Q3: Lists of IDs

A low powered microcontroller with restricted processing and only a few megabytes of storage is used to implement a electronic door lock. When a user swipes their access card we receive a 16 byte card ID in the form of a UUID, and we will allow the user to enter if their ID is on a list of known card IDs. This list of card IDs is generated in the cloud, but the lock must be able to operate even in cases where connectivity to the cloud is temporarily unavailable. It is not neccessary to retrieve any further data, we just want to check if the ID is on the list.

How would you suggest we store these on the microcontroller so that we can quickly check if the user is permitted? The list of valid access cards is calculated in the cloud, and changes infrequently compared to the frequency of checks. Explain briefly the pros and cons of various approaches, and suggest what you would consider the best option. It is not necessary to write any code, but if that is the best way to express your idea, go ahead!

Ultimately, my approach would be to store a cache of data with appropriate TTL metadata for each UUID record.
So, we would need to be mindful of the cache priority implemented when performing a ‘cloud-fetch’. I would assume any security system would record when last the UUID-respective access-card was used, e.g ‘lastLoginAt’ etc. I would assume it checks in such an order too. Since it is far more likely a recent user is using their card again– well at least by the day but I digress. Our microcontroller would cache all valid UUIDs in order of lastUsed; the cache amount of course being set for the appropriate strong limit, - a limit that is suitably storable without high-risk to performance.
Primary Cases for cache refresh

- Cache will refresh with every scan.
- every TimeToRefresh ; TTR < TTL*
- Or if it is listening to changes realistically.

On TTL,

- Cache’s TTL would be checked really on an offline scan. Then MC will perform a check if the cache is valid and then evaluate for valid access. If not deny on assumption due to offline. (‘orange I guess’).

Now given it is online, it should rely on the fetched data from the cloud to evaluate validity.

- (sub idea: I would hope there is a 'listening for changes’ level of connectivity. If so, to reduce computation on MC we can count for TTR cases on the Server or so. Otherwise, I'm sure the mc can handle a timer– even a washing machine can)

Else given it is not online, I reiterate, given the scan we check the TTL validate to current_date difference from fetched_At. If not within we deny access. If not we perform the same check with the substitute cache data.