DinoQR: Time-Secured, Realtime QR Codes

Ben Gershuny, Isaac Hilton-VanOsdall, Oscar Newman
Brown University
{benjamin_gershuny,isaac_hilton-vanosdall,oscar_newman}@brown.edu

Abstract

We sought to apply the existing an near-universally available technology of QR codes to more secure use-cases requiring the verification of location, identify, or time. We developed a protocol for real-time animated QR code generation, scanning, and verification to provide these qualities. With the hope of expanding the technology past strict QR codes, we implemented a custom QR encoding, decoding, and scanning pipeline in Python and OpenCV2. This was functional but slow due to the limitations of Python. We also built a live server, client, and scanner demo using existing QR Code libraries as a proof of concept of the protocol. Current results are promising and we hope to combine both efforts by rewriting our QR pipeline in a low-level language.

1. Introduction

Countless daily actions require verifying time, identity, or location. And many organizations, businesses, and services rely on that ability. Unfortunately, solving that problem typically requires specialized hardware (like ID cards, NFC, RFID, etc.) and software. While this is fine in certain settings, there are many that would benefit from a universally available approach. For example, venues could eliminate price-gouging secondhand markets for tickets by tying ticketing to identity and preventing transfers outside of their system. Likewise, a technology like this could be used to verify that home healthcare providers for the elderly visit their pateients when they say they do, and prevent fraud (this type of verification is now a legal requirement healthcare providers are struggling to meet). It could even be used to securely exchange secrets with trusted but potentially numerous third parties.

QR codes are a promising solution to this problem. They are universally avaialble. They can be both generated and scanned by anyone with a camera-enabled smartphone. Using modern web technology, there may not even be a need for users to download an app to scan and generate QR codes.

Unfortunately, QR codes as they exist are insecure for these applications. They can be copied, shared, replicated, and scanned by anyone at any time. They simply act as a static identifier or a shorthand for typing in some sort of code.

1.1. Problem Statement

In this work, we seek to create a secure, universally available, and flexible QR code pipline and protocol to provide the verificaiton of location, identity, and time to the scanner. We will achieve this through a custom encoding protocol which provides security using timestamps and cryptographic one-time passwords alongside an animated QR code displayed on a client, and a scanner and server capable of verifying those animated codes in realtime.

2. Related Work

Some research has focused on encoding high-density data in colored (rather than black-and-white) QR codes [4]. This research sought to create extremely robust high capacity QR codes using large amounts of data and multi-color-channel encoding. It identified both novel types of color interference and the best methods to avoid them.

A live event ticketing startup based in Amsterdam uses a similar approach of animating QR codes to disalow ticket resale outside of their closed system (We actually discovered them while researching this idea a few monthsa go) [1]. They seem to have developed a robust, proprietary system for ticketing specifically, though it's not clear what additional levels of security and identify verification their protocol provides.

Ivan Daniluk, a developer from Spain, developed a system of transmitting large packets of data via quickly animating QR codes [2]. His work on this project, and later adaption of the system to error-resistent Fountail Codes provides a great model for making QR codes not only spatially but temporally robust [3].

3. Methods

3.1. The Protocol

The protocol we designed encodes, every 10th of a second, three pieces of data totaling 24 bytes into a QR code. First, it encodes a universal resource ID (a 16 bit UUID). This is constant, and allows the verification server to identify a particular resource, such as a ticket or task to be copmleted, relevant to the applied domain. The second part is the last four bytes of the current Unix time in milliseconds. This is the most that is needed to verify codes are generated in realtime because of the third part, which is a Temporary One Time Password.

3.2. Encoding & Decoding

3.3. Scanning

3.4. Client and Server

4. Results

5. Comparaison to Existing Techniques

Contour-based detection
Do we need references??? Probably

References

[1] GUTS tickets — honest ticketing.

[2] Ivan Daniluk. Animated QR data transfer with gomobile and gopherjs.

[3] Ivan Daniluk. Fountain codes and animated QR.

[4] Zhibo Yang, Huanle Xu, Jianyuan Deng, Chen Change Loy, and Wing Cheong Lau. Robust and fast decoding of high-capacity color QR codes for mobile applications. 27(12):6093–6108.