June 27, 2019

EMBEDDING OF PAYLOAD DATA, OVERT SECURITY FEATURES, AND PUBLIC KEY FEATURES INTO AN IMAGE TO LATER VALIDATE THE AUTHENTICITY OF THE IMAGE

HP INC

Embedding of payload data, overt security features, and public key features into an image to later validate the authenticity of the image

This disclosure relates to the field of anti-counterfeiting and counterfeit detection via physically embedded security codes.

A system is disclosed which prevents counterfeiters from harvesting security codes using conventional code-scanning technologies, forcing a counterfeiter to obtain or synthesize an exact copy of a physically rendered security code, making counterfeiting of security codes more difficult. The disclosed system also enables the creation of apps designed specifically for security personnel that verify all levels of the marking, as opposed to a consumer-based solution that may only validate payloads before performing subsequent processing on a server.

Conventionally, embedded security codes, such as barcodes or RFID tags, rely only on the uniqueness of the readable code payload to allow the detection of counterfeit objects on which the codes are embedded. The problem with this is that counterfeiters can harvest the payload of authentic security codes using conventional code scanners, and this allows counterfeiters to easily create copies of authentic security codes, indistinguishable from the authentic security codes, merely by copying the payload into their own codes. This makes the detection of counterfeiting difficult, and does not deter counterfeiting as well as the system disclosed below.

The new approach embeds two levels of additional security/anti-counterfeiting information in the security codes in a manner that cannot be read by conventional code scanners. One embodiment is a QR code image with additional security/anti-counterfeiting information besides the standard QR payload embedded in the image.

The first level of additional security/anti-counterfeiting information is a secure ID embedded in the padding of the QR code, which is reflected in the QR code image. The padding of the QR code is ignored by conventional QR code scanning applications, so the secure ID cannot be easily read by counterfeiters hoping to harvest it. The secure ID is a set of bytes whose value can be generated randomly and stored separately, or it can be generated by applying a one-way hash algorithm (h1) to the payload to be embedded in the QR code.

$$\text{secureID} = h1(\text{payload})$$

In the example in Fig. 1, a version 4 QR code with high error correction, which has 34 bytes in which to store the payload, has 24 bytes of payload and an 6-byte security ID encoded in the last 6 bytes of padding.

Fig. 1: Encoding of a 6-byte security ID into QR code data

34 bytes of total space																																	
Payload																								**Padding**									
24 bytes used for payload in this example																											**6 bytes of padding used for Security ID**						
01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	10	21	22	23	24	25	26	27	28	29	30	31	32	33	34

h	t	t	p	:	/	/	h	p	.	c	o	m	/	A	B	C	D	E	1	2	3	4	5	U	U	U	U	S	S	S	S	S	S

UUUU = Unused padding bytes (Payload could grow into these bytes)
SSSSSS = Padding bytes used for SecurityID

A special QR code scanning application can be created to read the normally-ignored padding bytes and extract the security ID. If the security ID read from the padding bytes doesn't match the hashed payload (i.e. h1(payload)), then we know that the QR code was created by a counterfeiter who generated the QR code from a harvested payload. If the security ID read from the padding bytes does match the hashed payload, then we can move on to analyzing the second level of security/anti-counterfeiting information embedded in the QR code image.

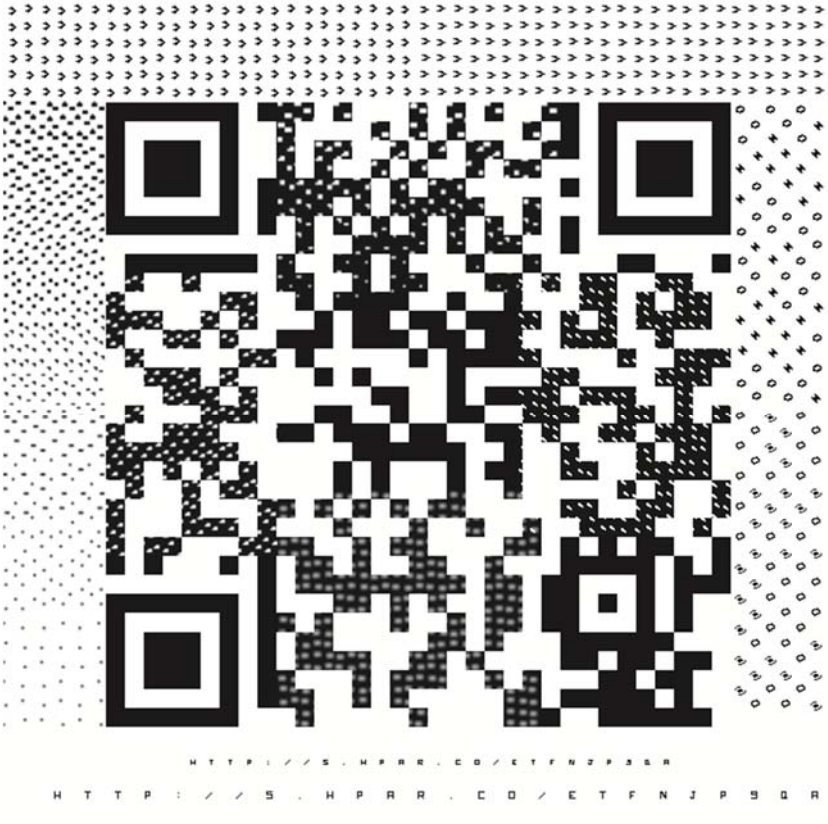
The second level of additional security/anti-counterfeiting information is public key features. Public key features are a fine-grained digital representation of a combination of the payload and a public key that gets embedded into the security code. The public key is a set of bytes whose value can be generated randomly and stored separately, or it can be generated by applying another one-way hash algorithm (h2) to the payload

publicKey = h2(payload)

Because of their fine-grained nature, public key features are difficult to copy or synthesize with enough fidelity to fool image scanning and analysis applications designed to look for authentic public key features.

In the QR code example, the public key features could be represented as small changes to the normal QR code image. The example in Fig. 2 is a QR code generated with public key features and a 6-byte security ID of 0x1308C7C81C1A.

Fig. 2: QR code image with an embedded security ID and embedded public key features



Specialized scanning and analysis applications can compare the actual public key features in the captured QR code image to the expected public key features in an authentic QR code with the same payload to create a distance measurement that estimates the likelihood that the captured QR code is authentic.

Disclosed by Matthew Gaubatz, Bruce Williams and Yuan-Jen Liu, HP Inc.