LABRYS FRONTIER SERIES

Extending Pix:

An approach to offline Dynamic QR Code generation

April
2023

https://doi.org/10.59262/9qu6ex

Extending Pix:

An approach to offline Dynamic QR Code generation

Labrys

Jonas de Abreu
Mariana Cunha e Melo

Labrys's mission is to connect technology, business, and policy to build, inspire, and enable society-focused technology at scale. Our ultimate goal is to create a world where technologists can develop society-focused technology, citizens can effectively debate what society needs from technology, and regulators can align incentives to strengthen an open, inclusive, and sustainable economy for all.

Center for Technology and Public Interest, SL
Carrer de Bailèn, 11, Barcelona, Spain, 08010
www.wearelabrys.com

Table of contents

Summary

The Pix Dynamic QR Code URI can be extended to allow for offline QR Code generation. The proposed solution involves generating URIs that can be used as a vehicle to transmit information from the client to the server, allowing the payee to generate their own URIs. The document also goes into detail about URI properties, encoding, and cryptography. The proposed design balances tradeoffs between the amount of data that can be transmitted and cryptographic guarantees, and uses commonly available cryptographic primitives to reduce implementation costs.

I. Pix Dynamic QR Code and its limitations: the case of batch QR Code generation

Pix is the fastest-growing real-time payments rail in the world. Designed, built, and run by the Brazilian Central Bank, Pix is predicated on a single settlement flow that is triggered by an extensible list of payment initiation methods. One of the options for initiating a Pix transaction is what is called a Dynamic QR Code, which takes the rail's extensibility to the next level. Dynamic QR Codes are based on a URL that the Payee PSP generates and that returns all payment information to the Payer. Its flexibility comes from three facts:

(i) The payment information comes from the Payee PSP at the time of the payment, meaning the entity can return

contextual and updated information every time a payer scans the QR Codes and calls the URL, including assigning transaction-specific UUIDs to improve payment reconciliation;

(ii) The Brazilian Central Bank adopted an open standard for the URL, which is an RFC 3986-compliant URI; and

(iii) The standard for Dynamic QR Codes is non-exhaustive, meaning that Payee PSPs can create fields beyond those specified as long as they use the *schema* prescribed for additional information.

Usually, a Pix Dynamic QR Code can only be generated by the Payee PSP because it creates and stores the *Dynamic QR Code payload* with all the payment information before creating the URI and the QR Code. Even though that is the intended chain of events for Dynamic QR Codes, consider the following scenario.

A utility company has tens of millions of bills to close every 1st day of the month. This company's previous process was to generate the bills using only their infrastructure and, after this huge process was finished, send a single file to the bank to have the payment documents created.

In this scenario, the company's process of closing the bills was almost independent of external connectivity, and their infrastructure was not designed with online access in mind. Changing the internal and mostly offline process to one that has to make a new request to the Payee PSP to get the Dynamic QR Code is not trivial. There are two reasons why:

1. The request to create the Dynamic QR Code can fail. If the Payee PSP API is not working during the bill closing time, for example, the process could be delayed, impacting customers or even exposing the company to regulatory sanctions.

2. Connecting the company's infrastructure to the internet to generate the Dynamic QR Codes may also require provisioning more hardware and people to monitor the infrastructure.

Reducing the implementation cost and improving the payment initiation resilience could be the difference between this company adopting Pix or not. And the way to do it is with Dynamic QR Code offline generation, where the payee generates their own Dynamic QR Codes.

The challenge with this approach is how to generate the URI without a specific integration between the payee and its PSP. After all, when the payer scans the QR Code and calls the URI, the Payee PSP must return all payment information related to that QR Code. The other fields required to generate a Pix Dynamic QR Code, such as the company name, city, and other boilerplate fields, don't need to change for each individual bill.

Before proposing a solution for this issue, it's worth going through the inner works of URIs properties and how they can be leveraged to solve this use case.

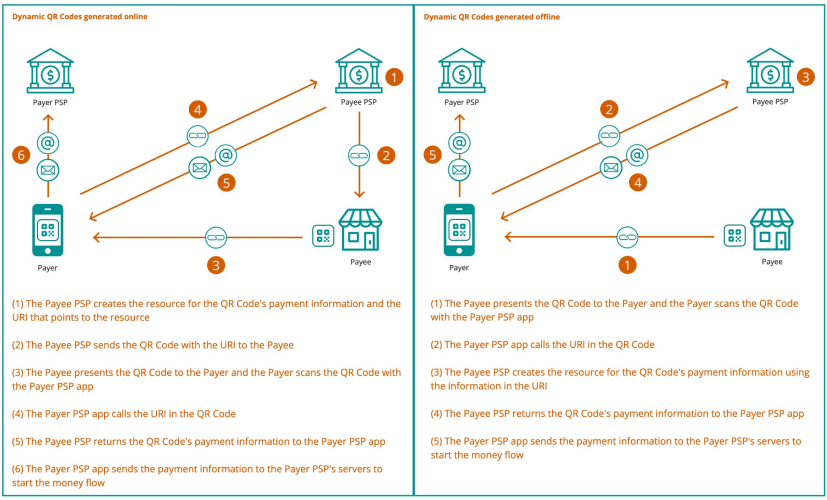
II. URI as identifier and as vehicle

The Pix standards are, with few exceptions, based on open standards. As mentioned, that is the case for the Pix Dynamic QR Code URI, which is an RFC 3986-compliant URI that identifies a Dynamic QR Code resource in the Payee PSP infrastructure. When the payer accesses this URI, the payee PSP returns the correspondent payload with the payment information.

Even though the purpose of a URI is to identify a resource in a server, it can have other functions. Specifically, the URI path can be any sequence of 71 characters without any specified pattern. Using this characteristic, a URI can also be used as a vehicle to transmit information from the client to the server. When the payer scans a Dynamic QR Code with their payment app and the app calls the URI, the Payee

PSP receives that URI, including any information that was inserted in its *path*. At this moment, the Payee PSP would be able to use that information to create the Dynamic QR Code payload and return it to the Payer PSP app to continue the payment flow. Note that at this point, the payment flow happens in the same way as the typical Dynamic QR Code flow.

Using URIs like that, the Payee could generate their own URIs knowing that once the Payer scanned the QR Code, the Payee PSP would be able to generate the appropriate payload and return it to the Payer. See below the comparison between the two flows.



Given the limitations the Pix QR Code's limitations for following the EMVco QR Code standard, the size of the URI is limited to 77 characters, limiting the amount of information the client can transmit to the payee PSP server. This limits general use cases, such as adding bill items to

the Dynamic QR Code. But this limitation does not affect more specific use cases that can leverage small amounts of information to create QR Codes, such as the scenario we mentioned above.

We'll dive into the specifics below. That includes detailed technical analysis, encoding, and cryptography. Feel free to skim through these parts.

III. An approach to offline Dynamic QR Code generation

To make the offline Pix QR Code generation possible, there are two pieces that need to be considered: the URI generation and how the payee PSP server is going to interpret that information to create the Dynamic QR Code payload. The latter is relatively simple since the payee PSP is only changing when the QR Code payload is created. We'll focus on ensuring our strategy to generate offline QR Code URIs respect the restrictions defined by the Pix Dynamic QR Codes.

Our proposed design attempts to balance tradeoffs between the amount of data that can be transmitted and the cryptographic guarantees. In this case, we preferred to provide stronger guarantees than more space so that this

solution is easier to use and adapt. We also focused on using commonly available, used, and understood cryptographic primitives to reduce implementation costs. Depending on product characteristics, it may be possible to reduce the guarantees to increase the amount of information transmitted.

IV. The solution for offline URI generation

Our solution specifies the URI path structure, the encoding, and the encryption.

IV.1. URI path structure

Our solution for the URI looks like this:

```
example.com.br/{KeyID}{CompactIV}
  {EncryptedInformation}{AuthenticationTag}
```

The Pix QR Code establishes a maximum limit for the URL size of 77 characters (Pix manual, page 9), so we minimized the space usage while not increasing too much the implementation cost.

The information is encrypted using AES-GCM with a 128 bits key. The KeyId is a 4-bit key identification to simplify key management operations, such as rotation. We rely on a compact, simple counter CompactIV to reduce in-transit IV size while keeping all the same encryption properties. The AuthenticationTag is 96-bit long to save space as well.

Our most conservative approach allows 128 bits of information (one block) to be transmitted. Small changes in the encoding can enable the transmission of 256 bits (two blocks).

IV.2. Encoding

Considering a domain with 15 characters, we have 62 characters of URI space left to use. Since we encrypt information, we need to encode the resulting bytes in some form, as the URI scheme has a restricted number of characters that we can use.

Our choice of encoding is Base 64 because it is available in every programming language and is widely known. This encoding uses 6 bits per character, inflating the resulting payload by 8/6. That gives us 372 bits (62 characters * 8 bits/character * 8/6) to work with.

Since the space available is small, other less common encoding schemes can be useful. One such scheme would be a Base 71 encoding, which uses all unreserved chars (page 27) to encode information. Base 71 would give us 412 bits, allowing you to encode two blocks instead of a single block using Base 64.

If it were possible to use UTF-8, it would allow us to encode up to 32 bits/character, more than eight times the information we can now. Unfortunately, with the current restrictions, that is not possible. URIs can only hold ASCII characters, and UTF-8 characters must be encoded, multiplying their size instead of compacting.

Even if URIs allowed UTF-8 characters without encoding, the EMVco manual also restricts that by defining that you count the number of characters by counting bytes.

From the EMVco manual:

> "4.1 Payload The length of the payload should not exceed 512 alphanumeric characters, and the number of characters should be reduced proportionally when multi-byte [Unicode] characters are used.
>
> Note that, as data object values with a format of S may contain characters coded as UTF-8 and depending on the alphabet being used there may not be a one-to-one mapping of characters to bytes, special consideration would be needed to determine the number of bytes in the payload".

The Pix manual or BRCode manual doesn't seem to add other restrictions.

IV.3. Encryption

a. KeyId

The 4-bit key id serves a dual purpose. It simplifies key management (especially key rotation) and allows us to use up to 16 keys simultaneously.

Since our compact IV limits the amount of URIs generated to around 16 million per key, additional keys substantially increase the number of URIs that can be generated.

b. CompactIV

AES-GCM requires a 96-bit initialization vector (IV). Encoding a random IV in the URI takes too much space that we can use for other purposes.

To avoid that, we use a counter IV (we can use GCM with non-random IVs) and limit the number of operations with one specific key to 2^24 (around 16 million IVs) to use only 24 bits to represent the IV. When using the IV, we must expand it back to 96 bits by padding it with left zeros.

It is essential to remember that even though GCM does not require random IVs, IVs must never repeat for the same key.

c. EncryptedInformation

While it may be evident that encrypting the URI information ensures that only authorized customers generate these URIs, this also serves another purpose.

The Pix Dynamic QR Code defines a part named pixURLAccessToken that, according to the Pix's security manual (page 22), must be:

> pixURLAccessToken: randomness and security
>
> - Minimum size of 120 random bits;
> - Maximum size may fit the available space as long as the other URL components are considered;
> - It must be impossible to deduct its value except by reading the QR Code [...]

Our EncryptedInformation achieves that. It has at least 128 bits and is indistinguishable from random bits. The value is impossible to be deducted without the encryption key. It is reasonable to argue that this would not comply with the third requirement. Still, since the pixURLAccessToken needs to be stored somewhere in the PSP infrastructure, it is reasonable to assume that storing a cryptographic key would not violate this requirement.

d. AuthenticationTag

The GCM authentication tag is one of the biggest pieces (96 bits) and is not easily reducible, as this is already the smallest size that NIST (section 5.2.1.2) recommends for general-purpose uses. It may be possible to reduce it to 64 or 32 bits, depending on the specific product. NIST has recommendations for that as well (Appendix C).

V. Conclusion

Extensibility is a crucial part of what makes Pix unique. It allows PSPs to extend Pix's functionalities to support new and unexpected use cases. Extensions have the potential to bring Pix to every corner of the payments market and effectively push for a country-wide digitalization of the economy. In this paper, we showed that extending a small piece of the Pix standard makes allows it to support a use case that was not initially considered on its protocol: offline dynamic QR Code generation. Solutions like that can reduce the cost and complexity for a company to join Pix, bring new business opportunities for PSPs, and benefit citizens.

About the authors

Jonas de Abreu is an experienced security engineer with a strong background in software engineering and security. He worked as an external consultant and instructor in the early stages of his career. Jonas spent almost a decade at Nubank, the leading fintech company in LATAM, where he made significant contributions as the Chief Information Security Officer and later as a Principal Engineer. He played a crucial role in shaping Nubank's infosec team and strategy and made a notable impact on the Brazilian Central Bank's decisions regarding the Pix rail. Jonas was responsible for Nubank's technical proposals during the Pix Forum, which were highly valued by industry experts and had a pivotal role in shaping the future of payment systems in Brazil.

About the authors

Mariana Cunha e Melo is an experienced lawyer with a background in strategic litigation, public policy, legal research, and regulated markets. She has worked with Supreme Court justices in Brazil, represented Google in high courts and strategic litigation cases, and built the internet law team at a top law firm. At Nubank, she helped structure the Public Policy team and led efforts to work with the Brazilian Central Bank on designing its Real-Time Payments rail. She has also worked in early-stage startups as a strategic projects owner and director of regulation and strategy. Mariana is also a writer and speaker on topics such as privacy, free speech online, and regulation of fintech companies, with numerous international events under her belt.