Specifiche Qrcode dinamico

CSR-D Lombardia

Revisione del Documento: **01**

Data revisione: **17/12/2025**

	Struttura	Data
Redatto da:	*Divisione - Servizi per le Entrate Regionali, Territorio, Ambiente, Mobilità – ARIA S.p.A.*	17/12/2025

Cronologia delle Revisioni

Revisione	Data	Sintesi delle Modifiche
01	17/12/2025	Prima versione del documento che descrive le specifiche tecniche del Qrcode dinamico utilizzato nel CSR Digitale

Tabella 1 - Cronologia delle revisioni

Limiti di utilizzo del documento
In base alla classificazione del documento.

Tabella 2 - Link di utilizzo del documento

Sommario

Indice delle Tabelle

Indice delle Figure

Non è stata trovata alcuna voce dell'indice delle figure.

1. Introduzione

Il presente documento descrive le specifiche tecniche per la lettura dei QRcode dinamici generati tramite l'SDK. Per evitare frodi il QRcode viene generato ad intervalli di tempo predefiniti e parametrizzabili. In base alla nostra esperienza un buon compromesso tra sicurezza ed usabilità da parte dell'utente prevede un tempo di rigenerazione di 30 secondi. I sistemi di validazione devono essere in grado di leggere il QRcode e decodificarlo in base alle caratteristiche di seguito descritte. La logica di generazione è inserita all'interno dell'SDK che decide, tramite opportune parametrizzazioni e configurazioni, se e quando generare il QRcode offline rispetto a quando richiedere i dati aggiornati online al sistema centrale.

2. Struttura del QRcode

I dati contenuti all'intero del QRcode sono descritti in questo JSON. Tutti i dati oltre al valore in posizione 15 sono riferiti alla firmaClient.

Pos	Nome Campo	Lunghezza (byte)	Tipo
1	Prefix	6	String
2	QRVer	6	String
3	Pnr	18	String
4	Partenza	16	String
5	Destinazione	16	String
6	CodArticolo	16	String
7	DataOraInizioValidita	8	Epoc
8	DataOraFineValidita	8	Epoc
9	FineValiditaOffline	8	Epoc
10	FineValiditaClientkey	8	Epoc
11	PubClientkey	130	String (chiave pubblica P-256 uncompressed)
12	IndiceChiave	6	String
13	FirmaServer	128	String (firma digitale server)
14	DataOraFineValiditaDynamic	8	Epoc
15	DataOraInizioValiditaDynamic	8	Epoc
16	FirmaClient		String (firma digitale client)

Tabella 3 - Json, dati contenuti all'interno del QRcode


```
{
   "qrVersionItems": [
      {
         "qrVer": "563035",
         "base": "Hex",
         "content": [
            {
               "pos": 1,
               "nomeCampo": "Prefix",
               "numZeroPadding": 6,
               "overrideCharForZeroPadding": "2D",
               "base": "Hex",
               "tipo": "String"
            },
```

```
{
    "pos": 2,
    "nomeCampo": "QRVer",
    "numZeroPadding": 6,
    "overrideCharForZeroPadding": "2D",
    "base": "Hex",
    "tipo": "String"
},
{
    "pos": 3,
    "nomeCampo": "Pnr",
    "numZeroPadding": 18,
    "overrideCharForZeroPadding": "2D",
    "base": "Hex",
    "tipo": "String"
},
{
    "pos": 4,
    "nomeCampo": "Partenza",
    "numZeroPadding": 16,
    "overrideCharForZeroPadding": "2D",
    "base": "Hex",
    "tipo": "String"
},
{
    "pos": 5,
    "nomeCampo": "Destinazione",
    "numZeroPadding": 16,
    "overrideCharForZeroPadding": "2D",
    "base": "Hex",
    "tipo": "String"
},
```

```
        {
            "pos": 6,
            "nomeCampo": "CodArticolo",
            "numZeroPadding": 16,
            "overrideCharForZeroPadding": "2D",
            "base": "Hex",
            "tipo": "String"
        },
        {
            "pos": 7,
            "nomeCampo": "DataOraInizioValidita",
            "numZeroPadding": 8,
            "overrideCharForZeroPadding": "2D",
            "base": "Hex",
            "tipo": "Epoc"
        },
        {
            "pos": 8,
            "nomeCampo": "DataOraFineValidita",
            "numZeroPadding": 8,
            "overrideCharForZeroPadding": "2D",
            "base": "Hex",
            "tipo": "Epoc"
        },
        {
            "pos": 9,
            "nomeCampo": "FineValiditaOffline",
            "numZeroPadding": 8,
            "overrideCharForZeroPadding": "2D",
            "base": "Hex",
            "tipo": "Epoc"
        },
```

```
{
    "pos": 10,
    "nomeCampo": "FineValiditaClientkey",
    "numZeroPadding": 8,
    "overrideCharForZeroPadding": "2D",
    "base": "Hex",
    "tipo": "Epoc"
},
{
    "pos": 11,
    "nomeCampo": "PubClientkey",
    "numZeroPadding": 130,
    "overrideCharForZeroPadding": "2D",
    "base": "Hex",
    "tipo": "String"
},
{
    "pos": 12,
    "nomeCampo": "IndiceChiave",
    "numZeroPadding": 6,
    "overrideCharForZeroPadding": "2D",
    "base": "Hex",
    "tipo": "String"
},
{
    "pos": 13,
    "nomeCampo": "FirmaServer",
    "numZeroPadding": 128,
    "overrideCharForZeroPadding": "2D",
    "base": "Hex",
    "tipo": "String"
},
```

```
            {
               "pos": 14,
               "nomeCampo": "FineValiditaQr",
               "numZeroPadding": 8,
               "overrideCharForZeroPadding": "2D",
               "base": "Hex",
               "tipo": "Epoc"
            },
            {
               "pos": 15,
               "nomeCampo": InizioValiditaQr ",
               "numZeroPadding": 8,
               "overrideCharForZeroPadding": "2D",
               "base": "Hex",
               "tipo": "Epoc"
            }
         ]
      }
   ],
   "result": "OK",
   "msgErr": null
}
```


2.1 Formato delle chiavi

Le chiavi Pubbliche sono in formato uncompressed NIST P-256 aka secp256r1 aka prime256v1 Coordinata X: 32 byte Coordinata Y: 32 byte.

3. Processo di decodifica del QRcode

Il processo di decodifica del QRcode prevede i seguenti passaggi. Tutti i dati sono convertiti in HEX, tranne le date che sono convertite in intero.

Per semplicità si utilizza un caso di esempio:

"QRcode":"4352535630352D2D415249415F5452454E5F44333743464639382D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D38363236694147E0694147E0694299B669490C2004695AA20EEEE4015718EFB6996164F00482B86129632B8AE0E748D3DA944AD13C7120CEE52CDB66923D9DB456CC915654BFD39E6910FD68E8DD6C516206125A024130310D868737F4F4CD2876D54E3F50ACCC5445E58C00215A979C3199B4E04F87694FBA3051D9529CEB08CA2F5FAA47591D20751BFF9292AD13DF47F34777E215C1B469414854694148363045022047952D379BFD27150C5513BAEEFD71A34D9ACAA32F0295E3F9C96BE267CDD27F022100BD36486E456FAB038F36DE63533CAFA2F53B09F155C4FBB3B4C6893CE19C456"

Step	Azioni	Valore di esempio	Note
1	Leggere il QRcode con il sistema di lettura		

3	Estrarre i dati del campo in posizione 1
	
	{
	"pos": 1,
	"nomeCampo": "Prefix",
	"numZeroPadding": 6,
	"overrideCharForZeroPadding": "2D",
	"base": "Hex",
	"tipo": "String"
	}
		435253	

4	Estrarre i dati del campo in posizione 2
	
	{
	"pos": 2,
	"nomeCampo": "QRVer",
	"numZeroPadding": 6,
	"overrideCharForZeroPadding": "2D",
	"base": "Hex",
	"tipo": "String"
	}
		563035	

5	Estrarre i dati del campo in posizione 3
	
	{
	"pos": 3,
	"nomeCampo": "Pnr",
	"numZeroPadding": 18,
	"overrideCharForZeroPadding": "2D",
	"base": "Hex",
	"tipo": "String"
	}
		2D2D415249415F5452454E5F4433374346463938	

Step	Azioni	Valore di esempio	Note

6	Estrarre i dati del campo in posizione 4
	
	{
	"pos": 4,
	"nomeCampo": "Partenza",
	"numZeroPadding": 16,
	"overrideCharForZeroPadding": "2D",
	"base": "Hex",
	"tipo": "String"
	}
		2D2D2D2D2D2D2D2D	

7	Estrarre i dati del campo in posizione 5
	
	{
	"pos": 5,
	"nomeCampo": "Destinazione",
	"numZeroPadding": 16,
	"overrideCharForZeroPadding": "2D",
	"base": "Hex",
	"tipo": "String"
	}
		2D2D2D2D2D2D2D2D	

8	Estrarre i dati del campo in posizione 6
	
	{
	"pos": 6,
	"nomeCampo": "CodArticolo",
	"numZeroPadding": 16,
	"overrideCharForZeroPadding": "2D",
	"base": "Hex",
	"tipo": "String"
	}
		2D2D2D2D38363236	

9	Estrarre i dati del campo in posizione 7
	
	{
	"pos": 7,
	"nomeCampo": "DataOraInizioValidita",
	"numZeroPadding": 8,
	"overrideCharForZeroPadding": "2D",
	"base": "Hex",
	"tipo": "Epoc"
	}
		694147E0	

10	Estrarre i dati del campo in posizione 8 { "pos": 8, "nomeCampo": "DataOraFineValidita",	694147E0	

Step	Azioni	Valore di esempio	Note
	"numZeroPadding": 8, "overrideCharForZeroPadding": "2D", "base": "Hex", "tipo": "Epoc" }		

11	Estrarre i dati del campo in posizione 9
	
	{
	"pos": 9,
	"nomeCampo": "FineValiditaOffline",
	"numZeroPadding": 8,
	
	"overrideCharForZeroPadding": "2D",
	"base": "Hex",
	"tipo": "Epoc"
	}
		694299B6	

12	Estrarre i dati del campo in posizione 10
	
	{
	"pos": 10,
	"nomeCampo": "FineValiditaClientkey",
	"numZeroPadding": 8,
	
	"overrideCharForZeroPadding": "2D",
	"base": "Hex",
	"tipo": "Epoc"
	}
		69490C20	

13	Estrarre i dati del campo in posizione 11
	
	{
	"pos": 11,
	"nomeCampo": "PubClientkey",
	"numZeroPadding": 130,
	
	"overrideCharForZeroPadding": "2D",
	"base": "Hex",
	"tipo": "String"
	}
		04695AA20EEEE4015718EFB6996164
		F00482B86129632B8AE0E748
		D3DA944AD13C7120CEE52CDB6692
		3D9DB456CC915654BFD39E6910FD6
		8E8DD6
		C516206125A02	

14	Estrarre i dati del campo in posizione 12
	
	{
	"pos": 12,
	"nomeCampo": "IndiceChiave",
	"numZeroPadding": 6,
	
	"overrideCharForZeroPadding": "2D",
	"base": "Hex",
	"tipo": "String"
	}
		413031	

Step	Azioni	Valore di esempio	Note

15	Estrarre i dati del campo in posizione 13
	
	{
	"pos": 13,
	"nomeCampo": "FirmaServer",
	"numZeroPadding": 128,
	"overrideCharForZeroPadding": "2D",
	"base": "Hex",
	"tipo": "String"
	}
		0D868737F4F4CD2876D54E3F50ACC C5445E58C00215A979C3199B4E 04F87694FBA3051D9529CEB08CA2F 5FAA47591D20751BFF9292AD13D F47F34777E215C1B4	

16	Estrarre i dati del campo in posizione 14
	
	{
	"pos": 14,
	"nomeCampo": "FineValiditaQr",
	"numZeroPadding": 8,
	"overrideCharForZeroPadding": "2D",
	"base": "Hex",
	"tipo": "Epoc"
	}
		69414854	

17	Estrarre i dati del campo in posizione 15
	
	{
	"pos": 15,
	"nomeCampo": "InizioValiditaQr",
	"numZeroPadding": 8,
	"overrideCharForZeroPadding": "2D",
	"base": "Hex",
	"tipo": "Epoc"
	}
		69414836	

18	Estrarre i dati del campo in posizione 16	3045022047952D379BFD27150C5513 BAEEFD71A34D9ACAA32F0295E3F9 C9 6BE267CDD27F022100BD36486E456 FAB038F36DE63533CAFA2F53B09F1 55 C4FBB3B4C6893CE19C4563	
19	Effettuare conversione del campo 1 da HEX ad ASCII	CRS	
20	Effettuare conversione del campo 2 da HEX ad ASCII	V05	
21	Effettuare conversione del campo 3 da HEX ad ASCII	--ARIA_TREN_D37CFF98	
22	Effettuare conversione del campo 4 da HEX ad ASCII	--------	

Step	Azioni	Valore di esempio	Note
23	Effettuare conversione del campo 5 da HEX ad ASCII	--------	In caso di titolo venduto in zone STIBM il campo conterrà riferimento partenza (es. Mi1) e destinazione (es. Mi4).
24	Effettuare conversione del campo 6 da HEX ad ASCII	----8626	
25	Effettuare conversione del campo 7 da HEX ad ASCII	694147E0 (hex) = 1762921440 (decimale) = 2025-11-12 04:24:00 UTC	
26	Effettuare conversione del campo 8 da HEX ad ASCII	694147E0 (hex) = 1762921440 (decimale) = 2025-11-12 04:24:00 UTC	
27	Effettuare conversione del campo 9 da HEX ad ASCII	694299B6 (hex) = 1762981302 (decimale) = 2025-11-12 21:01:42 UTC	
28	Effettuare conversione del campo 10 da HEX ad ASCII	69490C20 (hex) = 1763123232 (decimale) = 2025-11-14 12:27:12 UTC	
29	Effettuare conversione del campo 11 da HEX ad ASCII		
30	Effettuare conversione del campo 12 da HEX ad ASCII	A01	
31	Effettuare conversione del campo 13 da HEX ad ASCII		
32	Effettuare conversione del campo 14 da HEX ad ASCII	69414854 (hex) = 1762921556 (decimale) = 2025-11-12 04:25:56 UTC	
33	Effettuare conversione del campo 15 da HEX ad ASCII	69414836 (hex) = 1762921526 (decimale) = 2025-11-12 04:25:26 UTC	
34	Effettuare conversione del campo 16 da HEX ad ASCII		

4. Processo di validazione del QRcode

Una volta decodificato, il QRcode dovrà essere validato. Tutte le informazioni utili sono contenute all'interno dei dati decodificati.

Il processo di validazione prevede:

- Verifica della firmaClient (campo 16) utilizzando la PubClientKey (campo 11);
- Verifica della firmaServer (campo 13) utilizzando la PubServerKey (ottenuta tramite apposite api esposta dal sistema centrale, per i dettagli si vedano specifiche di integrazione).

Se entrambe le verifiche danno esito positivo si può passare alla verifica delle date di validità del QRcode. Per comprendere se il QRcode è scaduto od è valido occorre verificare che la data & ora corrente (es. del validatore) sia compresa tra la DataOraInizioValiditaDynamic (campo 14) e DataOraFineValiditaDynamic (campo 15).

5. Processo di validazione del QRcode

Qui di seguito si dettaglia il processo di parsing e validazione corredato da un esempio pratico.

A partire dall'immagine del QRcode formato Aztec,

Figure 1 - QRcode formato Aztec - (parsing con app Bynary Eye)

si può ottenere il contenuto che ha questa forma:

4352535630352D2D415249415F5452454E5F44333743464639382D2D2D2D2D2D2D2D2D2D2D
2D2D2D2D2D2D2D2D2D2D38363236694147E0694147E0694299B669490C2004695AA20EEEE40
15718EFB6996164F00482B86129632B8AE0E748D3DA944AD13C7120CEE52CDB66923D9DB4
56CC915654BFD39E6910FD68E8DD6C516206125A024130310D868737F4F4CD2876D54E3F50
ACCC5445E58C00215A979C3199B4E04F87694FBA3051D9529CEB08CA2F5FAA47591D20751
BFF9292AD13DF47F34777E215C1B469414854694148363045022047952D379BFD27150C5513
BAEEFD71A34D9ACAA32F0295E3F9C96BE267CDD27F022100BD36486E456FAB038F36DE63
533CAFA2F53B09F155C4FBB3B4C6893CE19C4563

Il contenuto può essere parsato seguendo le specifiche messe a disposizione

QrVer	Aka QrVer	Pos	NomeCampo	Numero di caratteri	Padding	Aka Padding	Base	Tipo
563035	V05	1	Prefix	6	stringa "2D" a SX	carattere "-" a SX	Hex	String
563035	V05	2	QRVer	6	stringa "2D" a SX	carattere "-" a SX	Hex	String
563035	V05	3	Pnr	40	stringa "2D" a SX	carattere "-" a SX	Hex	String
563035	V05	4	Partenza	16	stringa "2D" a SX	carattere "-" a SX	Hex	String
563035	V05	5	Destinazione	16	stringa "2D" a SX	carattere "-" a SX	Hex	String
563035	V05	6	CodArticolo	16	stringa "2D" a SX	carattere "-" a SX	Hex	String
563035	V05	7	DataOraInizioValidita	8	stringa "2D" a SX	carattere "-" a SX	Hex	Epoc
563035	V05	8	DataOraFineValidita	8	stringa "2D" a SX	carattere "-" a SX	Hex	Epoc
563035	V05	9	FineValiditaOffline	8	stringa "2D" a SX	carattere "-" a SX	Hex	Epoc

563035	V05	10	FineValiditaClientkey	8	stringa "2D" a SX	carattere "-" a SX	Hex	Epoc
563035	V05	11	PubClientkey	130	stringa "2D" a SX	carattere "-" a SX	Hex	String
563035	V05	12	IndiceChiave	6	stringa "2D" a SX	carattere "-" a SX	Hex	String
563035	V05	13	FirmaServer	128	no padding		Hex	String
563035	V05	14	FineValiditaQr	8	stringa "2D" a SX	carattere "-" a SX	Hex	Epoc
563035	V05	15	InizioValiditaQr	8	stringa "2D" a SX	carattere "-" a SX	Hex	Epoc
563035	V05	16	FirmaClient	fino alla fine	no padding		Hex	String

Lo schema indica in che posizione (campo POS), e quanto è lungo (Numero Caratteri) ogni singolo campo, il campo Padding indica la stringa di riempimento e la sia posizione all'interno del campo. Da attenzionare il campo FirmaServer.

Con questa regola il QRcode sopra è cosi parsato:

Prefix	435253
QRVer	563035
Pnr	2D2D415249415F5452454E5F4433374346463938
Partenza	2D2D2D2D2D2D2D2D
Destinazione	2D2D2D2D2D2D2D2D
CodArticolo	2D2D2D2D38363236
DataOraInizioValidita	694147E0
DataOraFineValidita	694147E0
FineValiditaOffline	694299B6
FineValiditaClientkey	69490C20
PubClientkey	04695AA20EEEE4015718EFB6996164F00482B86129632B8AE0E748D3DA944AD13C7120CEE52CDB66923D9DB456CC915654BFD39E6910FD68E8DD6C516206125A02
IndiceChiave	413031
FirmaServer	0D868737F4F4CD2876D54E3F50ACCC5445E58C00215A979C3199B4E04F87694FBA3051D9529CEB08CA2F5FAA47591D20751BFF9292AD13DF47F34777E215C1B4
FineValiditaQr	69414854
InizioValiditaQr	69414836
FirmaClient	3045022047952D379BFD27150C5513BAEEFD71A34D9ACAA32F0295E3F9C96BE267CDD27F022100BD36486E456FAB038F36DE63533CAFA2F53B09F155C4FBB3B4C6893CE19C4563

A questo punto è possibile iniziare a validare il QRcode secondo le due firme, client e server.

La firmaClient è quella che firma tutto il contenuto del messaggio, e la chiave pubblica per validarlo è contenuta all'interno del messaggio stesso al campo PubClientkey.

```
// verifica la firma del client messaggio è tutto senza la firma del client

 var = messaggio
HexToBytes("4352535630352D2D415249415F5452454E5F443337343464639382D2D2D2D2D2D2D2D2D2D2D2D2D2D2D
2D2D2D2D2D2D38363236694147E0694147E0694299B669490C2004695AA20EEEE4015718EFB6996164F00482B861
29632B8AE0E748D3DA944AD13C7120CEE52CDB66923D9DB456CC915654BFD39E6910FD68E8DD6C516206125A0241
30310D868737F4F4CD2876D54E3F50ACCC5445E58C00215A979C3199B4E04F87694FBA3051D9529CEB08CA2F5FAA
47591D20751BFF9292AD13DF47F34777E215C1B46941485469414836");

 var firmabyte =
HexToBytes("3045022047952D379BFD27150C5513BAEEFD71A34D9ACAA32F0295E3F9C96BE267CDD27F022100BD
36486E456FAB038F36DE63533CAFA2F53B09F155C4FBB3B4C6893CE19C4563");

 var pub =
"04695AA20EEEE4015718EFB6996164F00482B86129632B8AE0E748D3DA944AD13C7120CEE52CDB66923D9DB456C
C915654BFD39E6910FD68E8DD6C516206125A02";



var sdkVerifier = new SdkECDSASignVerify(
    messaggio: messaggio
    firma: firmabyte,
    pubkey: PubClientkey.Substring(2, 128));
var validoclient = sdkVerifier.VerificaFirma();
dove SdkECDSASignVerify è
using System;
using System.Security.Cryptography;
using System.Text;

namespace ECDSA
{
    internal class SdkECDSASignVerify
    {
    private readonly byte[] D;
    private readonly byte[] X;
    private readonly byte[] Y;
    private readonly byte[] firma;
    private readonly string pubkey;
    private readonly byte[] message;

public SdkECDSASignVerify(byte[] messaggio, byte[] firma, string pubkey)
{
    this.firma = firma;
    this.pubkey = pubkey;
    this.message = messaggio;
}
  internal bool VerificaFirma()
  {
      var key = ECDsa.Create(new ECParameters
      {
          Curve = ECCurve.NamedCurves.nistP256,
          Q = new ECPoint
          {
              X = HexToBytes(pubkey.Substring(0, pubkey.Length / 2)),
              Y = HexToBytes(pubkey.Substring(pubkey.Length / 2, pubkey.Length / 2))
          }
      });
```

```
            bool isSignatureValid = key.VerifyData(message, firma,
      HashAlgorithmName.SHA256, DSASignatureFormat.Rfc3279DerSequence);
     return isSignatureValid;
  }       private static byte[] FromHexString(string hex)
      {
          var numberChars = hex.Length;
          var hexAsBytes = new byte[numberChars / 2];
          for (var i = 0; i < numberChars; i += 2)
              hexAsBytes[i / 2] = Convert.ToByte(hex.Substring(i, 2), 16);

          return hexAsBytes;
      }
  }
}
```

Fatto questo può essere verificato il contenuto statico della QRcode, e i campi da attenzionare sono: IndiceChiave e FirmaServer e lunghezza FirmaServer (che e 128).

L'indice chiave, che in questo caso è A01, serve a identificare il certificato pubblico che sarà inviato alle CCA tramite API o altro flusso, possono essere aggiunti altri certificati pubblici in corso, al fine di garantire la sicurezza della parte statica. In ogni caso va mantenuta la lista dei certificati pubblici al fine di garantire la validità dei QRcode emessi in precedenza.


```
{
    "listaCertificatiPubblici": [
        {
            "codKey": "A01",
            "publicCertificate": "-----BEGIN CERTIFICATE-----
\r\nMIICFTCCAbugAwIBAgIUdXl1ALDZsBIAHPL+bACn5iAL0zYwCgYIKoZIzj0EAwIw\r\nYDELMAkGA1UEBhMCU
EwxDjAMBgNVBAgMBUl0YWx5MQ8wDQYDVQQHDAZNYXJjaGUx\r\nEzARBgNVBAoMClBsdXNlcnZpY2UxGzAZBgNVBA
MMElBsdXNlcnZpY2VaZXRlbWFR\r\ncjAeFw0yNDExMDYxNzA0MTZaFw0yNjEwMzExNzA0MTZaMGAxCzAJBgNVBAY
TAlBM\r\nMQ4wDAYDVQQIDAVJdGFseTEPMA0GA1UEBwwGTWFyY2hlMRMwEQYDVQQKDApQbHVz\r\nZXJ2aWNlMRsw
GQYDVQQDDBJQbHVzZXJ2aWNlWmV0ZW1hUXIwWTATBgcqhkjOPQIB\r\nBggqhkjOPQMBBwNCAATybRiTkowcM+ikL
QS4fzwLmHTVcQ+x8XRKQvMLgC0GvO5f\r\n/2+hYTOeGoP9st3R5f37SpP8ugPusRBXbtXhmrbpo1MwUTAdBgNVHQ
4EFgQUqPZ1\r\n2/zQNzeiOyz3+1kVqtVVJ74wHwYDVR0jBBgwFoAUqPZ12/zQNzeiOyz3+1kVqtVV\r\nJ74wDwY
DVR0TAQH/BAUwAwEB/zAKBggqhkjOPQQDAgNIADBFAiEAnEY6XRJ03E74\r\nY2FVbGIg+jzdkTzuJ2zNtcTbIFy7
G0QCIFRXThF6tHtloEAPmZGYG5BHYvIPrxed\r\nRLQ3J2cvwcwb\r\n-----END CERTIFICATE-----"
        }
    ],
    "result": "OK",
    "msgErr": null
}
```


Segue

```
    // verifico statico qr con forma server copresa
    var QrStaticoconfirma = "
4352535630352D2D415249415F5452454E5F44333743464639382D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D2D
38363236694147E0694147E0694299B669490C2004695AA20EEEE4015718EFB6996164F00482B86129632B8AE0E7
48D3DA944AD13C7120CEE52CDB66923D9DB456CC915654BFD39E6910FD68E8DD6C516206125A024130310D868737
F4F4CD2876D54E3F50ACCC5445E58C00215A979C3199B4E04F87694FBA3051D9529CEB08CA2F5FAA47591D20751B
FF9292AD13DF47F34777E215C1B4";
    var mypubcert = "-----BEGIN CERTIFICATE-----
\r\nMIICFTCCAbugAwIBAgIUdXl1ALDZsBIAHPL+bACn5iAL0zYwCgYIKoZIzj0EAwIw\r\nYDELMAkGA1UEBhMCUEwx
DjAMBgNVBAgMBUl0YWx5MQ8wDQYDVQQHDAZNYXJjaGUx\r\nEzARBgNVBAoMClBsdXNlcnZpY2UxGzAZBgNVBAMMElBs
dXNlcnZpY2VaZXRlbWFR\r\ncjAeFw0yNDExMDYxNzA0MTZaFw0yNjEwMzExNzA0MTZaMGAxCzAJBgNVBAYTAlBM\r\n
MQ4wDAYDVQQIDAVJdGFseTEPMA0GA1UEBwwGTWFyY2hlMRMwEQYDVQQKDApQbHVz\r\nZXJ2aWNlMRswGQYDVQQDDBJQ
bHVzZXJ2aWNlWmV0ZW1hUXIwWTATBgcqhkjOPQIB\r\nBggqhkjOPQMBBwNCAATybRiTkowcM+ikLQS4fzwLmHTVcQ+x
8XRKQvMLgC0GvO5f\r\n/2+hYTOeGoP9st3R5f37SpP8ugPusRBXbtXhmrbpo1MwUTAdBgNVHQ4EFgQUqPZ1\r\n2/zQ
NzeiOyz3+1kVqtVVJ74wHwYDVR0jBBgwFoAUqPZ12/zQNzeiOyz3+1kVqtVV\r\nJ74wDwYDVR0TAQH/BAUwAwEB/zAK
```

```
BggqhkjOPQQDAgNIADBFAiEAnEY6XRJ03E74\r\nY2FVbGIg+jzdkTzuJ2zNtcTbIFy7G0QCIFRXThF6tHtloEAPmZGY
G5BHYvIPrxed\r\nRLQ3J2cvwcwb\r\n-----END CERTIFICATE-----";


    ECDSASignVerify eCDSASignVerify = new ECDSASignVerify();
    var lunghezzafirma = 128;
    var validoserver = eCDSASignVerify.Verifica(qrSigned: QrStaticoconfirma, pubcert: new
X509Certificate2(Encoding.ASCII.GetBytes(mypubcert)), lunghezzafirma: lunghezzafirma);

dove ECDSASignVerify è

using System;
using System.Security.Cryptography;
using System.Security.Cryptography.X509Certificates;
using System.Text;

namespace ECDSA
{
    public class ECDSASignVerify
    {
        public bool Verifica(string qrSigned, X509Certificate2 pubcert, int lunghezzafirma =
122)
        {


                var messagestring = qrSigned.Substring(0, qrSigned.Length - lunghezzafirma);
                var message = HexToBytes(messagestring);
                var firmastring = qrSigned.Substring(qrSigned.Length - lunghezzafirma);
                var signatureder= HexToBytes(firmastring);

                var assert = pubcert.GetECDsaPublicKey().VerifyData(message, signatureder,
HashAlgorithmName.SHA256, DSASignatureFormat.IeeeP1363FixedFieldConcatenation);
                return assert;
        }

  byte[] HexToBytes(string hex)
  {
      // Rimuove eventuali spazi dalla stringa esadecimale
      hex = hex.Replace(" ", "");

      // Controlla che la stringa abbia una lunghezza pari
      if (hex.Length % 2 != 0)
      {
          throw new ArgumentException("La stringa esadecimale deve avere una lunghezza
pari.");
      }

      byte[] bytes = new byte[hex.Length / 2];
      for (int i = 0; i < hex.Length; i += 2)
      {
          // Converte ogni coppia di caratteri esadecimali in un byte
          bytes[i / 2] = Convert.ToByte(hex.Substring(i, 2), 16);
      }
      return bytes;
  }    }
}
```