Hello CBOR-LD

An introduction to the **C**oncise **B**inary **O**bject **R**epresentation for **L**inked **D**ata

The Problem

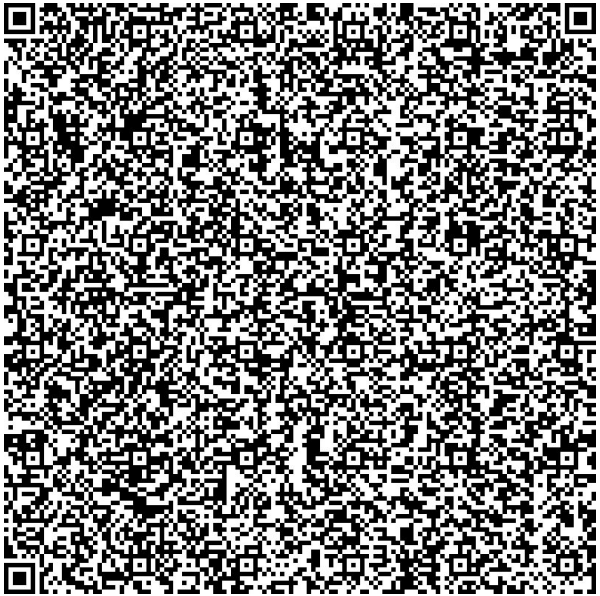
Credit: Jimmy McMillan

VCs and DID Docs are too damn big!*

* For a few important Verifiable Credential (VC) and Decentralized Identifier (DID) use cases - QR Codes, efficient storage, offline interactions.

Use Case: Present Verifiable Credential

Example of the Problem



JSON-LD Document
1,217 bytes

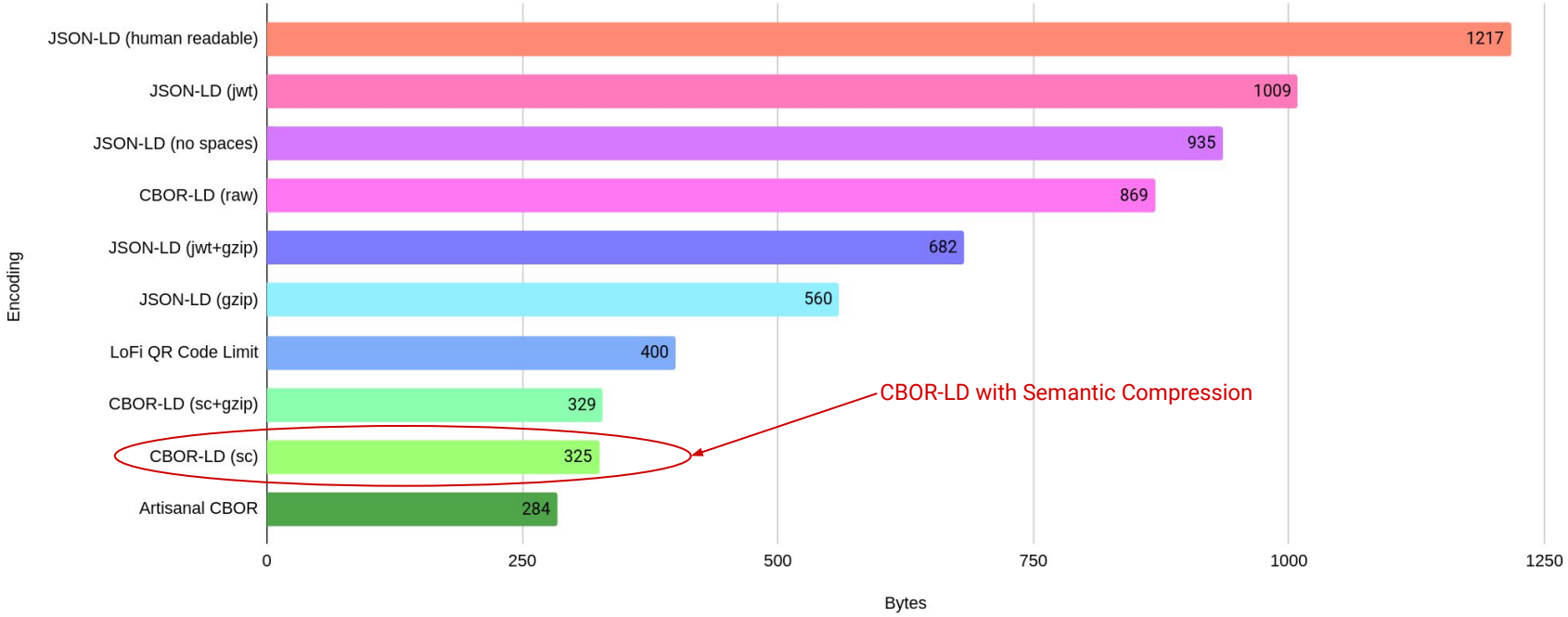
LoFi QR Code Display Limit
400 bytes

Goal

Compress JSON-LD* Documents using a general algorithm.

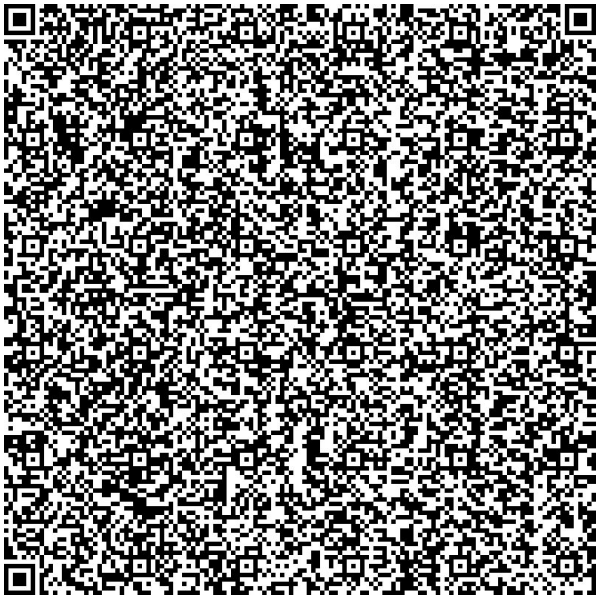
* This mechanism will not work for JSON-only documents for reasons explained later in the presentation

CBOR-LD Results*

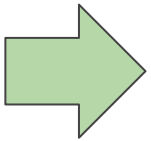


*Example data used: https://github.com/digitalbazaar/cborld/blob/main/examples/cit.jsonld

Verifiable Credentials on Low Fidelity Displays



JSON-LD Verifiable Credential
1,217 bytes



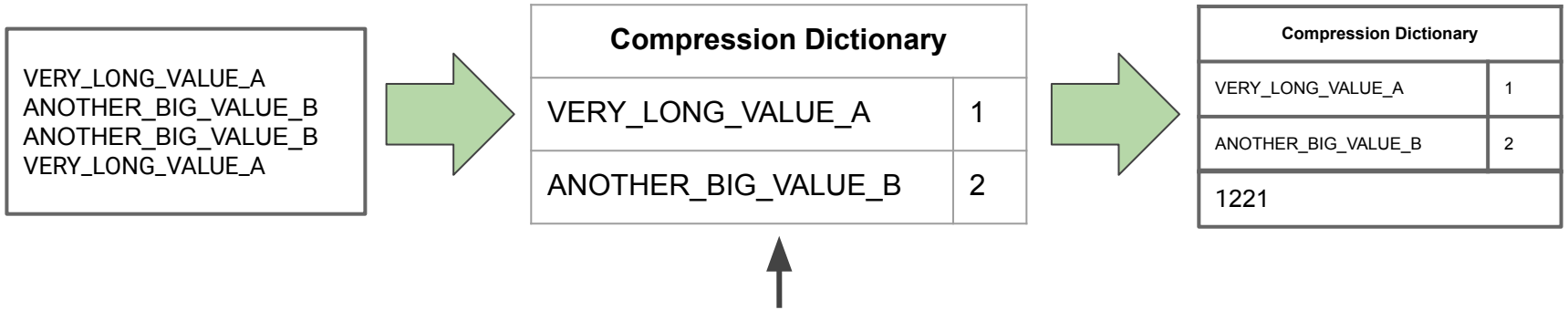
CBOR-LD Verifiable Credential
325 bytes

How does it work?

How Compression Works

Reduce data duplication

aka: Don't Repeat Yourself (DRY)



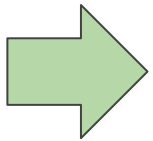
The key to good compression

CBOR-LD (no compression)

No dictionary. Just convert directly to CBOR.

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  "type": "Note",
  "summary": "A note",
  "content": "This is an example note.",
}
```

JSON-LD Document
140 bytes



```
d9              //  CBOR Tag - next 2 bytes
  0500            // CBOR Tag #1280 (Raw CBOR-LD)
    a4            // Map, 4 pairs
      68          // String, length: 8
        40636f6e74657874 // {Key:0}, "@context"
      78          // String, length next 1 byte
        25        // String, length: 37
          687474....616d73 // {Val:0}, "https://www.w3.org/ns/activitystreams"
      67          // String, length: 7
        73756d6d617279 // {Key:2}, "summary"
      66          // String, length: 6
        41206e6f7465 // {Val:2}, "A note"
      67          // String, length: 7
...
```

Raw CBOR-LD Document
111 bytes
(20% smaller)

CBOR-LD Semantic Compression

Build compression dictionary using @context.

Globally linkable, industry-standard dictionary

CBOR-LD Compression Dictionary for https://www.w3.org/2018/credentials/v1	
@context	0x01
issuanceDate	0x02
issuer	0x03
VerifiableCredential	0x04
… and so on...	

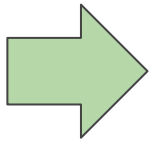
No need to include the dictionary in data!

CBOR-LD Semantic Compression

Semantic compression better than binary compression.

```
{
  "@context": "https://www.w3.org/ns/activitystreams",
  ...
}
```

53 bytes



```
01      // {Key:0}, 1 - @context
10      // {Val:0}, 16 - https://www.w3.org/ns/activitystreams
```

2 bytes

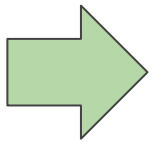
*This is the **key innovation** for getting compression ratios that are far **better than** pure binary compression.*

Term Compression

Compress known JSON-LD terms.

```
{
  "type": "VerifiablePresentation",
  ...
}
```

34 bytes



```
18      // Positive number, next 1 byte
  2b    // {Key:2}, 43 - type
  0e    // {Val:2}, 14 - VerifiablePresentation
```

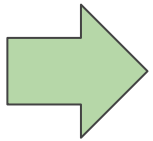
3 bytes

DateTime Compression

Compress xsd:dateTime values.

```
{
  "issuanceDate": "2020-07-14T19:23:24Z",
  ...
}
```

39 bytes



```
18       // Positive number, next 1 byte
  1e     // {Key:3}, 30 - issuanceDate
 1a      // Positive number, next 4 bytes
  5f0e062c // {Val:3}, 1594754604 - 2020-07-14T19:23:24Z
```

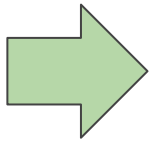
7 bytes

Binary Compression

Compress Multibase-encoded fields to raw bytes.

```
{
  "proofValue": "M55Q1ewxSHq5kS...Ui23IFCWA==",
  ...
}
```

107 bytes



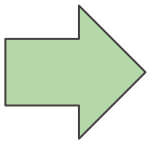
```
18      // Positive number, next 1 byte
  26    // {Key:2}, 38 - proofValue
 58     // Bytes, length next 1 byte
  40    // Bytes, length: 64
    e794357b...14258 // {Val:2} - EdDSA signature
```

68 bytes

URL Compression

Convert well-known URL patterns to binary.

```
{
  "issuer": "did:key:z6MkhNZxXHvf4YMbtZkEkgA9QAz6gN8f9ZtP47EdCEJMF5Hh",
  ...
}
```



```
18       // Positive number, next 1 byte
  20     // {Key:2}, 32 - issuer
 82      // {Val:2}, Array, 2 items
  19     // Positive number, next 2 bytes
   0400  // [0], 1024 - did:key
  58     // Bytes, length next 1 byte
   22    // Bytes, length: 34
    ed012b5f69...1cb9fa628 // [1] - (Ed25519 public key)
```

73 bytes

41 bytes

CBOR-LD Extensibility

Extensible encoders and decoders (codecs):

- Global URL/Datatype codecs (base64url DIDs?)
- Application-specific @context URLs
- Application-specific URL/Datatype codecs

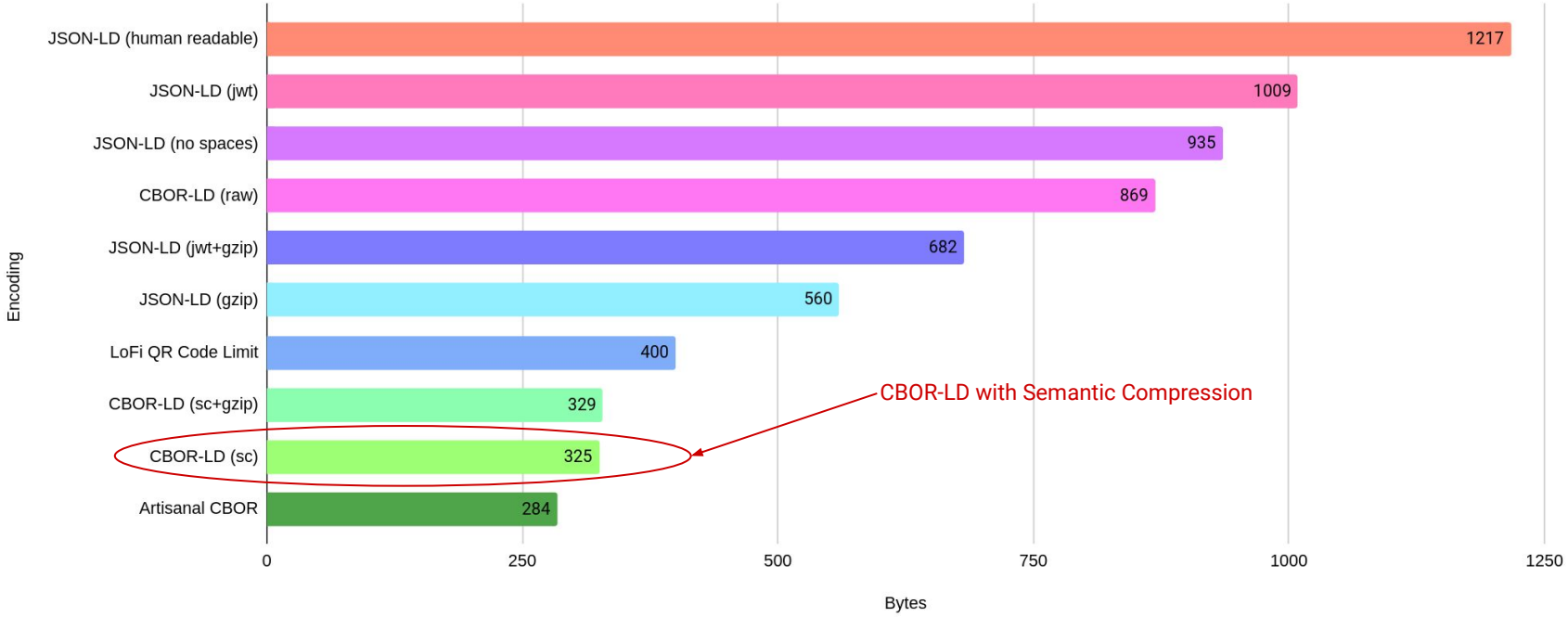
Benefits of CBOR-LD

- Compression ratios up to 40% better than gzip/zlib.
- Works on any compacted JSON-LD Document.
- LD Proof digital signatures do not need to change.
- Extensible registry-based codec model.
- Application-specific codecs.
- "Practical" set of design trade-offs.

Drawbacks of CBOR-LD

- As of July 2020 - it's new, expect bugs.
- Doesn't work for JSON-only data.
- Only works on compacted JSON-LD.
- Processing overhead on top of JSON-LD.
- Codecs are "more complex than necessary".
- Not as compact as Artisanal CBOR.

CBOR-LD Results



The Vision for CBOR-LD

- Eventually, byte-level semantic processing
- Go to CBOR-LD and stay there
- Semantic processing over fixed data structures
- Smaller data means faster calculations
- Push ability to reason (AI) to CPU register level

Questions?

Appendix

Links to Projects

- Implementation
  - https://github.com/digitalbazaar/cborld
- CBOR-LD Specification
  - https://digitalbazaar.github.io/cbor-ld-spec/