Applicable Cryptography

- ***Applicable Free Form Programmable Cryptography.***
- **Author**: Dara O Shayda
- **Artwork**: Tokyo Ghoul

Preface

The Free Form Programmable Cryptography is designed for applicable coding, and standalone application units with instant use and benefit for the public.

The Free Form Programmable Cryptography uses ideograms, pictograms and small Spoken phrases, however with semantics of the most powerful, standard and modern commercial cryptography functionality. Moreover, all computational components reside within multiple highly secured cloud systems suitable for scientific computing and law enforcement applications.

I greatly hope that this aspect of the Free Form Programmable Cryptography ends up being most useful and elevating for everyone on this planet, to learn and to protect!

Dara O Shayda

May 18 2026

Chief of Software

Computational Classnotes

Republic of Ireland

An **ideogram** or **ideograph** (from Greek *idéa* 'idea' + *gráphō* 'to write') is a symbol that is used within a given writing system to represent an idea or concept in a given language.[1] . Some ideograms are more arbitrary than others: some are only meaningful assuming preexisting familiarity with some convention; others more directly resemble their signifieds. Ideograms that represent physical objects by visually illustrating them are called *pictograms*.[3][4]

Digital Signature

Computational components for signing any objects

Digital Signature

hash[ ] vs. ?@ vs. ?[ ]

A **hash code** is a fixed-size numerical value generated from input data of arbitrary size, used primarily for indexing data in hash tables. It allows for efficient data retrieval and storage by mapping variable-length data to a fixed-length representation.

Think of the hash code as a unique digital signature for any data that once even if one bit of that data altered the entire new hash code would be different.

Example: "dara is good lookin" has the hash code:

"0x05ee501e3232083ca0f16b02011944e735f78db170c568c2ea85e56380ddca05"

Counterexample: "dara is good lookin." the ending red dot cause an entire change to the has code above:

"0x4700eceb9e5cbc887ad3dea1833081d51c3572c7886ea9c8a3f30b0c5ad4c614"

```
hash1 = □@"dara is good lookin";
hash2 = hash ["dara is good lookin"];
hash3 = hash ["dara is good looking."];

show hash1 also hash2 also hash3;

save as crypto;
```

Output

"hash1" → "(0x05ee501e3232083ca0f16b02011944e735f78db170c568c2ea85e56380ddca05)"

"hash2" → "(0x05ee501e3232083ca0f16b02011944e735f78db170c568c2ea85e56380ddca05)"

"hash3" → "(0x4700eceb9e5cbc887ad3dea1833081d51c3572c7886ea9c8a3f30b0c5ad4c614)"

?[ ]

```
hash4 = □[]"dara is goodest at being good!"];

show hash4;

save as crypto;
```

Output

"hash4" → "(0x3b3791bf07902593e5a1500a79b56da6bfaf2c25b3c6c2959c6c69654745f8c0)"

goodest

the word goodest is even gooder than the word good infact its actually the bestest

Source: https://www.urbandictionary.com/

Asymmetic Keys

Two keys one public and one private.

Asymmetic Keys

?

□□ is a mighty operator, function and a cloud data source all at once!

As an operator or function it is used with no arguments and it creates the Free Form Programming Language's (ff) default blockchain whic is currently set to Bloxberg 2.0.

In the ff script below you can see on the first line □□ acts as singleton operator or function and , and on the 3rd line as a cloud data source:

```
□□;

show □□;

save as blox;
```

Output

"□□" → {"type" -> "\"elliptic\"", "curve" -> "\"ethereum\"", "compressed" -> "False"}

```
□□;

tmp = "□□curve";

show tmp;

save as blox;
```

Output

"tmp" → ethereum

?: Public Key

?: Private Key

Asymmetic Keys

Encrypt: ??

- Symbol □□behaves like a binary operator
- □□ encrypts what is on its rhs e.g. □□m
- □□□□uses the key on its lhs to encrypt
- crypt = □□□□ m just like e.g. multiplication copies the resultant into the lhs of =

> □□□□ m behaves like a product by the operator □□incorporating is rhs and lhs into a product.

```
□□["rsa"] ;

m = "hi";

crypt = □□□□m;

show crypt;

save as rsa;
```

Output:

"crypt" ⟶ "<|

\"Cipher\" -> \"RSA\",

\"Data\" ->
ByteArray[\"UK0fWzfMEKdQ+aQnc5a3BX0C7ptg4aEa5mbXQhIRz+/17WVRIQ+atsjSET8Rin7BsIPaT
W851pky8dbTLNNg8vja0mI572KZJXRM9YGaFte2UkqwGl4OSEdmv+fXD7KbVi0Ps/uO/EskrksneIPSCn
IMaJ8VkSRjcNsXjuAoPvJ92mwvvfSSdMNxm0FJjuCMSgGZRfwwEK0XWviKcmV4eljBv2m9NT4Rk1e130
EXovnqtGS5XNyGzDQew8fHtgRreeTmSwOO7HCfrqmzYH14aBTRf+KuLLcaPpKo98PGeCmnDW56Cp
DWHsnYsOtkq/oY/9ACHwPk9hR5Hfl9O2l2Yg==\"],

\"OriginalForm\" -> String,

\"Padding\" -> \"PKCS1\"

|>"

Asymmetic Keys

Decrypt: ? ?

```
□□["rsa"] ;

bob = "hi";

crypt  = □□□□ob;

alice = crypt □□□□;

show bob also alice;

save as rsa;
```

Output:

"bob" ⟶ "(\"hi\")"

"alice" ⟶ "(\"hi\")"

Cipher?is non-commutative

□□□□ encrypts while □□□□ decrypts!

□□is called Cipher.

Cipher□□acts like a binary operator.

Asymmetic Keys

?

1. No matter , if the key on the left of the cipher □□ is public and the right side only the private key, how about a single key □ symbol which is replaced according to its corresponding location relative to □□. y = □ □□x encrypts x into y and x = y□□□ decrypts y back to x. If the cipher □□has asymmetric keys this works fine, and symmetric key ciphers by definition fits.

Block Chain: Bloxberg 2.0

Block Chain: Bloxberg 2.0

?, ?, ?, ?

A very simple and a very short but a very powerful program:

```
□□;

show (□□□);
show (□□□);

show □□also □□;

save as blox;
```

Output

"⬡□□" → "0x40d93f50D4a56Be60fDe352c01097E1E42405b69"

"⬡□□" → "0xa020807ff2c90743afbcb8c5f74c191cb6eb037f1613211d42a4bf3977e5f9f0"

"□□" → <|"Type" -> "EllipticCurve", "CurveName" -> "secp256k1", "Compressed" -> False,

"PublicByteArray" -> {4, 250, 117, 78, 63, 249, 35, 37, 119, 220, 59, 73, 128, 250, 4, 178, 253, 14, 126, 0, 19, 129, 44, 182, 137, 197, 161, 110, 203, 53, 24, 120, 64, 179, 9, 240, 194, 86, 37, 144, 38, 84, 88, 224, 184, 173, 242, 231, 142, 52, 104, 62, 26, 166, 48, 141, 145, 95, 228, 62, 101, 132, 235, 155, 110},

"PublicCurvePoint" ->
{113285473313324576753788119846615597644810720062194244047274842411781932480576,
80981563178412881780651944120220270359429660115039823663624727923423640787822}

|>

"□□" → <|"Type" -> "EllipticCurve", "CurveName" -> "secp256k1",

"PublicCurvePoint" ->
{113285473313324576753788119846615597644810720062194244047274842411781932480576,
80981563178412881780651944120220270359429660115039823663624727923423640787822},

"PrivateMultiplier" ->
724274817524094192198119960591994213342903056413246030766445660441431196287 84,

"Compressed" -> False,

"PublicByteArray" -> {4, 250, 117, 78, 63, 249, 35, 37, 119, 220, 59, 73, 128, 250, 4, 178, 253, 14, 126, 0, 19, 129, 44, 182, 137, 197, 161, 110, 203, 53, 24, 120, 64, 179, 9, 240, 194, 86, 37, 144, 38, 84, 88, 224, 184, 173, 242, 231, 142, 52, 104, 62, 26, 166, 48, 141, 145, 95, 228, 62, 101, 132, 235, 155, 110},

"PrivateByteArray" -> {160, 32, 128, 127, 242, 201, 7, 67, 175, 188, 184, 197, 247, 76, 25, 28, 182, 235, 3, 127, 22, 19, 33, 29, 66, 164, 191, 57, 119, 229, 249, 240}

|>

Block Chain: Bloxberg 2.0

??

```
□□;

□□ (
□□"dara1"->"dara tag1"
□"this is a ccndev test1"
□□"dara2"->"dara tag2"
□"this is a ccndev test2"
);

keys = "□□/bloxberg/keys";

show keys;

value = "□□/pdf";
cert = "□□/certificate";
crid ="□□/bloxberg/crid";
bodycrid = "□□/hashlist";

show value also cert also crid also bodycrid;

save as blox2;
```

Output

"keys" ⟶ "{@context, id, type, issuer, issuanceDate, credentialSubject, crid, cridType, metadataJson, proof}"

"value" ⟶ "https://www.wolframcloud.com/obj/ccn2/freeform/ff/blockchain/bloxberg2/cert_pdf/ff_free_blox2.pdf"

"cert" ⟶ "https://www.wolframcloud.com/obj/ccn2/freeform/ff/blockchain/bloxberg2/cert_json/ff_free_blox2.json"

"crid" ⟶ "9fc15cac82ea792549e70d7f7ed796b87933c5cae4785250939fc745b55c1cae"

"bodycrid" ⟶ "{9fc15cac82ea792549e70d7f7ed796b87933c5cae4785250939fc745b55c1cae, 54b6e8fda68070499e08cf95d11fa56db20f5bda24d1237f4f9bcdcfaaecddee}"


```
"[
  {
    \"@context\": [
      \"https: //www.w3.org/2018/credentials/v1\",
      \"https: //w3id.org/bloxberg/schema/research_object_certificate_v1\"
    ],
    \"id\": \"https: //bloxberg.org\",
    \"type\": [
      \"VerifiableCredential\",
      \"BloxbergCredential\"
    ],
    \"issuer\": \"https: //raw.githubusercontent.com/bloxberg-org/issuer_json_l2/master/issuer.json\",
    \"issuanceDate\": \"2026-05-26T00: 49: 31.743598+00: 00\",
    \"credentialSubject\": {
```

```
      \"id\": \"https:
//sepolia.arbiscan.io/address/0x3A2165222F8189A2Ed880e555868BeE1C67963e9\",
      \"issuingOrg\": {
        \"id\": \"https: //bloxberg.org\"
      }
    },
    \"crid\": \"9fc15cac82ea792549e70d7f7ed796b87933c5cae4785250939fc745b55c1cae\",
    \"cridType\": \"sha2-256\",
    \"metadataJson\": \"{
      \\\"dara1\\\": \\\"daratag1\\\",
      \\\"dara2\\\": \\\"daratag2\\\"
    }\",
    \"proof\": {
      \"type\": \"MerkleProof2019\",
      \"created\": \"2026-05-26T00: 49: 32.325863\",
      \"proofValue\":
\"z9nmN4tsD8AYm9oXySiMhLVX9vwX5XjRDJiYU4fJv9r5DoZFnXWHXSfG3wutGpnjLCTvoKVF8X5YSyikB6rFAyoNkUY3
U6SMNF42WNToyFRYZPPHbvU1pPvWcRKggfTMmJAkFTPr3Ef9iFSpKp2kiRc17PKfxBUHUsQ6cc65Yt6NyC92Td3RbpyWNT
EvXP5Zt7dS1dXNd56P9fJs1THhB59iQUvL4zvkb5uTGHsHA3yk7M53oxorzFDZgvH5CNnrgPBBcwVNWnkBUiLprjwPoWfo
7Se56buCJ5p37rr1mqaeEq8k9QYLAqKdB4hVpHEDAX1NWJQ75wa5ZDXvPegmpDxZ44PD9hokgnQGh8mh7f2rBgmnh9uqgv
pnjsgzAP1b3NxCP9C5yZZTWcNKfJmthUk4UuU\",
      \"proofPurpose\": \"assertionMethod\",
      \"verificationMethod\": \"ecdsa-koblitz-pubkey:
0x995b5A82500bc2558Fd37937e795f6468F14642c\",
      \"ens_name\": \"mpdl.berg\"
    }
  },
  {
    \"@context\": [
      \"https: //www.w3.org/2018/credentials/v1\",
      \"https: //w3id.org/bloxberg/schema/research_object_certificate_v1\"
    ],
    \"id\": \"https: //bloxberg.org\",
    \"type\": [
      \"VerifiableCredential\",
      \"BloxbergCredential\"
    ],
    \"issuer\": \"https: //raw.githubusercontent.com/bloxberg-
org/issuer_json_l2/master/issuer.json\",
    \"issuanceDate\": \"2026-05-26T00: 49: 31.743598+00: 00\",
```

```
    \"credentialSubject\": {
      \"id\": \"https:
//sepolia.arbiscan.io/address/0x3A2165222F8189A2Ed880e555868BeE1C67963e9\",
      \"issuingOrg\": {
        \"id\": \"https: //bloxberg.org\"
      }
    },
    \"crid\": \"54b6e8fda68070499e08cf95d11fa56db20f5bda24d1237f4f9bcdcfaaecddee\",
    \"cridType\": \"sha2-256\",
    \"metadataJson\": \"{
      \\\"dara1\\\": \\\"daratag1\\\",
      \\\"dara2\\\": \\\"daratag2\\\"
    }\",
    \"proof\": {
      \"type\": \"MerkleProof2019\",
      \"created\": \"2026-05-26T00: 49: 32.329260\",
      \"proofValue\":
\"z9nmN4tsCzSUK5dJjPu71x2JjnpQCKMnVFzLXbTeEx89caacHd1rFHwKMfoAWJzTYzUHJurvYiwpdoTx8FjdBebr31Kv
9oNvhCrubyXCoA5yxu5fKYKqo6uTQgLqmTwMxE7vuUXhWYhDcSvuXTTEguyPa3PyfgdyuAqDLmQgXKhT7d7TCF2qs67Epg
effJLikgE39z79smuQfmnAD9YWAhZ7XqayCxiabCLpG4jSDKgbdB8V4x2dAwCxf3V6xcsRf5GxbhnZzo9T2ATfYq74D67d
gpVSYPa8gWAtYWZst2EFKwLji2ZJZy2jBJvKg7uaFsypNpoW3g4UXYed8C5S1vLykfCHviz1gRBH9ru8msWs9TMp2uYoPX
VcsJ68Hyx25SsxP4MSpYKCpFABNny6BBamefx\",
      \"proofPurpose\": \"assertionMethod\",
      \"verificationMethod\": \"ecdsa-koblitz-pubkey:
0x995b5A82500bc2558Fd37937e795f6468F14642c\",
      \"ens_name\": \"mpdl.berg\"
    }
  }
]"
```

Example: Photo ID

```
□□;

□□;

//add name and other information;
□□(
□□"first name"->"dara"
□□"last name"->"shayda"
);

show "□□highlight";

save as photo_id;
```

Free Form Editor

Free Form Programming Language ff

Free Form

```
👶;

🔐;

//add name and other information;
⛓(
🔏"first name"->"dara"
🔏"last name"->"shayda"
);

show "👶/highlight";
```

Symbols: ∩ ∪ ⊕ ⊕ ⊗ × ✕ ✖ ✖ ✖ ∂ ∈ ∉ ⊂ → 🍇 🔐 🔑 🔑 ⬡ 👶 S T F A E H

Image

Zip File

Drag and drop a file (or click to browse)

Construct!

Powered by Wolfram

Example: Bardcodes

Any image/photo entering the Free Form language computational container is immediately processed for:

- Any faces? if yes, highlight and cutout the facial regions and compute them (their properties).
- Any barcodes or qr-codes? if yes, highlight the barcode regions and read them.
- Any texts? if yes, OCR the image.
- Add the necessary post-processing e.g. image annotations
- Finally collect at the resulting metadata tags and contents (to be hashed) ⚲ and URL export as String


```
//symbol to process barcodes and qrcodes;
□□□□□□□□;□□□□□□□□

□□;

//add name and other information;
□□ (
□□"first name"->"dara"
□□"last name"->"shayda"
);

show "□□□□□□□□/□□□□□□□□";

show "□□□□□□□□/□□□□□□□□ts";
```

```
save as blox_qr;
```

Output

"□□□□□□□□□□□□□/index"

"□□□□□□□□□□□□□/bartexts"

{"065656915641", "065515000037", "9771468413022", "037000377085", "8435039118747"}

Blockchain Certificate

"□ /certificate" (trimmed text)

https://www.wolframcloud.com/obj/ccn2/freeform/ff/blockchain/bloxberg2/cert_json/ff_free_blox_qr.json

There are currently 4 default mandatory items inside the hash list or the CRID and you can obtain the barcode related ones

Scanned bars as above

https://www.wolframcloud.com/obj/ccn2/freeform/ff/images/ff_free_blox_qr_bars.png

JSON containing if any other OCR texts computed from the image

https://www.wolframcloud.com/obj/ccn2/freeform/ff/images/ff_free_blox_qr_imagetext.json

See the bounding box JSON below addresses as "_boxes"

https://www.wolframcloud.com/obj/ccn2/freeform/ff/images/ff_free_blox_qr_boxes.json

The String values for the barcodes and the qrcodes

https://www.wolframcloud.com/obj/ccn2/freeform/ff/images/ff_free_blox_qr_barcodebox.json

"□□□□□□□□□□□□□□/⚲"

"⚲\"https://www.wolframcloud.com/obj/ccn2/freeform/ff/images/ff_free_\blox_qr_bars.png\"⚲\"https:

qr-codes or bar-codes bounding box scanning area

Any geometrical information pertinent to full understanding and computational access for a region of an image or a recognized object is computed and place in appropriate JSON structures.

"□□□□□□□□□□□□□□/boxes"

[

□

□

4.295e2,

7.305e2

□

□

8.395e2,

9.105e2

□

□

□

□

1.395e2,

3.925e2

[

[

2.745e2,

8.205e2

[

[

[

[

9.805e2,

3.085e2

[

[

9.855e2,

5.805e2

[

[

[

[

1.1775e3,

5.555e2

[

[

1.2515e3,

9.075e2

[

[

[

[

[1.565e2,

2.285e2

],

[

6.365e2,

2.395e2

]

]

Block Chain: Bloxberg 2.0

Example Photo Id + multiple images and barcodes

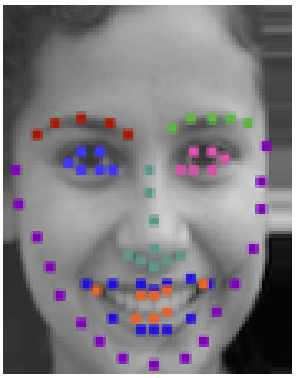
- USA, the state of Virginia (proposed?) Personal Identity card
- With qrcodes, barcodes and multiple images and text.
- Symbol □□ does all the facial computing (neural nets), barcodes, qrcodes and image text
- This will empower a universal blockchained or bloxberged Digital object that could and should enable a new de-centralized identification card system.

```
□□;

□□;

//□□fills in numerous □content + □□etadata into the blockchain;
□□;

show "□□index";

save as photo_id;
```

"□□/index"

"□□/barsindex"

"□□/highlight"



"□□/text"

{

"image_text":"Vi soe VA, USA FEDERAL *\nLYQIM1A IDENTIFICATION CARD

_#n* a\n716700487 e\nTESSIER 8 Gender: Female, Orientation: Straight,

Age: 35,\nBELLA Fi Race: White\/ Non-Hispanic, Votes: Democrat,\nAd

srw aD *B88\"795-a202 5 Job: Industrial Machinery Repair Worker,

Security\n74 IN RD: 5 5 â??Answers: {The Truman Show, Omaha, White},\n

Dato of birt ry Marital status: married, Salutation: Mrs.\nOeio4rt988

A\na ED Height: 5â?? 6â?□ \nF BRO Weight: 136 ssn: Q\nNONE\n\ntek

items tebe IMT |"

}

"□□/bartexts"

{

"@<LF><FS><CR>ANSI

6360050101DL00300203DLDAQ3265188<LF>DAALOTT,ERIC,B,<LF>DAG763 TEST

STREET<LF>DAINEW YORK CITY<LF>DAJSC<LF>DAK10005 <LF>DARD <LF>DAS

    <LF>DAT   <LF>DAU601<LF>DAW170<LF>DAYBRO <LF>DAZBRO

<LF>DBA20241004<LF>DBB19911004<LF>DBC1<LF>DBD20140101<LF>DBG2<LF>DBH1

}

Color Keys

Blockchain by Bloxberg 2.0

CCN cloud systems

Image Processing + Bloxberg 2.0 Blockchain Digitals

{

"□□/face", "□□/face1", "□□/highlight", "□□/faces", "□□/index", "□□/text",

"□□/facials", "□□/boxes", "□□/bartexts", "□□/barsindex", "□□/♀", "□□/□□", "□□

processed", "(□□)", "□□/type", "□□/curve", "□□/compressed", "(□□)",

"(⬡□□)", "⬡□□/type", "□□/type", "(□□)", "(⬡□□)", "⬡□□/type", "□□/type",

"□ /bloxberg/contexts", "□ /bloxberg/context1/@context",

"□ /bloxberg/context1/id", "□ /bloxberg/context1/type",

"□ /bloxberg/context1/issuer", "□ /bloxberg/context1/issuancedate",

"□ /bloxberg/context1/credentialsubject", "□ /bloxberg/context1/crid",

"□ /bloxberg/context1/cridtype", "□ /bloxberg/context1/metadatajson",

"□ /bloxberg/context1/proof", "□ /bloxberg/context2/@context",

"□ /bloxberg/context2/id", "□ /bloxberg/context2/type",

"□ /bloxberg/context2/issuer", "□ /bloxberg/context2/issuancedate",

"□ /bloxberg/context2/credentialsubject", "□ /bloxberg/context2/crid",

"□ /bloxberg/context2/cridtype", "□ /bloxberg/context2/metadatajson",

"□ /bloxberg/context2/proof", "□ /bloxberg/context3/@context",

"□ /bloxberg/context3/id", "□ /bloxberg/context3/type",

"□ /bloxberg/context3/issuer", "□ /bloxberg/context3/issuancedate",

"□ /bloxberg/context3/credentialsubject", "□ /bloxberg/context3/crid",

"□ /bloxberg/context3/cridtype", "□ /bloxberg/context3/metadatajson",

"□ /bloxberg/context3/proof", "□ /bloxberg/context4/@context",

"□ /bloxberg/context4/id", "□ /bloxberg/context4/type",

"□ /bloxberg/context4/issuer", "□ /bloxberg/context4/issuancedate",

"□ /bloxberg/context4/credentialsubject", "□ /bloxberg/context4/crid",

"□ /bloxberg/context4/cridtype", "□ /bloxberg/context4/metadatajson",

"□ /bloxberg/context4/proof", "□ /bloxberg/context5/@context",

"□ /bloxberg/context5/id", "□ /bloxberg/context5/type",

"□ /bloxberg/context5/issuer", "□ /bloxberg/context5/issuancedate",

"□ /bloxberg/context5/credentialsubject", "□ /bloxberg/context5/crid",

"□ /bloxberg/context5/cridtype", "□ /bloxberg/context5/metadatajson",

"□ /bloxberg/context5/proof", "□ /bloxberg/context6/@context",

"□ /bloxberg/context6/id", "□ /bloxberg/context6/type",

"□ /bloxberg/context6/issuer", "□ /bloxberg/context6/issuancedate",

"□ /bloxberg/context6/credentialsubject", "□ /bloxberg/context6/crid",

"□ /bloxberg/context6/cridtype", "□ /bloxberg/context6/metadatajson",

"□ /bloxberg/context6/proof", "□ /bloxberg/context7/@context",

"□ /bloxberg/context7/id", "□ /bloxberg/context7/type",

"□ /bloxberg/context7/issuer", "□ /bloxberg/context7/issuancedate",

"□ /bloxberg/context7/credentialsubject", "□ /bloxberg/context7/crid",

"□ /bloxberg/context7/cridtype", "□ /bloxberg/context7/metadatajson",

"□ /bloxberg/context7/proof", "□ /contents", "□ /hashlist", "□ /key_eth",

"□ /value", "□ /pdf", "□ /fullcredentialobject",

"□ /certificate", "□ /metadata",

}

"□□/⚲"

"⚲\"https://www.wolframcloud.com/obj/ccn2/freeform/ff/images/ff_free_blox_id_face.png\"⚲\"\
https://www.wolframcloud.com/obj/ccn2/freeform/ff/images/ff_free_blox_\ id_facehighlight.png\"⚲\"\
https://www.wolframcloud.com/obj/ccn2/freeform/ff/images/ff_free_blox_id_facehighlights.png\"⚲\"\
https://www.wolframcloud.com/obj/ccn2/freeform/ff/images/ff_free_blox_id_index.png\"⚲\"\
https://www.wolframcloud.com/obj/ccn2/freeform/ff/images/ff_free_blox_id_imagetext.json\"⚲\"\
https://www.wolframcloud.com/obj/ccn2/freeform/ff/images/ff_free_blox_id_facialpoints.json\"⚲\"\
https://www.wolframcloud.com/obj/ccn2/freeform/ff/images/ff_free_blox_id_barcodebox.json\""

"□□/□□"

"□□\"group\"->\"ff\"□□\"user\"->\"free\"□□\"ffname\"->\"blox_\ id\"□□\"publisher\"->\"CCN\"□□
\"license\"->\"CC-BY-NC-ND 4.0\""

Full Credential Object

"□ /fullcredentialobject"

"([{\"@context\":[\"https://www.w3.org/2018/credentials/v1\",\"https://w

3id.org/bloxberg/schema/research_object_certificate_

v1\"],\"id\":\"https://bloxberg.org\",\"type\":[\"VerifiableCredential

\",\"BloxbergCredential\"],\"issuer\":\"https://raw.githubusercontent.c

om/bloxberg-org/issuer_json_

l2/master/issuer.json\",\"issuanceDate\":\"2026-06-02T02:59:59.

448151+00:00\",\"credentialSubject\":{\"id\":\"https://sepolia.arbisca

n.io/address/0xAF3aD7B80e90C0Ca651a8e673E193eCC690BADA8\",\"issuingOrg

\":{\"id\":\"https://bloxberg.org\"}},\"crid\":\"9d05f7aa5009599ded40f

f901de8fbce8bedb00004c0e6ad77fd128c7fac6806\",\"cridType\":\"sha2-256\",\"m

etadataJson\":\"{\\\"group\\\":\\\"ff\\\", \\\"user\\\":\\\"free\\\",

\\\"ffname\\\":\\\"blox_id\\\", \\\"publisher\\\":\\\"CCN\\\",

\\\"license\\\":\\\"CC-BY-NC-ND

4.0\\\"}\",\"proof\":{\"type\":\"MerkleProof2019\",\"created\":\"2026-0

6-02T02:59:59.

947487\",\"proofValue\":\"zLBHrm1GqmftAu35gTndQtHSErc2pdrx2dQ69dF9Fqvj

vqCY71iNnFEaW1kSw5rVcT6XXiWJHY7uUQyKhjPsg32KpHfyMR2GZZkEeY7YvcRRQSAjkK

sduAX2KXxWnFFBWFbn27m7m9ritdwzTzbbGFPRNfAqSeGhDKxTqLp4wYgqfAWAuFxDnG3n

hyUDCAP1c9BohuH45o3f64tEnAH3LQmJkzGKKixRqz9TtypnELSJwzeMPTWZNPTihHhL2P

A9N9YvMt3RtbJGaWzRAwebi7Tf2FwujSNmCX7Cygb13 ...

...

@context\":[\"https://www.w3.org/2018/creden

tials/v1\",\"https://w3id.org/bloxberg/schema/research_object_

certificate_

v1\"],\"id\":\"https://bloxberg.org\",\"type\":[\"VerifiableCredential

\",\"BloxbergCredential\"],\"issuer\":\"https://raw.githubusercontent.c

om/bloxberg-org/issuer_json_

l2/master/issuer.json\",\"issuanceDate\":\"2026-06-02T02:59:59.

448151+00:00\",\"credentialSubject\":{\"id\":\"https://sepolia.arbisca

n.io/address/0xAF3aD7B80e90C0Ca651a8e673E193eCC690BADA8\",\"issuingOrg

\":{\"id\":\"https://bloxberg.org\"}},\"crid\":\"5b57331ee65751d0024a3

f47940cdecdeebf27e070c670cdee7e472aaaa274f3\",\"cridType\":\"sha2-256\",\"m

etadataJson\":\"{\\\"group\\\":\\\"ff\\\", \\\"user\\\":\\\"free\\\",

\\\"ffname\\\":\\\"blox_id\\\", \\\"publisher\\\":\\\"CCN\\\",

\\\"license\\\":\\\"CC-BY-NC-ND

4.0\\\"}\",\"proof\":{\"type\":\"MerkleProof2019\",\"created\":\"2026-0

6-02T02:59:59.\

951710\",\"proofValue\":\"zLBHrm1GqUpCCwm ....

...

"□ /pdf"

"({\"https://www.wolframcloud.com/obj/ccn2/freeform/ff/blockchain/bloxberg2/cert_pdf/ff_free_blox_id_1.pdf\",

\"https://www.wolframcloud.com/obj/ccn2/freeform/ff/blockchain/bloxberg2/cert_pdf/ff_free_blox_id_2.pdf\",

\"https://www.wolframcloud.com/obj/ccn2/freeform/ff/blockchain/bloxber2/cert_pdf/ff_free_blox_id_3.pdf\",

\"https://www.wolframcloud.com/obj/ccn2/freeform/ff/blockchain/bloxberg2/cert_pdf/ff_free_blox_id_4.pdf\",

\"https://www.wolframcloud.com/obj/ccn2/freeform/ff/blockchain/bloxberg2/cert_pdf/ff_free_blox_id_5.pdf\",

\"https://www.wolframcloud.com/obj/ccn2/freeform/ff/blockchain/bloxberg2/cert_pdf/ff_free_blox_id_6.pdf\",

\"https://www.wolframcloud.com/obj/ccn2/freeform/ff/blockchain/bloxberg2/cert_pdf/ff_free_blox_id_7.pdf\"})"

Certificate 1

Certificate 7