Digital Signature

Computational components for signing any objects

- hash[ ], □ [ ], □ @
- Digital Signature

hash[ ], □ [ ], □ @

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 = □@"dara is good lookin";
hash3 = hash["dara is good looking"];

show hash1 also hash2 also hash3;

save as myhash;
```

Output

hash1⟶("0x05ee501e3232083ca0f16b02011944e735f78db170c568c2ea85e56380ddca05")

hash2⟶("0x05ee501e3232083ca0f16b02011944e735f78db170c568c2ea85e56380ddca05")

hash3⟶("0x4700eceb9e5cbc887ad3dea1833081d51c3572c7886ea9c8a3f30b0c5ad4c614")

Digital Signature

- Your digital signature is the proof that you signed something
- Your digital signature attests to the fact that your data was not tampered with
- In asymmetric key system you ought to sign with the private key □□ or any symbol representing one
- message: any expression from symbols to numerals to expressions to images to audio and video and so on
- □□: Free Form language symbol for digital signature in binary operator form in this case. key□□message computes a signature for the message given this key. Operator form □□ should be thought of a multiplication a □ b where two left and right operands act on each other to compute a product. In a □□b, the key a acts on the message/data b to output a signature.

Sign

```
□□"rsa";

message = "hi";

signature = □□□□ssage;

save as signature1;
```

Cloud Object Browser: signature1

https://www.wolframcloud.com/obj/ccn2/AI/freeform/blockchain/publishedtest/certs_free_signature1

Verify Signature

- Signature, generally performed in another program in another system and in the past. To access that signature Free Form language follows the path paradigm e.g. "/certs/free/signature1"
- base = "/certs/free/signature1"; states that the symbol base represents the path on the rhs; base can by any other symbol/word
- s = "base/signature/value"; allows the access to the external signature value again in path form and s presents the accessed signature

> Paths starting with / are external paths e.g. "/certs/free/signature1" created by other programs and networks , while paths like "base/signature/value" are local to the current Free Form language script.

- key = "base/□□"; in paired key cryptography systems, assumption is everyone has access to the public key □□. This public key is then used to verify if the computed signature was indeed performed by the secret private key □□ .
- q = (✍□□ s ⚲ message)□□key; this time the rhs the public key acts on the lhs namely (✍□□ s ⚲ message) and places returns the evaluation boolean result in q.
- (✍□□ s ⚲ message) is a generalization of list in other programming languages. Symbols ✍ □□ and ⚲ "pin" contents into the list and depending the pin symbol the content is interpreted
- ⚲ message pins a message the subject matter of interest in this signature computations
- ✍□□ s pins the signature from a previous program who allegedly signed the message namely ⚲ message

```
base = "/certs/free/signature1";


s = "base/signature/value";


key = "base/□□";


message = "hi";


q = (✍□□s □ message)□□key;


show q;


save as verify1;
```

Output:

"q" ⟶ True

Cloud Object Browser: verify1

https://www.wolframcloud.com/obj/ccn2/AI/freeform/blockchain/publishedtest/certs_free_verify1

FAIL: Verify Signature

```
base = "/certs/free/signature1";
```

```
s = "base/signature/value";

key = "base/□□";

message = "hi!";

q = ( ✍□□s □ message  ) □□ey;

show q;

save as verify1-2;
```

Output:

"q" ⟶ False

Cloud Object Browser: verify1-2

https://www.wolframcloud.com/obj/ccn2/AI/freeform/blockchain/publishedtest/ff_free_verify1-2