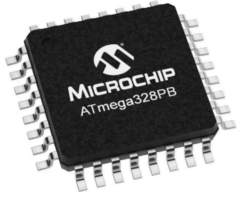
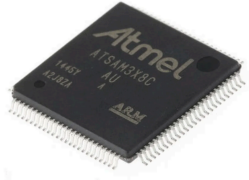
Next Generation Crypto

Light-weight crypto.
Quantum-robust crypto
Tokenization.
Zero-knowledge.
Homomorphic Encryption.
zkSnarks, Range-proofs
IPFS

Prof Bill Buchanan OBE
https://asecuritysite.com/

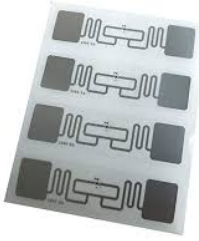
A Computer?





Light-weight crypto

- Conventional cryptography. Servers and Desktops. Tablets and smart phones.
- Light-weight cryptography. Embedded Systems. RFID and Sensor Networks.



Thus often light-weight cryptography methods balance performance (throughput) against **power drain and GE (gate equivalents).** Along with this the method must also have a low requirement for RAM (where the method requires the usage of running memory to perform its operation) and ROM (where the method is stored on the device). In order to assess the strengths of various methods we often **define the area** that the cryptography function will use on the device — and which is defined in μm^2.

View methods

Light-weight crypto

Cipher	Key bits	Block bits	Cycles per block	Throughput at 100 kHz (Kbps)	Logic process	Area (GEs)
Block ciphers						
Present	80	64	32	200.00	0.18 µm	1,570
AES	128	128	1,032	12.40	0.35 µm	3,400
Hight	128	64	34	188.20	0.25 µm	3,048
Clefia	128	128	36	355.56	0.09 µm	4,993
mCrypton	96	64	13	492.30	0.13 µm	2,681
DES	56	64	144	44.40	0.18 µm	2,309
DESXL	184	64	144	44.40	0.18 µm	2,168
Stream ciphers						
Trivium[5]	80	1	1	100.00	0.13 µm	2,599
Grain[5]	80	1	1	100.00	0.13 µm	1,294

*AES: Advanced Encryption Standard; DES: Data Encryption Standard; DESXL: lightweight DES with key whitening.

function will use on the device — and which is defined in μm^2.

View methods

Light-weight crypto

Cipher	Key size (bits)	Block size (bits)	Encryption (cycles/ block)	Throughput at 4 MHz (Kbps)	Decryption (cycles/ block)	Relative throughput (% of AES)	Code size (bytes)	SRAM size (bytes)	Relative code size (% of AES)
Hardware-oriented block ciphers									
DES	56	64	8,633	29.6	8,154	38.4	4,314	0	152.4
DESXL	184	64	8,531	30.4	7,961	39.4	3,192	0	112.8
Hight	128	64	2,964	80.3	2,964	104.2	5,672	0	200.4
Present	80	64	10,723	23.7	11,239	30.7	936	0	33.1
Software-oriented block ciphers									
AES	128	128	6,637	77.1	7,429	100.0	2,606	224	100.0
IDEA	128	64	2,700	94.8	15,393	123.0	596	0	21.1
TEA	128	64	6,271	40.8	6,299	53.0	1,140	0	40.3
SEA	96	96	9,654	39.7	9,654	51.5	2,132	0	75.3
Software-oriented stream ciphers									
Salsa20	128	512	18,400	111.3	NA	144.4	1,452	280	61.2
LEX	128	320	5,963	214.6	NA	287.3	1,598	304	67.2

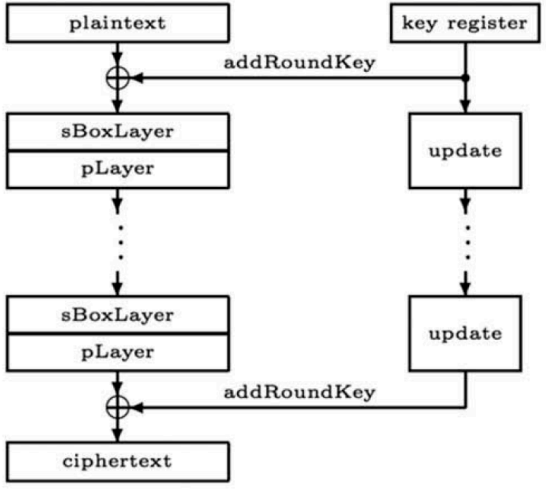
*IDEA: International Data Encryption Algorithm; TEA: Tiny Encryption Algorithm; SEA: Scalable Encryption Algorithm.

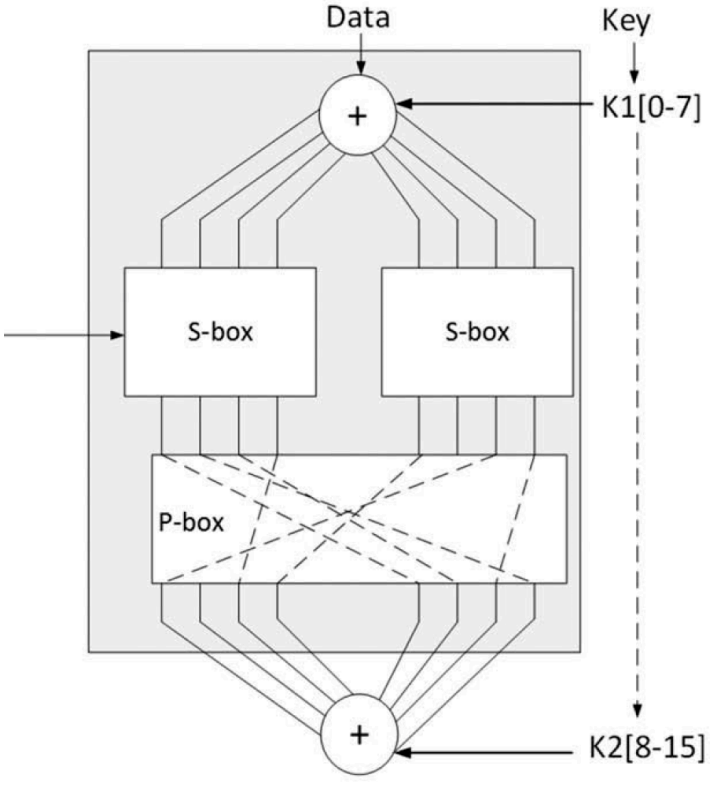
function will use on the device" and which is defined in *pm*".

View methods

Light-weight crypto (PRESENT)

```
generateRoundKeys()
for i = 1 to 31 do
    addRoundKey(STATE, K_i)
    sBoxLayer(STATE)
    pLayer(STATE)
end for
addRoundKey(STATE, K_32)
```





x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
$S[x]$	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

View methods

Light-weight crypto (Finalists)

	Area (μm^2 - synthesis over 22nm)	Area (kGE)
TinyJambu	716	3.6
Grain128-AEAD	861	4.3
GIFT-COFB	1,618	8.1
Romulus	1,961	9.8
ASCON	21,93	11
Xoodyak	2,387	11.9
Photon-Beetle	2,523	12.6
ISAP	3,080	15.4
Elephant	3,458	17.3
Sparkle	7,897	739.5

Name	Type	Variant	Underlying Primitive	State (Bits)	Key (Bits)	Mode	Rate/Block (Bits)	Tag (Bits)	Security (Bits)
Ascon	Sponge	Ascon-128	Ascon-p	320	128	Duplex	64	128	128
		Ascon-128a	Ascon-p	320	128	Duplex	128	128	128
Elephant	Sponge	Jumbo	Spongent	176	128	Elephant	176	64	127
		Dumbo	Spongent	160	128	Elephant	160	64	112
		Delirium	Keccak	200	128	Elephant	176	128	127
GIFT-COFB	Block	GIFT-COFB	GIFT-128	192	128	COFB	128	128	128
Grain-128AEAD	Stream	Grain-128AEAD	N/A	256	128	N/A	1	64	128
ISAP	Sponge	ISAP-A-128	Ascon-p	320	128	ISAP	64	128	128
		ISAP-K-128	Keccak	400	128	ISAP	144	128	128
		ISAP-K-128A	Keccak	400	128	ISAP	144	128	128
		ISAP-A-128A	Ascon-p	320	128	ISAP	64	128	128
PHOTON-Beetle	Sponge	PHOTON-Beetle-AEAD[128]	PHOTON256	256	128	Beetle	128	256	121
		PHOTON-Beetle-AEAD	PHOTON256	256	128	Beetle	32	256	128
Romulus	Block	Romulus-M	Skinny-128-384	384	128	COFB	128	128	128
		Romulus-N	Skinny-128-384	384	128	COFB	128	128	128
		Romulus-T	Skinny-128-384	384	128	COFB	128	128	128
SPARKLE	Sponge	SCHWAEMM256-128	SPARKLE	384	128	SPARKLE	256	128	120
		SCHWAEMM128-128	SPARKLE	256	128	SPARKLE	128	128	120
		SCHWAEMM192-192	SPARKLE	384	192	SPARKLE	192	192	184
		SCHWAEMM256-256	SPARKLE	512	256	SPARKLE	256	256	248
TinyJambu	Sponge	TinyJambu	TinyJambu	128	128	TinyJambu	32	64	120
Xoodyak	Sponge	Xoodyak	Xoodoo	384	128	Cyclist	352	128	128

Elsadek, S. Aftabjahani, D. Gardner, E. MacLean, J. R. Wallrabenstein, and E. Y. Tawfik, "Hardware and energy efficiency evaluation of nist lightweight cryptography standardization finalists," in 2022 IEEE International Symposium on Circuits and Systems (ISCAS). IEEE, 2022, pp. 133–137.
H. Madushan, I. Salam, and J. Alawatugoda, "A review of the nist lightweight cryptography finalists and their fault analyses," Electronics, vol. 11, no. 24, p. 4199, 2022.

View methods

Light-weight crypto (Finalists)

Algorithm	Impl.	Primary	Flag	Size	h(8)	h(16)	h(32)	h(64)	h(128)	Benchmark
reference	$nacl_ref$	yes	O3	18774	768	768	772	1364	1968	1
sparkle	rhys	yes	O1	7912	1036	1036	1468	2272	3884	2
Xoodyak	XKCP-AVR8	yes	O3	2604	1284	1288	1924	3192	5732	2.9
gimli	rhys	yes	O3	19554	1284	1920	2544	3804	6312	3.2
ascon	rhys	yes	O3	2178	2972	3552	4736	7088	11784	6
drygascon	rhys	no	O3	15500	4604	4600	6540	10360	17912	9.1
photon-beetle	$avr8_speed$	yes	O3	2948	2372	2364	6940	16084	34172	17.4
skinny	rhys	yes	O2	9784	7048	10556	13976	20952	34896	17.7

Algorithm	Impl.	Primary	Flag	Size	Enc(0:8)	Dec(0:8)	Enc(128:129)	Dec(128:128)	Bench.(128)	Bench.(8)
sparkle	rhys	yes	O3	12290	1276	1316	4648	5072	4.7	3.3
Xoodyak	XKCP-AVR8	yes	O3	4560	2596	2608	7184	7128	3.3	1.6
knot	$avr8_speed$	no	Os	1664	2124	2140	8144	8160	2.9	2
ascon	rhys	no	O3	5180	1240	1284	8056	8488	2.8	3.3
GIFT-COFB	rhys	yes	O1	23312	1852	1892	8220	8776	2.7	2.2
saeaes	ref	no	O3	17062	1208	1212	8992	9004	2.6	3.4
hyena	rhys	yes	O3	293860	1912	1964	8960	9396	2.5	2.2
elephant	rhys	no	O3	13106	1924	1948	9260	9796	2.4	2.2
estate	ref	yes	O3	9434	1424	1448	10276	10292	2.3	2.9
romulus	rhys	no	O3	19346	1632	1676	10152	10568	2.2	2.5
spook	rhys	no	O3	12942	2984	2968	10272	10708	2.2	1.4
tinyjambu	rhys	yes	O3	9174	1232	1288	10364	10888	2.2	3.4
subterranean	rhys	yes	Os	6042	3372	3460	10288	10944	2.2	1.2
orange	rhys	yes	O3	12140	2500	2536	11200	11620	2	1.7
gimli	rhys	yes	O3	21272	1920	1956	11944	12360	1.9	2.2
skinny	rhys	no	O1	12452	1604	1644	12960	14372	1.7	2.6
photon-beetle	$avr8_speed$	yes	Os	3536	2444	2472	20076	20092	1.2	1.7
reference	rhys	yes	O2	7874	4152	4156	23812	23764	1	1
grain128aead	rhys	yes	O2	9532	3992	3980	30396	30124	0.8	1
isap	rhys	no	O2	3824	20212	20256	42936	43372	0.5	0.2

E. Bellini and Y. J. Huang, “Randomness testing of the nist light weight cipher finalist candidates,” in NIST Lightweight Cryptography Workshop, May, 2022.

View methods

Next Generation Crypto

Light-weight crypto.
Quantum-robust crypto
Tokenization.
Zero-knowledge.
Homomorphic Encryption.
zkSnarks, Range-proofs

Prof Bill Buchanan OBE
https://asecuritysite.com/pqc

Quantum-robust crypto

- **Lattice-based cryptography** [Lattice]—This classification shows great potential and is leading to new cryptography methods, such as for fully homomorphic encryption, and code obfuscation.
- **Code-based cryptography** [McEliece]—This classification was created in 1978 with the McEliece cryptosystem but has barely been using in real applications. The McEliece method uses linear codes that are used in error correcting codes, and involves matrix-vector multiplication. An example of a linear code is Hamming code.
- **Multivariate polynomial cryptography** [UOV]—This classification involves the difficulty of solving systems of multivariate polynomials over finite fields. Unfortunately, many of the methods that have been proposed have already been broken.
- **Hash-based signatures** [GMSS]—This classification involves creating digital signatures using hashing methods. The drawback is that a signer needs to keep a track of all of the messages that have been signed, and that there is a limit to the number of signatures that can be produced. New methods, though, integrate into Merkle Trees, which allows for the signer to use the same keys to sign multiple entities.

Quantum-robust crypto

NEWS

NIST Announces First Four Quantum-Resistant Cryptographic Algorithms

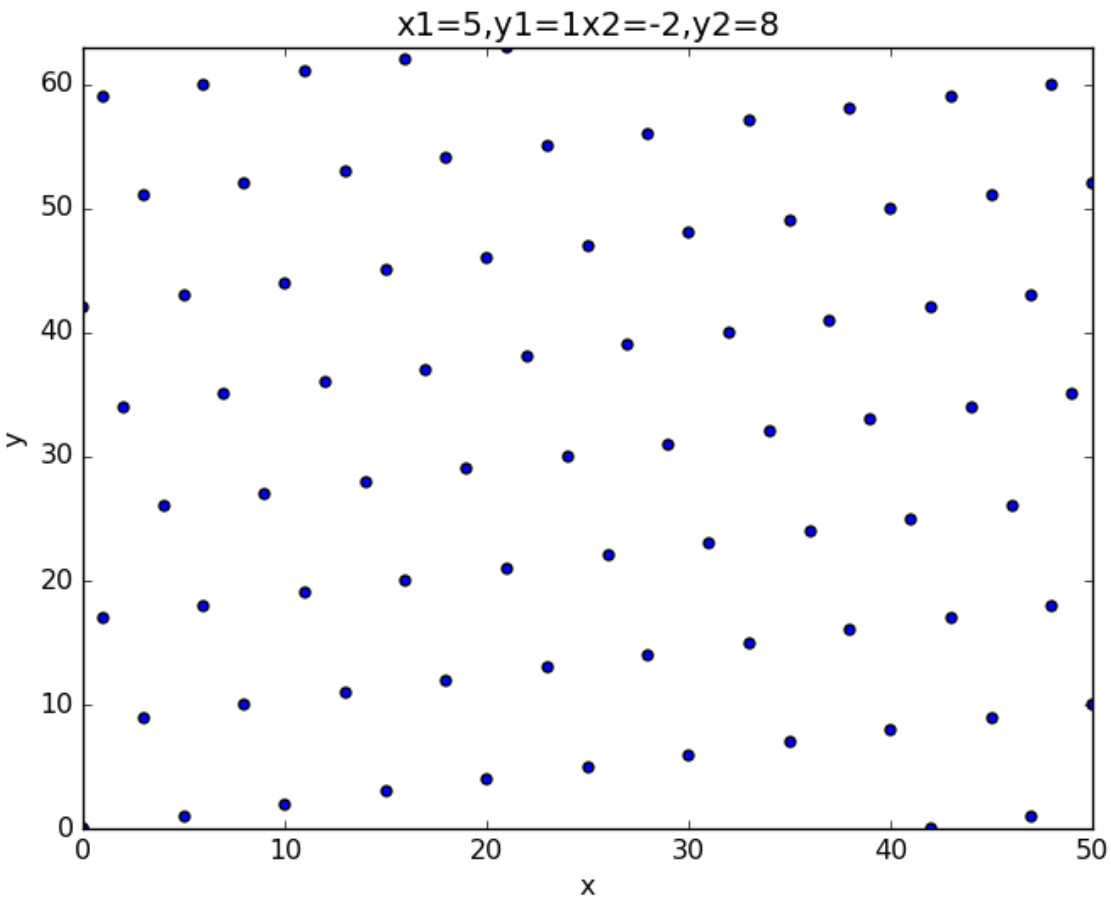
Federal agency reveals the first group of winners from its six-year competition.

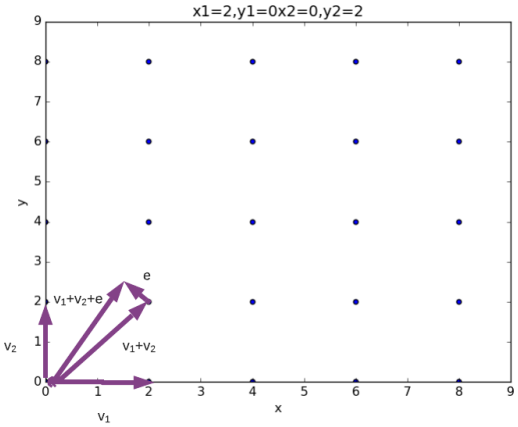
For general encryption, used when we access secure websites, NIST has selected the CRYSTALS-Kyber algorithm. Among its advantages are comparatively small encryption keys that two parties can exchange easily, as well as its speed of operation.

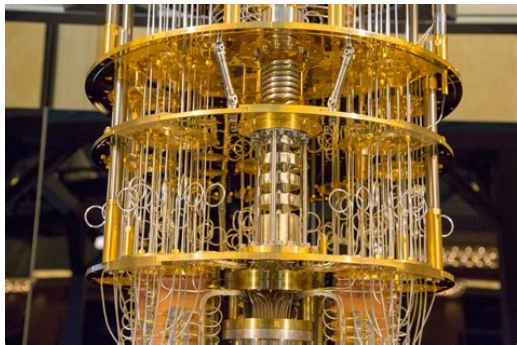
For digital signatures, often used when we need to verify identities during a digital transaction or to sign a document remotely, NIST has selected the three algorithms CRYSTALS-Dilithium , FALCON and SPHINCS+ (read as "Sphincs plus"). Reviewers noted the high efficiency of the first two, and NIST recommends CRYSTALS-Dilithium as the primary algorithm, with FALCON for applications that need smaller signatures than Dilithium can provide. The third, SPHINCS+, is somewhat larger and slower than the other two, but it is valuable as a backup for one chief reason: It is based on a different math approach than all three of NIST's other selections.

PQC Robust Crypto - Lattice

Short Vector Problem and Closest Vector Problem







PQC Robust Crypto – Lattice Methods (LWE)

This is a method defined by Oded Regev in 2005 [here] and is known as LWE (Learning With Errors). First we select a random series of values for our public key (A). For example, let's select 20 random values from 0 to 100:

```
[80, 86, 19, 62, 2, 83, 25, 47, 20, 58, 45, 15, 30, 68, 4, 13, 8, 6, 42, 92]
```

Next we add create a list (B) and were the elements are $B_i = A_i s + e_i \pmod q$, and where s is a secret value, and e is a list of small random values (the error values). If we take a prime number (q) of 97, and an error array (e) of:

```
[3, 3, 4, 1, 3, 3, 4, 4, 1, 4, 3, 3, 2, 2, 3, 2, 4, 4, 1, 3]
```

we generate a list (B) of:

```
[15, 45, 2, 20, 13, 30, 32, 45, 4, 3, 34, 78, 55, 51, 23, 67, 44, 34, 17, 75]
```

The A and B list will be our public key, and s will be our secret key. We can now distribute A and B to anyone who wants to encrypt a message for us (but keep s secret). To encrypt we take samples from the A and B lists, and take a message bit (M), and then calculate two values:

$u = \sum(A_{samples}) \pmod q$

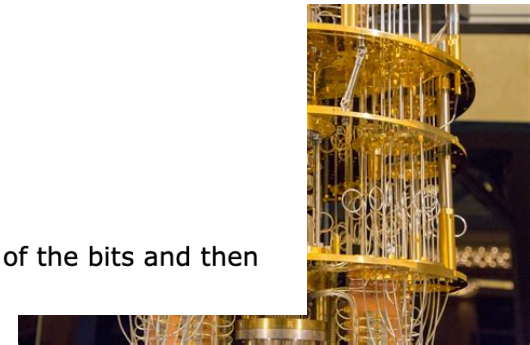
$v = \sum(B_{samples}) + \frac{q}{2}M \pmod q$

The encrypted message is (u, v). To decrypt, we calculate:

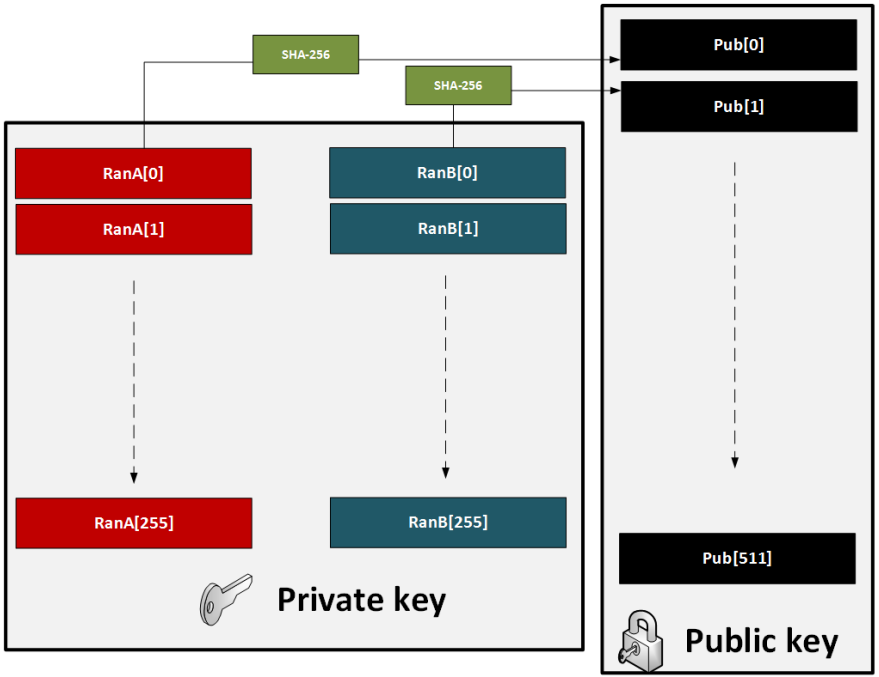
$Dec = v - su \pmod q$

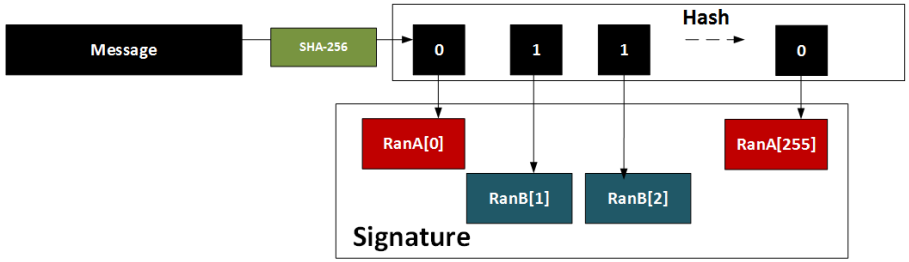
If Dec is less than $\frac{q}{2}$, the message is a zero, else it is a 1.

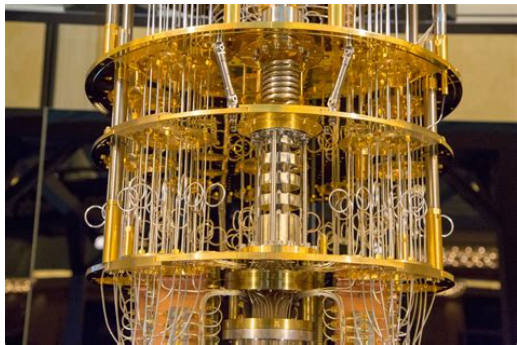
For homomophic encryption, we can perform a single-bit adder function by generating (u,v) for each of the bits and then performing (for the two values for bit_0 - $v1_0, u1_0, v2_0$ and $u2_0$:



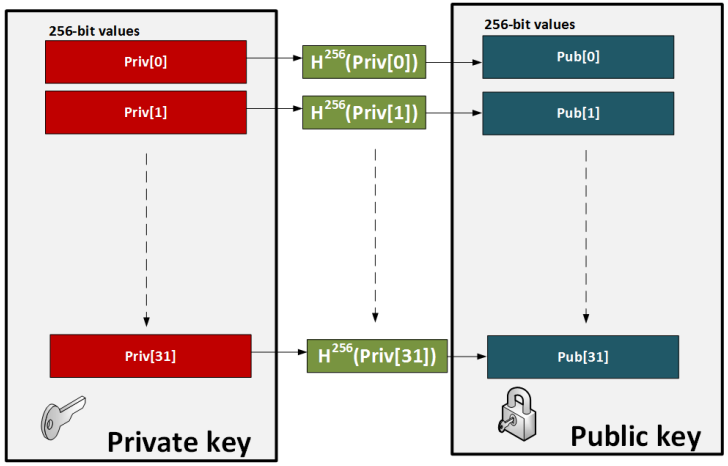
PQC Robust Crypto – Hash-based (Lamport)

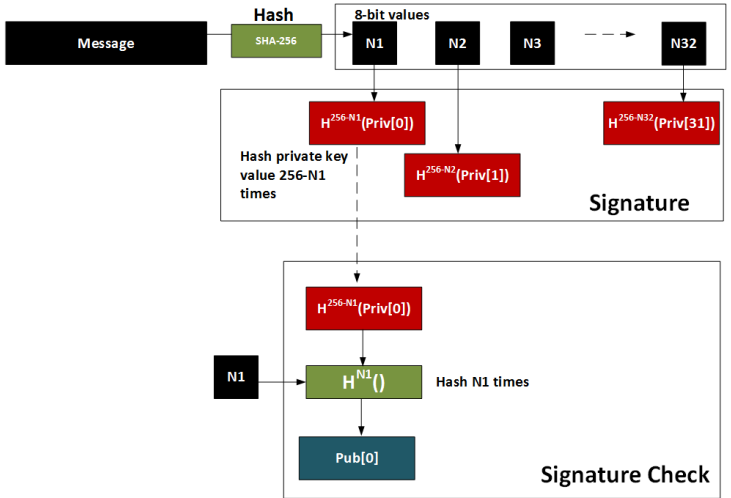


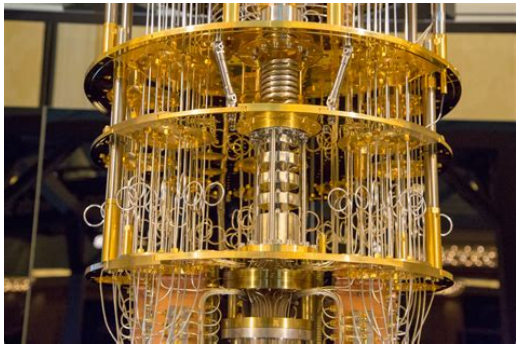




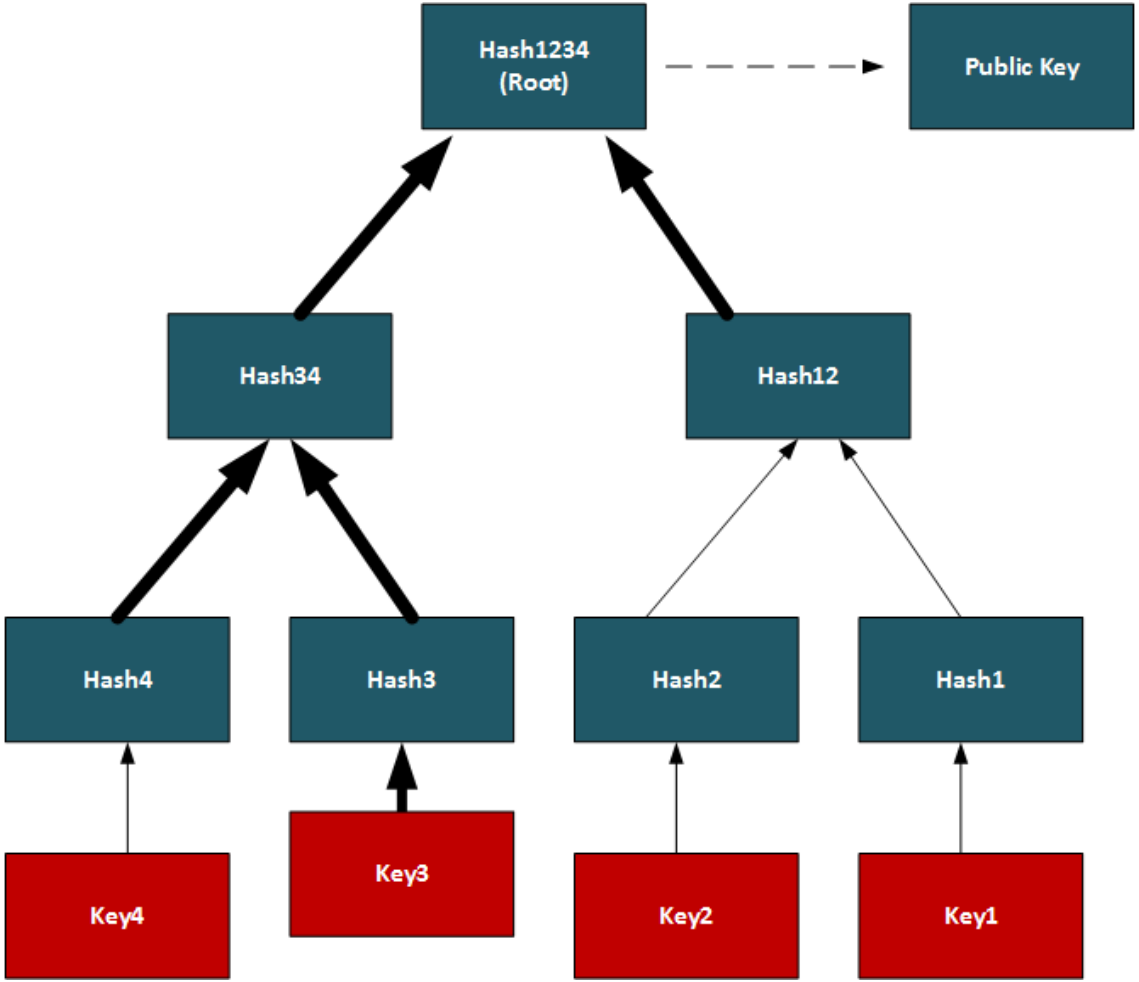
PQC Robust Crypto – Hash-based (W-OTS)

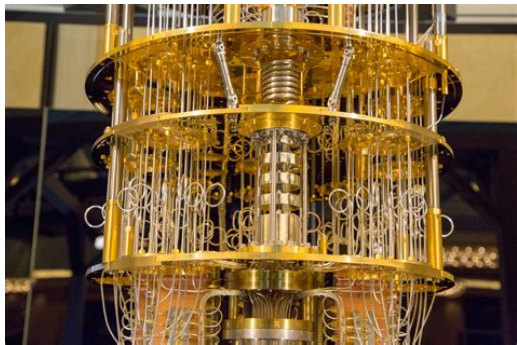






PQC Robust Crypto – Merkle signature scheme (MSS)



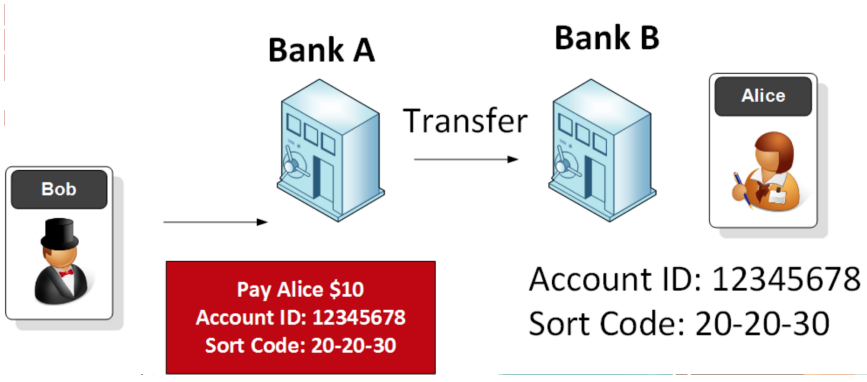


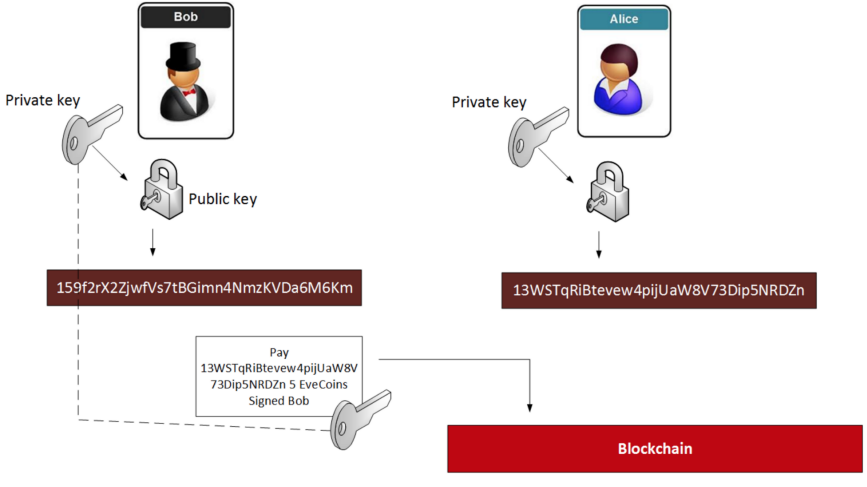
Next Generation Crypto

Light-weight crypto.
Quantum-robust crypto
Tokenization.
Zero-knowledge.
Homomorphic Encryption.
zkSnarks, Range-proofs

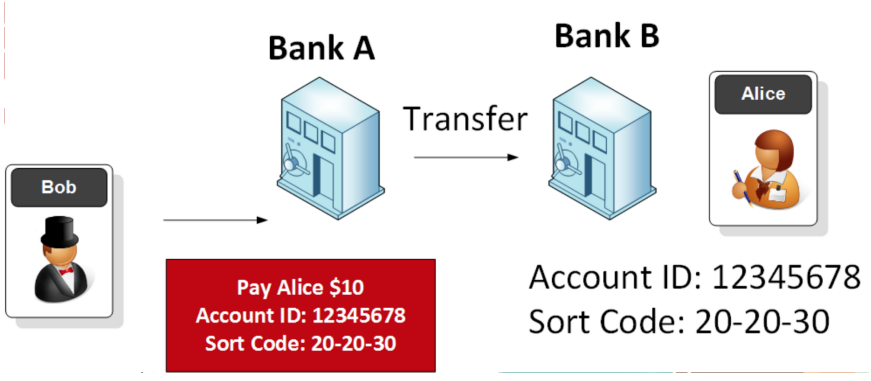
Prof Bill Buchanan OBE
https://asecuritysite.com/tokens

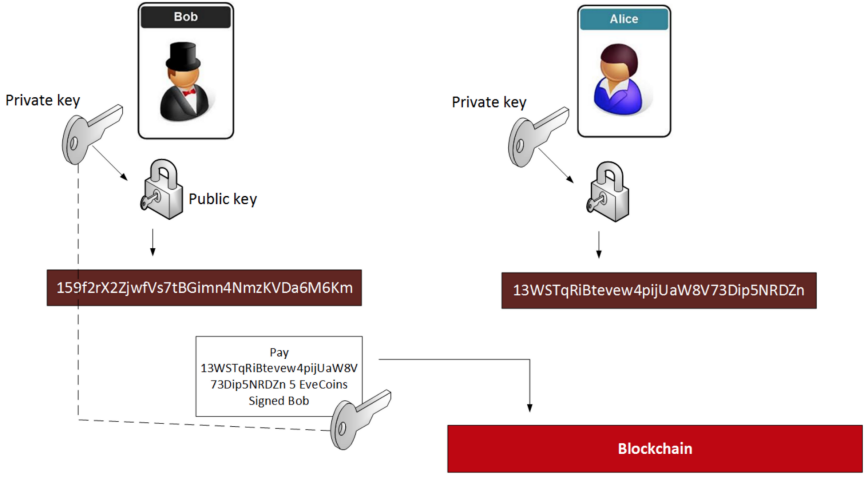
Blockchain Act



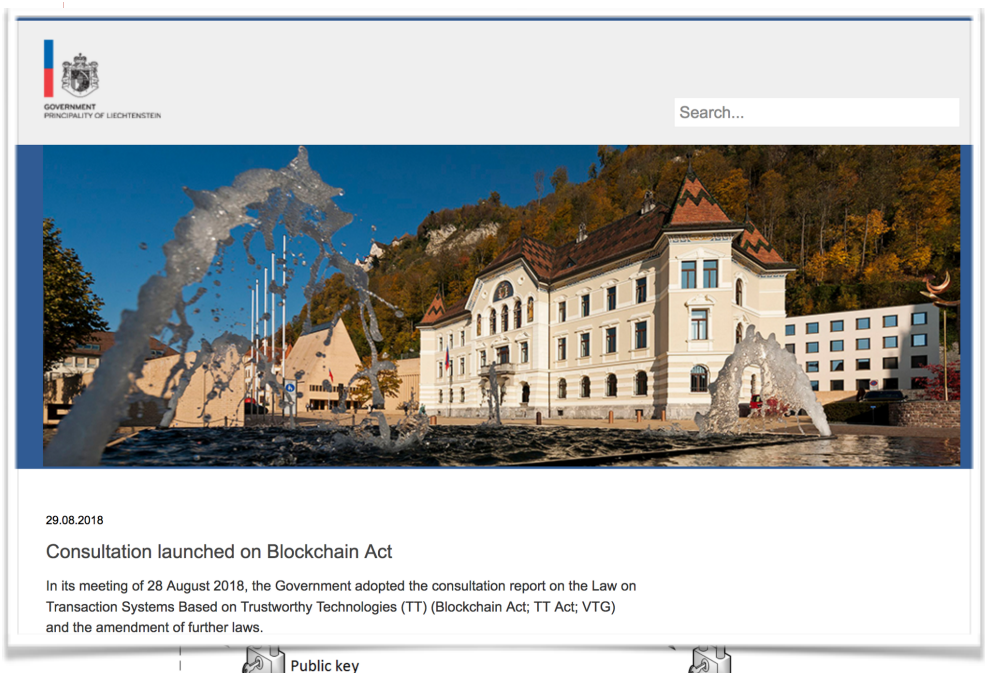


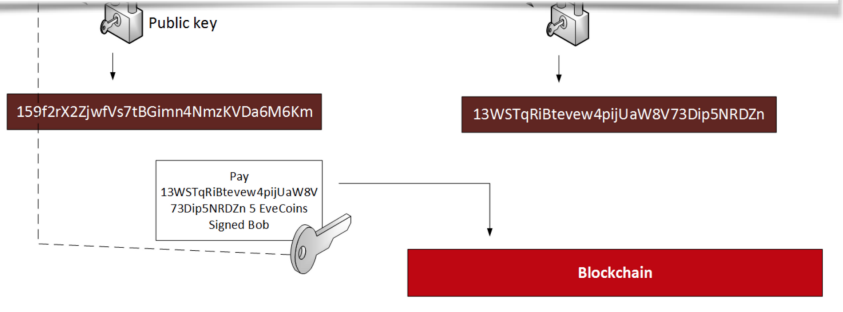
Blockchain Act



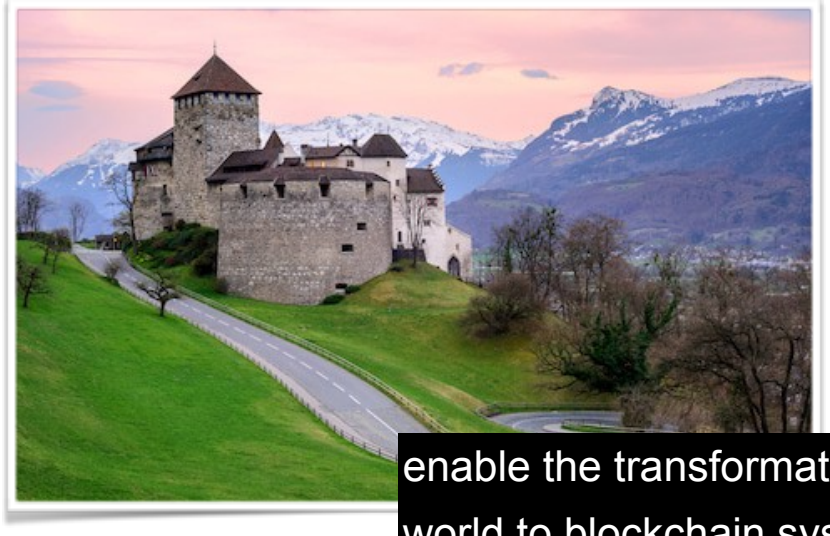


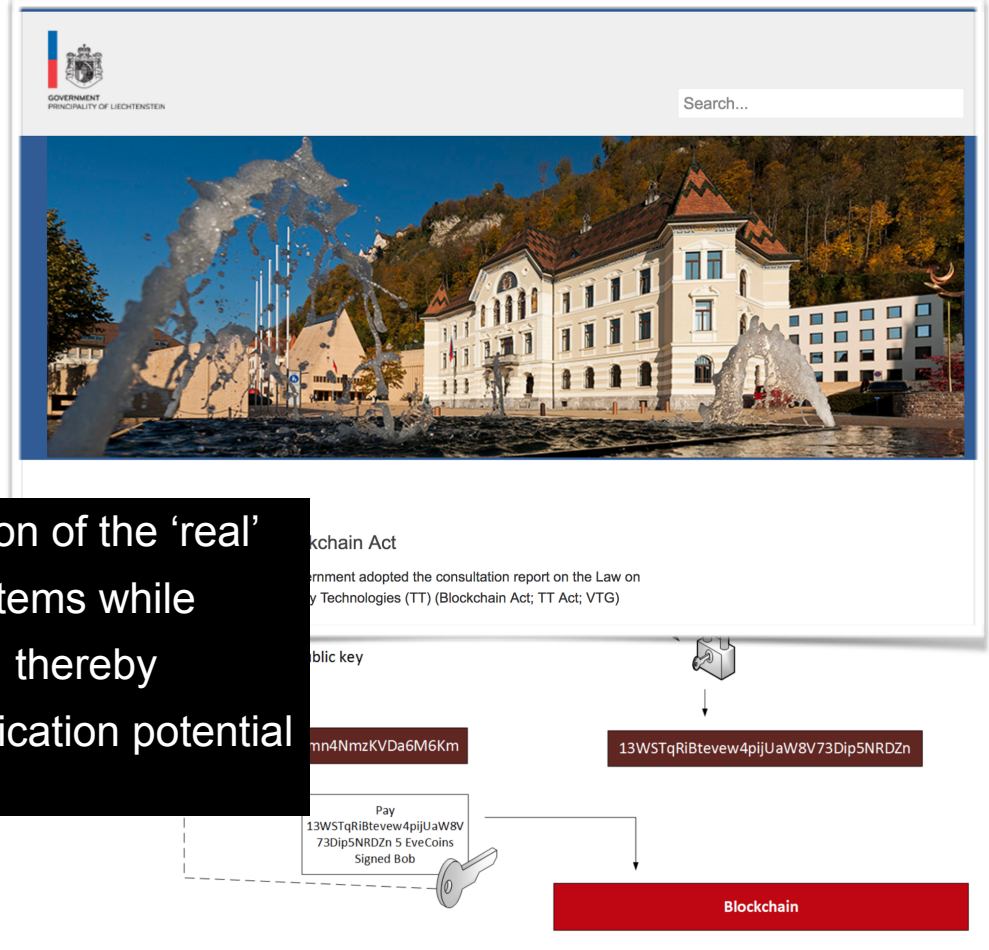
Blockchain Act





Blockchain Act

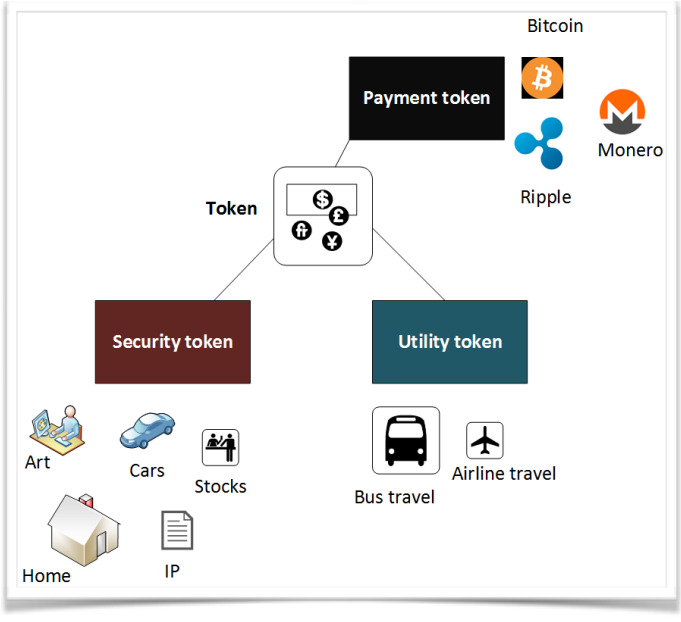




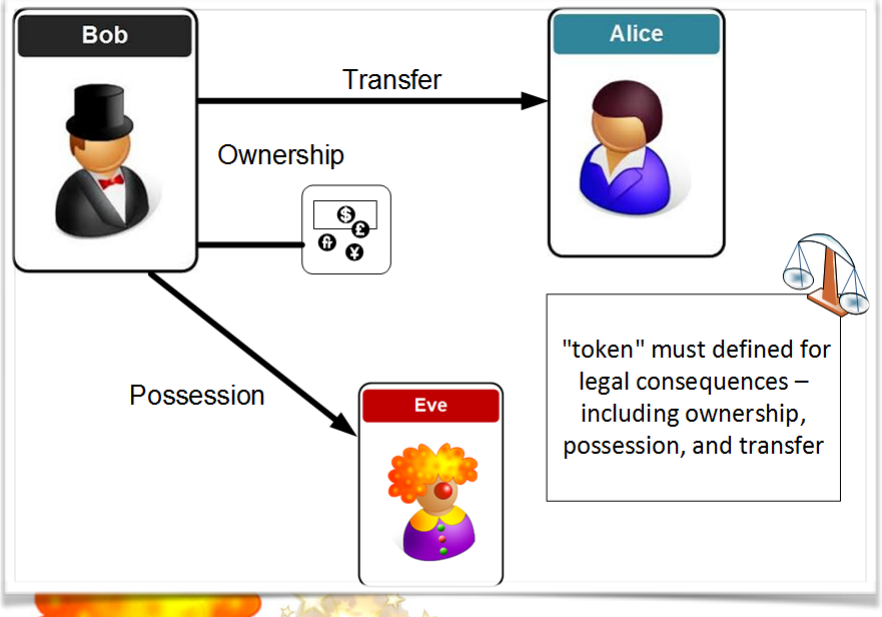
enable the transformation of the ‘real’ world to blockchain systems while ensuring legal certainty, thereby opening up the full application potential of the token economy.

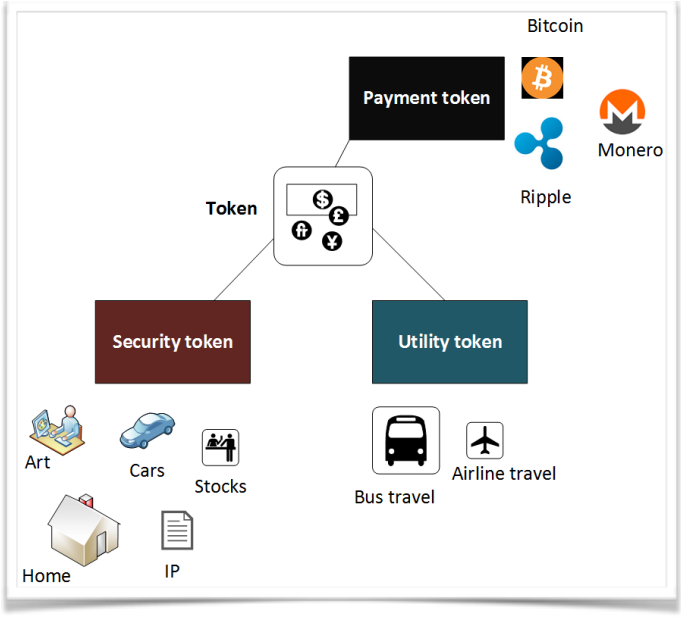
Blockchain Act

Blockchain Act

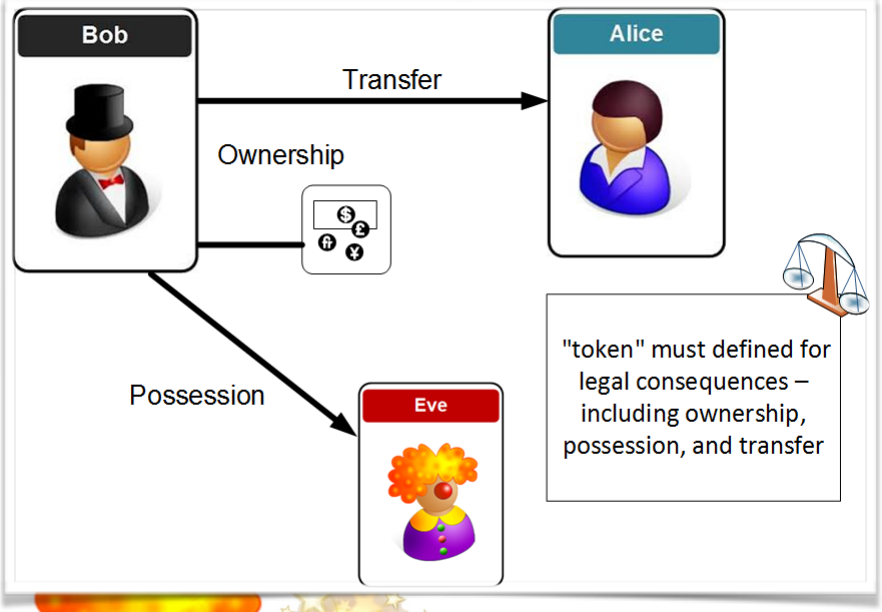


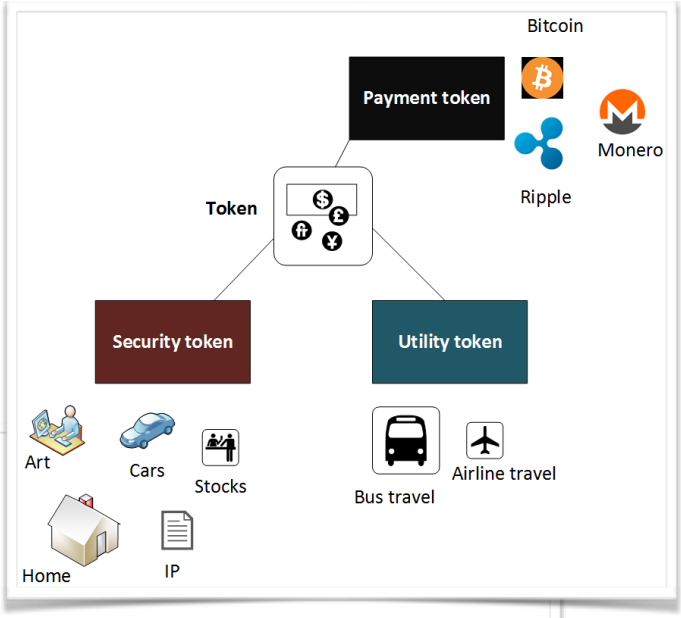
Blockchain Act

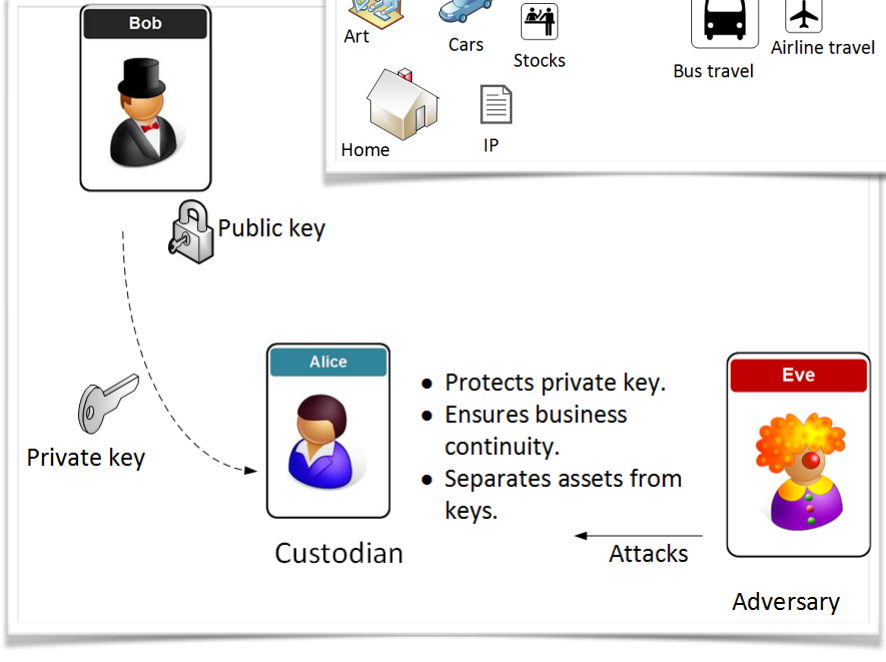




Blockchain Act







Blockchain Act

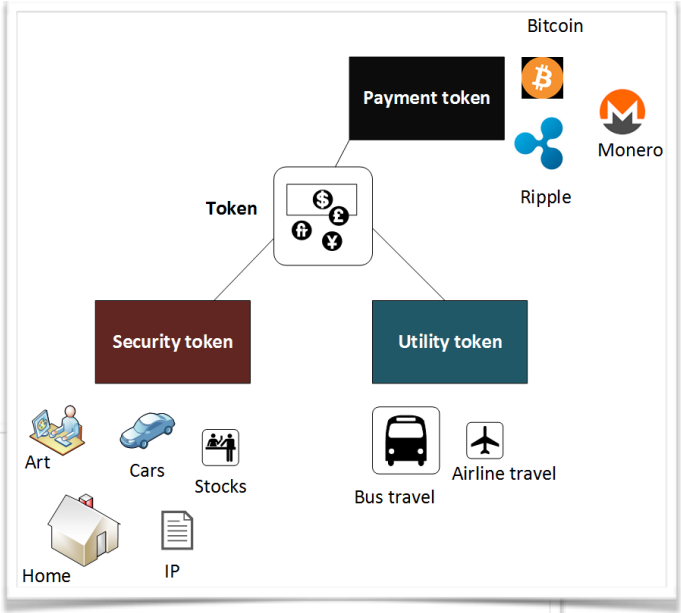
- **Definitions (Art. 5 VE-VTG):** This defines a token as something that defines claims of a person to the rights to goods.

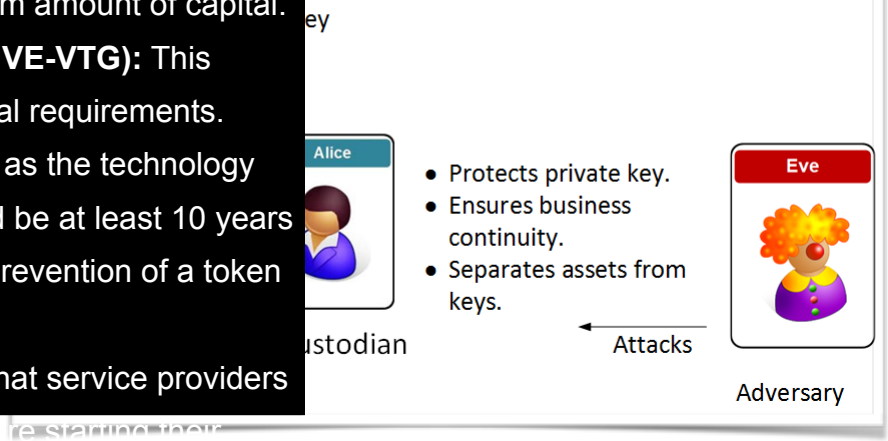
Rights of disposal (Art. 6 ff. VE-VTG): This defines the rights to transfer tokens, and is normally defined by the owner of a private key signing the transaction. A disposition is defined as the transfer of the disposition authorization on the token. Within the Act, it is defined that a buyer has the rights to dispose of a token, even if the seller was not authorized to dispose of the same token.

Requirements for VT service providers (Art. 13 ff. VE-VTG): This defines the entities who will perform services within the VT. These entities must provide an organisational structure, control mechanisms and a minimum amount of capital.

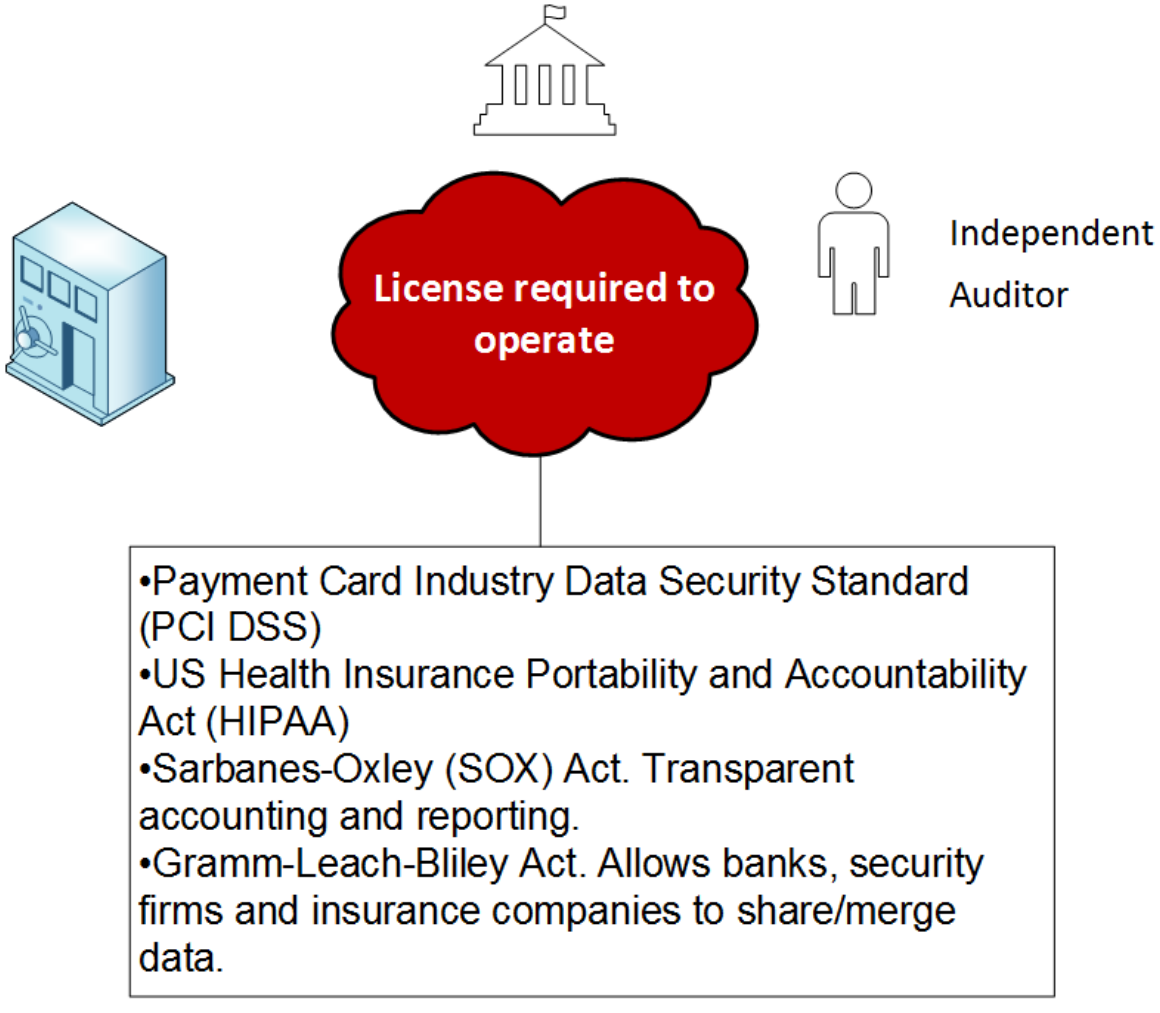
Basic information on the issuance of tokens (Art. 28 ff. VE-VTG): This defines the assurance in the issuing of tokens and their legal requirements. They must provide a minimum amount of information, such as the technology used, the purpose of the token, and any risks. There should be at least 10 years of issuance, and to also prevent token cloning, along with prevention of a token not being released with the same rights.

Obligation to register (Art. 36 ff. VE-VTG): This defines that service providers

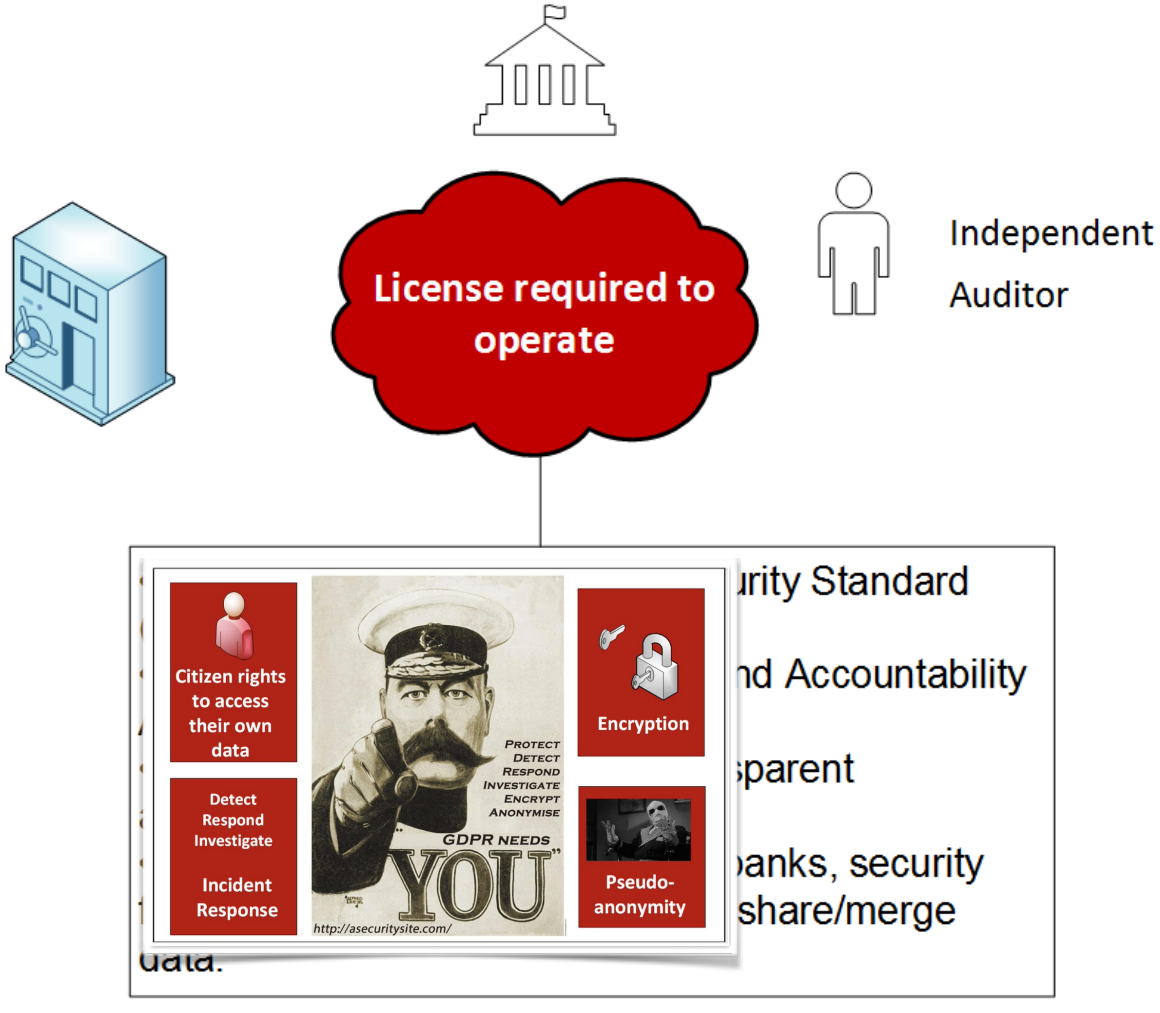




Audit Compliance



Audit Compliance



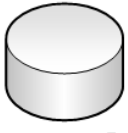
Surrogate identifiers

Personally Identifiable Information (PII)

PAN – Primary Account Number

ID=543 611 041

Name: Bobby Smith
Address: 10 Eve Row
Date of Birth: 5/5/55



Surrogate mapping table

Real	Surrogate
ID=543 611 041	ID=741 534 011
ID=533 841 943	ID= 666 001 845

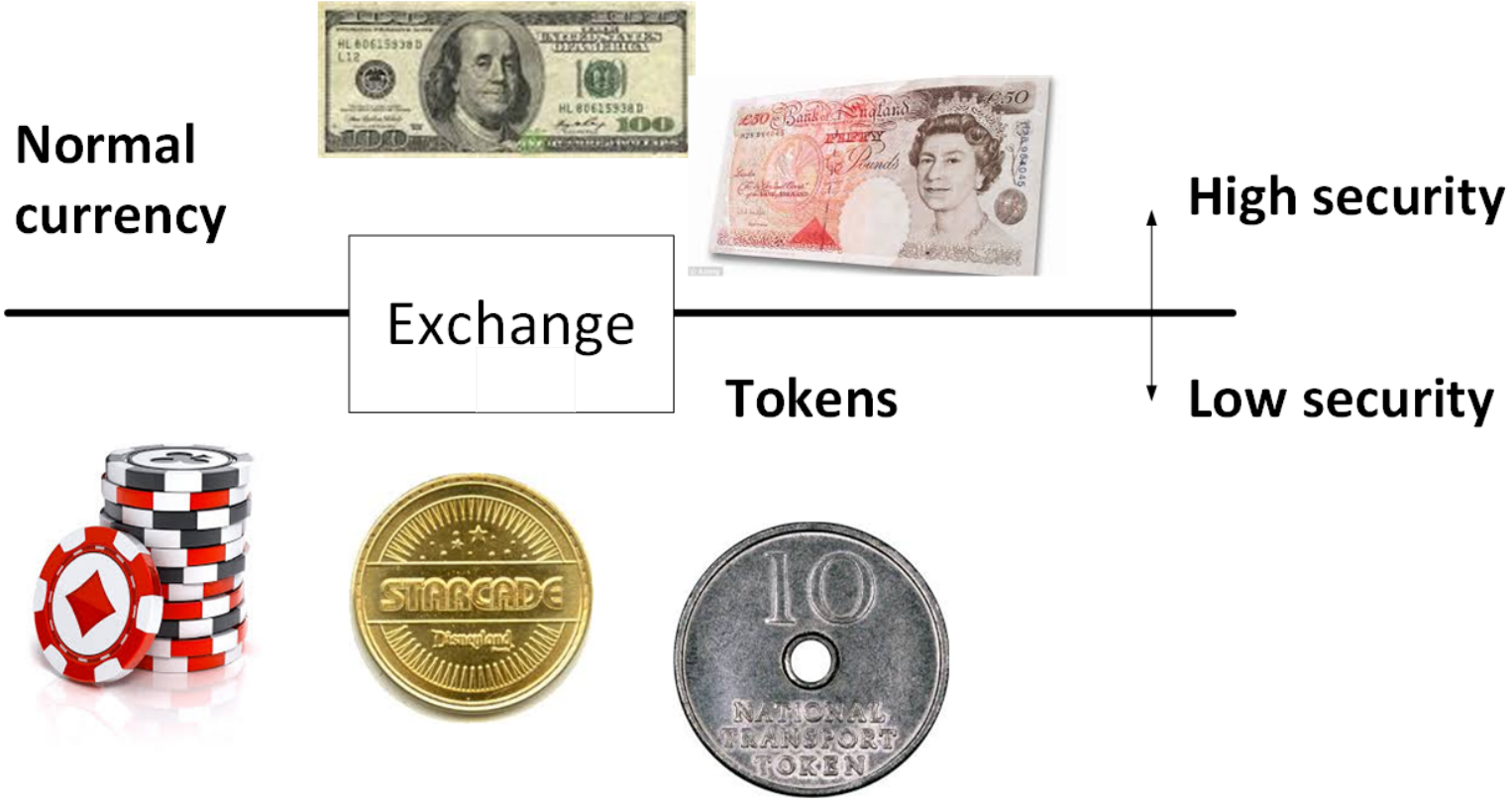
Transactions

Surrogate Identifier

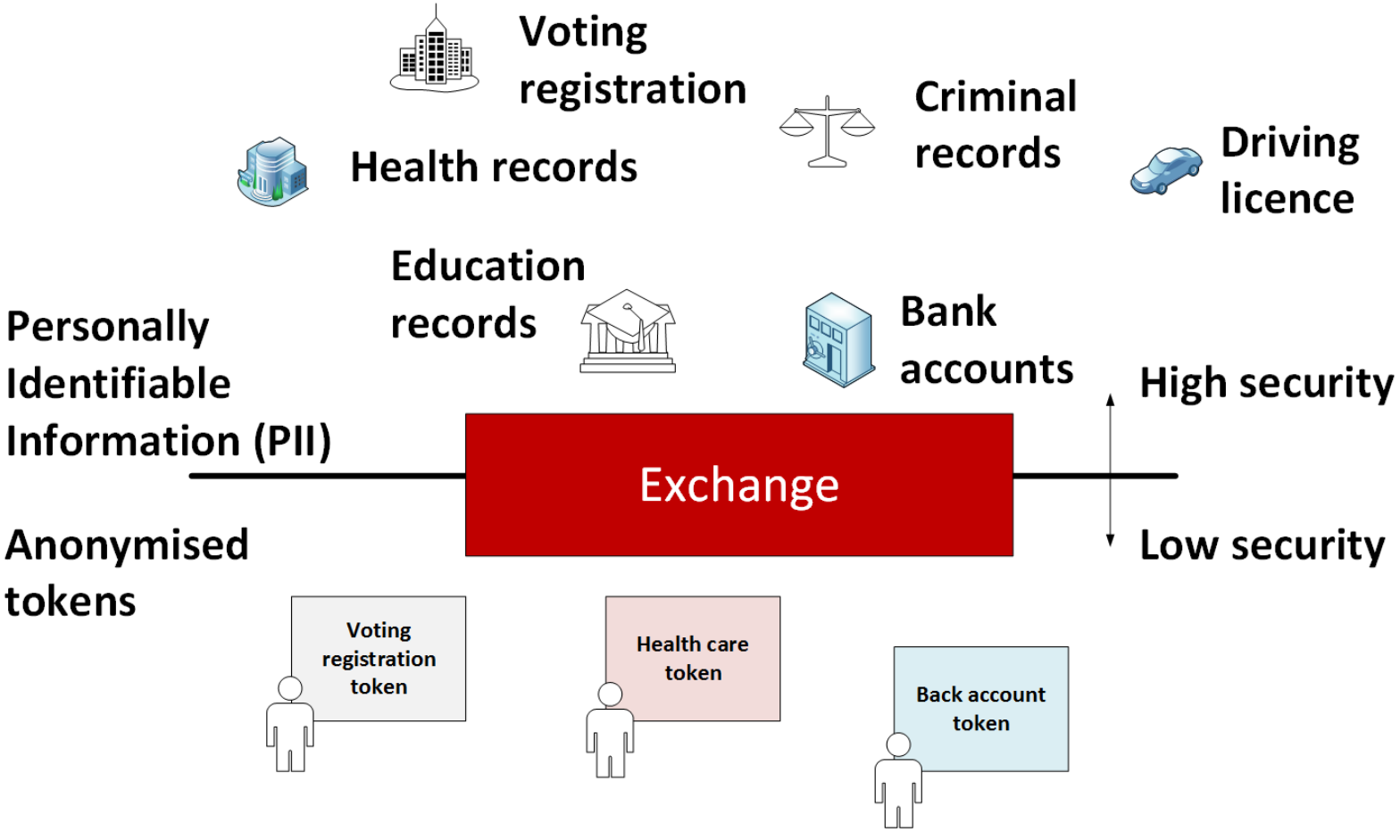
ID=741 534 011

ID	Transaction
741 534 001	Pay 666 001 845 $10
532 550 423	Pay 741 534 011 $190

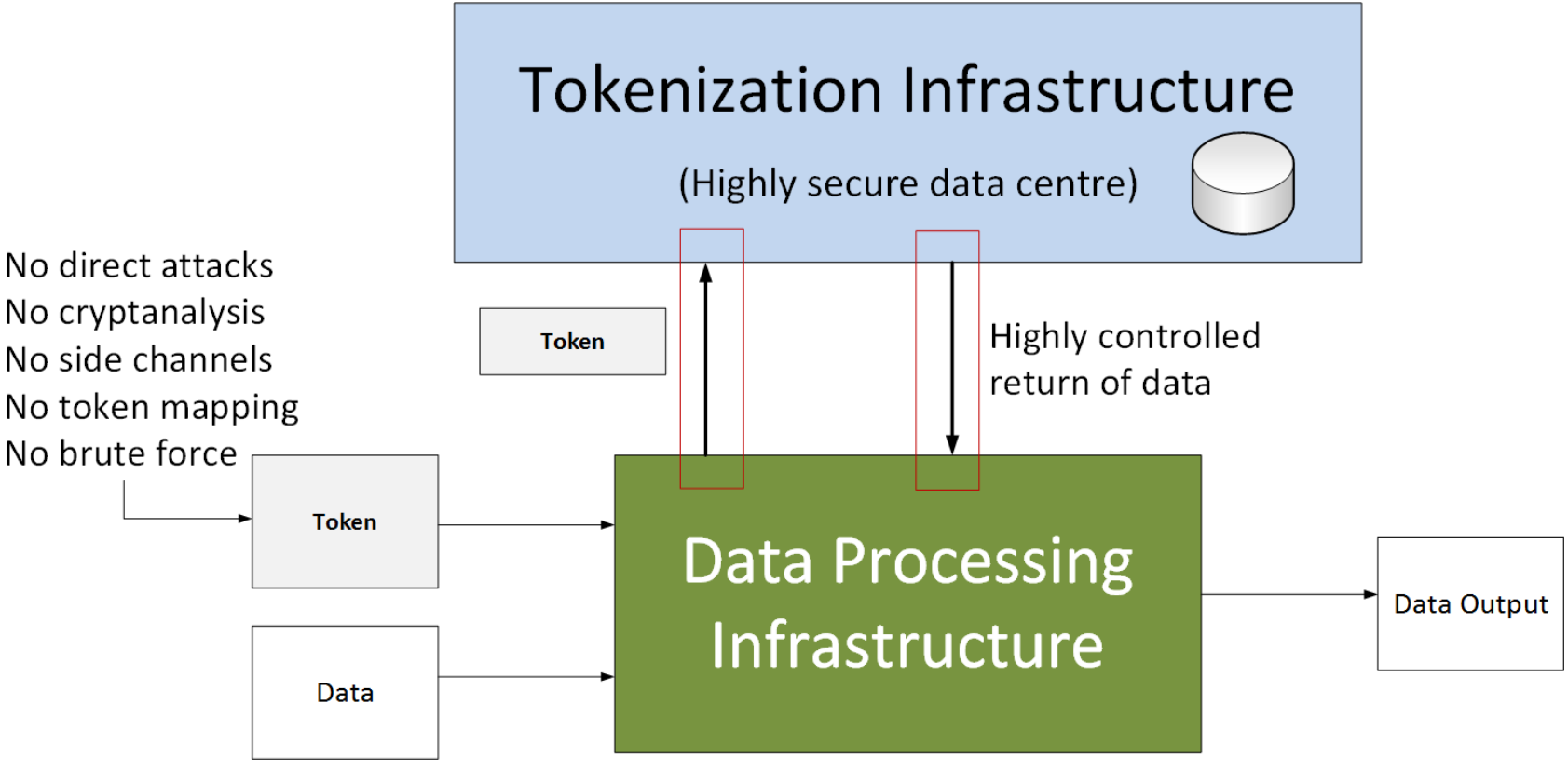
Tokenization with currency



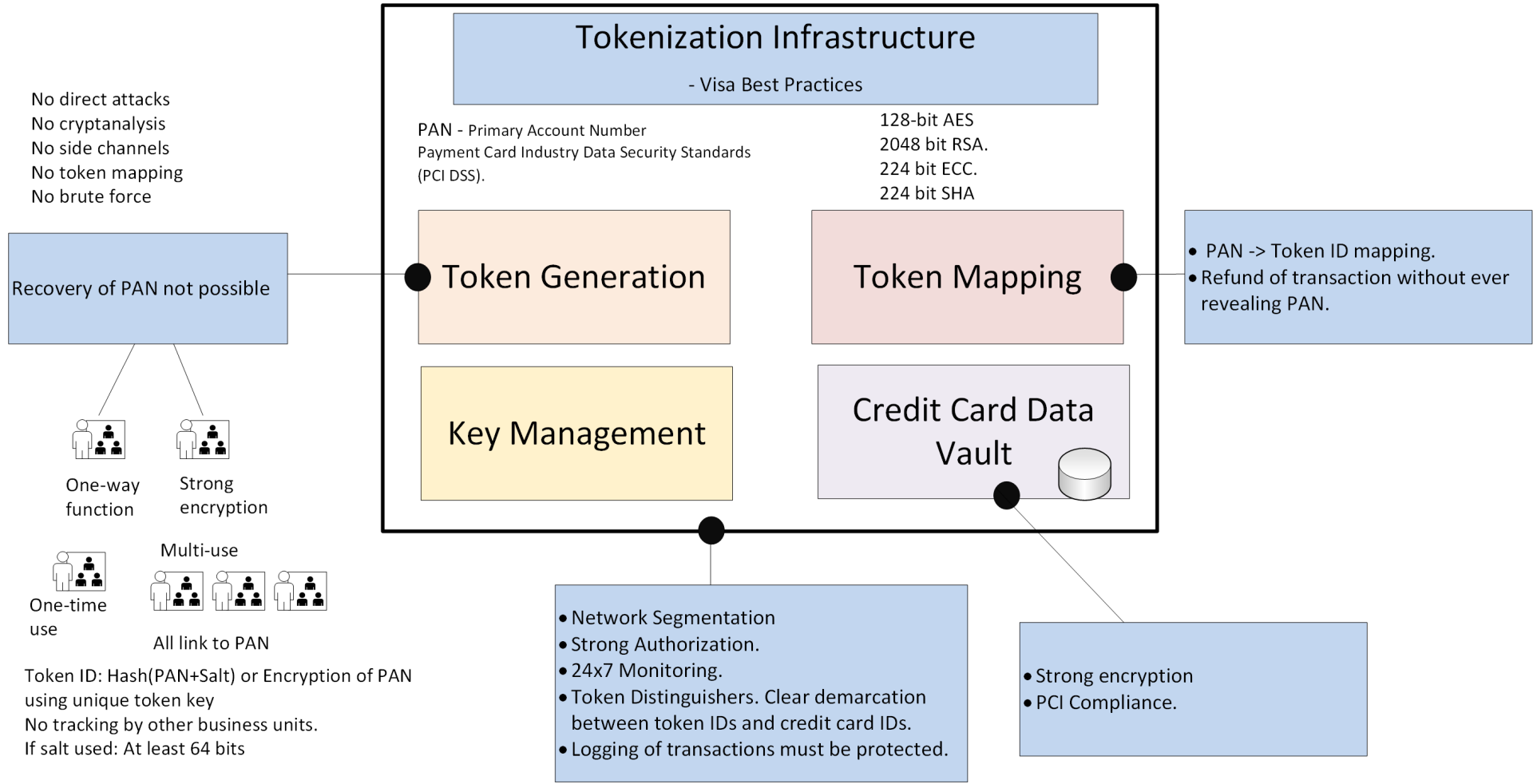
Tokenization with data



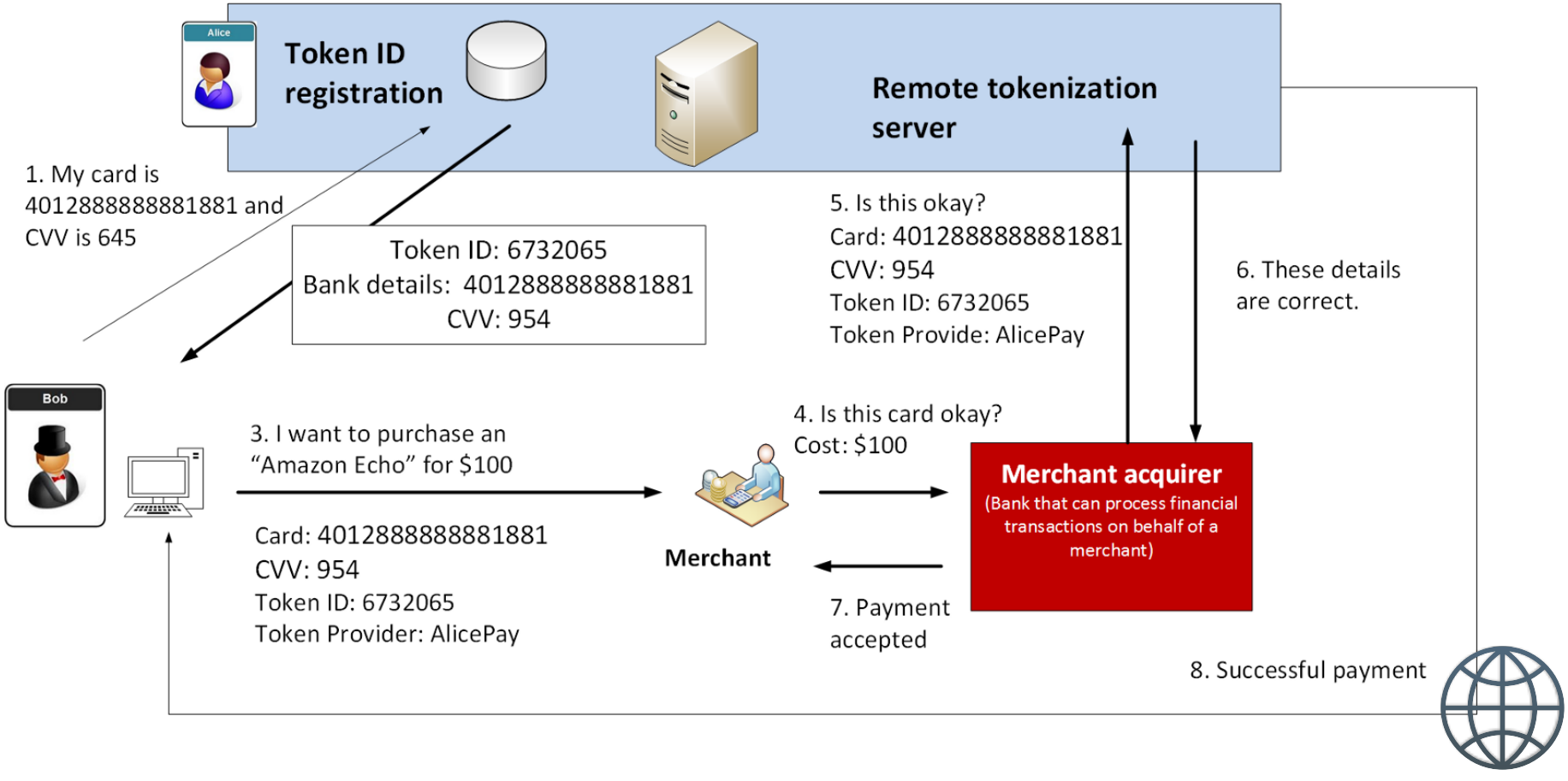
Tokenization with data



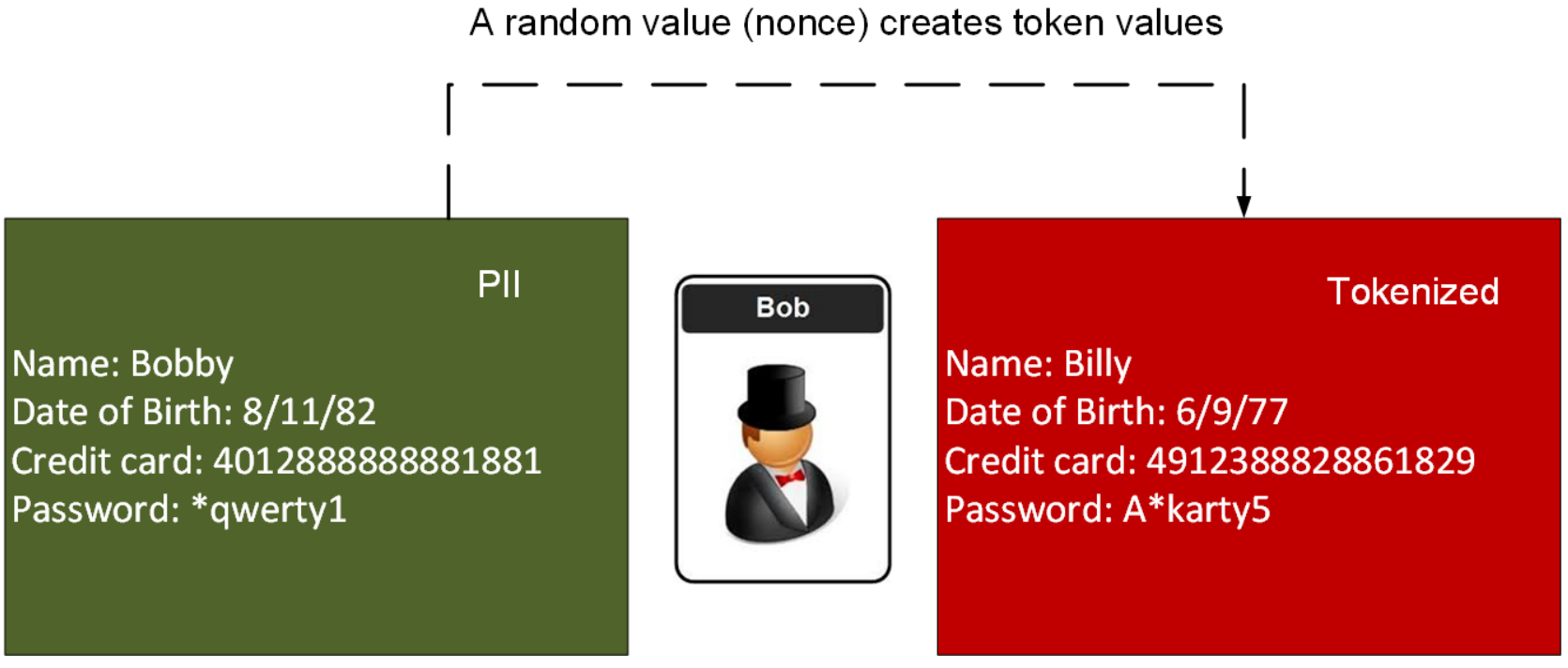
Visa Best Practice for Tokenization



Token Mapping



Token Mapping



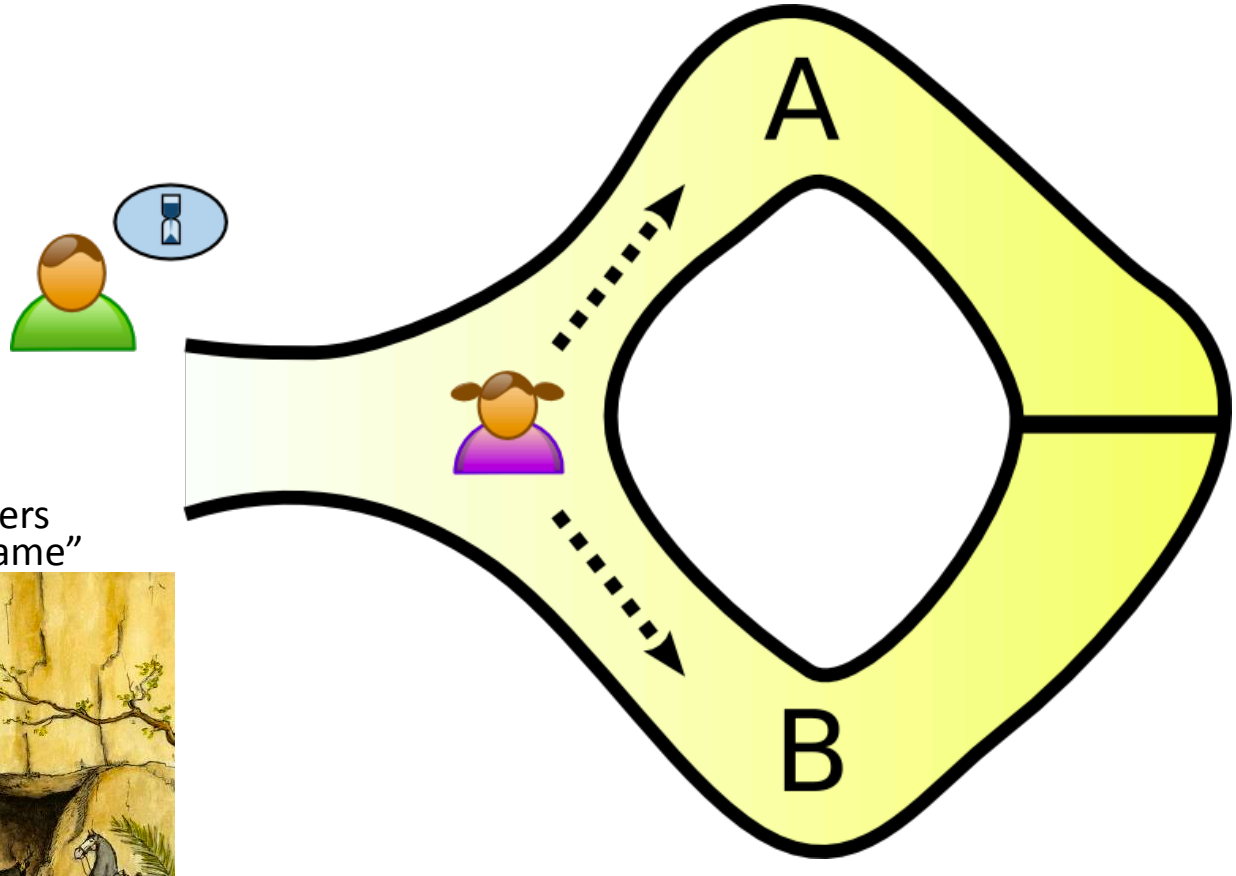
Next Generation Crypto

Light-weight crypto.
Quantum-robust crypto
Tokenization.
Zero-knowledge.
Homomorphic Encryption.
zkSnarks, Range-proofs

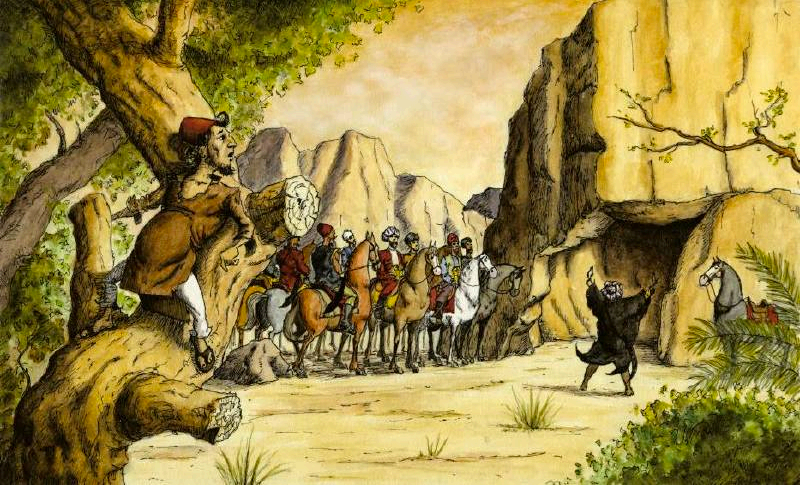
Prof Bill Buchanan OBE
http://asecuritysite.com/zero

Zero-knowledge Proof

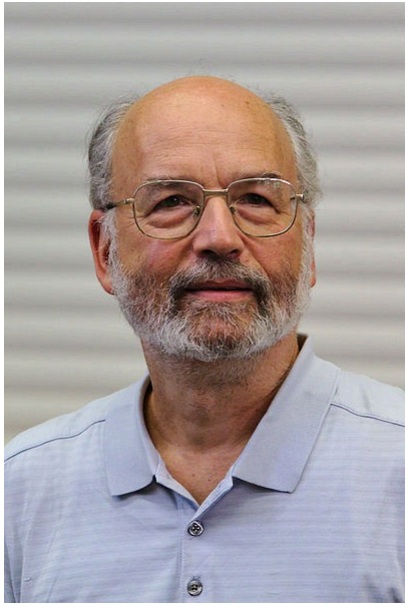
- Peggy is the prover.
- Victor is the verifier.



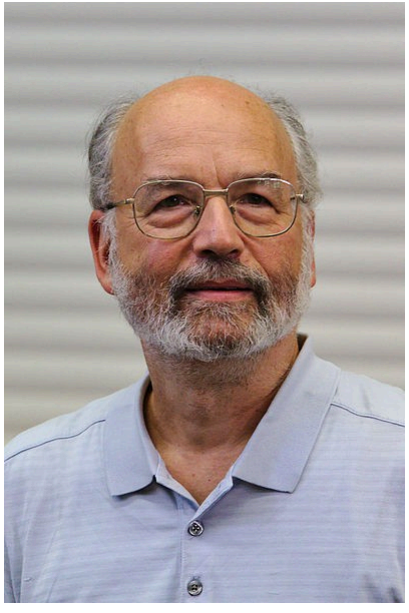
Ali Baba - poor woodcutter - discovers the secret of a thieves as “open sesame”



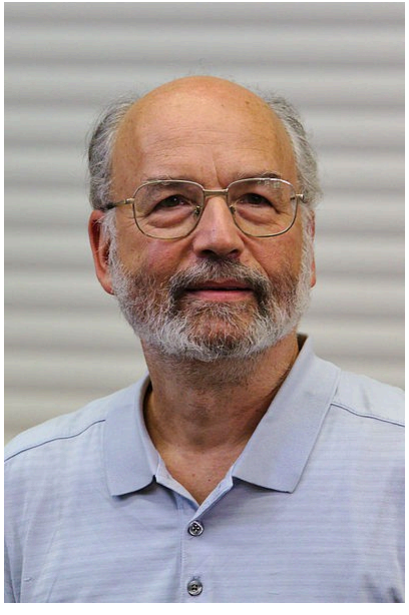
Zero Knowledge Proof: Fiat-Shamir

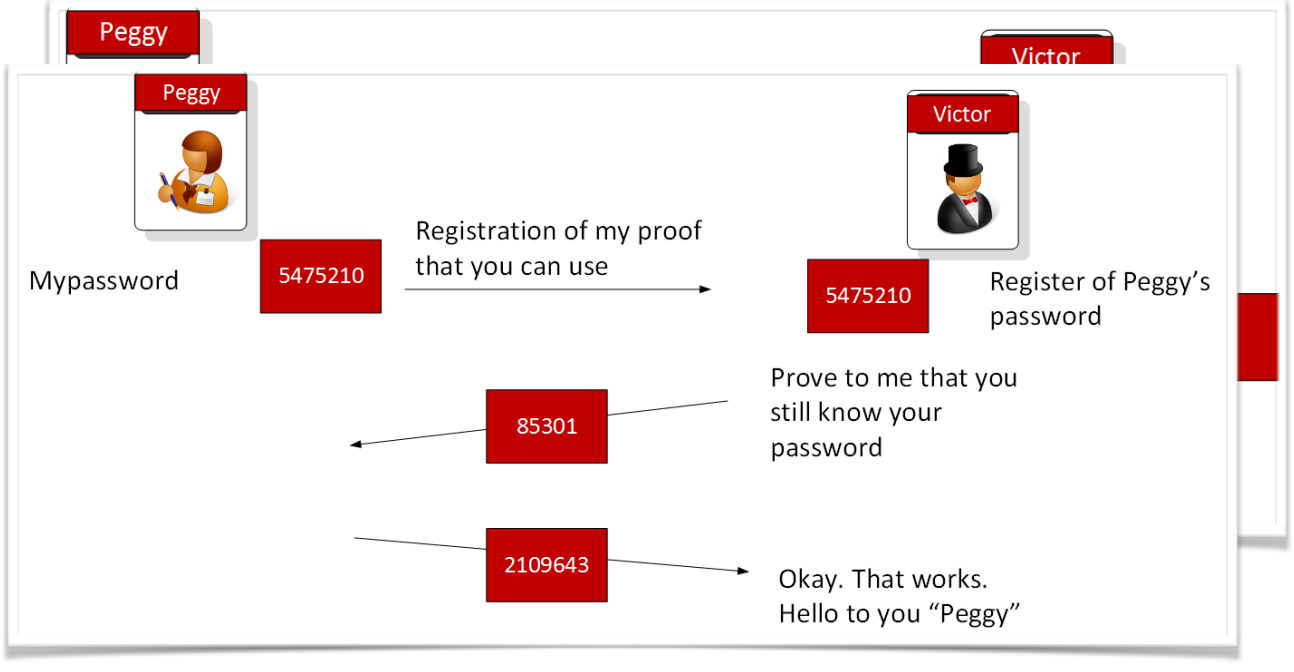


Zero Knowledge Proof: Fiat-Shamir

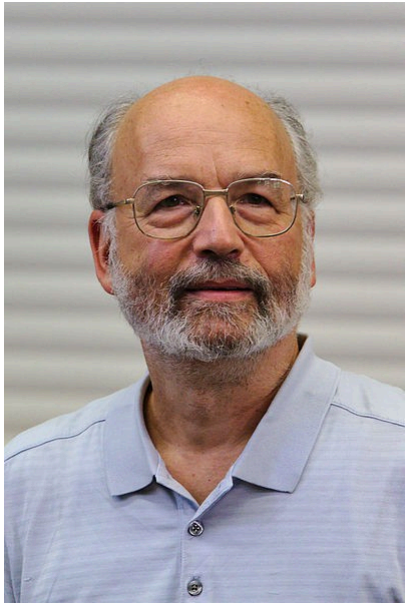


Zero Knowledge Proof: Fiat-Shamir





Zero Knowledge Proof: Fiat-Shamir



Peggy

Victor

pass= Mypassword
x = Hash(pass)
$y = g^x$

$y = g^x$

$y = g^x$

Peggy select *v*

$t = g^v$

Victor generates a random value (c)

c

$r = v - c\,x$

r

Victor computes:
$g^r y^c$

If same as t, Peggy is verified

Bob proves and Alice verifies - Non-interactive random oracle access

Bob is the prover and knows the value of *x*

Bob and Alice agree on G and p

Alice is the verifier

Shared Secret: $y = G^x$

Prove to me you still know *x*!

Random value: *v*

Commitment: $t=g^v$

Challenge: c =Hash(g,y,t)

Response: r =v – c x (mod p)

$$g^r y^c = g^{v-cx} y^c = g^{v-cx} (g^x)^c = g^{v-cx+cv} = t$$

(r,c)

Check $t' = g^r y^c$

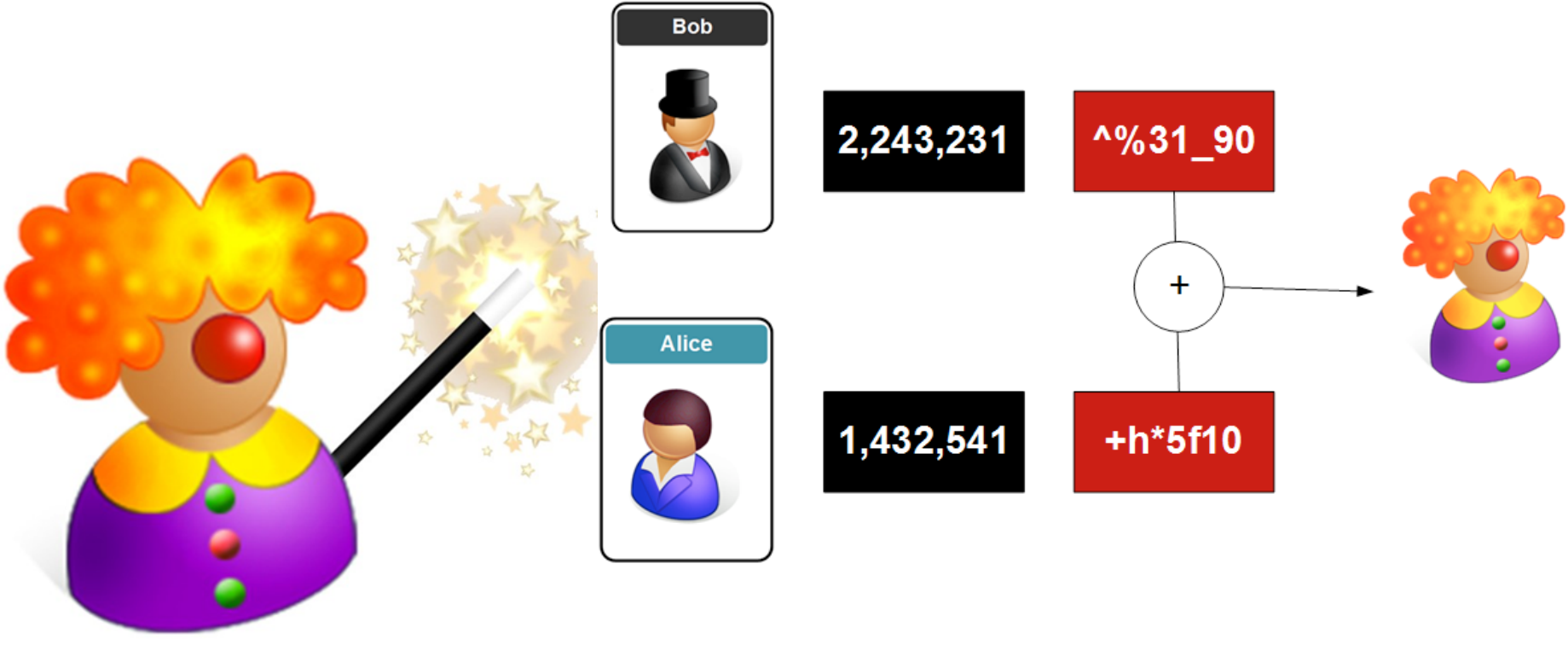
Recheck c = H(g,y,t')

Next Generation Crypto

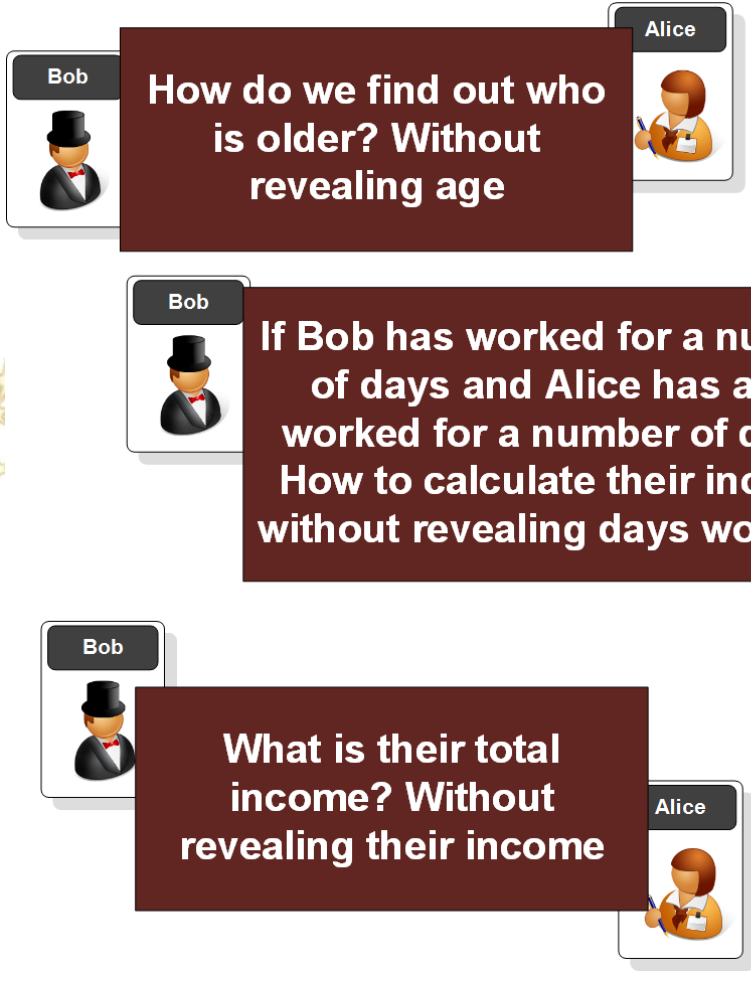
Light-weight crypto.
Quantum-robust crypto
Tokenization.
Zero-knowledge.
Homomorphic Encryption.
zkSnarks, Range-proofs

Prof Bill Buchanan OBE
http://asecuritysite.com/homomorphic

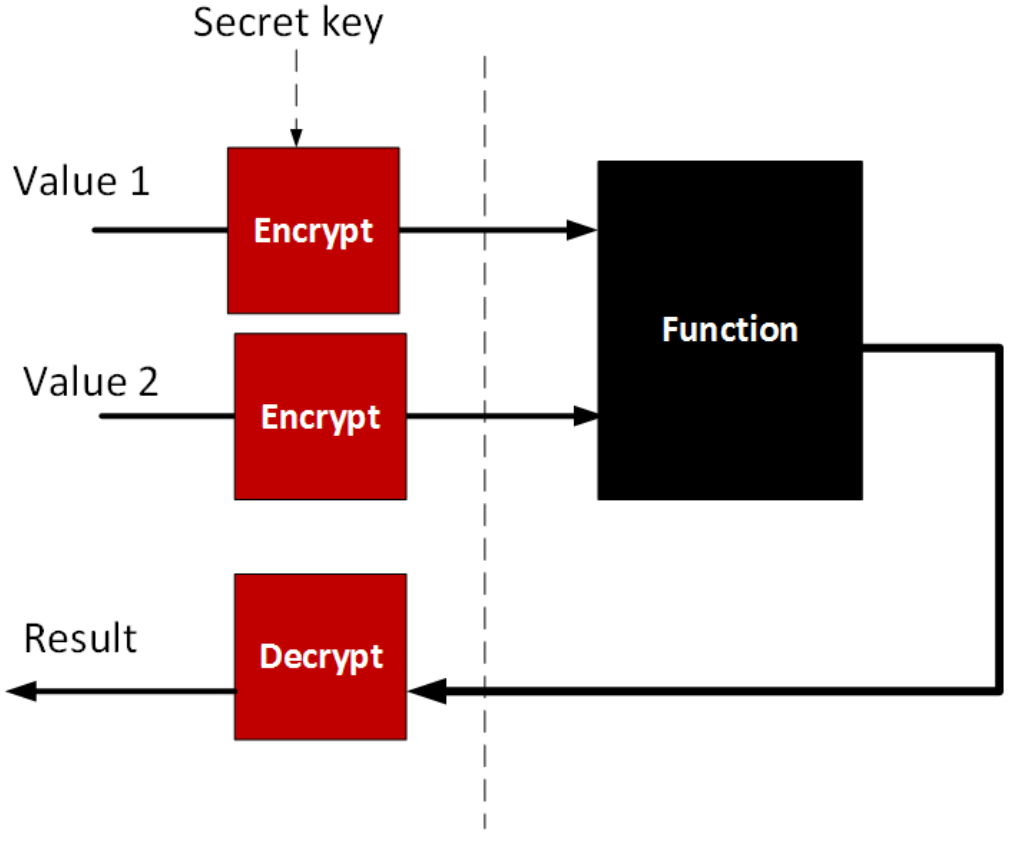
Homomorphic Encryption



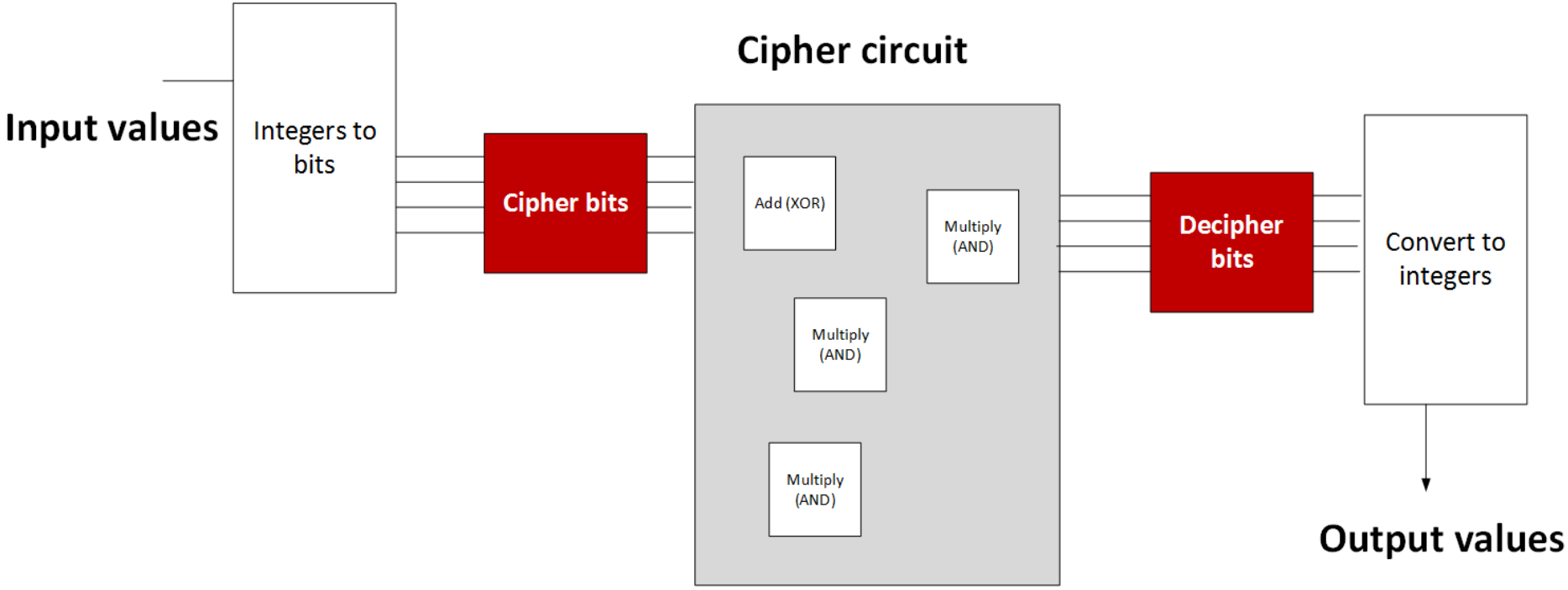
Homomorphic Encryption



Full Homomorphic Encryption - DGHV (Dijk, Gentry, Halevi and Vaikuntanathan



Full Homomorphic Encryption - DGHV (Dijk, Gentry, Halevi and Vaikuntanathan



Full Homomorphic Encryption - DGHV (Dijk, Gentry, Halevi and Vaikuntanathan

$$c = p \times q + 2 \times r + m$$

$$d = (c \mod p) \mod 2$$

Next Generation Crypto

Light-weight crypto.
Quantum-robust crypto
Tokenization.
Zero-knowledge.
Homomorphic Encryption.
zkSnarks, Range-proofs

Prof Bill Buchanan OBE
http://asecuritysite.com/zero

zCash

Benchmark	Time in seconds	Min	Max	Change	Trend
time createjoinsplit	44.64126	44.61028	44.98179	−0.65	−4.48%
time parameterloading	2.64027	2.63953	2.70091	−0.16	−2.60%
time solveequihash	30.78956	14.24525	107.95399	−32.47	−19.73%
time solveequihash 2 threads	49.32094	15.41534	115.41706	−0.46	−3.35%
time validatelargetx	0.60553	0.60509	0.60841	−0.06	0.10%
time verifyequihash	0.00155	0.00154	0.00371	−0.58	0.09%
time verifyjoinsplit	0.02776	0.02775	0.05627	−1.38	−4.41%
Average				−5.11	−4.91%

Zcash uses ZKP

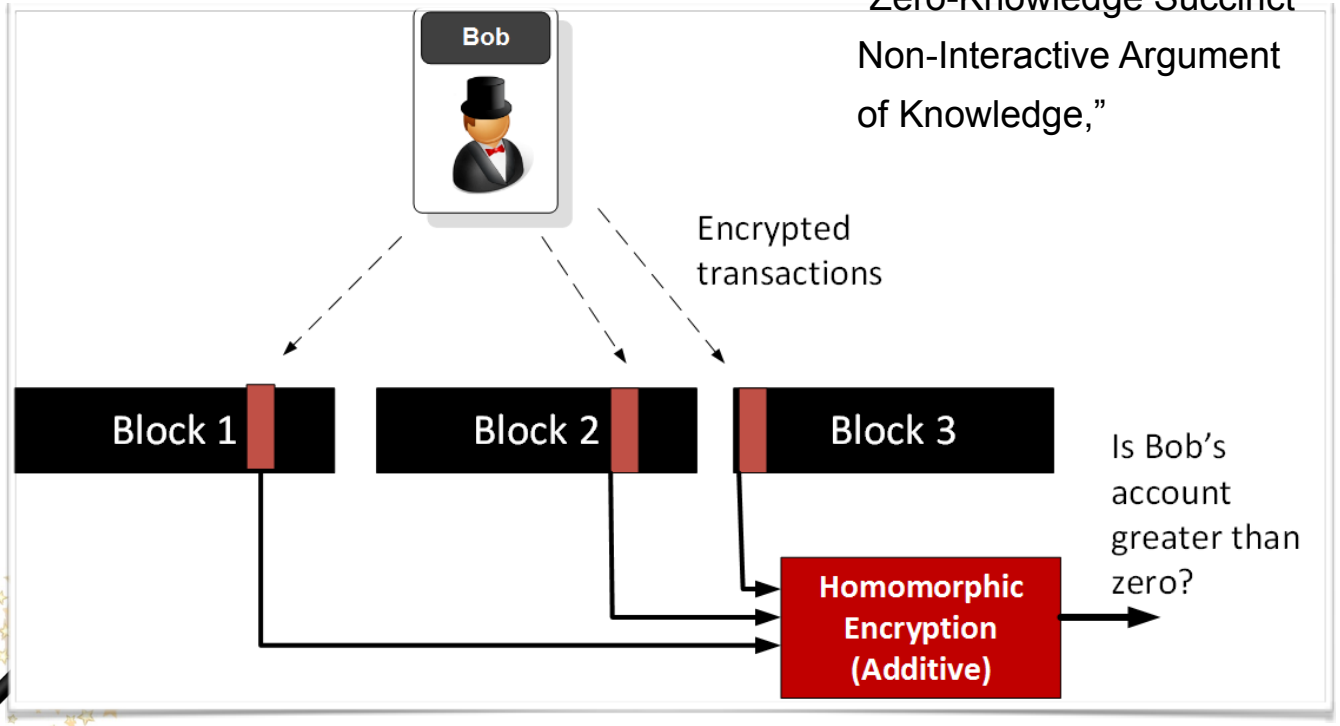
Humans that were at fault with a new Zcoin hack (Zcash) and which involved making transactions of £561,000 ($699,000), and with an associated profit of £349,000 ($435,000).

It happened with a single additional character in the source code, and allowed the creation of a Zerocoin spend transaction, and which was able to extract 370,000 Zcoins.

zk-SNARK stands for “Zero-Knowledge Succinct Non-Interactive Argument of Knowledge,”

Zero-knowledge Proofs

zk-SNARK stands for "Zero-Knowledge Succinct Non-Interactive Argument of Knowledge,"



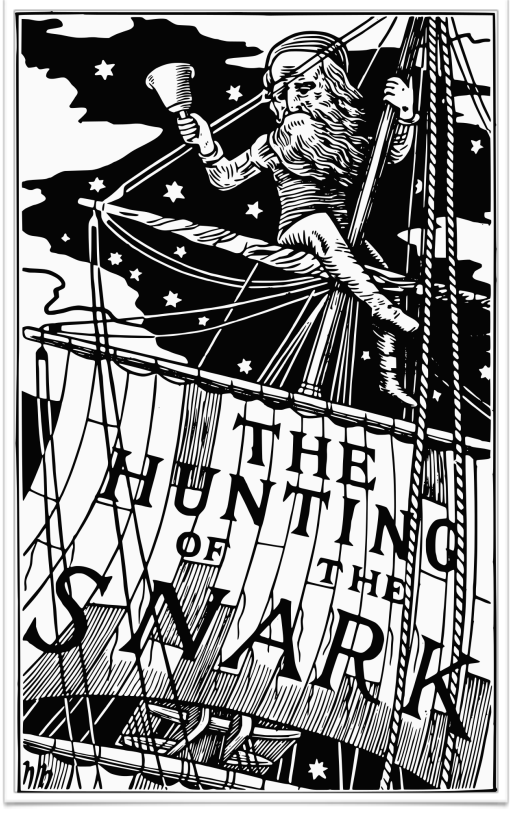
Pal Link

SPIRITUS PARTNERS

THE DATA LAB

NHS National Services Scotland

zkSNARK



Hidden Homomorphic

With HH we aim to prove that Alice knows two values (x, y) to equal a given answer (ans)

Let's say you want to prove that
Alice can produce two number
which adds up to 8. Overcomes the
major problem of Blockchain -
which is the lack of privacy in
computations and transac

Blind Evaluation

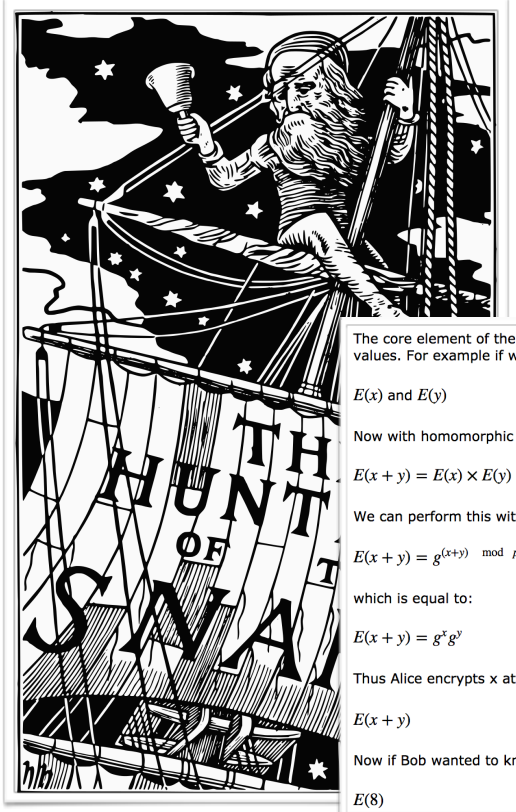
Bob doesn't want Alice to known the values that he is using and Alice doesn't want Bob to know the method she is using to compute a result

zkSNARK

Hidden Homomorphic

With HH we aim to prove that Alice knows two values (x, y) to equal a given answer (ans)

Let's say you want to prove that Alice can produce two number which adds up to 8. Overcomes the major problem of Blockchain -



The core element of the method is the usage of HH (Homomorphic Hiding), and which allows us to perform a mathematical operation on encrypted values. For example if we have a value x and a value y, and we want to encrypt them we get:

$E(x)$ and $E(y)$

Now with homomorphic encryption we can take the encrypted values and multiply them to get the addition:

$$E(x+y) = E(x) \times E(y)$$

We can perform this with discrete logs and where:

$$E(x+y) = g^{(x+y) \mod p-1}$$

which is equal to:

$$E(x+y) = g^x g^y$$

Thus Alice encrypts x at g^x, and y with g^y, and then just multiply the values together to get:

$E(x+y)$

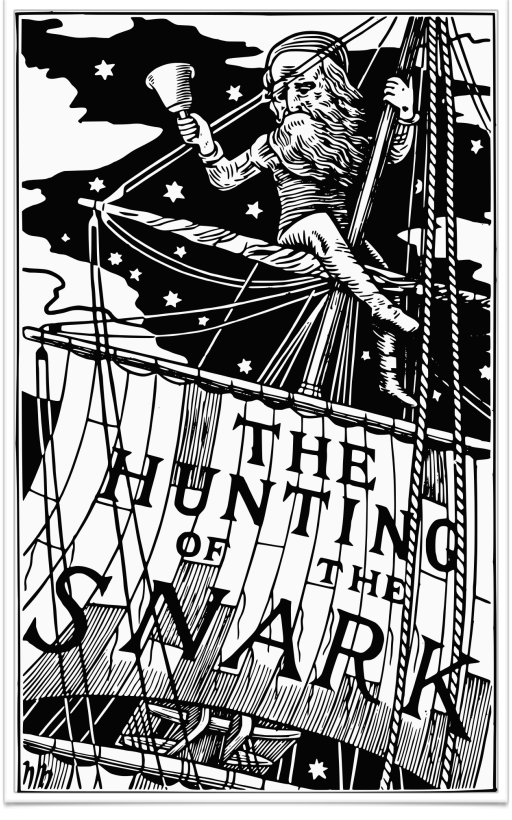
Now if Bob wanted to know if x plus y equals 8, Bob will then encrypt:

$E(8)$

Blind Evaluation

ant Alice to known
t he is using and
want Bob to know
e is using to
ult

zkSNARK



Hidden Homomorphic

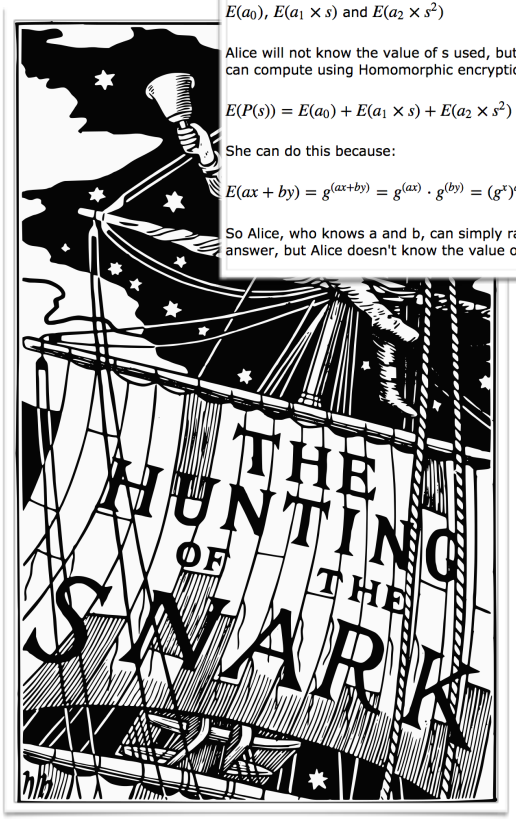
With HH we aim to prove that Alice knows two values (x, y) to equal a given answer (ans)

Let's say you want to prove that
Alice can produce two number
which adds up to 8. Overcomes the
major problem of Blockchain -
which is the lack of privacy in
computations and transac

Blind Evaluation

Bob doesn't want Alice to known the values that he is using and Alice doesn't want Bob to know the method she is using to compute a result

zkSNARK



For this we use polynomials, and where we have an equation such as:

$P(x) = a_0 + a_1 x + a_2 x^2$

Bob sends all the elements - known as hidings - of the computation for a value of s:

$E(a_0)$, $E(a_1 \times s)$ and $E(a_2 \times s^2)$

Alice will not know the value of s used, but she knows the "wiring" of the function. In this case she knows that it is a simple adding operation, so she can compute using Homomorphic encryption:

$E(P(s)) = E(a_0) + E(a_1 \times s) + E(a_2 \times s^2)$

She can do this because:

$E(ax + by) = g^{(ax+by)} = g^{(ax)} \cdot g^{(by)} = (g^x)^a \cdot (g^y)^b = E(x)^a \cdot E(y)^b$

So Alice, who knows a and b, can simply raise the values received to the polynomial factors and multiply and return to Bob. Bob then knows the answer, but Alice doesn't know the value of s used.

Hidden Homomorphic

we aim to prove that Alice
o values (x, y) to equal a
swer (ans)

that

ber

mes the

major problem of Blockchain -

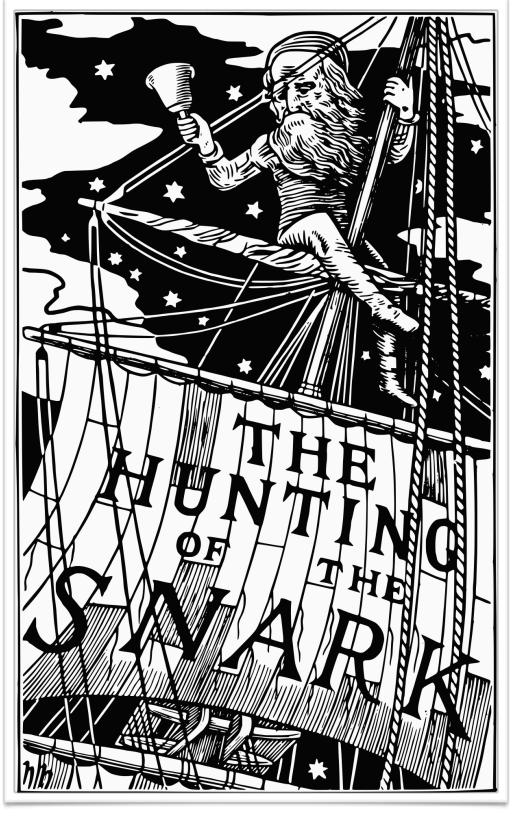
which is the lack of privacy in

computations and transac

Blind Evaluation

Bob doesn't want Alice to known the values that he is using and Alice doesn't want Bob to know the method she is using to compute a result

zkSNARK



Hidden Homomorphic

With HH we aim to prove that Alice knows two values (x, y) to equal a given answer (ans)

Let's say you want to prove that
Alice can produce two number
which adds up to 8. Overcomes the
major problem of Blockchain -
which is the lack of privacy in
computations and transac

Blind Evaluation

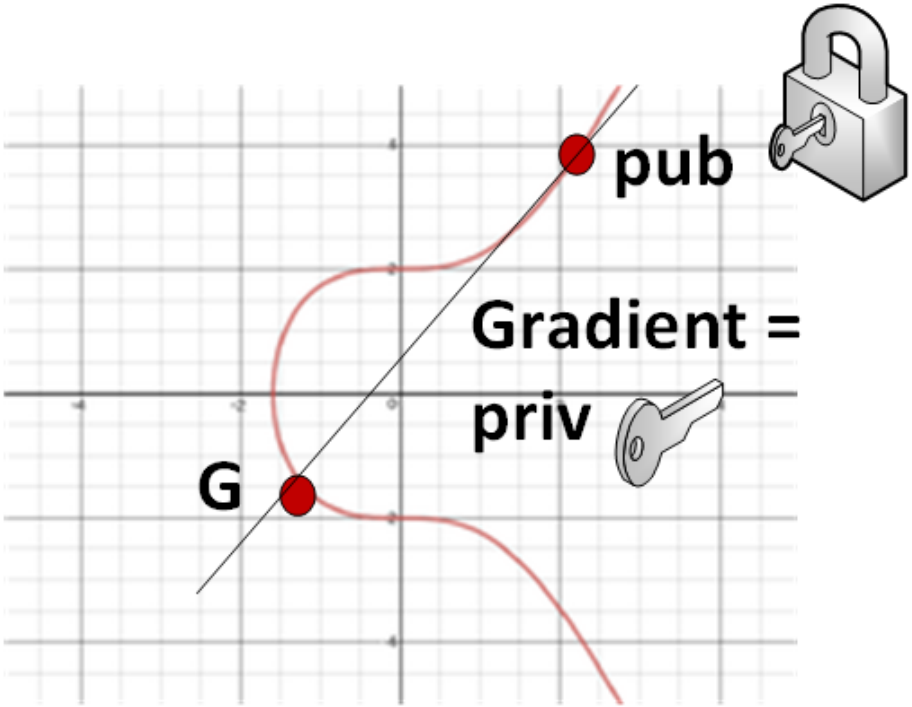
Bob doesn't want Alice to known the values that he is using and Alice doesn't want Bob to know the method she is using to compute a result

Next Generation Crypto

Light-weight crypto.
Quantum-robust crypto
Tokenization.
Zero-knowledge.
Homomorphic Encryption.
zkSnarks, Range-proofs

Prof Bill Buchanan OBE
http://asecuritysite.com/zero

Public and private keys with ECC



Private key:
0xc9f4f55bdeb5ba0bd337f2dbc952a5439e20ef
9af6203d25d014e7102d86aaeeL
Public key:
0xc44370819cb3b7b57b2aa7edf550a9a5410c23
4d27aff497458bbbfec8b6a327,
0x52a1a3e222cd89cbd2764b69bd9b0ea5c4fd6c
a28861e1f2140eeff9c2e76487

G:
(506626302227734366957871889516853432625060345377759417550018736038911672924 0L
,
326705100207588169780830851305070431844712733806592432759389043357573374824 2

Pedersen Commitment and Mimblewimble

Pedersen Commitment and Mimblewimble

Adding a blinding value

For this, we can obscure the values in a transaction by adding a **blinding value**. Let's say that we have a transaction value of v, we can now define this as a point on the elliptic curve (H) as:

$$v \times H$$

If we have three transactions ($v1$, $v2$ and $v3$), we can create a sum total of:

$$\text{Total} = v1 \times H + v2 \times H + v3 \times H = (v1 + v2 + v3) \times H$$

We can thus determine the sum of the transactions. But we could eventually find-out the values of $v1$, $v2$ and $v3$, as they would always appear as the same value when multiplied by H. We can now add a blinding factor by adding a second point on another elliptic curve (G) and a private key (r). A transaction value is then (as defined as a Pedersen Commitment):

$$C = v \times H + r \times G$$

Pedersen Commitment and Mimblewimble

```
MIMBLEWIMBLE
Tom Elvis Jedusor
19 July, 2016

\****/
Introduction
/****\

Bitcoin is the first widely used financial system for which all the necessary
data to validate the system status can be cryptographically verified by anyone.
However, it accomplishes this feat by storing all transactions in a public
database called "the blockchain" and someone who genuinely wishes to check
this state must download the whole thing and basically replay each transaction,
check each one as they go. Meanwhile, most of these transactions have not
affected the actual final state (they create outputs that are destroyed
a transaction later).
```

Adding a blinding value

…y adding a **blinding**
…, we can now define this

…l create a sum total of:

…But we could eventually
…lways appear as the same
…ding factor by adding a
second point on another elliptic curve (G) and a private key (r). A transaction value is then (as defined as a Pedersen Commitment):

$$C = v \times H + r \times G$$

Pedersen Commitment and Mimblewimble

MIMBLEWIMBLE

Tom E…

19 Ju…

****…

Introd…

/****…

Bitco…

data …

Howeve…

databa…

this …

check …

affect…

a tran…

Ignotus Peverell

ignopeverell

Block or report user

Sign in to view email

Overview Repositories 5 Stars 2 Followers 135 Following 0

Popular repositories

grin
Forked from mimblewimble/grin
Minimal implementation of the MimbleWimble protocol.
Rust ★ 1

site-test
Test repository for a slightly more elaborate Grin Jekyll site
HTML ★ 1 2

hacker
Forked from pages-themes/hacker
Hacker is a Jekyll theme for GitHub Pages
CSS 1

enumset
Forked from Lymia/enumset
A library for defining enums that can be used in compact bit sets.
Rust

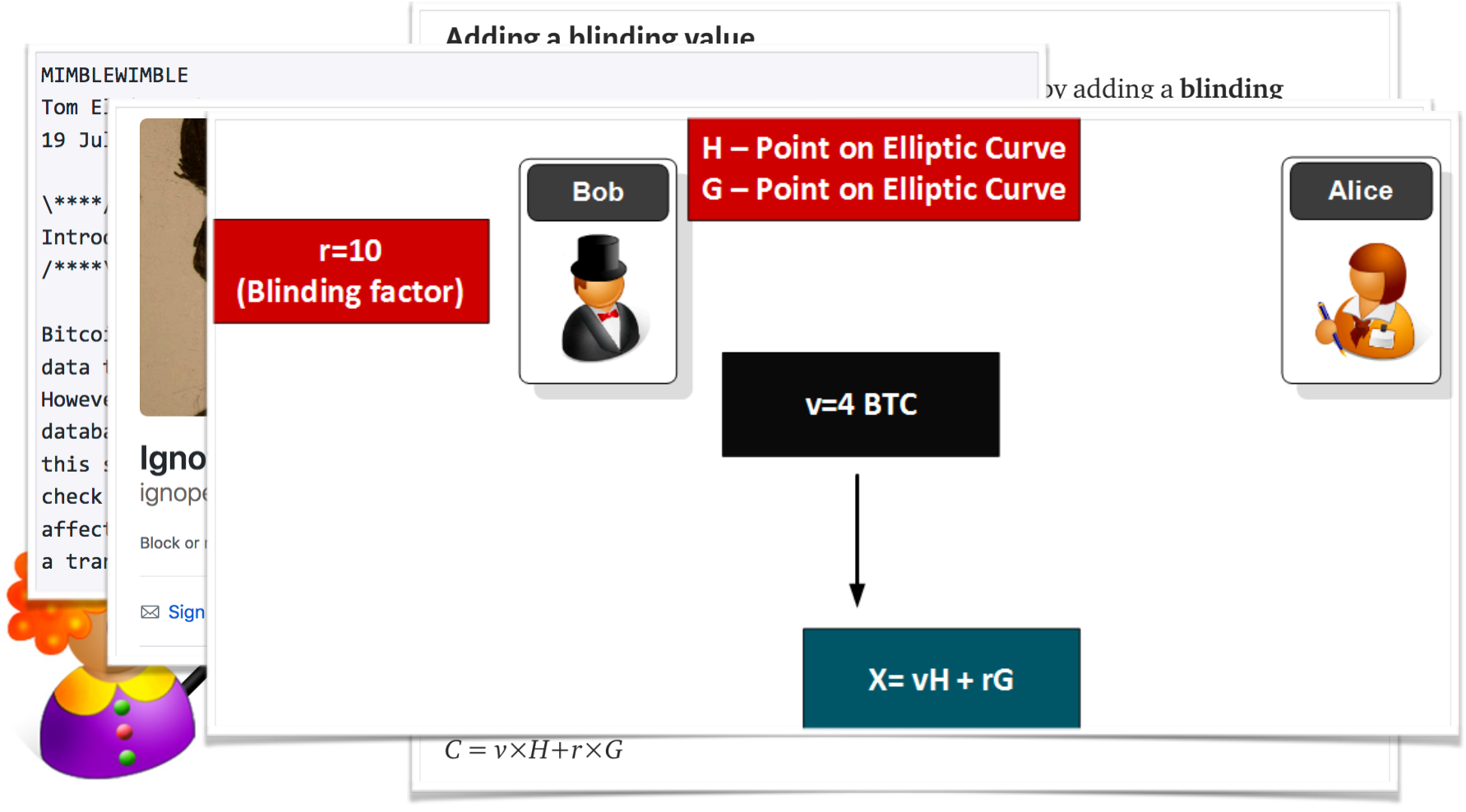
Adding a blinding value

…by adding a **blinding**

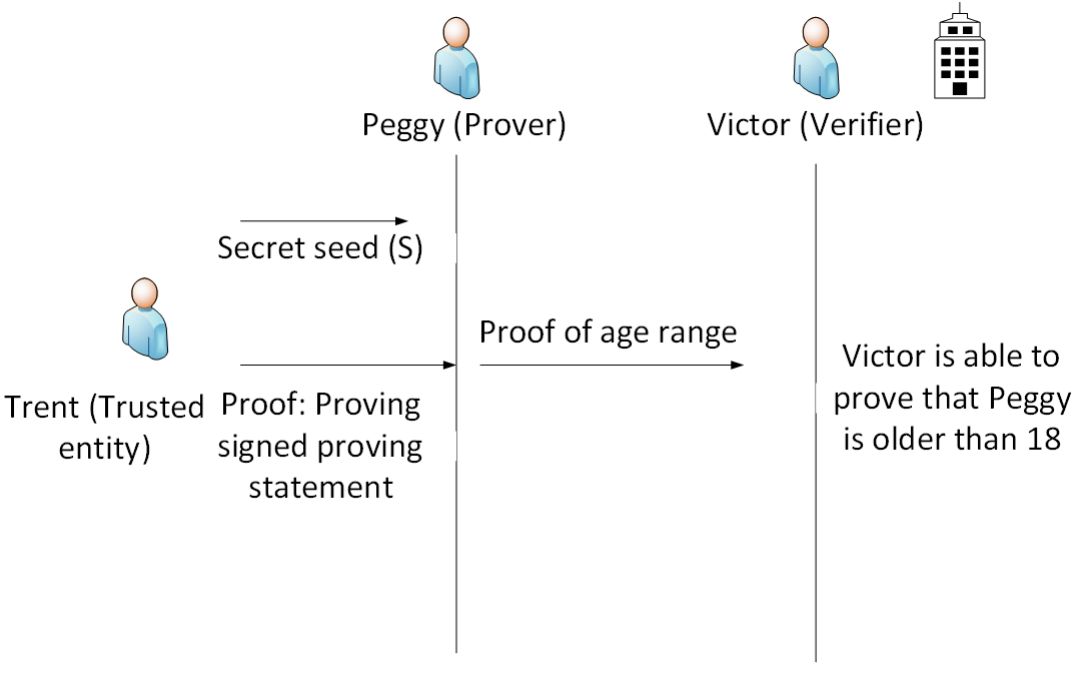
value is then (as defined as a Pedersen Commitment):

$$C = v \times H + r \times G$$

Pedersen Commitment and Mimblewimble



Range Proof

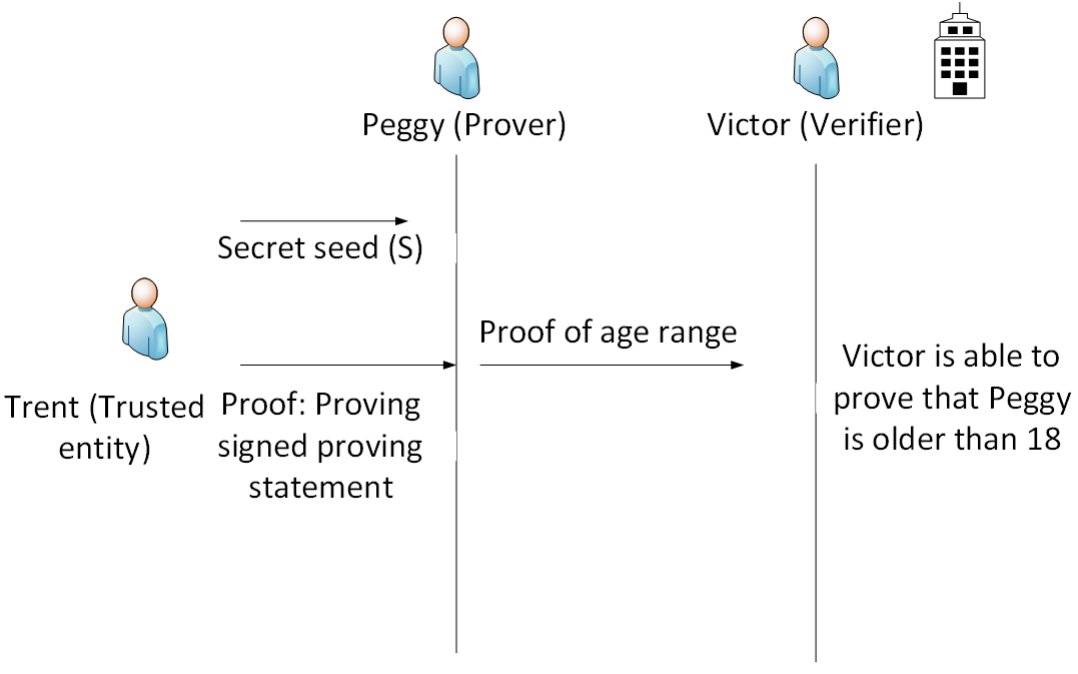

Peggy (Prover)
Victor (Verifier)
Secret seed (S)
Trent (Trusted entity)
Proof: Proving signed proving statement
Proof of age range
Victor is able to prove that Peggy is older than 18

Range Proof

Bulletproofs: Short Proofs for Confidential Transactions and More

Benedikt Bünz*[1], Jonathan Bootle†[2], Dan Boneh‡[1],
Andrew Poelstra§[3], Pieter Wuille¶[3], and Greg Maxwell‖

[1]Stanford University
[2]University College London
[3]Blockstream

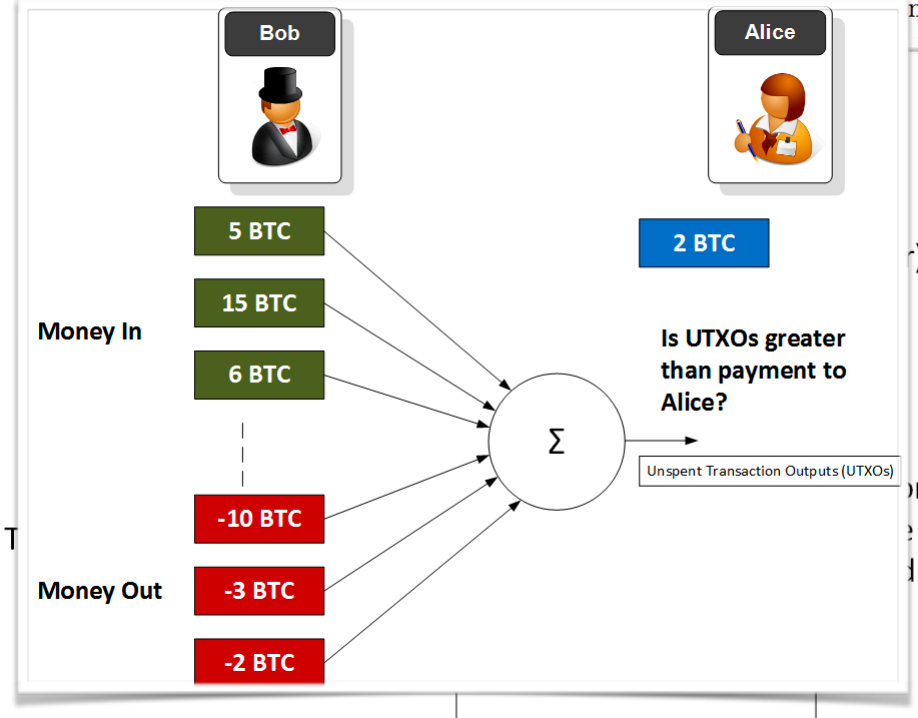


Range Proof

Bulletproofs: Short Proofs for Confidential Transactions and More

Benedikt Bünz*[1], Jonathan Bootle†[2], Dan Boneh‡[1],
Andrew Poelstra§[3], Pieter Wuille¶[3], and Greg Maxwell‖

[1]Stanford University
[2]University College London



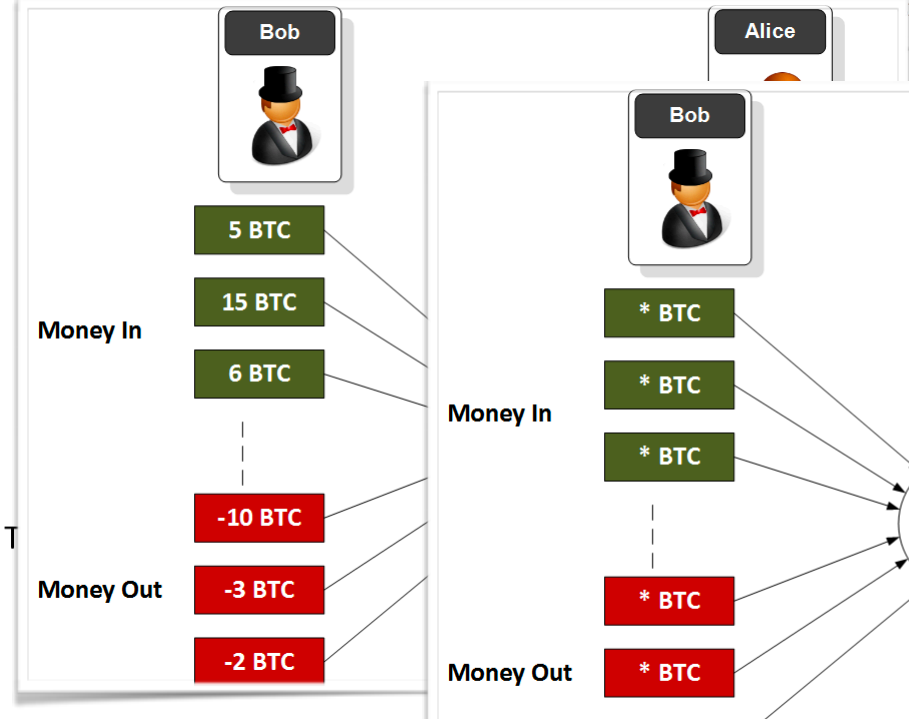
r is able to
that Peggy
der than 18

Range Proof

Bulletproofs: Short Proofs for Confidential Transactions and More

Benedikt Bünz*[1], Jonathan Bootle†[2], Dan Boneh‡[1],
Andrew Poelstra§[3], Pieter Wuille¶[3], and Greg Maxwell‖

[1]Stanford University
[2]University College London



Monero has seen an 80% reduction in transaction size, and which has also led to a significant reduction in the transaction fees.

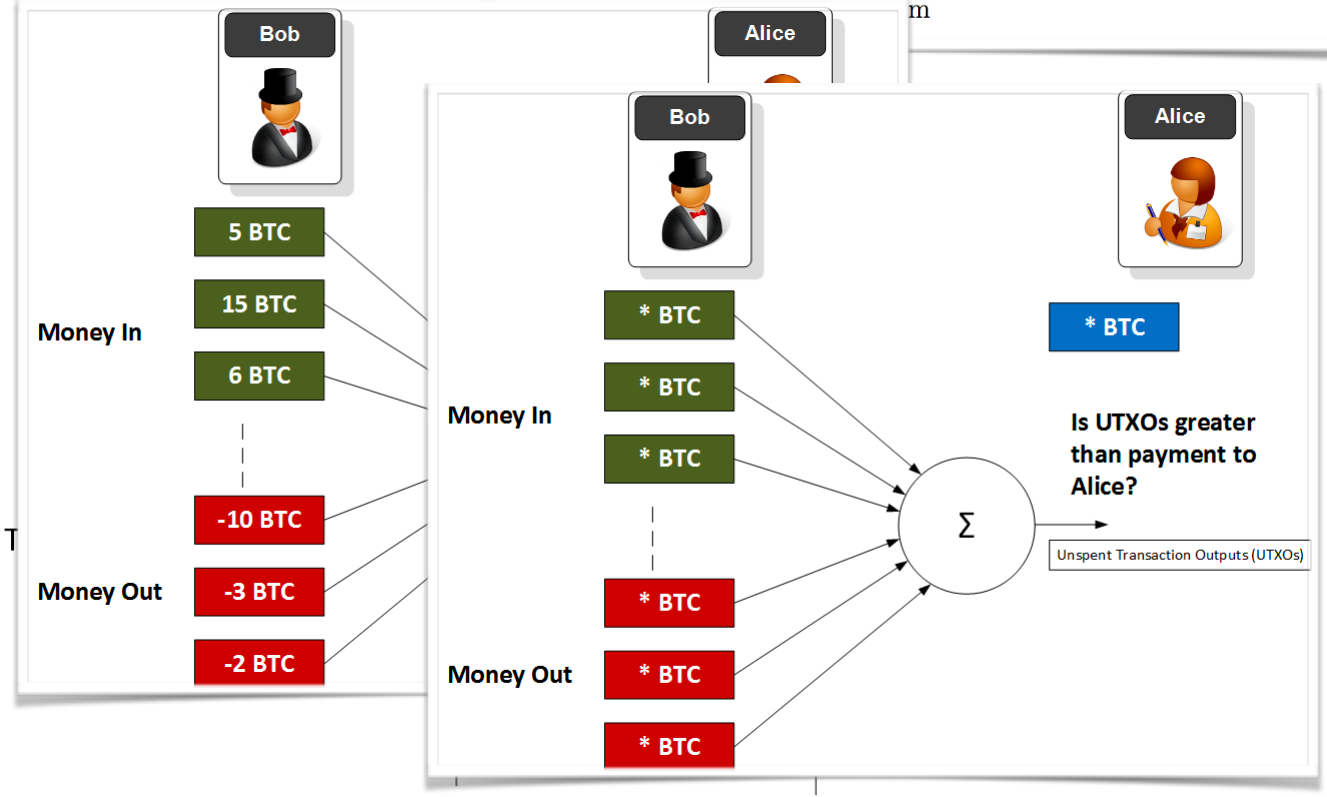
Range Proof

Bob is applying for a loan from Alice, and now needs to prove that he has a salary of between $60,000 and $100,000, and that his employer — Trent — has at least $1million in the bank ... merged into a single (and short) bullet proof

Bulletproofs: Short Proofs for Confidential Transactions and More

Benedikt Bünz*[1], Jonathan Bootle†[2], Dan Boneh‡[1],
Andrew Poelstra§[3], Pieter Wuille¶[3], and Greg Maxwell‖

[1]Stanford University
[2]University College London



Monero has seen an 80%

reduction in transaction

size, and which has also led

to a significant reduction in

the transaction fees.

Range Proof

Bulletproofs: Short Proofs for Confidential Transactions and More

Benedikt Bünz*[1], Jonathan Bootle†[2], Dan Boneh‡[1],
Andrew Poelstra§[3], Pieter Wuille¶[3], and Greg Maxwell‖

[1]Stanford University
[2]University College London

Bob is applyin

from Alice, an

to prove that h

of between $6

$100,000, and

employer — Tr

least $1million

... merged into

short) bullet p

- Significant reduction in the size of the signature as opposed to other CT methods (such as zk-SNARKs and zk-STARKs).
- Significantly reduced transaction fees with shorter signatures.
- Supports MPC (Multiparty Computation) and where many parties can come together to create a single range proof, without revealing their secrets.
- Allow for the aggregation of range proofs and produces a single, and short, signature.
- Fast verification of proofs (and which are faster than most range proof methods, but still slower than zk-SNARKs).
- Design to be setup for blockchain integration.
- No need to setup a trust infrastructure. This often involves creating an initial set of encryption keys which are then used for trusted signatures. These keys should be used only once, and then deleted. If these keys are not deleted, there is a risk to future trustworthiness of the whole infrastructure.

Monero has seen an 80%

reduction in t

size, and whi

to a significa

the transactio

Bob is applyin

from Alice, an

to prove that h

of between $6

$100,000, and

employer — Tr

least $1millio

... merged into

short) bullet p

Range

- With bulletproofs, we can ask our financial institutions to prove that they have liquidity … “Prove that you have more than $1 billion of liquidity”, and if they failed to prove this, we would quickly move to audit them. Fraud on a large-scale basis would thus be detected in seconds.

nd More

University College London

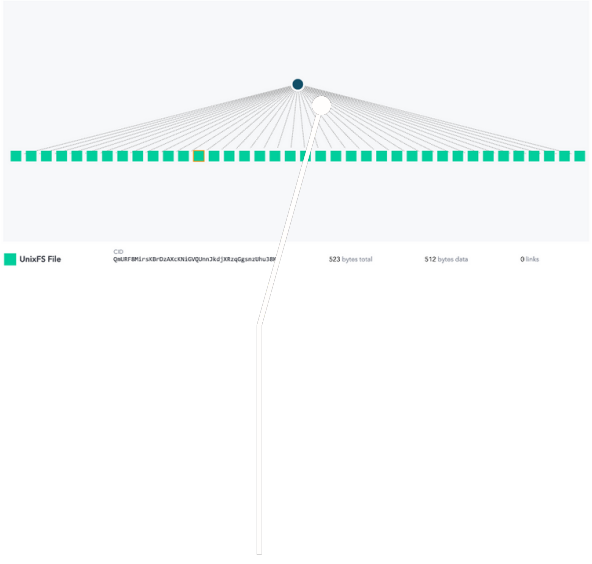
- Significant reduction in the size of the signature as opposed to other CT methods (such as zk-SNARKs and zk-STARKs).
- Significantly reduced transaction fees with shorter signatures.
- Supports MPC (Multiparty Computation) and where many parties can come together to create a single range proof, without revealing their secrets.
- Allow for the aggregation of range proofs and produces a single, and short, signature.
- Fast verification of proofs (and which are faster than most range proof methods, but still slower than zk-SNARKs).
- Design to be setup for blockchain integration.
- No need to setup a trust infrastructure. This often involves creating an initial set of encryption keys which are then used for trusted signatures. These keys should be used only once, and then deleted. If these keys are not deleted, there is a risk to future trustworthiness of the whole infrastructure.

IPFS

Light-weight crypto.
Quantum-robust crypto
Tokenization.
Zero-knowledge.
Homomorphic Encryption.
zkSnarks, Range-proofs
IPFS

Prof Bill Buchanan OBE
http://asecuritysite.com/zero

IPFS



Protobuf UnixFS View on IPFS Gateway

CID QmZqkuqX1qTspb1GgmnzyRFetf1uMyA3CemvvgPZD39sPo
SIZE 33 KB
LINKS 3
DATA

```
▶Object {type: "directory", data: undefined, blockSizes: Array[0]}
```

	PATH	CID
0	1004 - Batman - alt.txt	QmY2JkRD74B1Sb8NRN8EpC1117h6VsZEX7oV16HH78aT11
1	1004 - Batman - transcript.txt	QmQM43UESfCSSQfFAheJEq5jduc2fgx1HVX1Mi3zWC66uc
2	1004 - Batman.png	QmbBnZBudfuGbbzDRTAhSVEYsoxCAe5yBimuuYvYu3BMHG

Protobuf UnixFS View on IPFS Gateway

CID QmY2JkRD74B1Sb8NRN8EpC1117h6VsZEX7oV16HH78aT11
SIZE 185 B
LINKS 0
DATA

```
▼Object {type: "file", data: Buffer[174], blockSizes: Array[0]}
  type: "file"
▼ data: Buffer[174]
    0: 73
    1: 39
    2: 109
    3: 32
    4: 114
    5: 101
    6: 97
    7: 108
    8: 108
    9: 121
    10: 32
```

CID INFO

QmZqkuqX1qTspb1GgmnzyRFetf1uMyA3CemvvgPZD39sPo

base58btc - cidv0 - dag-pb - sha2-256-256-aae5699dbbf2a3...

BASE - VERSION - CODEC - MULTIHASH

MULTIHASH

0x1220aae5699dbbf2a37aa4da529084e2cd82ebacd63c6468becce651522616878112

HASH DIGEST

0x12 = sha2-256
0x20 = 256 bits

Next Generation Crypto

Light-weight crypto.
Quantum-robust crypto
Tokenization.
Zero-knowledge.
Homomorphic Encryption.
zkSnarks, Range-proofs

Prof Bill Buchanan OBE
http://asecuritysite.com/encryption