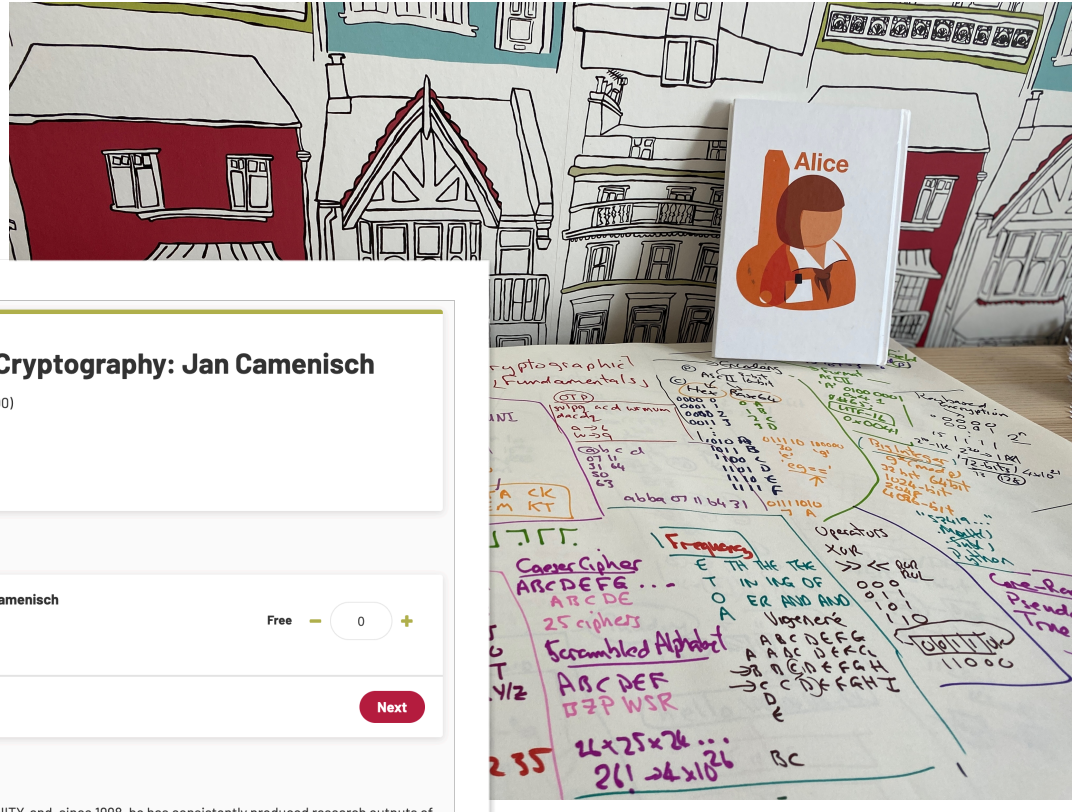
Next Generation Crypto

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

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

Next Generation Crypto



Light-weight crypto

Qu
Tok
Zer
Hor
zkS
IPF

Pr
htt

World-leaders in Cryptography: Jan Camenisch

30th Apr 2024 6pm - 7pm (GMT+01:00)

This is a live-stream event

Add to Calendar

Tickets

World-leaders in Cryptography: Jan Camenisch

E-TICKET

World-leaders in Cryptography: Jan Camenisch

Free − 0 +

Next

HOSTED BY

Edinburgh Napier

Event Details

Jan is the CTO and a Cryptographer at DFINITY, and, since 1998, he has consistently produced research outputs of rigour, novelty and sheer brilliance [here]. He was recently awarded the Levchin Prize at Real World Crypto 2024 - along with Anna Lysyanskaya.

Next Gene

Light-weight

Qu

Tok

Zer

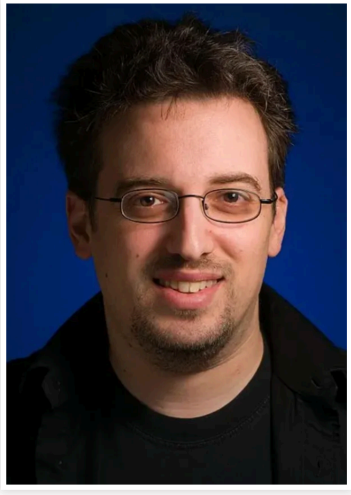
Hor

zkS

IPF

Pr

htt



World-leaders in Cryptography: Daniel J Bernstein

8th May 2024 5pm - 6pm (GMT+01:00)

This is a live-stream event

Add to Calendar

HOSTED BY

Edinburgh Napier

Tickets

World-leaders in Cryptography: Daniel J Bernstein

E-TICKET

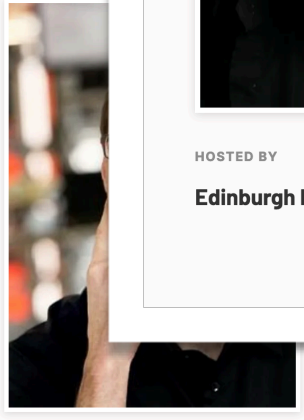
World-leaders in Cryptography: Daniel J Bernstein

Free − 0 +

Next

Event Details

Daniel J Bernstein (djb) was born in 1971. He is a USA/German citizen and a Personal Professor at Eindhoven University of Technology and a Research Professor at the University of Illinois at Chicago.



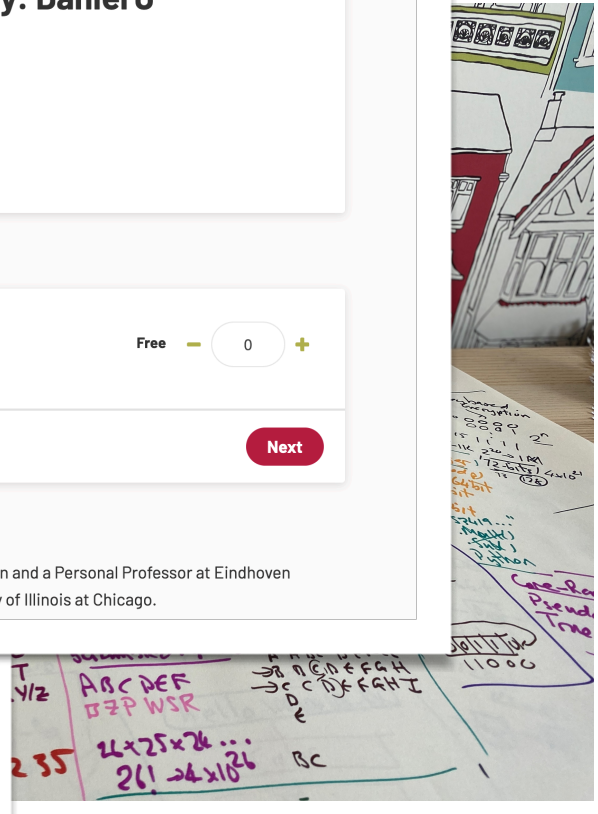
HOSTED BY

Edinburgh Napier

Next

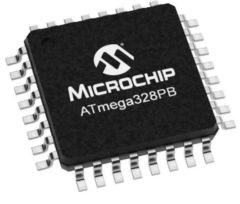
Event Details

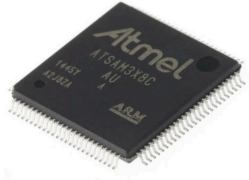
Jan is the CTO and a Cryptographer at DFINITY, and, since 1998, he has consistently produced research outputs of rigour, novelty and sheer brilliance [here]. He was recently awarded the Levchin Prize at Real World Crypto 2024 - along with Anna Lysyanskaya.



No	Date	Subject	Lab
2	23 Jan 2024	Ciphers and Fundamentals [Unit]	[Lab] [Demo]
3	30 Jan 2024	Symmetric Key [Unit]	[Lab]
4	6 Feb 2024	Hashing and MAC [Unit]	[Lab Bruce Schneier]
5	13 Feb 2024	Asymmetric (Public) Key [Unit]	[Lab]
6	20 Feb 2024	Key Exchange [Unit]	[Lab Whitfield Diffie]
7	27 Feb 2024	Reading Week (Revision lecture)	Mini-project [Here] / Coursework
8	5 Mar 2024	Digital Signatures and Certificates [Unit]	[Lab Vincent Rijmen]
9	12 Mar 2024	Test (Units 1-5) 40% of overall mark [Here]	
10	19 Mar 2024	Tunnelling [Unit]	[Lab Marty Hellman
11	9 Apr 2024	Blockchain [Unit]	[Lab] Troy Hunt
12	16 Apr 2024	Future Cryptography [Unit]	[Lab]
13	23 Apr 2024	Host/Cloud Security [Unit]	[Lab]
14	30 Apr 2024		Jan Camenish
15	7 May 2024	Coursework Hand-in - 60% of overall mark (Sunday, 12 May 2024) [Coursework]	Daniel J Bernstein

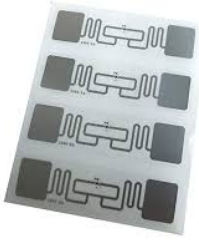
A Computer?





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

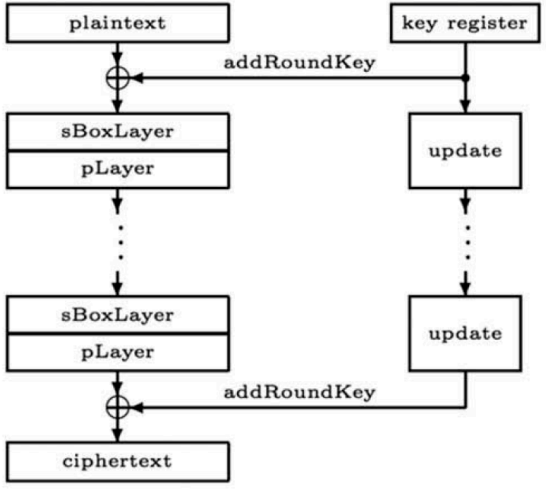
*IDEA: International Data Encryption Algorithm; TEA: Tiny Encryption Algorithm; SEA: Scalable Encryption Algorithm.

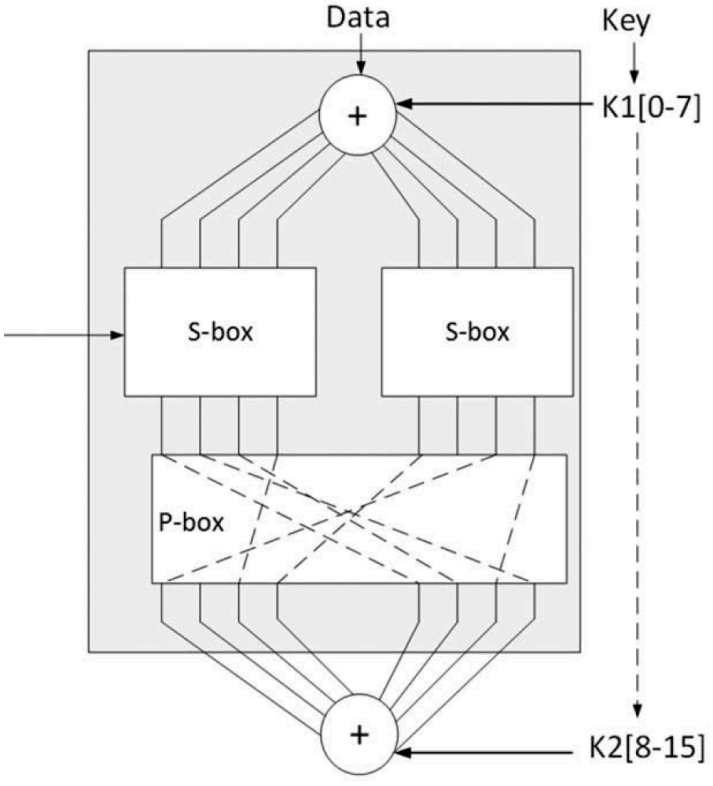
function will use on the device" and which is defined in *pm*".

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

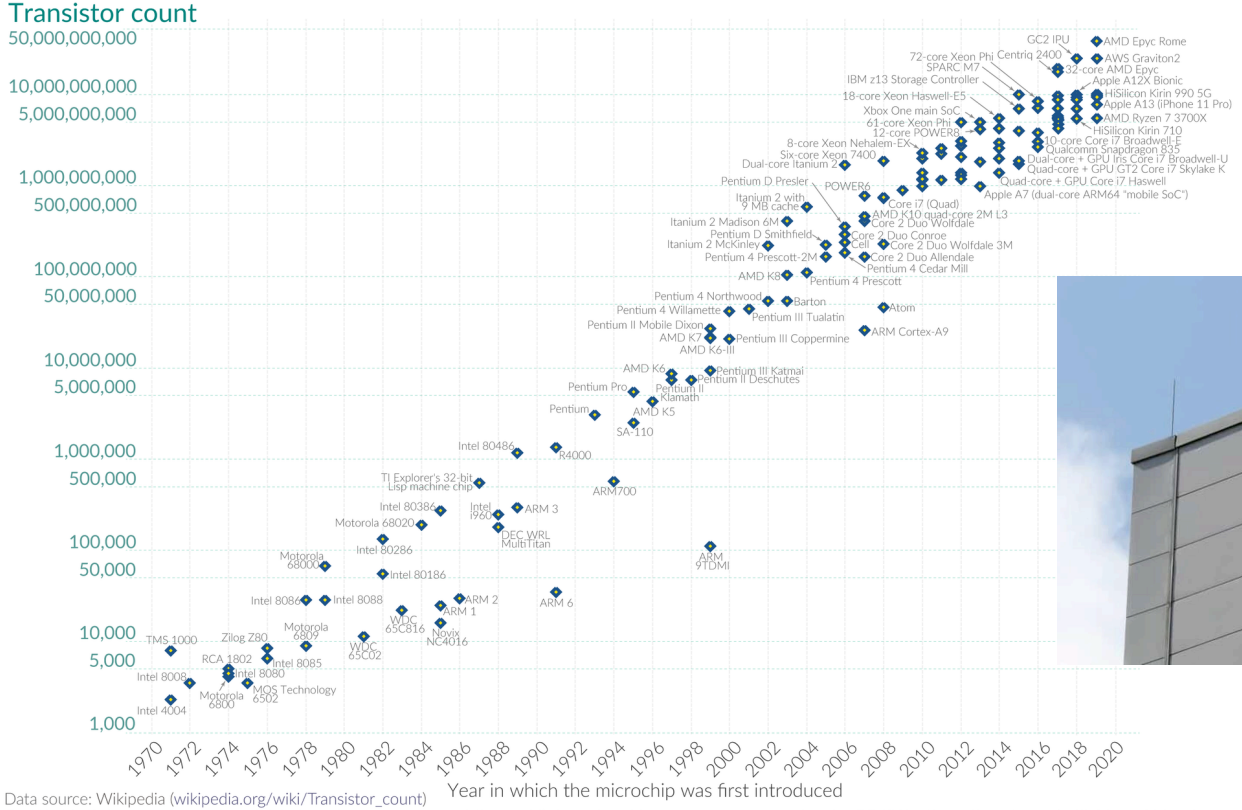
Prof Bill Buchanan OBE
https://asecuritysite.com/pqc

Moore's Law

Moore's Law: The number of transistors on microchips doubles every two years

Our World in Data

Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years.
This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.

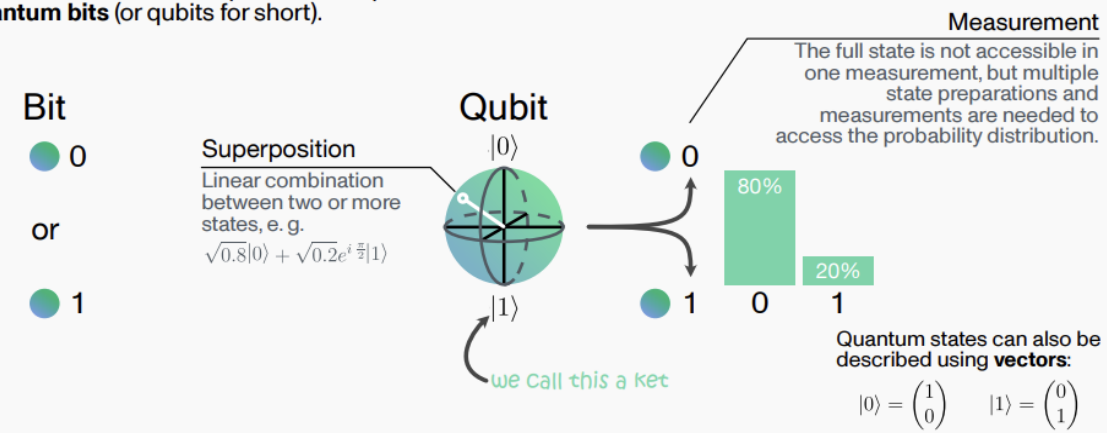


Data source: Wikipedia (wikipedia.org/wiki/Transistor_count)
OurWorldinData.org – Research and data to make progress against the world's largest problems.

Qubits.

Bits and Qubits

Instead of classical bits, quantum computers use **quantum bits** (or qubits for short).

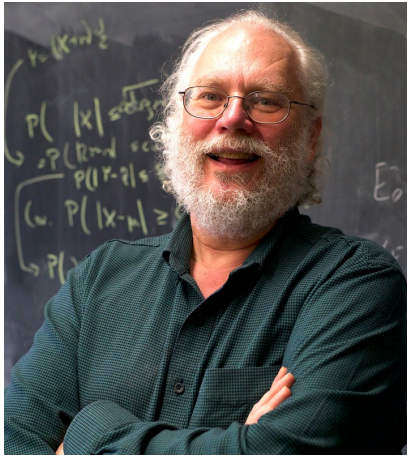


One-Qubit Gates

Gate		Matrix
X	**Pauli-X** is a 180° rotation around the x-axis; also known as the quantum NOT gate	$\begin{pmatrix}0 & 1\\1 & 0\end{pmatrix}$
Y	**Pauli-Y** is a 180° rotation around the y-axis	$\begin{pmatrix}0 & -i\\i & 0\end{pmatrix}$
Z	**Pauli-Z** is a 180° rotation around the z-axis	$\begin{pmatrix}1 & 0\\0 & -1\end{pmatrix}$
H	**Hadamard** maps $\|0\rangle$ to $\|+\rangle$ and $\|1\rangle$ to $\|-\rangle$; used to create an equal superposition	$\frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\1 & -1\end{pmatrix}$
S	**S** is a 90° rotation around the z-axis; S S = Z ; The inverse $S^\dagger$ rotates in the opposite direction	$\begin{pmatrix}1 & 0\\0 & i\end{pmatrix}$

Superposition

Shor's algorithm



See Appendix 1 for details + Colab notebook for Shor.

Factoring through order-finding

The following method succeeds in finding a factor of N with probability at least $1/2$, provided N is odd and not a prime power.

Factor-finding method

1. Choose $a \in \{2, \ldots, N-1\}$ at random.
2. Compute $d = \gcd(a, N)$. If $d \geq 2$ then output d and stop.
3. *Compute the order* r of a modulo N.
4. If r is even, then compute $d = \gcd(a^{r/2} - 1, N)$. If $d \geq 2$, output d and stop.
5. If this step is reached, the method has failed.

Main idea

1. By the definition of the order, we know that $a^r \equiv 1 \pmod{N}$.

$$a^r \equiv 1 \pmod{N} \quad \Leftrightarrow \quad N \text{ divides } a^r - 1$$

2. If r is even, then

$$a^r - 1 = (a^{r/2} + 1)(a^{r/2} - 1)$$

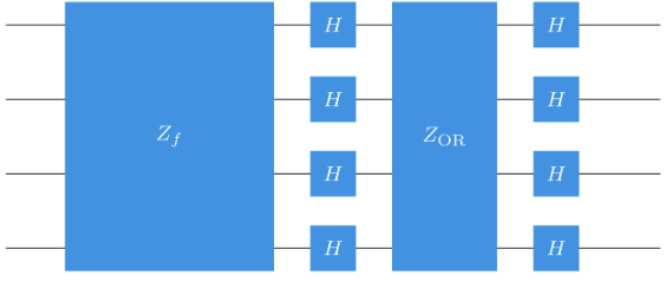
Each prime dividing N must therefore divide either $(a^{r/2} + 1)$ or $(a^{r/2} - 1)$.
For a random a, at least one of the prime factors of N is likely to divide $(a^{r/2} - 1)$.

Grover's algorithm.

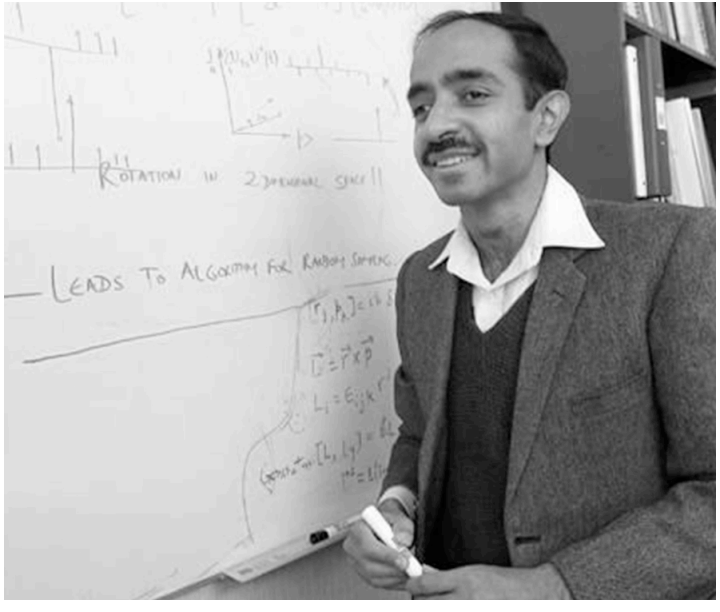
Grover's algorithm

1. Initialize an n qubit register Q to the all-zero state $|0^n\rangle$ and then apply a Hadamard operation to each qubit of Q.
2. Apply t times the unitary operation $G = H^{\otimes n} Z_{\mathrm{OR}} H^{\otimes n} Z_f$ to the register Q
3. Measure the qubits of Q with respect to standard basis measurements and output the resulting string.

The operation $G = H^{\otimes n} Z_{\mathrm{OR}} H^{\otimes n} Z_f$ iterated in step 2 will be called the *Grover operation* throughout the remainder of this lesson. Here is a quantum circuit representation of the Grover operation:



Here the Z_f operation is depicted as being larger than Z_{OR} as a way to suggest that it is likely to be the more costly operation (but this is only meant a visual clue and not something with a formal meaning). In particular, when we're working within the query model Z_f requires one query while Z_{OR} requires no queries — and if instead we have a Boolean circuit for the function f and convert it to a quantum circuit for Z_f, we can reasonably expect that the resulting quantum circuit will be larger and more complicated than one for Z_{OR}.



See Appendix 2 for details + Colab notebook for Grover.

Why worry?

President Biden Signs Memo to Combat Quantum Computing Threat

FORT MEADE, Md. — The White House announced today that President Joe Biden has signed a National Security Memorandum (NSM) aimed at maintaining U.S. leadership in quantum information sciences and to mitigate the risks of quantum computing to the Nation's security.

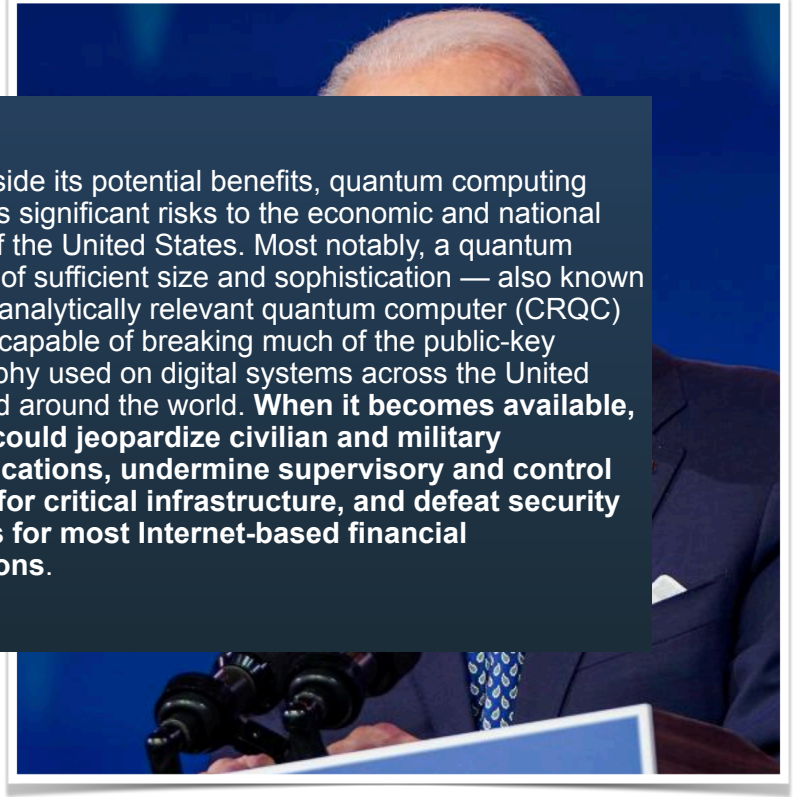
"Promoting United States Leadership in Quantum Computing While Mitigating Risks to Vulnerable Cryptographic Systems" - also known as NSM-10 - directs U.S. Government agencies to migrate vulnerable cryptographic systems to quantum-resistant cryptography as part of multi-year effort. As the National Manager for National Security Systems, the Director of NSA will oversee this process across the 50-plus government departments and agencies using National Security Systems (NSS) - systems that contain classified information or are otherwise critical to military or intelligence operations.

Why worry?

President Biden Signs Memo to Combat Quantum Computing Threat

FORT MEADE, Md. — The White House announced today that President Joe Biden has signed a National Security Memorandum (NSM) aimed at maintaining U.S. leadership in quantum information sciences and to mitigate the risks of quantum computing to the Nation's security.

"Promoting United States Leadership in Quantum Computing While Mitigating Risks to Vulnerable Cryptographic Systems" - also known as NSM-10 - directs U.S. Government agencies to migrate vulnerable cryptographic systems to quantum-resistant cryptography as part of multi-year effort. As the National Manager for National Security Systems, the Director of NSA will oversee this process across the 50-plus government departments and agencies using National Security Systems (NSS) - systems that contain classified information or are otherwise critical to military or intelligence operations.



Yet alongside its potential benefits, quantum computing also poses significant risks to the economic and national security of the United States. Most notably, a quantum computer of sufficient size and sophistication — also known as a cryptanalytically relevant quantum computer (CRQC) — will be capable of breaking much of the public-key cryptography used on digital systems across the United States and around the world. **When it becomes available, a CRQC could jeopardize civilian and military communications, undermine supervisory and control systems for critical infrastructure, and defeat security protocols for most Internet-based financial transactions**.

Why worry?

President Biden Signs Memo to Combat Quantum Computing Threat

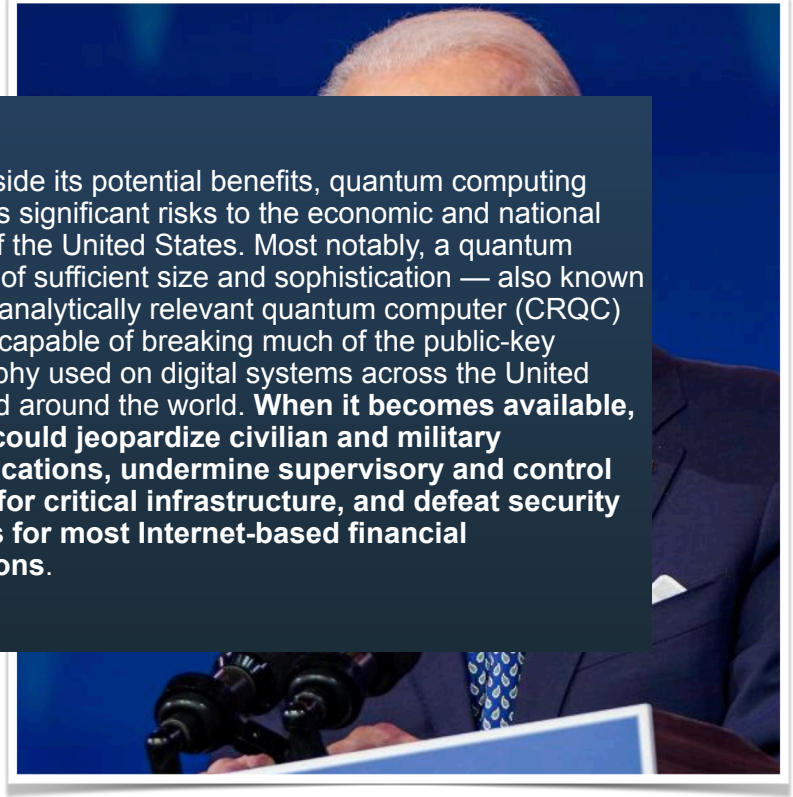
Within 90 days of the release of the first set of NIST standards for quantum-resistant cryptography referenced in subsection 3(a) of this memorandum, and on an annual basis thereafter, as needed, the Secretary of Commerce, through the **Director of NIST, shall release a proposed timeline for the deprecation of quantum-vulnerable cryptography in standards**, with the goal of moving the maximum number of systems off quantum-vulnerable cryptography within a decade of the publication of the initial set of standards. The Director of NIST shall work with the appropriate technical standards bodies to encourage interoperability of commercial cryptographic approaches.

t President
M) aimed a
s and to
urity.

While
known as
rable
s part of
rity System
lus
rity Systems
herwise

critical to military or intelligence operations.

Yet alongside its potential benefits, quantum computing also poses significant risks to the economic and national security of the United States. Most notably, a quantum computer of sufficient size and sophistication — also known as a cryptanalytically relevant quantum computer (CRQC) — will be capable of breaking much of the public-key cryptography used on digital systems across the United States and around the world. **When it becomes available, a CRQC could jeopardize civilian and military communications, undermine supervisory and control systems for critical infrastructure, and defeat security protocols for most Internet-based financial transactions**.



Why worry?

Pres
Cor

Within 90 days
standards for q
subsection 3(a)
thereafter, as n
the **Director of**
the deprecatio
standards, wit
systems off qua
decade of the p
Director of NIS
standards bodi
cryptographic a

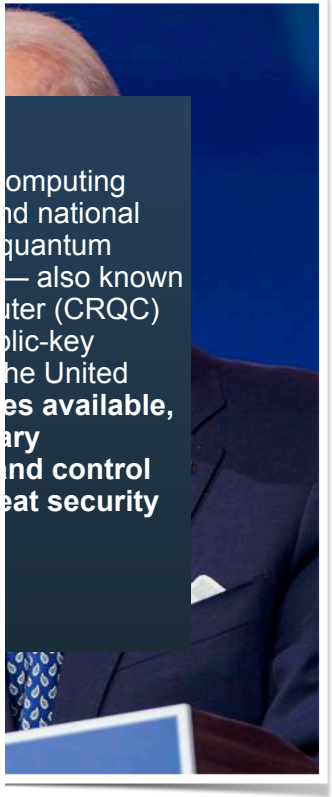
critical to mili

QUANTUM-READINESS: MIGRATION TO POST-QUANTUM CRYPTOGRAPHY

NIST NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY U.S. DEPARTMENT OF COMMERCE

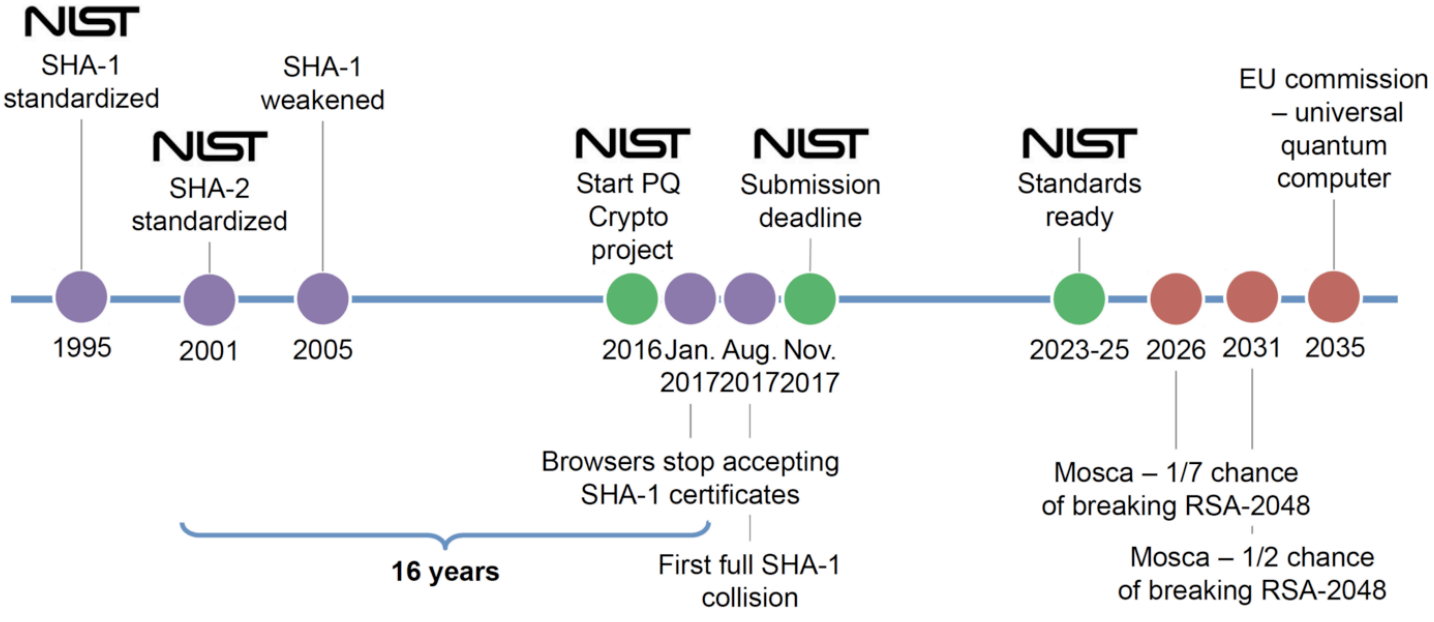
BACKGROUND

The Cybersecurity and Infrastructure Security Agency (CISA), the National Security Agency (NSA), and the National Institute of Standards and Technology (NIST) created this factsheet to inform organizations — especially those that support Critical Infrastructure — about the impacts of quantum capabilities, and to encourage the early planning for migration to post-quantum cryptographic standards by developing a Quantum-Readiness Roadmap. NIST is working to publish the first set of post-quantum cryptographic (PQC) standards, to be released in 2024, to protect against future, potentially adversarial, cryptanalytically-relevant quantum computer (CRQC) capabilities. A CRQC would have the potential to break public-key systems (sometimes referred to as asymmetric cryptography) that are used to protect information systems today.



omputing
nd national
quantum
— also known
ter (CRQC)
lic-key
he United
es available,
ary
nd control
eat security

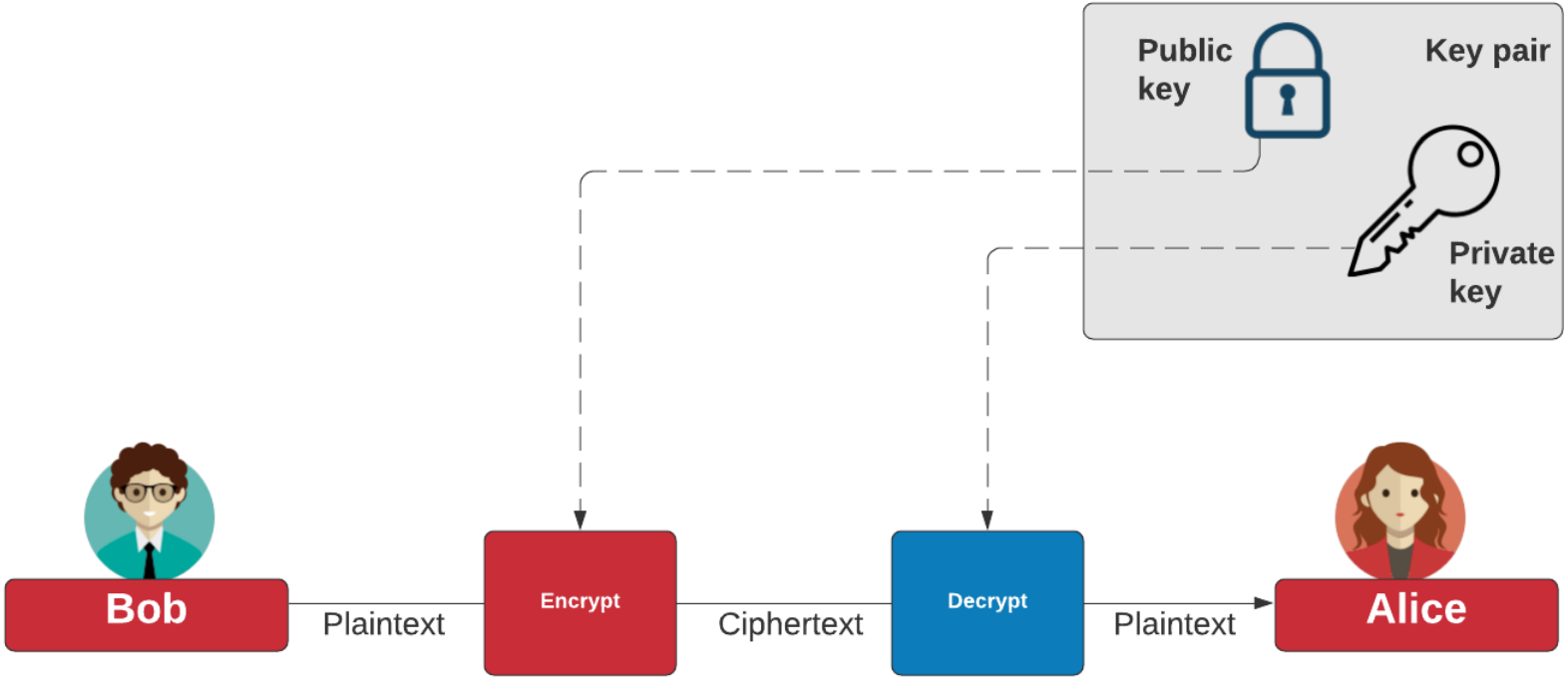
KEM and Digital Signatures/Encryption



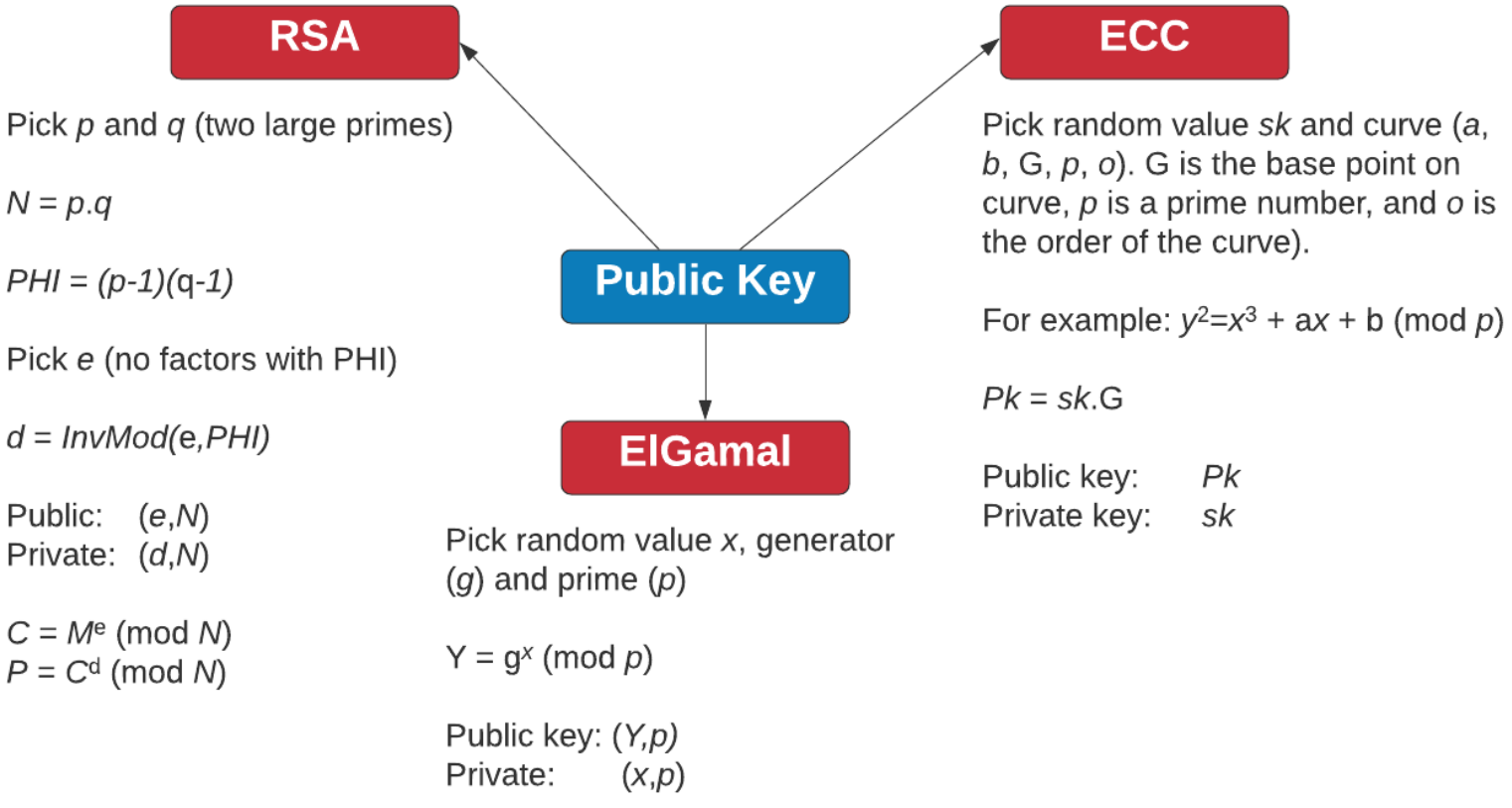
asecuritysite.com/pqc/

NIST

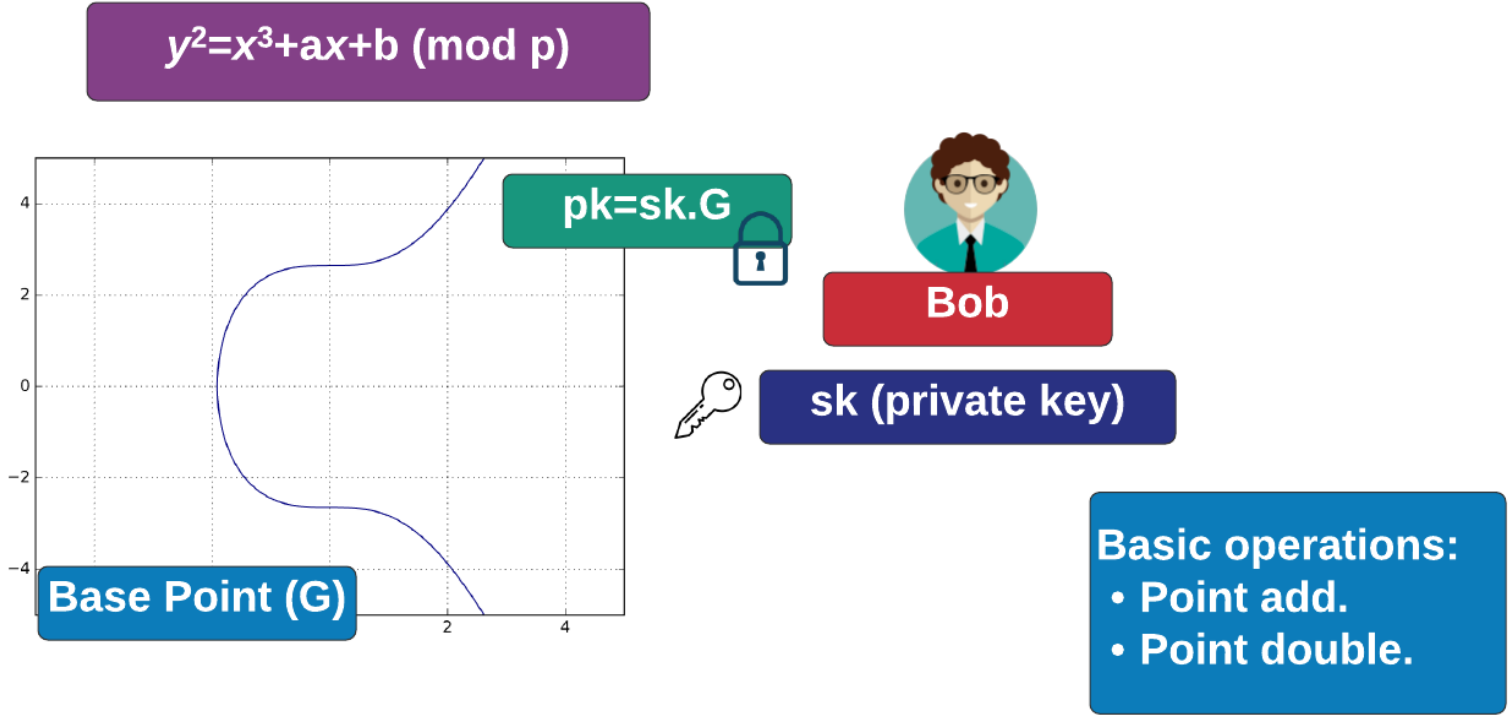
Public Key Encryption/Key Exchange



Existing Public Key Methods

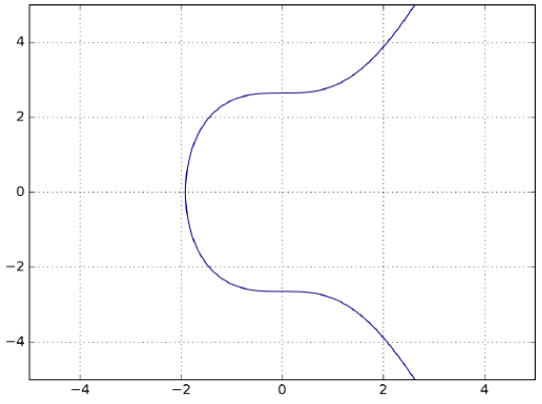


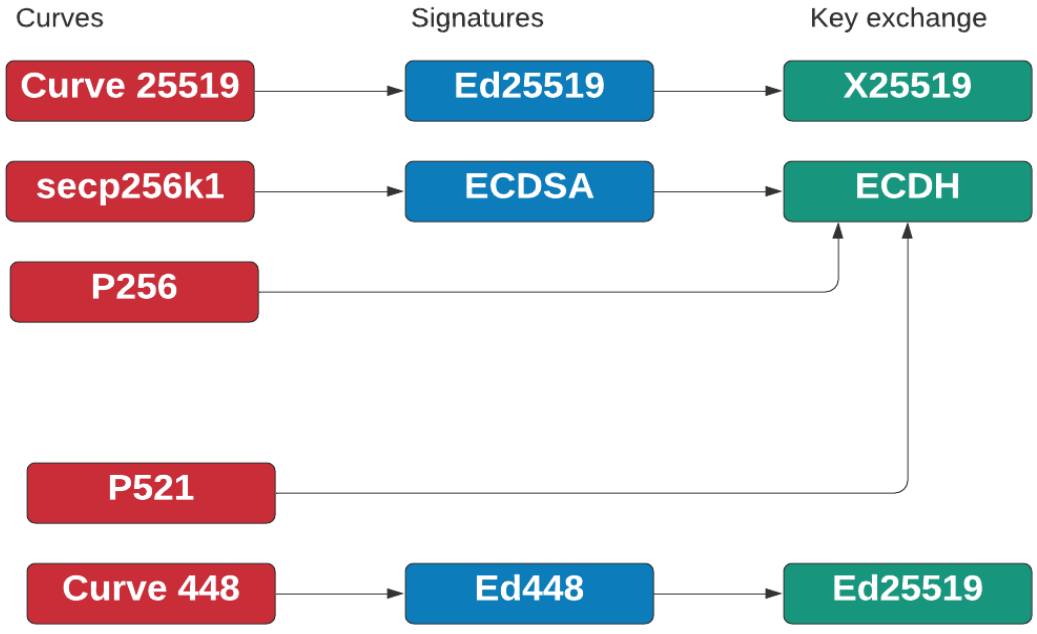
Elliptic Curve Outline



https://asecuritysite.com/ecc/ecc_types

Existing Signature and Key Exchange Methods

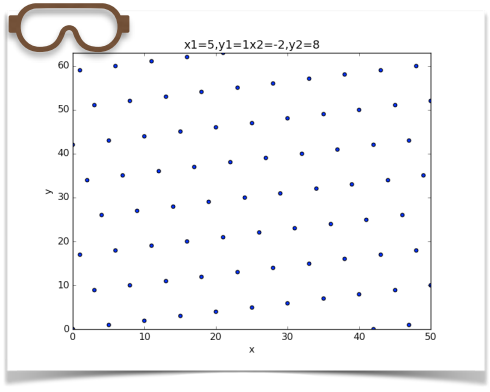


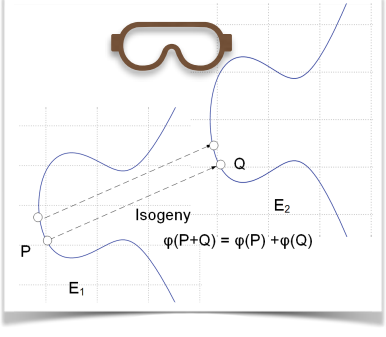


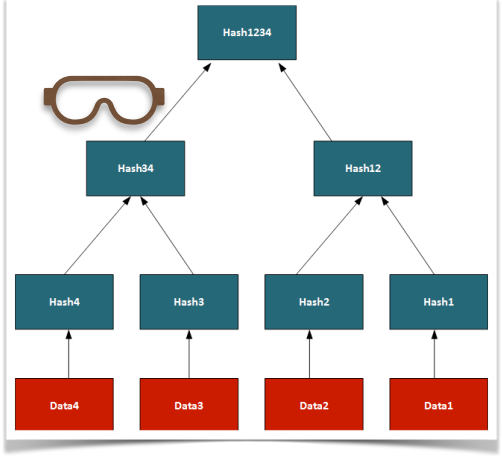
https://asecuritysite.com/ecc/ecc_types

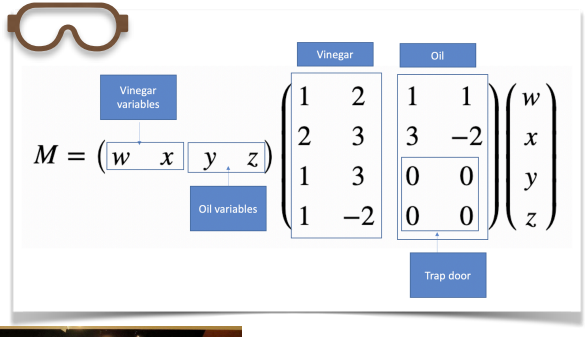
Quantum Robust Methods

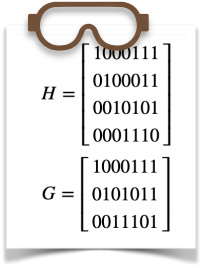
- **Hash-based/symmetric-based**. This includes Merkle signatures, SPHINCS and Picnic, and are used to create digital signatures. They cannot be used for public-key encryption and involve creating a range of private/public key pairs, and which are only used once. We must keep a track of the key pairs that we have used, in order to not use the again.
- **Code-based**. This includes McEliece and Niederreiter, and have been studied for decades.
- **Multivariate**. This involves multivariate quadratic methods. An example of this is the oil-and-vinegar method.
- **Lattice-based**. This includes NTRU (Nth degree TRUncated polynomial ring), learning with errors (LWE), Ring LWE, and Learning with Rounding. These have some of the best attributes for creating digital signatures, key exchange and encryption, and with reasonably small encryption keys and ciphertext sizes.
- **Isogenies**. This includes Supersinglar elliptic curve isogenies. These are interesting but are rather slow at the current time.











asecuritysite.com/pqc/

KEM and Digital Signatures/Encryption

For Public-Key Encryption and KEMs (Key Exchange) we have:

- **Classic McEliece**. This has been around for around 40 years, and has been shown to be fairly resistant to attack. It produces a fairly long encryption key, but produces a fairly small amount of ciphertext.
- **CRYSTALS-KYBER (Lattice)**. Uses LWE (Learning with Errors) with lattice methods. A new lattice attack was discovered within the period of the assessment, but it is hoped that an updated version of KYBER can be produced for the final assessment. NIST have some worried about its side-channel robustness, and is a strong contender for KEM.
- **NTRU (Lattice)**. This is a traditional structured lattice based approach, and has been around for longer than the other lattice methods — showing that it is perhaps more robust against attack and against IP claims.
- **SABER (Lattice)**. This is based on modular learning with rounding, and uses lattice methods. SABER has excellent performance, and is possibly near production ready. NIST's only recommendation is that updates should perhaps consider side-channel attacks.

Alternative: BIKE (Code-based); FrodoKEM (LWE); HQC (Code-based); NTRU Prime (Lattice/NTRU); and SIKE (Isogenies).

Digital signatures:

- **CRYSTALS-DILITHIUM (Lattice)**. At present, CRYSTALS (Cryptographic Suite for Algebraic Lattices) supports two quantum robust mechanisms: Kyber for key-encapsulation mechanism (KEM) and key exchange; and Dilithium for a digital signature algorithm. CRYSTALS Dilithium uses lattice-based Fiat-Shamir schemes, and produces one of the smallest signatures of all the post-quantum methods, and with relatively small public and private key sizes.
- **FALCON (Lattice)**. Falcon is one of the finalists for the NIST standard for PQC (Post Quantum Cryptography), along with NTRU (Nth degree-truncated polynomial ring units) and CRYSTALS-DILITHIUM. It is derived from NTRU and is a lattice-based method for quantum robust digital signing. Falcon is based on the Gentry, Peikert and Vaikuntanathan method for generating lattice-based signature schemes, along with a trapdoor sampler - Fast Fourier sampling.
- **SPHINCS+ (Hash-based signature).**

Alternative: GeMSS (Multivariate); Picnic (Hash-based ZKP); and SPHINCS+ (Hash-based stateless).

asecuritysite.com/pqc/

NIST

KEM and Public Key Encryption (Lattice)

CRYSTALS-KYBER (Lattice) - Key Encapsulation Mechanism (KEM)

[PQC Home][Home]

Post Quantum Cryptography

@asecuritysite.com

CRYSTALS (Cryptographic Suite for Algebraic Lattices) supports two quantum robust mechanisms: Kyber for key-encapsulation mechanism (KEM) and key exchange; and Dilithium for a digital signature algorithm. CRYSTALS-Kyber uses LWE (Learning with Errors) with lattice methods. A new lattice attack was discovered within the period of the assessment [1], but it is hoped that an updated version of Kyber can be produced for the final assessment. NIST have some worried about its side-channel robustness, but is a strong contender for KEM. Overall a KEM allows a symmetric key to be passed using public key methods. In this case, Alice will generate her key pair of a public key (pk) and a private key (sk). She then passes her public key to Bob, and then Bob creates a cipher text (ct) with Alice's public key. He passes the ciphertext to Alice, and who decrypts with her private key. This will reveal the key that Bob wants Alice to use. Kyber512 has a security level of AES-128, Kyber738 maps to AES-192, and Keyber1024 to AES-256. [Kyber KEM] [SABER KEM] [NTRU KEM] [McEliece KEM]

Parameters

type:

Kyber512

Determine

```
Public key size: 800
Secret key size: 1632
Ciphertext  size: 768
Public key (only showing 1/8 of key):
62656d5bcca4c509a6acc43528e60832352cc3c144a92492d4371f389452377263bff7027300b42655268a116ee1878e55172cf2b9bb0e7b
965f71c2185121070571659243423815 3b91b2f3d82c140abb48980061b383f43b8c654b8d9f8c97c0458eae
Secret key (only showing 1/8 of key):
2adb01c695cd22396bdc42a251b98e8d49980323c513595c6b3ab09fd89091802f991227098442ca1a8468b82f2d630c6363599400a13607
4191e8add930784a1285bc4ba1871aa00651537d365577607036b4b3b37891e6d5589ce6465a331d346b506439a85e464f5b1809d6a79f67
8bb167bba0fae9916b06056585c9a91c40a471928e304edd3337c3a60d615a1a236c16e313c880082a6ecc5ed90559c577bc01a27046312f
c2cb05f83a30ad29904c1236e3492f4902a55d11844f209e39f0bb8b986b5a02c43540cb
Cipher text (only showing 1/8 of ciphertext):
a896d3bd27520c91a5ea634c7628d60e29cd02ea6b2b999d28ab75c6f4e7d4881e965e533a9d042da4cf96ad76f00e03dfc80d5eb17f734d
aa07bf9a45fbf35cf3b920f5ac6b48fbc369f575b8895baea1c2ba79b37beb0b7291bb7627de97d3
Key (A): 1912f2a6b8cb95cd7462f798acc76591114bba2bd38bc6f008060d161379ebf0
Key (B): 1912f2a6b8cb95cd7462f798acc76591114bba2bd38bc6f008060d161379ebf0
Keys are the same
```

asecuritysite.com/pqc/

Lattice: NTRU

NTRU (Nth degree TRUncated polynomial ring) - Key Encapsulation Mechanism (KEM)

[Lattice Home][Home]

The NTRU (Nth degree TRUncated polynomial ring) public key method is a lattice-based method which is quantum robust. It uses the shortest vector problem in a lattice, and has an underpinning related to the difficulty of factorizing certain polynomials. NTRUhps2048509 has a security level of AES-128, NTRUhps2048677 maps to AES-192, and NTRUhps4096821 to AES-256. [Kyber KEM] [SABER KEM] [NTRU KEM] [McEliece KEM]

Parameters

type:

ntruhps2048509

Determine

```
NTRU for KEM NTRU

Public key size: 699
Secret key size: 935
Ciphertext  size: 699

Public key (only showing 1/8 of key):
18c206f7503f34ac695d17a1d32fc904b5f15c2bfd45a61b828121c5a07499bce251551b470efc1bef9fbb748242fd7e6c0c1b70bae398b3
eb257c611110080b8e61a78f7142590a81e337aa97aec21555424cac360bb5
Secret key (only showing 1/8 of key):
84be10b8d900ee55235719e71842499b2258d5ba2a174d4e034d8f9e37ecdc08c29420a9e568e1dfcb6a3c00cc97bbaa492e241d6e20d200
a15a043caff213ba5ed045354ae5a82338c08784c604aebb47c10b2fc366315033f21aa57faf6fa83e8501051e0f17742e7a5e533cb7ca3f
2fe5132c
Cipher text:
c0ba6be9dbfd1f7cfeb63d8e712a144d572d50d74379445784517030a8ec2a509145f2e8d956b3e27b876f4836df3d11a1743a702bea03ad
d783a76d1f50d4ba988c7f11660f922d2a483849eb42e68a2cd04644088705

Key (A): a64423c9182a760c265c38f1796be8f3a25fc4e9a8f1a9a33ce607da0294978a
Key (B): a64423c9182a760c265c38f1796be8f3a25fc4e9a8f1a9a33ce607da0294978a
Keys are the same
```

KEM and Public Key Encryption: Dilithium

[PQC Home][Home]

Post Quantum Cryptography

@asecuritysite.com

At present, CRYSTALS (Cryptographic Suite for Algebraic Lattices) supports two quantum robust mechanisms: Kyber for key-encapsulation mechanism (KEM) and key exchange; and Dilithium for a digital signature algorithm. CRYSTALS Dilithium uses lattice-based Fiat-Shamir schemes, and produces one of the smallest signatures of all the post-quantum methods, and with relatively small public and private key sizes. The three main implements for the parameters used are: Dilithium 2, Dilithium 3 and Dilithium 5. Overall, Dilithium 3 is equivalent to a 128-bit signature, and is perhaps the starting point for an implementation.

Parameters

Type:

Dilithium 2

Determine

```
CRYPTO_PUBLICKEYBYTES = 1312
CRYPTO_SECRETKEYBYTES = 2528
CRYPTO_BYTES = 2420
Signature Length = 2479

Alice Public key (only showing 1/8 of key):
253e20a27d1336ad7cc9bff15a77259da306cd2e6609890978b2a057f2710c8cf6960b566ad08bea6a78d56382a7feae8015b3cca6adec4c
e0f744db6434f0a321cded8d718fbca75f531e655d2b14aacd24ef02b2e28e906d31d7077e9843bfe00557f9be9fbbe1363545bbda4f6ec9
fb4985cb53d1098a2468836b2628787ef4ee5b61c81afd823124136a7f336d73281a66b91db28003889190fb5c189f3fcf559d88
Alice Secret key (only showing 1/8 of key):
253e20a27d1336ad7cc9bff15a77259da306cd2e6609890978b2a057f2710c8cf6960b566ad08bea6a78d56382a7feae8015b3cca6adec4c
e0f744db6434f0a321cded8d718fbca75f531e655d2b14aacd24ef02b2e28e906d31d7077e9843bfe00557f9be9fbbe1363545bbda4f6ec9
fb4985cb53d1098a2468836b2628787ef4ee5b61c81afd823124136a7f336d73281a66b91db28003889190fb5c189f3fcf559d88f610e4f6
ba1fdbdf0ee35d2909d20dee7276bd29a2c07d9337002a970122d61fa538fa623fe5e3a2ff58dc4760f464256a28965315cce6a58d57a61f
c38366f7eb65ef6f38a6d38ccf16bbb2ee6dd83dcf3b855023ef58e7c9daefb11dfd1dd8e31fd663491d544792137ede4cb296750d1f25e0
a726abcb1f9767c0fac839aea4c1dac3a43a800e9b18d621abc3a0f9501a3f0bb3526d16

Message:
c0f80eb870016ca0702134aeb80db576cfe5e36cfb1a2b8e39fd686ccb500ce3d442d0e4576e1103798b86161515f00420d66bbe5170d280
c146dc
Signature (1/8 of actual size):
14a3b28dd36f185ee98c5662d0c170b07567d125ecc20476e2f938dfb8072e1c46f0ee14a94a71f20cf9cf4b32220d45f0759781f04e339e
44200fdf1f3040fefaaad357438748a4cb22384fc78615cf691c797242dbc75872454e3b9f6d6fc18dadf95d7f16f02a2ff27d28cfdc18c7
```

Digital Signature: SPHINCS+

[PQC Home][Home]

Post Quantum Cryptography

@asecuritysite.com

SPHINCS is a stateless hash-based signature scheme, which is quantum robust. It was proposed by Bernstein et al. in 2015 [paper] and updated in [1]. SPHINCS+ 256 128-bit has a public key size of 32 bytes, a private key size of 64 bytes, and a signature of 17KB. It has been shown to operate at speeds of hundreds of hashes per second on a 4-core 3.5GHz processor. In this case we will implement using SHA-256. Other hashing methods are Haraka and SHAKE-256, for 128-bit, 192-bit and 256-bit versions [article].

Method	Public key size	Private key size	Signature size	Security level
SPHINCS+ SHA-256 128-bit	32	64	17,088	1 (128-bit)
SPHINCS+ SHA-256 192-bit	48	96	35,664	3 (192-bit)
SPHINCS+ SHA-256 256-bit	64	128	49,856	5 (256-bit)

Parameters

Type:

SPHINCS+ SHA-256 128

Determine

```
NAME: SPHINCS+
CRYPTO_PUBLICKEYBYTES = 32
CRYPTO_SECRETKEYBYTES = 64
CRYPTO_BYTES = 17088
Signature Length + Msg = 17147

Alice Public key: 7371f3aac2f0717f8fe5c529df4bd093eef5abadc881fbcbd8ce1b24f4960eb6
Alice Secret key:
83b0e9b3e4d4ef8ae4860408700e21d6dd84caf184e6dc5ab0688510bb0b943b7371f3aac2f0717f8fe5c529df4bd093eef5abadc881fbcb
d8ce1b24f4960eb6

Message:
83b0e9b3e4d4ef8ae4860408700e21d6dd84caf184e6dc5ab0688510bb0b943b7371f3aac2f0717f8fe5c529df4bd093408c239b6796c1c6
183226
Signature (Showing 1/64th of signature):
cb7b222abb1d152dbb84909e32dda90c13d44d985ff022515974f3c91f36223d749ee0055650f408d36580322cc12e1aed22d2cf4757cd44
```

asecuritysite.com/pqc/

Four Generations of HE

PQC Fourth Round Candidate Key-Establishment Mechanisms (KEMs)

The following candidate KEM algorithms will advance to the fourth round:

Public-Key Encryption/KEMs
BIKE
Classic McEliece
HQC
SIKE

In the first round, we have:

- **Code-based Signatures:** CROSS (Codes and Restricted Objects Signature Scheme); Enhanced pqsigRM; FuLeeca; LESS (Linear Equivalence Signature Scheme) and MEDS (Matrix Equivalence Digital Signature Wave).
- **Isogenies:** SQIsign.
- **Lattice based:** EagleSign; EHTv3 and EHTv4; HAETAE; HAWK; HuFu (Hash-and-Sign Signatures From Powerful Gadgets); Raccoon; and SQUIRRELS (Square Unstructured Integer Euclidean Lattice Signature).
- **MPC in the head:** MIRA; MiRitH (MinRank in the Head); MQOM (MQ on my Mind); PERK; RYDE; and SDitH (Syndrome Decoding in the Head).
- **Multivariate Signatures (Oil and Vinegar):** 3WISE; Biscuit; DME-Sign; HPPC (Hidden Product of Polynomial Composition); MAYO; PROV (PRovable unbalanced Oil and Vinegar); QR-UOV; SNOVA; TUOV (Triangular Unbalanced Oil and Vinegar); UOV (Unbalanced Oil and Vinegar); and VOX.
- **Symmetric-based Signatures:** AIMer; Ascon-Sign; FAEST; and SPHINCS-alpha.

Doing a quick count, we have:

Multivariate: 11; Lattice: 7; Code-based: 5; MPC-in-the-head: 5; Symmetric-based: 4; and Isogenies: 1.

KEM/Public Key Encryption

			Key Gen	Factor	Encap	Factor	Decap	Factor	Score (KG)	Score (Enc)	Score (Dec)	Overall
1	Kyber512-90s	Lattice (LWE)	30377	1	32386	1	19056	1	10	10	10	30
2	Kyber512	Lattice (LWE)	39455	1.3	44574	1.4	29462	1.5	10	10	10	30
3	Kyber768-90s	Lattice (LWE)	41042	1.4	44543	1.4	28685	1.5	10	10	10	30
4	Kyber1024-90s	Lattice (LWE)	53080	1.7	60272	1.9	41897	2.2	10	10	8	28
5	LightSaber-KEM	Lattice (LWE+R)	58722	1.9	59047	1.8	52131	2.7	10	10	8	28
6	Kyber768	Lattice (LWE)	59143	1.9	64410	2	45868	2.4	10	8	8	26
7	Kyber1024	Lattice (LWE)	75026	2.5	85229	2.6	65614	3.4	8	8	8	24
8	ntrulpr653	Lattice	82693	2.7	85179	2.6	92217	4.8	8	8	8	24
9	ntrulpr761	Lattice	86244	2.8	86863	2.7	94738	5	8	8	8	24
10	Saber-KEM	Lattice (LWE+R)	88626	2.9	92828	2.9	85405	4.5	8	8	8	24
11	ntrulpr857	Lattice	102071	3.4	108231	3.3	124951	6.6	8	8	8	24
12	FireSaber-KEM	Lattice (LWE+R)	126301	4.2	133791	4.1	128222	6.7	8	8	8	24
13	ntrulpr1277	Lattice	130320	4.3	135241	4.2	154612	8.1	8	8	8	24
14	NTRU-HPS-2048-509	Lattice	177211	5.8	47477	1.5	36974	1.9	8	10	10	28
15	HQC-128	Code-based	186524	6.1	305875	9.4	521661	27.4	8	8	5	21
16	NTRU-HRSS-701	Lattice	287042	9.4	44868	1.4	59252	3.1	8	10	8	26
17	NTRU-HPS-2048-677	Lattice	299514	9.9	65027	2	56855	3	8	8	8	24
18	HQC-192	Code-based	401412	13.2	690709	21.3	1096878	57.6	5	5	5	15
19	NTRU-HPS-4096-821	Lattice	406679	13.4	72840	2.2	71483	3.8	5	8	8	21
20	HQC-256	Code-based	697629	23	1213311	37.5	1966753	103.2	5	5	3	13
21	BIKE-L1	Code-based	734851	24.2	118519	3.7	5131009	269.3	5	8	3	16
22	sntrup653	Lattice	915914	30.2	75309	2.3	66592	3.5	5	8	8	21
23	sntrup761	Lattice	1051296	34.6	82427	2.5	70008	3.7	5	8	8	21
24	FrodoKEM-640-AES	Lattice (LWE)	1148982	37.8	1567672	48.4	1488040	78.1	5	5	5	15
25	sntrup857	Lattice	1237221	40.7	95152	2.9	94318	4.9	5	8	8	21
26	BIKE-L3	Code-based	2091161	68.8	299495	9.2	13853177	727	5	8	3	16
27	FrodoKEM-976-AES	Lattice (LWE)	2606237	85.8	4092307	126.4	3143510	165	5	3	3	11
28	sntrup1277	Lattice	2853083	93.9	122815	3.8	113776	6	5	8	8	21
29	FrodoKEM-640-SHAKE	Lattice (LWE)	3148720	103.7	4267582	131.8	4093162	214.8	3	3	3	9
30	FrodoKEM-1344-AES	Lattice (LWE)	4664746	153.6	5829221	180	5379988	282.3	3	3	3	9

1	Kyber512-90s	Lattice (LWE)	30
2	Kyber512	Lattice (LWE)	30
3	Kyber768-90s	Lattice (LWE)	30
4	Kyber1024-90s	Lattice (LWE)	28
5	LightSaber-KEM	Lattice (LWE+R)	28
6	NTRU-HPS-2048-509	Lattice	28
7	Kyber768	Lattice (LWE)	26
8	NTRU-HRSS-701	Lattice	26
9	Kyber1024	Lattice (LWE)	24
10	ntrulpr653	Lattice	24
11	ntrulpr761	Lattice	24
12	Saber-KEM	Lattice (LWE+R)	24
13	ntrulpr857	Lattice	24
14	FireSaber-KEM	Lattice (LWE+R)	24
15	ntrulpr1277	Lattice	24
16	NTRU-HPS-2048-677	Lattice	24
17	HQC-128	Code-based	21
18	NTRU-HPS-4096-821	Lattice	21
19	sntrup653	Lattice	21
20	sntrup761	Lattice	21
21	sntrup857	Lattice	21
22	sntrup1277	Lattice	21
23	Classic-McEliece-348864f	Code-based	18
24	BIKE-L1	Code-based	16
25	BIKE-L3	Code-based	16
26	Classic-McEliece-348864	Code-based	16
27	HQC-192	Code-based	15
28	FrodoKEM-640-AES	Lattice (LWE)	15
29	HQC-256	Code-based	13
30	NTRU-HPS-4096-1229	Lattice	13
31	NTRU-HRSS-1373	Lattice	13
32	Classic-McEliece-6688128	Code-based	13

31	SIDH-p434	Isogeny	5654091	186.1	11647942	359.7	4520865	237.2	3	3	3	9
32	SIKE-p434	Isogeny	6197253	204	10097567	311.8	10904398	572.2	3	3	3	9
33	FrodoKEM-976-SHAKE	Lattice (LWE)	7021826	231.2	9243416	285.4	9109536	478	3	3	3	9
34	SIDH-p503	Isogeny	7956605	261.9	15930604	491.9	6316223	331.5	3	3	3	9
35	SIKE-p503	Isogeny	8671491	285.5	14285437	441.1	15090345	791.9	3	3	3	9
36	SIKE-p434-compressed	Isogeny	11151196	367.1	16091778	496.9	11552400	606.2	3	3	3	9
37	SIDH-p434-compressed	Isogeny	11209821	369	16467316	508.5	5101392	267.7	3	3	3	9
38	NTRU-HPS-4096-1229	Lattice	12406657	408.4	460803	14.2	741261	38.9	3	5	5	13
39	FrodoKEM-1344-SHAKE	Lattice (LWE)	12601112	414.8	16107704	497.4	15400715	808.2	3	3	3	9
40	SIKE-p503-compressed	Isogeny	15026278	494.7	22251822	687.1	15891889	834	3	3	3	9
41	SIDH-p503-compressed	Isogeny	15242166	501.8	22995082	710	7088573	372	3	3	3	9
42	NTRU-HRSS-1373	Lattice	15922068	524.1	340073	10.5	895579	47	3	5	5	13
43	SIKE-p610	Isogeny	16516384	543.7	30973808	956.4	30794480	1616	3	3	0	6
44	SIDH-p610	Isogeny	17128219	563.9	32146290	992.6	13851706	726.9	3	3	3	9
45	SIDH-p751	Isogeny	23617554	777.5	48813951	1507.3	19233242	1009.3	3	0	0	3
46	SIKE-p751	Isogeny	26485988	871.9	42886151	1324.2	47206045	2477.2	3	0	0	3
47	SIDH-p610-compressed	Isogeny	30775700	1013.1	44161525	1363.6	15152561	795.2	0	0	3	3
48	SIKE-p610-compressed	Isogeny	33401019	1099.5	42572480	1314.5	32317016	1695.9	0	0	0	0
49	SIKE-p751-compressed	Isogeny	44187613	1454.6	68414496	2112.5	49102481	2576.7	0	0	0	0
50	SIDH-p751-compressed	Isogeny	44403930	1461.8	67441010	2082.4	21077729	1106.1	0	0	0	0
51	Classic-McEliece-348864f	Code-based	349054107	11490.7	61805	1.9	184433	9.7	0	10	8	18
52	Classic-McEliece-348864	Code-based	472549759	15556.2	81172	2.5	188381	9.9	0	8	8	16
53	Classic-McEliece-6688128	Code-based	899451646	29609.6	228951	7.1	529520	27.8	0	8	5	13
54	Classic-McEliece-460896f	Code-based	1109936192	36538.7	140594	4.3	430599	22.6	0	8	5	13
55	Classic-McEliece-6960119f	Code-based	1323860114	43581	187544	5.8	473180	24.8	0	8	5	13
56	Classic-McEliece-6688128f	Code-based	1400749128	46112.2	206238	6.4	511853	26.9	0	8	5	13
57	Classic-McEliece-8192128f	Code-based	1440024054	47405.1	202961	6.3	520936	27.3	0	8	5	13
58	Classic-McEliece-460896	Code-based	1496892143	49277.2	146998	4.5	426035	22.4	0	8	5	13
59	Classic-McEliece-6960119	Code-based	1587651773	52264.9	187845	5.8	464536	24.4	0	8	5	13
60	Classic-McEliece-8192128	Code-based	2654019753	87369.4	203049	6.3	526242	27.6	0	8	5	13

https://asecuritysite.com/pqc/

KEM/Public Key Encryption

		Level	Public	Factor	Cipher	Factor	Secret	Factor	Shared	Factor
1	SIDH-p434-compressed	1	197	1	197	1.5	28	1	110	6.9
2	SIKE-p434-compressed	1	197	1	236	1.8	350	12.5	16	1
3	SIDH-p503-compressed	2	225	1.1	225	1.8	32	1.1	126	7.9
4	SIKE-p503-compressed	2	225	1.1	280	2.2	407	14.5	24	1.5
5	SIDH-p610-compressed	3	274	1.4	274	2.1	39	1.4	154	9.6
6	SIKE-p610-compressed	3	274	1.4	336	2.6	491	17.5	24	1.5
7	SIDH-p434	1	330	1.7	330	2.6	28	1	110	6.9
8	SIKE-p434	1	330	1.7	346	2.7	374	13.4	16	1
9	SIDH-p751-compressed	5	335	1.7	335	2.6	48	1.7	188	11.8
10	SIKE-p751-compressed	5	335	1.7	410	3.2	602	21.5	32	2
11	SIDH-p503	2	378	1.9	378	3	32	1.1	126	7.9
12	SIKE-p503	2	378	1.9	402	3.1	434	15.5	24	1.5
13	SIDH-p610	3	462	2.3	462	3.6	39	1.4	154	9.6
14	SIKE-p610	3	462	2.3	486	3.8	524	18.7	24	1.5
15	SIDH-p751	5	564	2.9	564	4.4	48	1.7	188	11.8
16	SIKE-p751	5	564	2.9	596	4.7	644	23	32	2
17	LightSaber-KEM	1	672	3.4	736	5.8	1568	56	32	2
18	NTRU-HPS-2048-509	1	699	3.5	699	5.5	935	33.4	32	2
19	Kyber512	1	800	4.1	768	6	1632	58.3	32	2
20	Kyber512-90s	1	800	4.1	768	6	1632	58.3	32	2
21	ntrulpr653	1	897	4.6	1025	8	1125	40.2	32	2
22	NTRU-HPS-2048-677	3	930	4.7	930	7.3	1234	44.1	32	2
23	Saber-KEM	3	992	5	1088	8.5	2304	82.3	32	2
24	sntrup653	1	994	5	897	7	1518	54.2	32	2
25	ntrulpr761	2	1039	5.3	1167	9.1	1294	46.2	32	2
26	NTRU-HRSS-701	3	1138	5.8	1138	8.9	1450	51.8	32	2
27	sntrup761	2	1158	5.9	1039	8.1	1763	63	32	2
28	Kyber768	3	1184	6	1088	8.5	2400	85.7	32	2
29	Kyber768-90s	3	1184	6	1088	8.5	2400	85.7	32	2
30	ntrulpr857	3	1184	6	1312	10.3	1463	52.3	32	2
29	FrodoKEM-640-SHAKE	Lattice (LWE)	3148720	103.7	4267582	131.8	4093162	214.8	3	3	3	9
30	FrodoKEM-1344-AES	Lattice (LWE)	4664746	153.6	5829221	180	5379988	282.3	3	3	3	9

1	Kyber512-90s	Lattice (LWE)	30									
2	Kyber512	Lattice (LWE)	30									
3	Kyber768-90s	Lattice (LWE)	30									
4	Kyber1024-90s	Lattice (LWE)	28									
5	LightSaber-KEM	Lattice (LWE+R)	28									
6	NTRU-HPS-2048-509	Lattice	28									
7	Kyber768	Lattice (LWE)	26									
8	NTRU-HRSS-701	Lattice	26									
9	Kyber1024	Lattice (LWE)	24									
10	ntrulpr653	Lattice	24									
11	ntrulpr761	Lattice	24									
12	Saber-KEM	Lattice (LWE+R)	24									
13	ntrulpr857	Lattice	24									
14	FireSaber-KEM	Lattice (LWE+R)	24									
15	ntrulpr1277	Lattice	24									
16	NTRU-HPS-2048-677	Lattice	24									
17	HQC-128	Code-based	21									
18	NTRU-HPS-4096-821	Lattice	21									
19	sntrup653	Lattice	21									
20	sntrup761	Lattice	21									
21	sntrup857	Lattice	21									
22	sntrup1277	Lattice	21									
23	Classic-McEliece-348864f	Code-based	18									
24	BIKE-L1	Code-based	16									
25	BIKE-L3	Code-based	16									
26	Classic-McEliece-348864	Code-based	16									
27	HQC-192	Code-based	15									
28	FrodoKEM-640-AES	Lattice (LWE)	15									
29	HQC-256	Code-based	13									
30	NTRU-HPS-4096-1229	Lattice	13									
31	NTRU-HRSS-1373	Lattice	13									
32	Classic-McEliece-6688128	Code-based	13									

31	SIDH-p434	Isogeny	5654091	186.1	11647942	359.7	4520865	237.2	3	3	3	9
32	SIKE-p434	Isogeny	6197253	204	10097567	311.8	10904398	572.2	3	3	3	9
33	FrodoKEM-976-SHAKE	Lattice (LWE)	7021826	231.2	9243416	285.4	9109536	478	3	3	3	9
34	SIDH-p503	Isogeny	7956605	261.9	15930604	491.9	6316223	331.5	3	3	3	9
35	SIKE-p503	Isogeny	8671491	285.5	14285437	441.1	15090345	791.9	3	3	3	9
36	SIKE-p434-compressed	Isogeny	11151196	367.1	16091778	496.9	11552400	606.2	3	3	3	9
37	SIDH-p434-compressed	Isogeny	11209821	369	16467316	508.5	5101392	267.7	3	3	3	9
38	NTRU-HPS-4096-1229	Lattice	12406657	408.4	460803	14.2	741261	38.9	3	5	5	13
39	FrodoKEM-1344-SHAKE	Lattice (LWE)	12601112	414.8	16107704	497.4	15400715	808.2	3	3	3	9
40	SIKE-p503-compressed	Isogeny	15026278	494.7	22251822	687.1	15891889	834	3	3	3	9
41	SIDH-p503-compressed	Isogeny	15242166	501.8	22995082	710	7088573	372	3	3	3	9
42	NTRU-HRSS-1373	Lattice	15922068	524.1	340073	10.5	895579	47	3	5	5	13
43	SIKE-p610	Isogeny	16516384	543.7	30973808	956.4	30794480	1616	3	3	0	6
44	SIDH-p610	Isogeny	17128219	563.9	32146290	992.6	13851706	726.9	3	3	3	9
45	SIDH-p751	Isogeny	23617554	777.5	48813951	1507.3	19233242	1009.3	3	0	0	3
46	SIKE-p751	Isogeny	26485988	871.9	42886151	1324.2	47206045	2477.2	3	0	0	3
47	SIDH-p610-compressed	Isogeny	30775700	1013.1	44161525	1363.6	15152561	795.2	0	0	3	3
48	SIKE-p610-compressed	Isogeny	33401019	1099.5	42572480	1314.5	32317016	1695.9	0	0	0	0
49	SIKE-p751-compressed	Isogeny	44187613	1454.6	68414496	2112.5	49102481	2576.7	0	0	0	0
50	SIDH-p751-compressed	Isogeny	44403930	1461.8	67441010	2082.4	21077729	1106.1	0	0	0	0
51	Classic-McEliece-348864f	Code-based	349054107	11490.7	61805	1.9	184433	9.7	0	10	8	18
52	Classic-McEliece-348864	Code-based	472549759	15556.2	81172	2.5	188381	9.9	0	8	8	16
53	Classic-McEliece-6688128	Code-based	899451646	29609.6	228951	7.1	529520	27.8	0	8	5	13
54	Classic-McEliece-460896f	Code-based	1109936192	36538.7	140594	4.3	430599	22.6	0	8	5	13
55	Classic-McEliece-6960119f	Code-based	1323860114	43581	187544	5.8	473180	24.8	0	8	5	13
56	Classic-McEliece-6688128f	Code-based	1400749128	46112.2	206238	6.4	511853	26.9	0	8	5	13
57	Classic-McEliece-8192128f	Code-based	1440024054	47405.1	202961	6.3	520936	27.3	0	8	5	13
58	Classic-McEliece-460896	Code-based	1496892143	49277.2	146998	4.5	426035	22.4	0	8	5	13
59	Classic-McEliece-6960119	Code-based	1587651773	52264.9	187845	5.8	464536	24.4	0	8	5	13
60	Classic-McEliece-8192128	Code-based	2654019753	87369.4	203049	6.3	526242	27.6	0	8	5	13

https://asecuritysite.com/pqc/

KEM/Public Key Encryption

		Level	Public	Factor	Cipher	Factor	Secret	Factor	Shared	Factor
1	SIDH-p434-compressed	1	197	1	197	1.5	28	1	110	6.9
2	SIKE-p434-compressed	1	197	1	236	1.8	350	12.5	16	1
3	SIDH-p503-compressed	2	225	1.1	225	1.8	32	1.1	126	7.9
4	SIKE-p503-compressed	2	225	1.1	280	2.2	407	14.5	24	1.5
5	SIDH-p610-compressed	3	274	1.4	274	2.1	39	1.4	154	9.6
6	SIKE-p610-compressed	3	274	1.4	336	2.6	491	17.5	24	1.5
7	SIDH-p434	1	330	1.7	330	2.6	28	1	110	6.9
8	SIKE-p434	1	330	1.7	346	2.7	374	13.4	16	1
9	SIDH-p751-compressed	5	335	1.7	335	2.6	48	1.7	188	11.8
10	SIKE-p751-compressed	5	335	1.7	410	3.2	602	21.5	32	2
11	SIDH-p503	2	378	1.9	378	3	32	1.1	126	7.9
12	SIKE-p503	2	378	1.9	402	3.1	434	15.5	24	1.5
13	SIDH-p610	3	462	2.3	462	3.6	39	1.4	154	9.6
14	SIKE-p610	3	462	2.3	486	3.8	524	18.7	24	1.5
15	SIDH-p751	5	564	2.9	564	4.4	48	1.7	188	11.8
16	SIKE-p751	5	564	2.9	596	4.7	644	23	32	2
17	LightSaber-KEM	1	672	3.4	736	5.8	1568	56	32	2
18	NTRU-HPS-2048-509	1	699	3.5	699	5.5	935	33.4	32	2
19	Kyber512	1	800	4.1	768	6	1632	58.3	32	2
20	Kyber512-90s	1	800	4.1	768	6	1632	58.3	32	2
21	ntrulpr653	1	897	4.6	1025	8	1125	40.2	32	2
22	NTRU-HPS-2048-677	3	930	4.7	930	7.3	1234	44.1	32	2
23	Saber-KEM	3	992	5	1088	8.5	2304	82.3	32	2
24	sntrup653	1	994	5	897	7	1518	54.2	32	2
25	ntrulpr761	2	1039	5.3	1167	9.1	1294	46.2	32	2
26	NTRU-HRSS-701	3	1138	5.8	1138	8.9	1450	51.8	32	2
27	sntrup761	2	1158	5.9	1039	8.1	1763	63	32	2
28	Kyber768	3	1184	6	1088	8.5	2400	85.7	32	2
29	Kyber768-90s	3	1184	6	1088	8.5	2400	85.7	32	2
30	ntrulpr857	3	1184	6	1312	10.3	1463	52.3	32	2
29	FrodoKEM-640-SHAKE	Lattice (LWE)	3148720	103.7	4267582	131.8	4093162	214.8	3	3	3	9
30	FrodoKEM-1344-AES	Lattice (LWE)	4664746	153.6	5829221	180	5379988	282.3	3	3	3	9

1	Kyber512-90s	Lattice (LWE)	30
2	Kyber512	Lattice (LWE)	30
3	Kyber768-90s	Lattice (LWE)	30
4	Kyber1024-90s	Lattice (LWE)	28
5	LightSaber-KEM	Lattice (LWE+R)	28
6	NTRU-HPS-2048-509	Lattice	28
7	Kyber768	Lattice (LWE)	26
8	NTRU-HRSS-701	Lattice	26
9	Kyber1024	Lattice (LWE)	24
10	ntrulpr653	Lattice	24
11	ntrulpr761	Lattice	24
12	Saber-KEM	Lattice (LWE+R)	24
13	ntrulpr857	Lattice	24
14	FireSaber-KEM	Lattice (LWE+R)	24
15	ntrulpr1277	Lattice	24
16	NTRU-HPS-2048-677	Lattice	24
17	HQC-128	Code-based	21
18	NTRU-HPS-4096-821	Lattice	21
19	sntrup653	Lattice	21
20	sntrup761	Lattice	21
21	sntrup857	Lattice	21
22	sntrup1277	Lattice	21
23	Classic-McEliece-348864f	Code-based	18
24	BIKE-L1	Code-based	16
25	BIKE-L3	Code-based	16
26	Classic-McEliece-348864	Code-based	16
27	HQC-192	Code-based	15
28	FrodoKEM-640-AES	Lattice (LWE)	15
29	HQC-256	Code-based	13
30	NTRU-HPS-4096-1229	Lattice	13
31	NTRU-HRSS-1373	Lattice	13
32	Classic-McEliece-6688128	Code-based	13

31	NTRU-HPS-4096-821	5	1230	6.2	1230	9.6	1590	56.8	32	2
32	FireSaber-KEM	5	1312	6.7	1472	11.5	3040	108.6	32	2
33	sntrup857	3	1322	6.7	1184	9.3	1999	71.4	32	2
34	BIKE-L1	1	1541	7.8	1573	12.3	5223	186.5	32	2
35	Kyber1024	5	1568	8	1568	12.3	3168	113.1	32	2
36	Kyber1024-90s	5	1568	8	1568	12.3	3168	113.1	32	2
37	NTRU-HPS-4096-1229	5	1842	9.4	1842	14.4	2366	84.5	32	2
38	ntrulpr1277	5	1847	9.4	1975	15.4	2231	79.7	32	2
39	sntrup1277	5	2067	10.5	1847	14.4	3059	109.3	32	2
40	HQC-128	1	2249	11.4	4481	35	2289	81.8	64	4
41	NTRU-HRSS-1373	5	2401	12.2	2401	18.8	2983	106.5	32	2
42	BIKE-L3	3	3083	15.6	3115	24.3	10105	360.9	32	2
43	HQC-192	3	4522	23	9026	70.5	4562	162.9	64	4
44	HQC-256	5	7245	36.8	14469	113	7285	260.2	64	4
45	FrodoKEM-640-AES	1	9616	48.8	9720	75.9	19888	710.3	16	1
46	FrodoKEM-640-SHAKE	1	9616	48.8	9720	75.9	19888	710.3	16	1
47	FrodoKEM-976-AES	3	15632	79.4	15744	123	31296	1117.7	24	1.5
48	FrodoKEM-976-SHAKE	3	15632	79.4	15744	123	31296	1117.7	24	1.5
49	FrodoKEM-1344-AES	5	21520	109.2	21632	169	43088	1538.9	32	2
50	FrodoKEM-1344-SHAKE	5	21520	109.2	21632	169	43088	1538.9	32	2
51	Classic-McEliece-348864	1	261120	1325.5	128	1	6452	230.4	32	2
52	Classic-McEliece-348864f	1	261120	1325.5	128	1	6452	230.4	32	2
53	Classic-McEliece-460896	3	524160	2660.7	188	1.5	13568	484.6	32	2
54	Classic-McEliece-460896f	3	524160	2660.7	188	1.5	13568	484.6	32	2
55	Classic-McEliece-6688128	5	1044992	5304.5	240	1.9	13892	496.1	32	2
56	Classic-McEliece-6688128f	5	1044992	5304.5	240	1.9	13892	496.1	32	2
57	Classic-McEliece-6960119	5	1047319	5316.3	226	1.8	13908	496.7	32	2
58	Classic-McEliece-6960119f	5	1047319	5316.3	226	1.8	13908	496.7	32	2
59	Classic-McEliece-8192128	5	1357824	6892.5	240	1.9	14080	502.9	32	2
60	Classic-McEliece-8192128f	5	1357824	6892.5	240	1.9	14080	502.9	32	2

https://asecuritysite.com/pqc/

Digital Signatures

	Type	Method	Key gen	Factor	Sign	Factor	Verify	Score (Dec)	Overall	Score (Enc)	Score (Dec)	Overall
1	picnic3_L1	Hash/ZKP	21187	1	23348413	100.7	19186836	261.3	10	3	3	16
2	picnic_L3_full	Hash/ZKP	22571	1.1	10877666	46.9	12483100	170	10	5	5	20
3	picnic3_L3	Hash/ZKP	24184	1.1	47229332	203.7	36701854	499.9	10	3	5	18
4	picnic3_L5	Hash/ZKP	25035	1.2	75838669	327.1	58815102	801	10	3	5	18
5	picnic_L1_full	Hash/ZKP	25052	1.2	4854620	20.9	5917140	80.6	10	5	8	23
6	picnic_L5_full	Hash/ZKP	25414	1.2	15670149	67.6	13711927	186.7	10	5	5	20
7	picnic_L1_UR	Hash/ZKP	26617	1.3	9290846	40.1	7755920	105.6	10	5	5	20
8	picnic_L1_FS	Hash/ZKP	26839	1.3	8025732	34.6	6777198	92.3	10	5	8	23
9	picnic_L3_UR	Hash/ZKP	36216	1.7	23156076	99.9	21778246	296.6	10	5	5	20
10	picnic_L5_UR	Hash/ZKP	41418	2	39313637	169.5	34446115	469.1	8	3	5	16
11	picnic_L5_FS	Hash/ZKP	44942	2.1	37066573	159.9	28836125	392.7	8	3	5	16
12	Dilithium2-AES	Lattice	67697	3.2	231878	1	73424	1	8	10	10	28
13	picnic_L3_FS	Hash/ZKP	72919	3.4	20939121	90.3	18323702	249.6	8	5	3	16
14	Dilithium3-AES	Lattice	104515	4.9	374297	1.6	113536	1.5	8	10	10	28
15	Dilithium2	Lattice	116511	5.5	342726	1.5	112506	1.5	8	10	10	28
16	Dilithium5-AES	Lattice	164148	7.7	416647	1.8	166482	2.3	8	10	8	26
17	Dilithium3	Lattice	191331	9	534254	2.3	180350	2.5	8	8	8	24
18	Dilithium5	Lattice	307765	14.5	610807	2.6	417971	5.7	5	8	8	21
19	SPHINCS+-Haraka-128f-s	Hash	1114859	52.6	27587176	119	1521957	20.7	5	3	5	13
20	SPHINCS+-Haraka-128f-r	Hash	1296477	61.2	31847101	137.3	2308807	31.4	5	3	5	13
21	SPHINCS+-Haraka-192f-s	Hash	1733557	81.8	47717755	205.8	2509526	34.2	5	3	5	13
22	SPHINCS+-Haraka-192f-r	Hash	2034468	96	61820381	266.6	3691384	50.3	5	3	5	13
23	SPHINCS+-SHA256-128f-s	Hash	3088404	145.8	72191077	311.3	8962488	122.1	3	3	3	9
24	SPHINCS+-SHA256-192f-s	Hash	3920103	185	119085653	513.6	12269960	167.1	3	3	3	9
25	SPHINCS+-SHAKE256-128f-s	Hash	3993469	188.5	118778065	512.2	11837565	161.2	3	3	3	9
26	SPHINCS+-SHA256-128f-r	Hash	4576725	216	115180200	496.7	16236483	221.1	3	3	3	9
27	SPHINCS+-Haraka-256f-s	Hash	4656137	219.8	89990034	388.1	2534462	34.5	3	3	5	11
28	SPHINCS+-SHAKE256-192f-s	Hash	5766316	272.2	166899175	719.8	16662894	226.9	3	3	3	9
29	SPHINCS+-SHA256-192f-r	Hash	6343609	299.4	187556226	808.9	22966293	312.8	3	3	3	9
30	SPHINCS+-Haraka-256f-r	Hash	6398014	302	119210092	514.1	3793534	51.7	3	3	5	11
31	SPHINCS+-SHAKE256-128f-r	Hash	6831602	322.4	176444474	760.9	21769720	296.5	3	3	3	9
32	SPHINCS+-SHAKE256-192f-r	Hash	9735939	459.5	284106213	1225.2	30835025	420	3	0	3	6
33	SPHINCS+-SHA256-256f-s	Hash	10771651	508.4	220637782	951.5	12168947	165.7	3	3	3	9
34	SPHINCS+-SHAKE256-256f-s	Hash	15684219	740.3	318508169	1373.6	16422940	223.7	3	0	3	6
35	Falcon-512	Lattice	24656358	1163.7	1085984	4.7	183949	2.5	0	8	8	16
36	SPHINCS+-SHAKE256-256f-r	Hash	27044805	1276.5	564867404	2436.1	32709637	445.5	0	0	3	3
37	SPHINCS+-SHA256-256f-r	Hash	28940978	1366	571268028	2463.7	29935214	407.7	0	0	3	3
38	SPHINCS+-Haraka-256s-s	Hash	69028778	3258.1	998765490	4307.3	1330020	18.1	0	0	5	5
39	Falcon-1024	Lattice	74741342	3527.7	2204927	9.5	359553	4.9	0	8	8	16
40	SPHINCS+-Haraka-128s-r	Hash	78280311	3694.7	622976746	2686.7	1634310	22.3	0	0	5	5
41	SPHINCS+-Haraka-128s-s	Hash	79319904	3743.8	573371215	2472.7	665009	9.1	0	0	8	8
42	SPHINCS+-Haraka-256s-r	Hash	97499330	4601.8	1332794011	5747.8	2003259	27.3	0	0	5	5
43	Rainbow-V-Circumzenithal	MQ	104248763	4920.4	130818975	564.2	147868082	2013.9	3	0	3	0
44	SPHINCS+-Haraka-192s-s	Hash	107353443	5066.9	1055976583	4554	846444	11.5	0	0	5	5
45	SPHINCS+-Haraka-192s-r	Hash	122640669	5788.5	1322115681	5701.8	1335915	18.2	0	0	5	5
46	SPHINCS+-SHA256-256s-s	Hash	164581875	7768.1	2072066784	8936	5872652	80	0	0	5	5
47	SPHINCS+-SHA256-128s-s	Hash	177182865	8362.8	1487638276	6415.6	2787373	38	0	0	5	5
48	Rainbow-V-Compressed	MQ	200893415	9481.9	1728316371	7453.6	150730999	2052.9	3	0	0	0
49	SPHINCS+-SHAKE256-128s-s	Hash	245590268	11591.6	1847092410	7965.8	3595958	49	0	0	5	5
50	SPHINCS+-SHA256-192s-s	Hash	253977643	11987.4	2457142599	10596.7	4148624	56.5	0	0	5	5
51	SPHINCS+-SHAKE256-256s-s	Hash	255886895	12077.5	3085396286	13306.1	8930086	121.6	0	0	3	3
52	SPHINCS+-SHA256-128s-r	Hash	280035862	13217.3	2271055091	9794.2	5982596	81.5	0	0	5	5
53	SPHINCS+-SHAKE256-192s-s	Hash	360685288	17023.9	3486117790	15034.3	5642190	76.8	0	0	5	5
54	SPHINCS+-SHA256-192s-r	Hash	397446085	18759	3710077050	16000.1	8174834	111.3	0	0	3	3
55	SPHINCS+-SHAKE256-256s-r	Hash	428926274	20244.8	664184504	2864.4	16498665	224.7	0	0	3	3
56	SPHINCS+-SHAKE256-128s-r	Hash	446377018	21068.4	3274888436	14123.3	6274742	85.5	0	0	5	5
57	Rainbow-I-Classic	MQ	492275357	23234.8	6220698	26.8	6573578	89.5	0	5	5	10
58	Rainbow-I-Compressed	MQ	542931029	25625.7	235804438	1016.9	7520590	102.4	0	0	3	3
59	Rainbow-I-Circumzenithal	MQ	544285874	25689.6	6271496	27	7651253	104.2	0	5	3	8
60	SPHINCS+-SHA256-256s-r	Hash	635319074	29986.3	1889802242	8150	15630510	212.9	0	0	3	3
61	SPHINCS+-SHAKE256-192s-r	Hash	661757883	31234.1	1701951015	7339.9	10728453	146.1	0	0	3	3
62	Rainbow-V-Classic	MQ	1707296542	80582.3	132498942	571.4	139224699	1896.2	0	3	0	3
63	Rainbow-III-Classic	MQ	2306828504	108879.4	59391493	256.1	61864256	842.6	0	3	3	6
64	Rainbow-III-Circumzenithal	MQ	3264803255	154094.6	59377096	256.1	70084535	954.5	0	3	3	6
65	Rainbow-III-Compressed	MQ	3292975486	155424.3	3593929070	15499.2	75246328	1024.8	0	0	0	0

1	Dilithium2-AES	28
2	Dilithium3-AES	28
3	Dilithium2	28
4	Dilithium5-AES	26
5	Dilithium3	24
6	picnic_L1_full	23
7	picnic_L1_FS	23
8	Dilithium5	21
9	picnic_L3_full	20
10	picnic_L5_full	20
11	picnic_L1_UR	20
12	picnic_L3_UR	20
13	picnic3_L3	18
14	picnic3_L5	18
15	picnic3_L1	16
16	picnic_L5_UR	16
17	picnic_L5_FS	16
18	picnic_L3_FS	16
19	Falcon-512	16
20	Falcon-1024	16
21	SPHINCS+-Haraka-12	13
22	SPHINCS+-Haraka-12	13
23	SPHINCS+-Haraka-19	13
24	SPHINCS+-Haraka-19	13
25	SPHINCS+-Haraka-25	11
26	SPHINCS+-Haraka-25	11
27	Rainbow-I-Classic	10
28	SPHINCS+-SHA256-12	9
29	SPHINCS+-SHA256-19	9

https://asecuritysite.com/pqc/

Digital Signatures

	Method	Level	Pub key(B)	Factor	Pri key (B)	Factor	Sig (B)	Factor
1	SPHINCS+-Haraka-128f-robust	1	32	1	64	1.3	17088	258.9
2	SPHINCS+-Haraka-128f-simple	1	32	1	64	1.3	17088	258.9
3	SPHINCS+-Haraka-128s-robust	1	32	1	64	1.3	7856	119
4	SPHINCS+-Haraka-128s-simple	1	32	1	64	1.3	7856	119
5	SPHINCS+-SHA256-128f-robust	1	32	1	64	1.3	17088	258.9
6	SPHINCS+-SHA256-128f-simple	1	32	1	64	1.3	17088	258.9
7	SPHINCS+-SHA256-128s-robust	1	32	1	64	1.3	7856	119
8	SPHINCS+-SHA256-128s-simple	1	32	1	64	1.3	7856	119
9	SPHINCS+-SHAKE256-128f-robust	1	32	1	64	1.3	17088	258.9
10	SPHINCS+-SHAKE256-128f-simple	1	32	1	64	1.3	17088	258.9
11	SPHINCS+-SHAKE256-128s-robust	1	32	1	64	1.3	7856	119
12	SPHINCS+-SHAKE256-128s-simple	1	32	1	64	1.3	7856	119
13	picnic_L1_FS	1	33	1	49	1	34036	515.7
14	picnic_L1_UR	1	33	1	49	1	53965	817.7
15	picnic_L1_full	1	35	1.1	52	1.1	32065	485.8
16	picnic3_L1	1	35	1.1	52	1.1	14612	221.4
17	SPHINCS+-Haraka-192f-robust	3	48	1.5	96	2	35664	540.4
18	SPHINCS+-Haraka-192f-simple	3	48	1.5	96	2	35664	540.4
19	SPHINCS+-Haraka-192s-robust	3	48	1.5	96	2	16224	245.8
20	SPHINCS+-Haraka-192s-simple	3	48	1.5	96	2	16224	245.8
21	SPHINCS+-SHA256-192f-robust	3	48	1.5	96	2	35664	540.4
22	SPHINCS+-SHA256-192f-simple	3	48	1.5	96	2	35664	540.4
23	SPHINCS+-SHA256-192s-robust	3	48	1.5	96	2	16224	245.8
24	SPHINCS+-SHA256-192s-simple	3	48	1.5	96	2	16224	245.8
25	SPHINCS+-SHAKE256-192f-robust	3	48	1.5	96	2	35664	540.4
26	SPHINCS+-SHAKE256-192f-simple	3	48	1.5	96	2	35664	540.4
27	SPHINCS+-SHAKE256-192s-robust	3	48	1.5	96	2	16224	245.8
28	SPHINCS+-SHAKE256-192s-simple	3	48	1.5	96	2	16224	245.8
29	picnic_L3_FS	3	49	1.5	73	1.5	76776	1163.3
30	picnic_L3_UR	3	49	1.5	73	1.5	121849	1846.2

29	SPHINCS+-SHA256-192f-r	Hash	6343609	299.4	187556226	808.9	22966293	312.8	3	3	3	9
30	SPHINCS+-Haraka-256f-r	Hash	6398014	302	119210092	514.1	3793534	51.7	3	3	5	11
31	SPHINCS+-SHAKE256-128f-r	Hash	6831602	322.4	176444474	760.9	21769720	296.5	3	3	3	9
32	SPHINCS+-SHAKE256-192f-r	Hash	9735939	459.5	284106213	1225.2	30835025	420	3	0	3	6
33	SPHINCS+-SHA256-256f-s	Hash	10771651	508.4	220637782	951.5	12168947	165.7	3	3	3	9
34	SPHINCS+-SHAKE256-256f-s	Hash	15684219	740.3	318508169	1373.6	16422940	223.7	3	0	3	6
35	Falcon-512	Lattice	24656358	1163.7	1085984	4.7	183949	2.5	0	8	8	16
36	SPHINCS+-SHAKE256-256f-r	Hash	27044805	1276.5	564867404	2436.1	32709637	445.5	0	0	3	3
37	SPHINCS+-SHA256-256f-r	Hash	28940978	1366	571268028	2463.7	29935214	407.7	0	0	3	3
38	SPHINCS+-Haraka-256s-s	Hash	69028778	3258.1	998765490	4307.3	1330020	18.1	0	0	5	5
39	Falcon-1024	Lattice	74741342	3527.7	2204927	9.5	359553	4.9	0	8	8	16
40	SPHINCS+-Haraka-128s-r	Hash	78280311	3694.7	622976746	2686.7	1634310	22.3	0	0	5	5
41	SPHINCS+-Haraka-128s-s	Hash	79319904	3743.8	573371215	2472.7	665009	9.1	0	0	8	8
42	SPHINCS+-Haraka-256s-r	Hash	97499330	4601.8	1332794011	5747.8	2003259	27.3	0	0	5	5
43	Rainbow-V-Circumzenithal	MQ	104248763	4920.4	130818975	564.2	147868082	2013.9	3	0	3	0
44	SPHINCS+-Haraka-192s-s	Hash	107353443	5066.9	1055976583	4554	846444	11.5	0	0	5	5
45	SPHINCS+-Haraka-192s-r	Hash	122640669	5788.5	1322115681	5701.8	1335915	18.2	0	0	5	5
46	SPHINCS+-SHA256-256s-s	Hash	164581875	7768.1	2072066784	8936	5872652	80	0	0	5	5
47	SPHINCS+-SHA256-128s-s	Hash	177182865	8362.8	1487638276	6415.6	2787373	38	0	0	5	5
48	Rainbow-V-Compressed	MQ	200893415	9481.9	1728316371	7453.6	150730999	2052.9	3	0	0	0
49	SPHINCS+-SHAKE256-128s-s	Hash	245590268	11591.6	1847092410	7965.8	3595958	49	0	0	5	5
50	SPHINCS+-SHA256-192s-s	Hash	253977643	11987.4	2457142599	10596.7	4148624	56.5	0	0	5	5
51	SPHINCS+-SHAKE256-256s-s	Hash	255886895	12077.5	3085396286	13306.1	8930086	121.6	0	0	3	3
52	SPHINCS+-SHA256-128s-r	Hash	280035862	13217.3	2271055091	9794.2	5982596	81.5	0	0	5	5
53	SPHINCS+-SHAKE256-192s-s	Hash	360685288	17023.9	3486117790	15034.3	5642190	76.8	0	0	5	5
54	SPHINCS+-SHA256-192s-r	Hash	397446085	18759	3710077050	16000.1	8174834	111.3	0	0	3	3
55	SPHINCS+-SHAKE256-256s-r	Hash	428926274	20244.8	664184504	2864.4	16498665	224.7	0	0	3	3

56	SPHINCS+-SHAKE256-128s-r	Hash	446377018	21068.4	3274888436	14123.3	6274742	85.5	0	0	5	5
57	Rainbow-I-Classic	MQ	492275357	23234.8	6220698	26.8	6573578	89.5	0	5	5	10
58	Rainbow-I-Compressed	MQ	542931029	25625.7	235804438	1016.9	7520590	102.4	0	0	3	3
59	Rainbow-I-Circumzenithal	MQ	544285874	25689.6	6271496	27	7651253	104.2	0	5	3	8
60	SPHINCS+-SHA256-256s-r	Hash	635319074	29986.3	1889802242	8150	15630510	212.9	0	0	3	3
61	SPHINCS+-SHAKE256-192s-r	Hash	661757883	31234.1	1701951015	7339.9	10728453	146.1	0	0	3	3
62	Rainbow-V-Classic	MQ	1707296542	80582.3	132498942	571.4	139224699	1896.2	0	3	0	3
63	Rainbow-III-Classic	MQ	2306828504	108879.4	59391493	256.1	61864256	842.6	0	3	3	6
64	Rainbow-III-Circumzenithal	MQ	3264803255	154094.6	59377096	256.1	70084535	954.5	0	3	3	6
65	Rainbow-III-Compressed	MQ	3292975486	155424.3	3593929070	15499.2	75246328	1024.8	0	0	0	0

1	Dilithium2-AES	28
2	Dilithium3-AES	28
3	Dilithium2	28
4	Dilithium5-AES	26
5	Dilithium3	24
6	picnic_L1_full	23
7	picnic_L1_FS	23
8	Dilithium5	21
9	picnic_L3_full	20
10	picnic_L5_full	20
11	picnic_L1_UR	20
12	picnic_L3_UR	20
13	picnic3_L3	18
14	picnic3_L5	18
15	picnic3_L1	16
16	picnic_L5_UR	16
17	picnic_L5_FS	16
18	picnic_L3_FS	16
19	Falcon-512	16
20	Falcon-1024	16
21	SPHINCS+-Haraka-12	13
22	SPHINCS+-Haraka-12	13
23	SPHINCS+-Haraka-19	13
24	SPHINCS+-Haraka-19	13
25	SPHINCS+-Haraka-25	11
26	SPHINCS+-Haraka-25	11
27	Rainbow-I-Classic	10
28	SPHINCS+-SHA256-12	9
29	SPHINCS+-SHA256-19	9

https://asecuritysite.com/pqc/

Digital Signatures

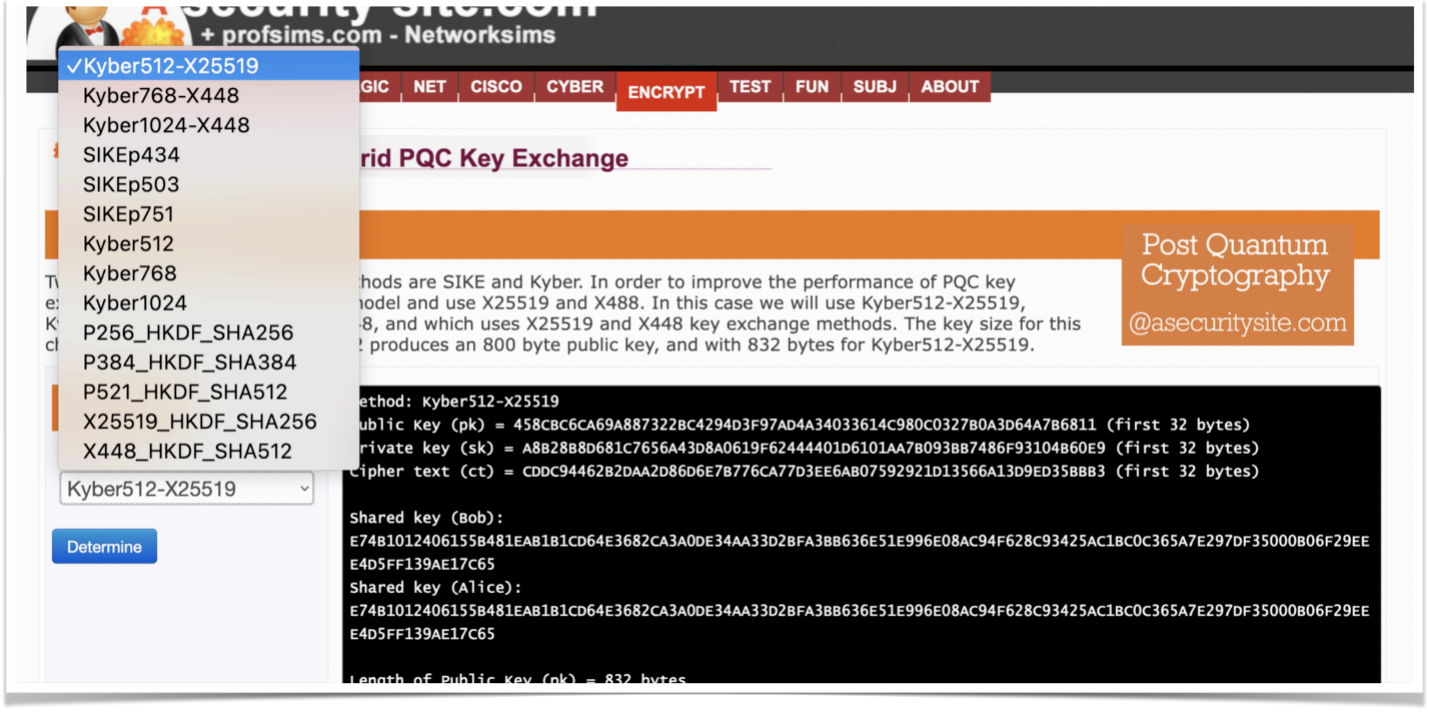
	Method	Level	Pub key(B)	Factor	Pri key (B)	Factor	Sig (B)	Factor
1	SPHINCS+-Haraka-128f-robust	1	32	1	64	1.3	17088	258.9
2	SPHINCS+-Haraka-128f-simple	1	32	1	64	1.3	17088	258.9
3	SPHINCS+-Haraka-128s-robust	1	32	1	64	1.3	7856	119
4	SPHINCS+-Haraka-128s-simple	1	32	1	64	1.3	7856	119
5	SPHINCS+-SHA256-128f-robust	1	32	1	64	1.3	17088	258.9
6	SPHINCS+-SHA256-128f-simple	1	32	1	64	1.3	17088	258.9
7	SPHINCS+-SHA256-128s-robust	1	32	1	64	1.3	7856	119
8	SPHINCS+-SHA256-128s-simple	1	32	1	64	1.3	7856	119
9	SPHINCS+-SHAKE256-128f-robust	1	32	1	64	1.3	17088	258.9
10	SPHINCS+-SHAKE256-128f-simple	1	32	1	64	1.3	17088	258.9
11	SPHINCS+-SHAKE256-128s-robust	1	32	1	64	1.3	7856	119
12	SPHINCS+-SHAKE256-128s-simple	1	32	1	64	1.3	7856	119
13	picnic_L1_FS	1	33	1	49	1	34036	515.7
14	picnic_L1_UR	1	33	1	49	1	53965	817.7
15	picnic_L1_full	1	35	1.1	52	1.1	32065	485.8
16	picnic3_L1	1	35	1.1	52	1.1	14612	221.4
17	SPHINCS+-Haraka-192f-robust	3	48	1.5	96	2	35664	540.4
18	SPHINCS+-Haraka-192f-simple	3	48	1.5	96	2	35664	540.4
19	SPHINCS+-Haraka-192s-robust	3	48	1.5	96	2	16224	245.8
20	SPHINCS+-Haraka-192s-simple	3	48	1.5	96	2	16224	245.8
21	SPHINCS+-SHA256-192f-robust	3	48	1.5	96	2	35664	540.4
22	SPHINCS+-SHA256-192f-simple	3	48	1.5	96	2	35664	540.4
23	SPHINCS+-SHA256-192s-robust	3	48	1.5	96	2	16224	245.8
24	SPHINCS+-SHA256-192s-simple	3	48	1.5	96	2	16224	245.8
25	SPHINCS+-SHAKE256-192f-robust	3	48	1.5	96	2	35664	540.4
26	SPHINCS+-SHAKE256-192f-simple	3	48	1.5	96	2	35664	540.4
27	SPHINCS+-SHAKE256-192s-robust	3	48	1.5	96	2	16224	245.8
28	SPHINCS+-SHAKE256-192s-simple	3	48	1.5	96	2	16224	245.8
29	picnic_L3_FS	3	49	1.5	73	1.5	76776	1163.3
30	picnic_L3_UR	3	49	1.5	73	1.5	121849	1846.2
31	picnic_L3_full	3	49	1.5	73	1.5	71183	1078.5
32	picnic3_L3	3	49	1.5	73	1.5	35028	530.7
33	SPHINCS+-Haraka-256f-robust	5	64	2	128	2.6	49856	755.4
34	SPHINCS+-Haraka-256f-simple	5	64	2	128	2.6	49856	755.4
35	SPHINCS+-Haraka-256s-robust	5	64	2	128	2.6	29792	451.4
36	SPHINCS+-Haraka-256s-simple	5	64	2	128	2.6	29792	451.4
37	SPHINCS+-SHA256-256f-robust	5	64	2	128	2.6	49856	755.4
38	SPHINCS+-SHA256-256f-simple	5	64	2	128	2.6	49856	755.4
39	SPHINCS+-SHA256-256s-robust	5	64	2	128	2.6	29792	451.4
40	SPHINCS+-SHA256-256s-simple	5	64	2	128	2.6	29792	451.4
41	SPHINCS+-SHAKE256-256f-robust	5	64	2	128	2.6	49856	755.4
42	SPHINCS+-SHAKE256-256f-simple	5	64	2	128	2.6	49856	755.4
43	SPHINCS+-SHAKE256-256s-robust	5	64	2	128	2.6	29792	451.4
44	SPHINCS+-SHAKE256-256s-simple	5	64	2	128	2.6	29792	451.4
45	picnic_L5_FS	5	65	2	97	2	132860	2013
46	picnic_L5_UR	5	65	2	97	2	209510	3174.4
47	picnic_L5_full	5	65	2	97	2	126290	1913.5
48	picnic3_L5	5	65	2	97	2	61028	924.7
49	Falcon-512	1	897	28	1281	26.1	690	10.5
50	Dilithium2	2	1312	41	2528	51.6	2420	36.7
51	Dilithium2-AES	2	1312	41	2528	51.6	2420	36.7
52	Falcon-1024	5	1793	56	2305	47	1330	20.2
53	Dilithium3	3	1952	61	4000	81.6	3293	49.9
54	Dilithium3-AES	3	1952	61	4000	81.6	3293	49.9
55	Dilithium5	5	2592	81	4864	99.3	4595	69.6
56	Dilithium5-AES	5	2592	81	4864	99.3	4595	69.6
57	Rainbow-I-Circumzenithal	1	60192	1881	103648	2115.3	66	1
58	Rainbow-I-Compressed	1	60192	1881	64	1.3	66	1
59	Rainbow-I-Classic	1	161600	5050	103648	2115.3	66	1
60	Rainbow-III-Circumzenithal	3	264608	8269	626048	12776.5	164	2.5
61	Rainbow-III-Compressed	3	264608	8269	64	1.3	164	2.5

56	SPHINCS+-SHAKE256-128s-r	Hash	446377018	21068.4	3274888436	14123.3	6274742	85.5	0	0	5	5
57	Rainbow-I-Classic	MQ	492275357	23234.8	6220698	26.8	6573578	89.5	0	5	5	10
58	Rainbow-I-Compressed	MQ	542931029	25625.7	235804438	1016.9	7520590	102.4	0	0	3	3
59	Rainbow-I-Circumzenithal	MQ	544285874	25689.6	6271496	27	7651253	104.2	0	5	3	8
60	SPHINCS+-SHA256-256s-r	Hash	635319074	29986.3	1889802242	8150	15630510	212.9	0	0	3	3
61	SPHINCS+-SHAKE256-192s-r	Hash	661757883	31234.1	1701951015	7339.9	10728453	146.1	0	0	3	3
62	Rainbow-V-Classic	MQ	1707296542	80582.3	132498942	571.4	139224699	1896.2	0	3	0	3
63	Rainbow-III-Classic	MQ	2306828504	108879.4	59391493	256.1	61864256	842.6	0	3	3	6
64	Rainbow-III-Circumzenithal	MQ	3264803255	154094.6	59377096	256.1	70084535	954.5	0	3	3	6
65	Rainbow-III-Compressed	MQ	3292975486	155424.3	3593929070	15499.2	75246328	1024.8	0	0	0	0

1	Dilithium2-AES	28
2	Dilithium3-AES	28
3	Dilithium2	28
4	Dilithium5-AES	26
5	Dilithium3	24
6	picnic_L1_full	23
7	picnic_L1_FS	23
8	Dilithium5	21
9	picnic_L3_full	20
10	picnic_L5_full	20
11	picnic_L1_UR	20
12	picnic_L3_UR	20
13	picnic3_L3	18
14	picnic3_L5	18
15	picnic3_L1	16
16	picnic_L5_UR	16
17	picnic_L5_FS	16
18	picnic_L3_FS	16
19	Falcon-512	16
20	Falcon-1024	16
21	SPHINCS+-Haraka-12	13
22	SPHINCS+-Haraka-12	13
23	SPHINCS+-Haraka-19	13
24	SPHINCS+-Haraka-19	13
25	SPHINCS+-Haraka-25	11
26	SPHINCS+-Haraka-25	11
27	Rainbow-I-Classic	10
28	SPHINCS+-SHA256-12	9
29	SPHINCS+-SHA256-19	9

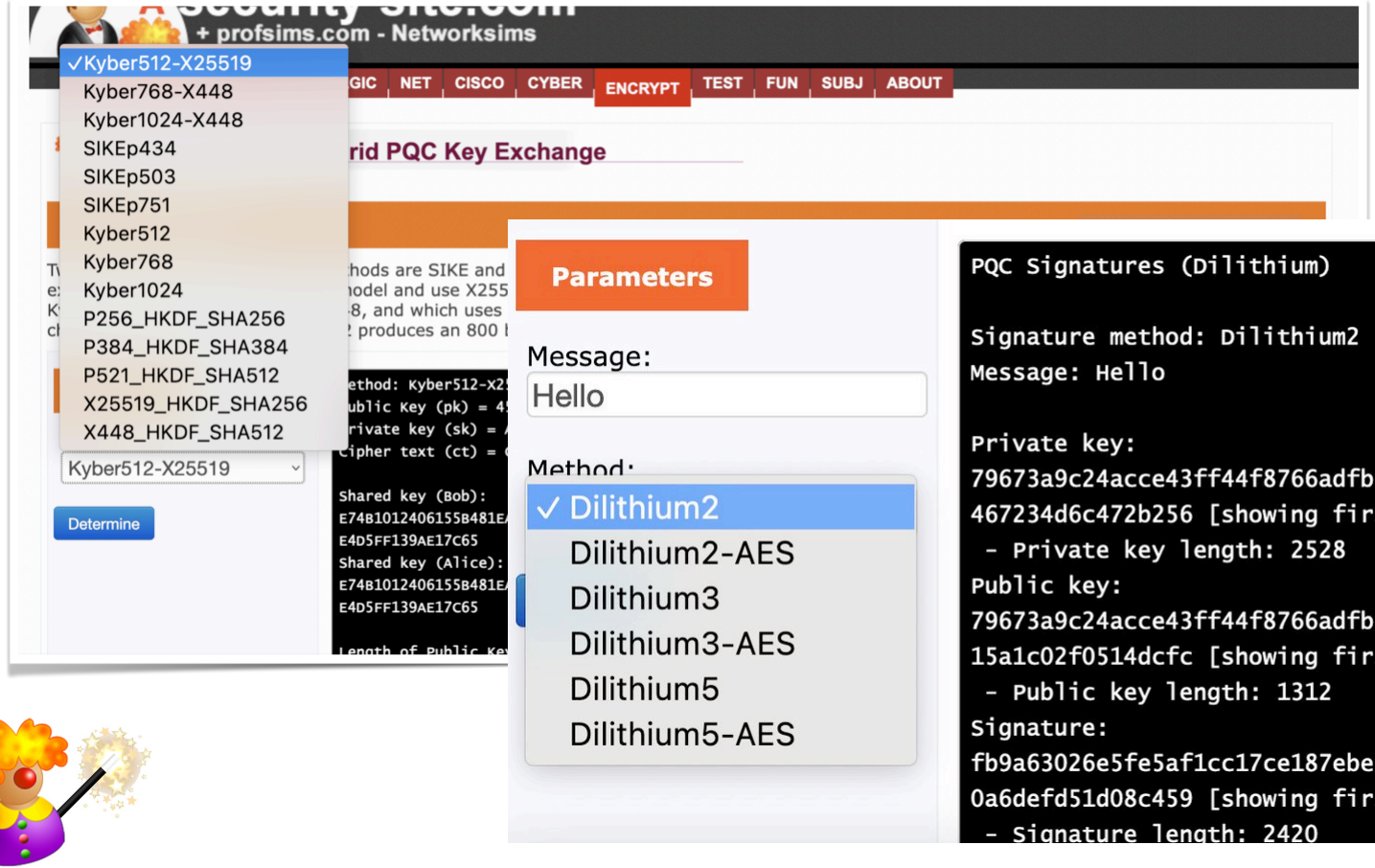
https://asecuritysite.com/pqc/

Hybrid Key Exchange/Digital Signatures



Hybrid

Hybrid Key Exchange/Digital Signatures



Hybrid

Key Exchange/Signatures

Type	Public key size (B)	Secret key size (B)	Ciphertext size (B)	
Kyber512	800	1,632	768	Learning with errors (Lattice)
Kyber738	1,184	2,400	1,088	Learning with errors (Lattice)
Kyber1024	1,568	3,168	1,568	Learning with errors (Lattice)
LightSABER	672	1,568	736	Learning with rounding (Lattice)
SABER	992	2,304	1,088	Learning with rounding (Lattice)
FireSABER	1,312	3,040	1,472	Learning with rounding (Lattice)
McEliece348864	261,120	6,452	128	Code based
McEliece460896	524,160	13,568	188	Code based
McEliece6688128	1,044,992	13,892	240	Code based
McEliece6960119	1,047,319	13,948	226	Code based
McEliece8192128	1,357,824	14,120	240	Code based
NTRUhps2048509	699	935	699	Lattice
NTRUhps2048677	930	1,234	930	Lattice
NTRUhps4096821	1,230	1,590	1,230	Lattice
SIKEp434	330	44	346	Isogeny
SIKEp503	378	56	402	Isogeny
SIKEp751	564	80	596	Isogeny
SIDH	564	48	596	Isogeny

Hybrid

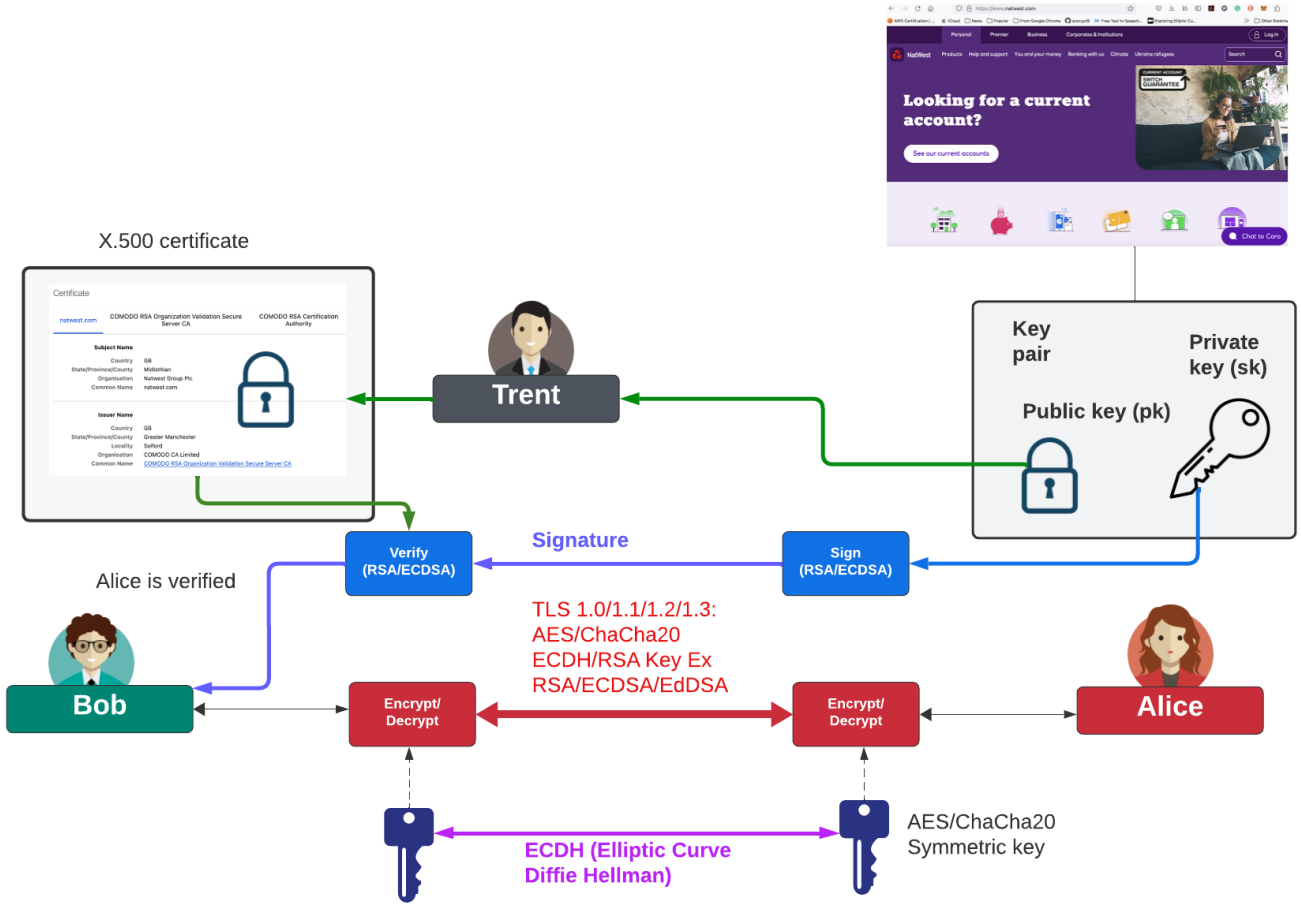
Key Exchange/Signatures

Type	Public key size (B)	Secret key size (B)	Ciphertext size (B)	
Kyber512	800	1,632	768	Learning with errors (Lattice)
Kyber738	1,184	2,400	1,088	Learning with errors (Lattice)
Kyber1024	1,568	3,168	1,568	Learning with errors (Lattice)
LightSABER	672	1,568	736	Learning with rounding (Lattice)
SABER	992	2,304	1,088	Learning with rounding (Lattice)
FireSABER	1,312	3,040	1,472	Learning with rounding (Lattice)
McEliece348864	261,120	6,452	128	Code based
McEliece460896	524,160	13,568	188	Code based
McEliece6688128	1,044,992	13,892	240	Code based
McEliece6960119	1,047,319			
McEliece8192128	1,357,824			
NTRUhps2048509	699			
NTRUhps2048677	930			
NTRUhps4096821	1,230			
SIKEp434	330			
SIKEp503	378			
SIKEp751	564			
SIDH	564			

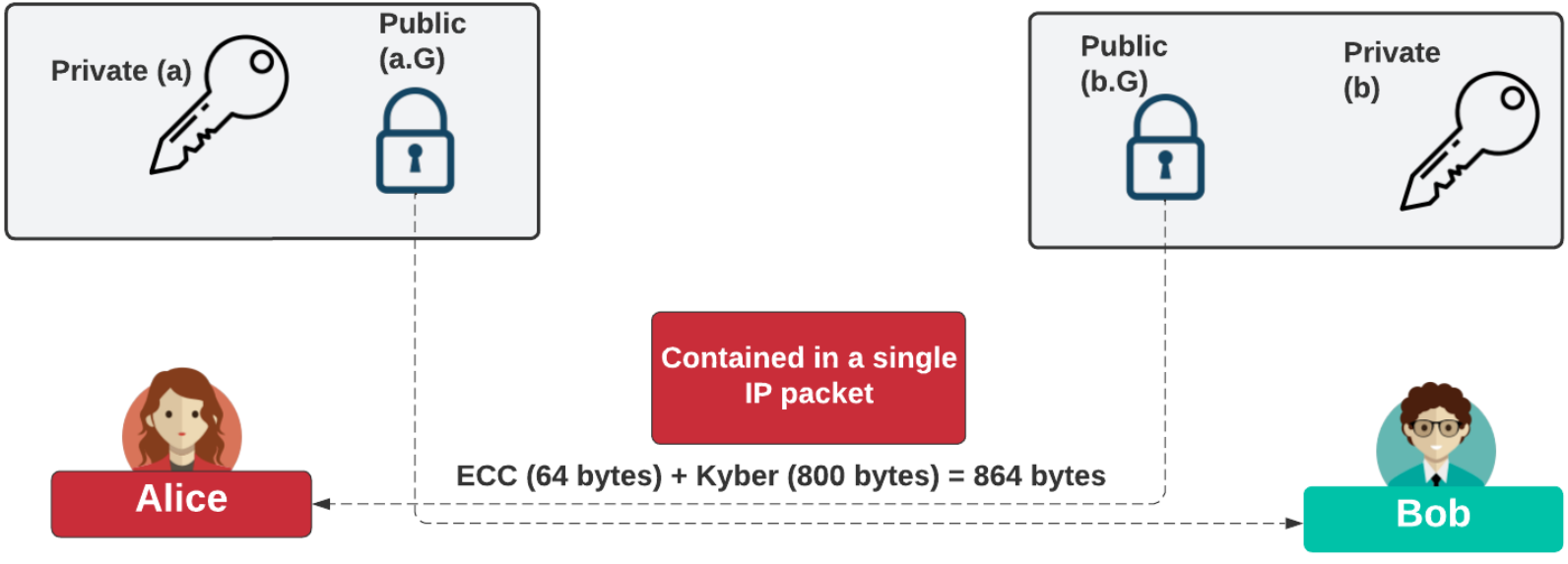
Method	Public key size	Private key size	Signature size	Security level
Crystals Dilithium 2 (Lattice)	1,312	2,528	2,420	1 (128-bit) Lattice
Crystals Dilithium 3	1,952	4,000	3,293	3 (192-bit) Lattice
Crystals Dilithium 5	2,592	4,864	4,595	5 (256-bit) Lattice
Falcon 512 (Lattice)	897	1,281	690	1 (128-bit) Lattice
Falcon 1024	1,793	2,305	1,330	5 (256-bit) Lattice
Rainbow Level Ia (Oil-and-Vineger)	161,600	103,648	66	1 (128-bit) Multivariate (UOV)
Rainbow Level IIIa	861,400	611,300	164	3 (192-bit) Multivariate (UOV)
Rainbow Level Vc	1,885,400	1,375,700	204	5 (256-bit) Multivariate (UOV)
Sphincs SHA256-128f Simple	32	64	17,088	1 (128-bit) Hash-based
Sphincs SHA256-192f Simple	48	96	35,664	3 (192-bit) Hash-based
Sphincs SHA256-256f Simple	64	128	49,856	5 (256-bit) Hash-based
Picnic 3 Full	49	73	71,179	3 (192-bit) Symmetric
GeMSS 128	352,188	16	33	1 (128-bit) Multivariate (HFEv-)
GeMSS 192	1,237,964	24	53	1 (128-bit) Multivariate (HFEv-)
RSA-2048	256	256	256	
ECC 256-bit	64	32	256	

Hybrid

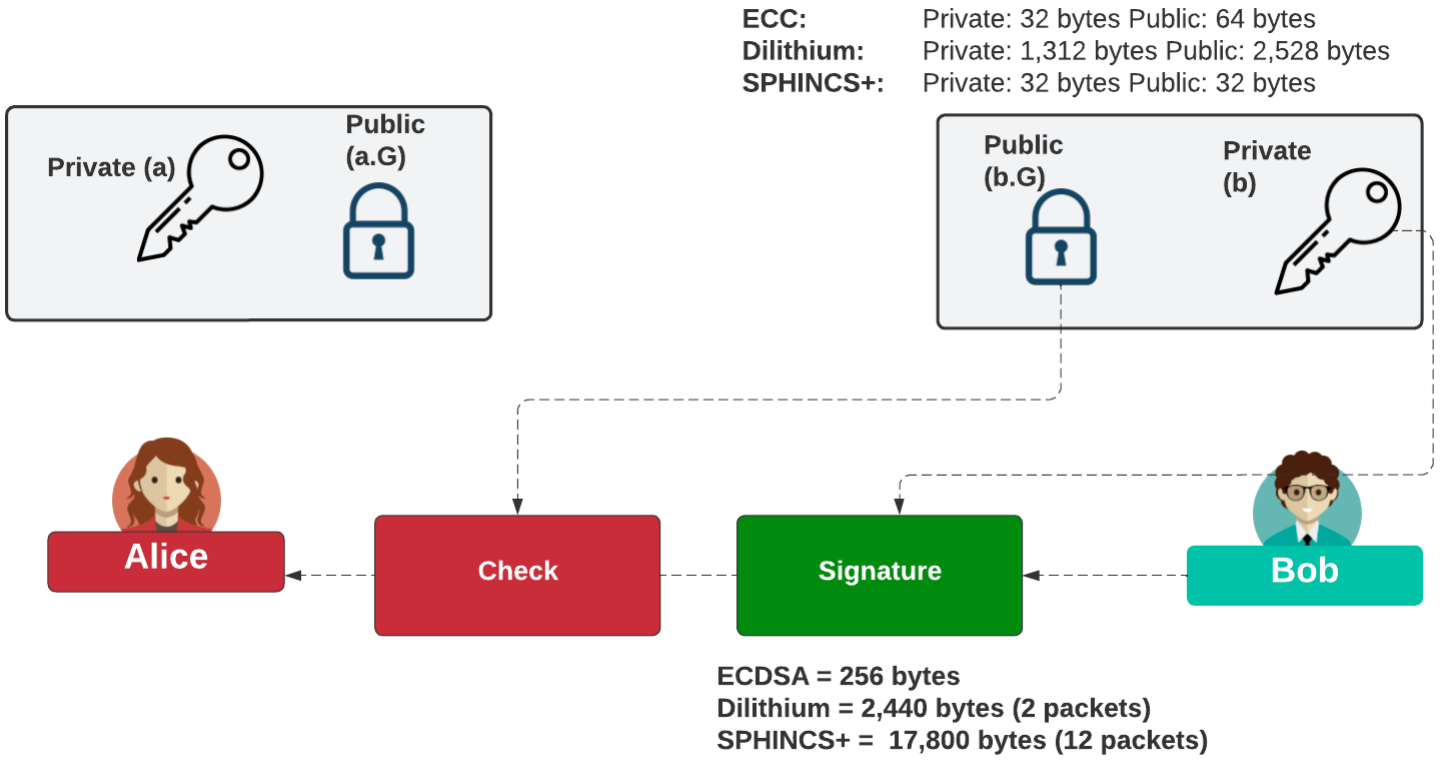
Key Exchange and Digital Signatures



Key Exchange



Digital Signatures

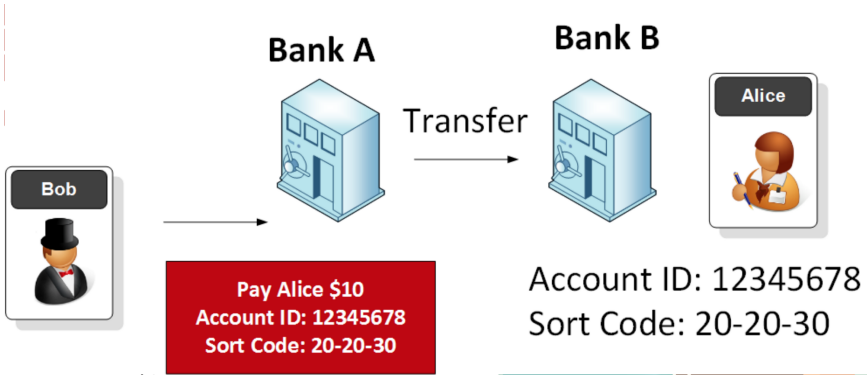


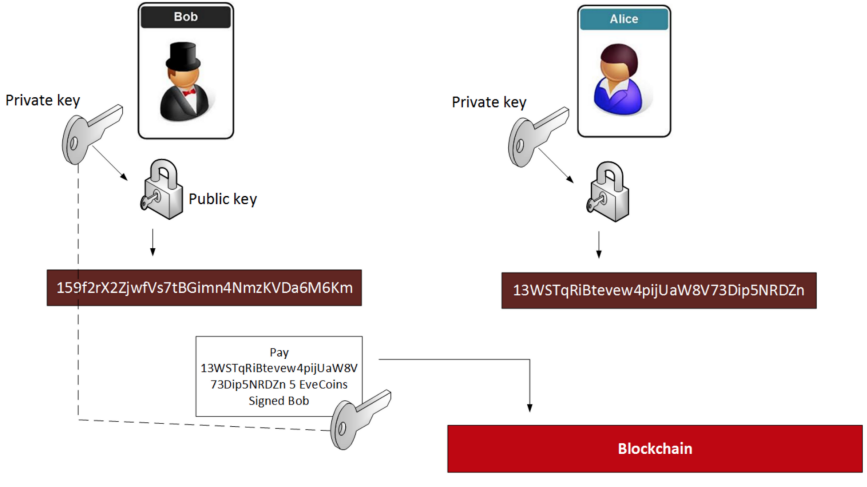
Next Generation Crypto

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

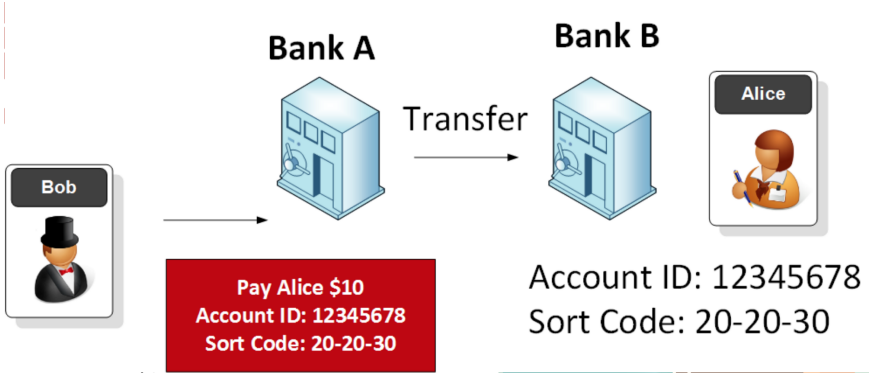
Prof Bill Buchanan OBE
https://asecuritysite.com/tokens

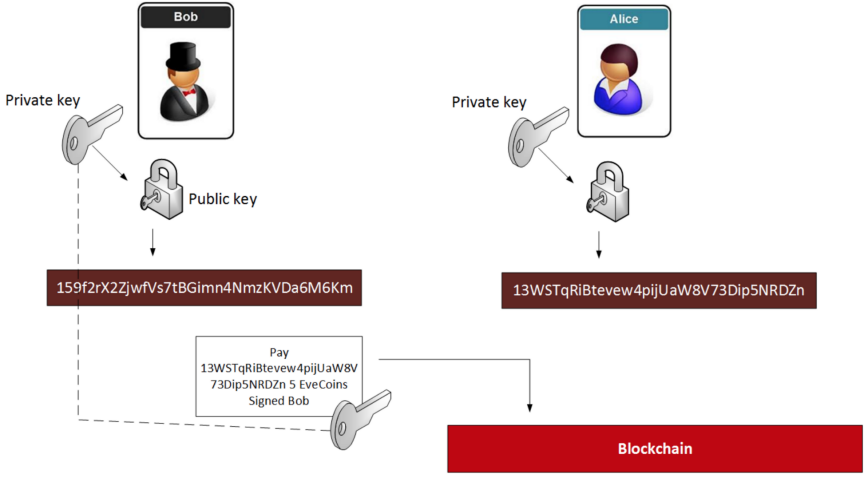
Blockchain Act



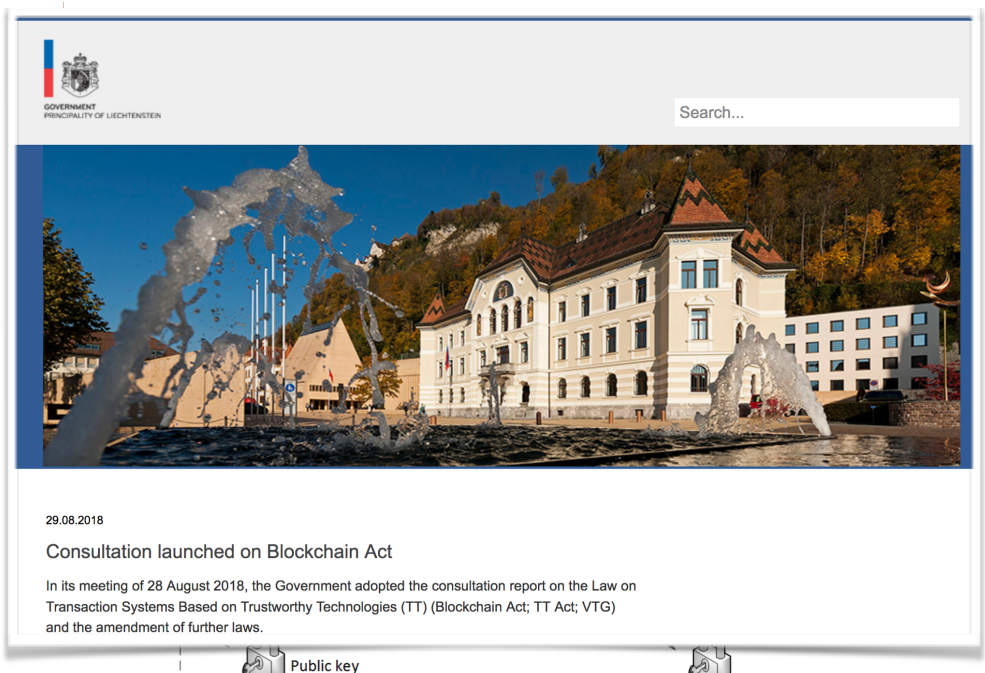


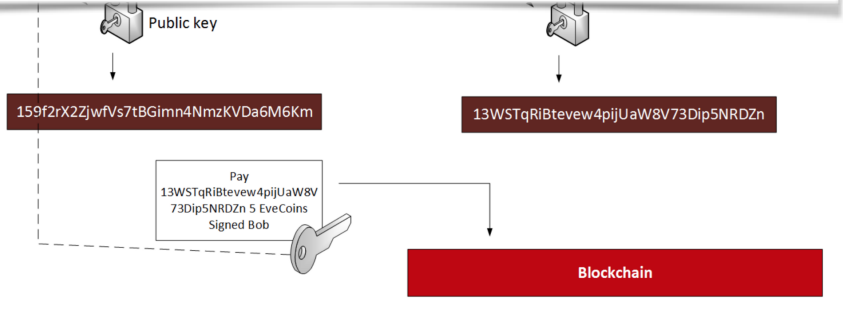
Blockchain Act



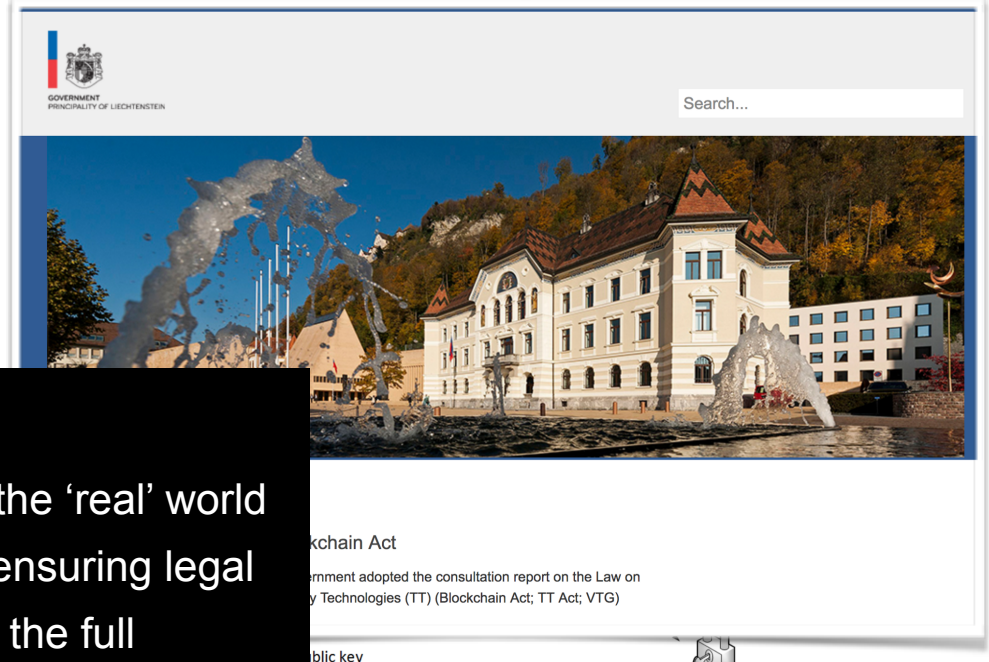


Blockchain Act





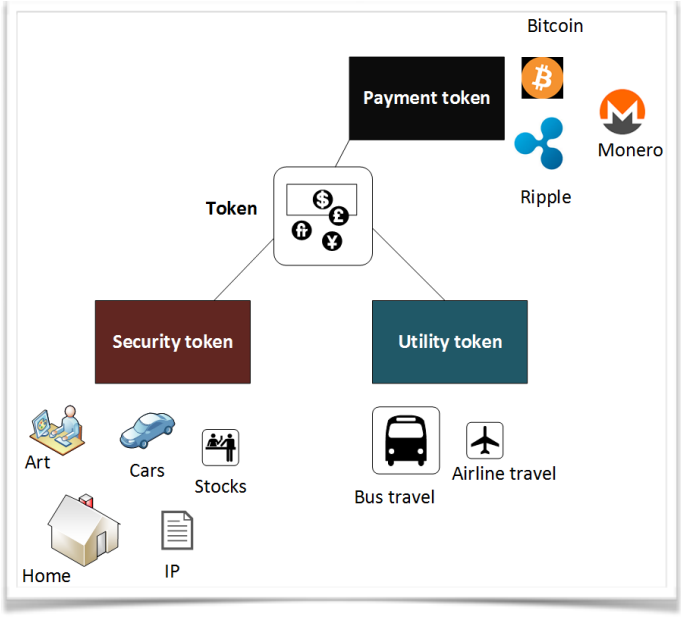
Blockchain Act



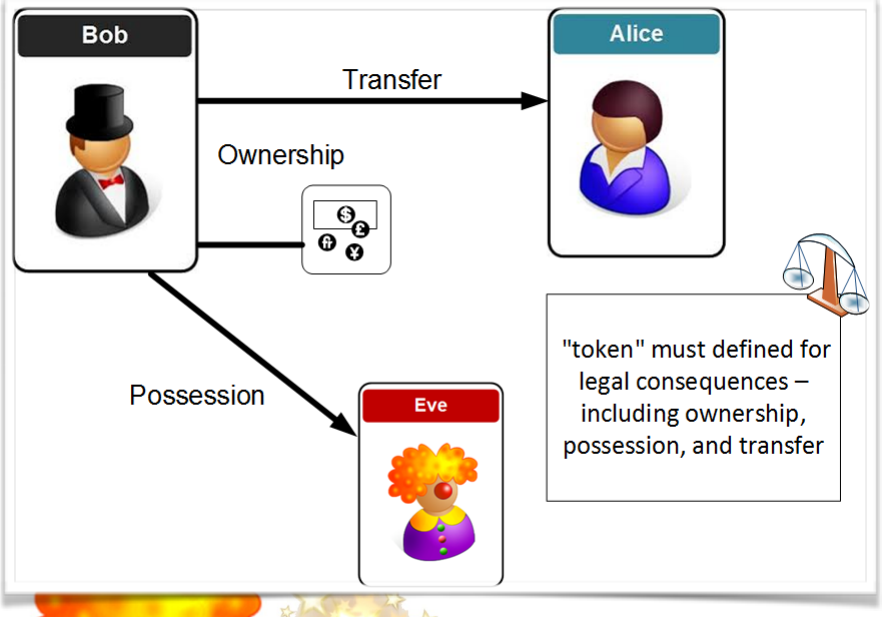
enable the transformation of the ‘real’ world to blockchain systems while ensuring legal certainty, thereby opening up the full application potential of the token economy.

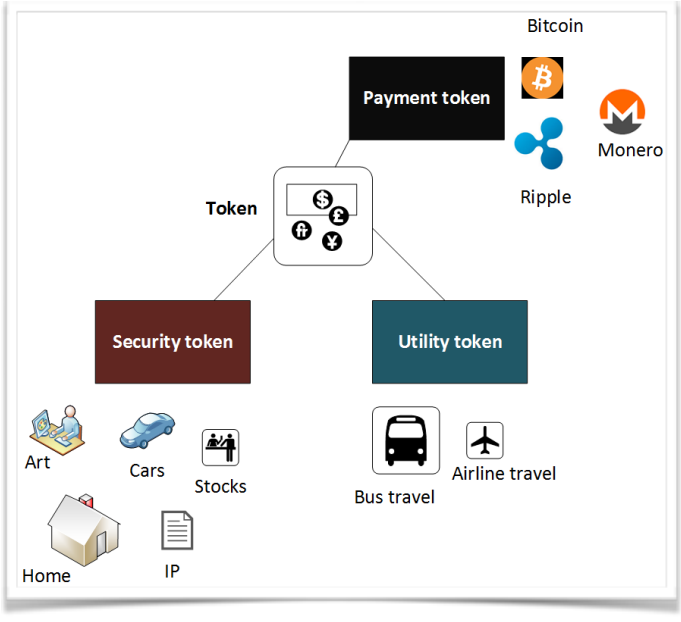
Blockchain Act

Blockchain Act

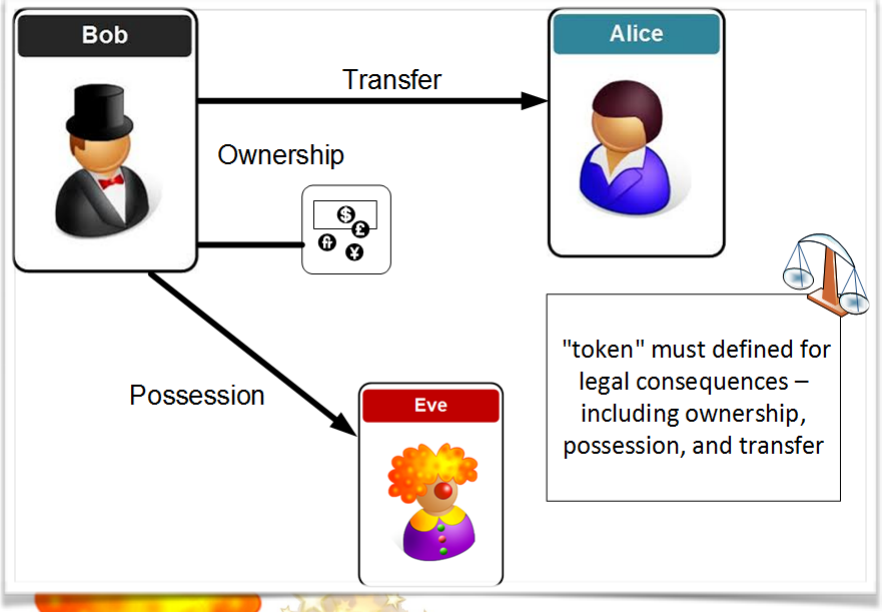


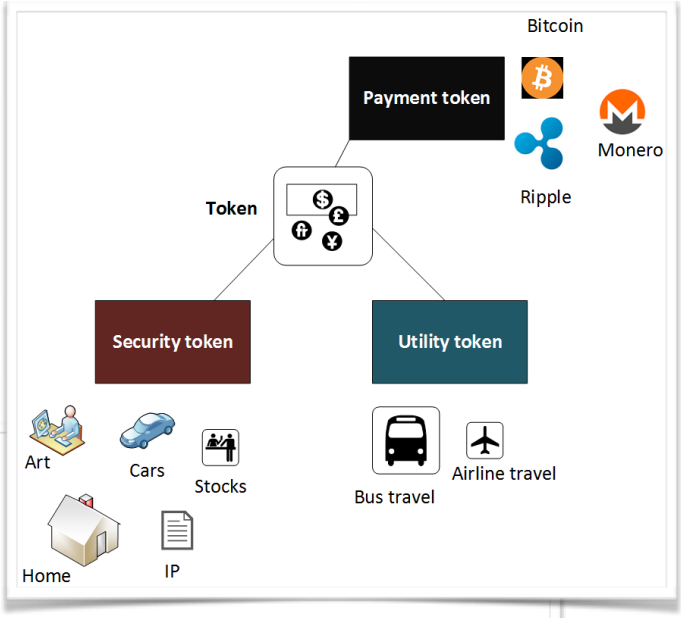
Blockchain Act

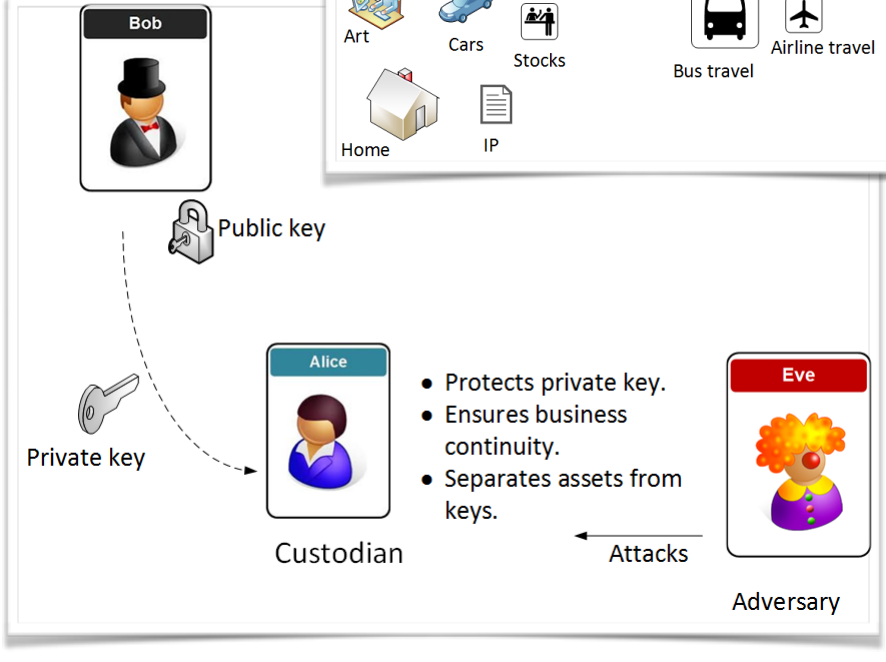




Blockchain Act







Blockchain Act

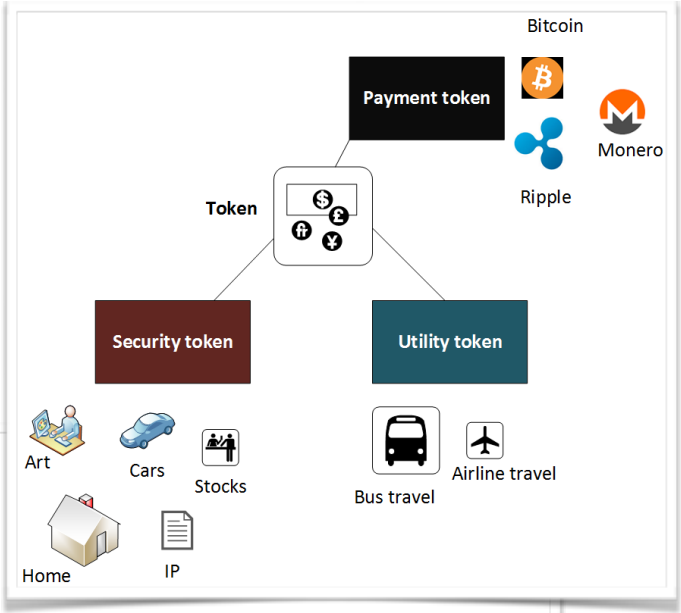
- **Definitions (Art. 5 VE-VTG):** This defines a token as something that defines claims of a person to the rights to goods.

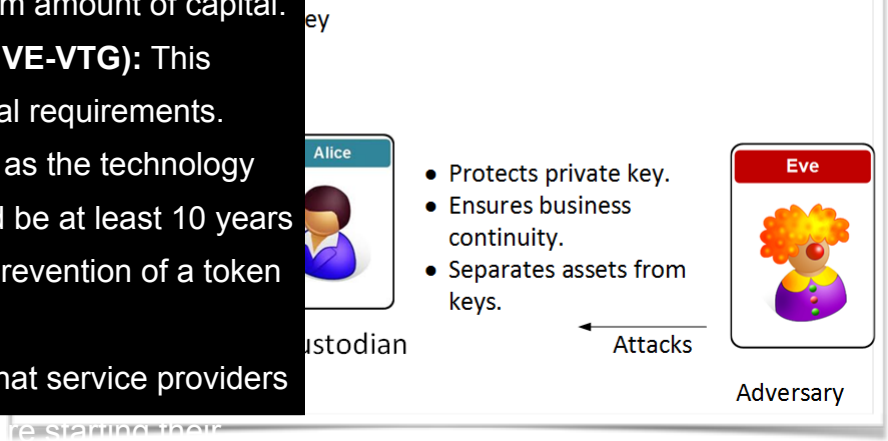
Rights of disposal (Art. 6 ff. VE-VTG): This defines the rights to transfer tokens, and is normally defined by the owner of a private key signing the transaction. A disposition is defined as the transfer of the disposition authorization on the token. Within the Act, it is defined that a buyer has the 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.

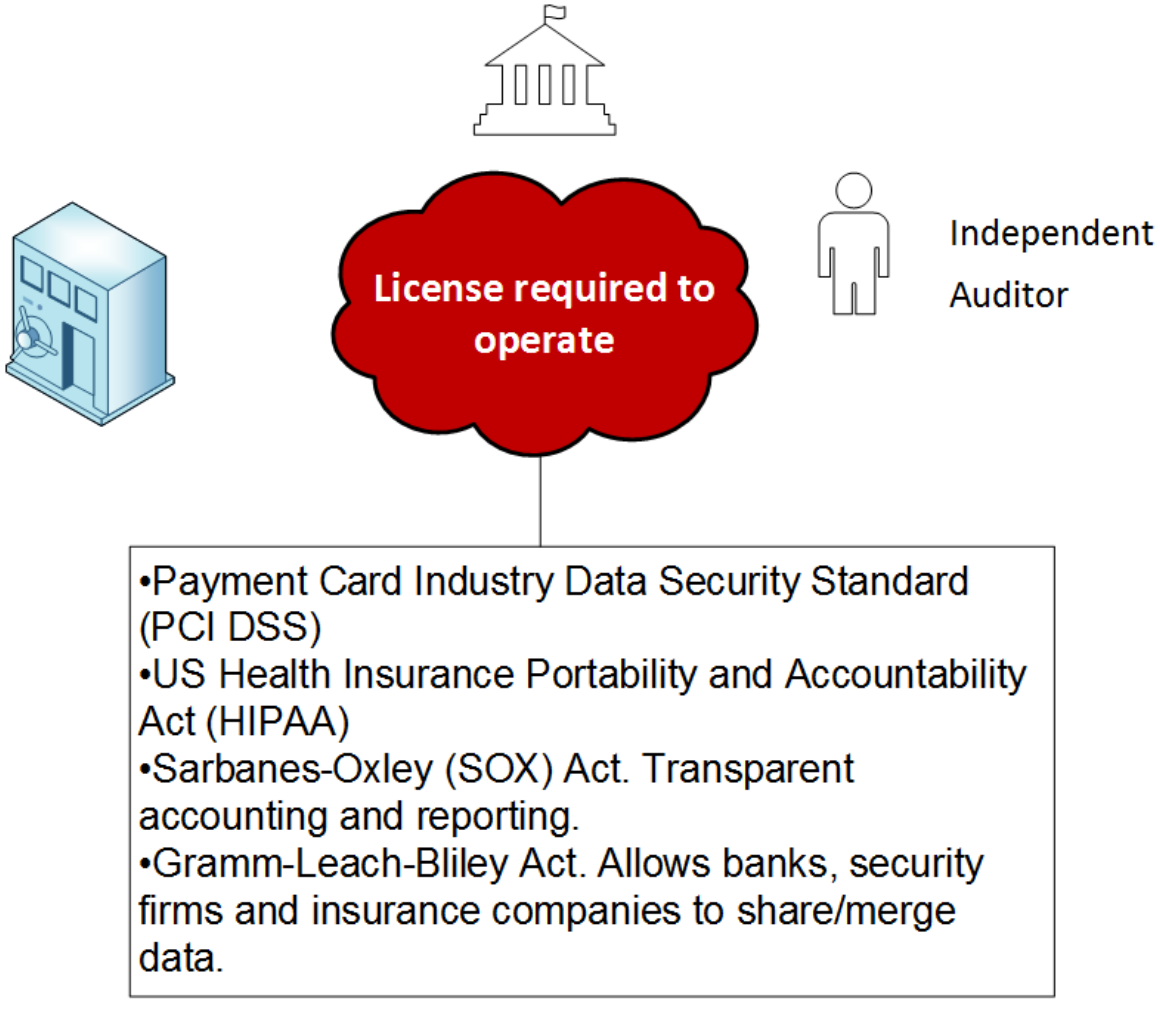
Basic information on the issuance of tokens (Art. 28 ff. VE-VTG): This defines the assurance in the issuing of tokens and their legal requirements. They must provide a minimum amount of information, such as the technology used, the purpose of the token, and any risks. There should be at least 10 years 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

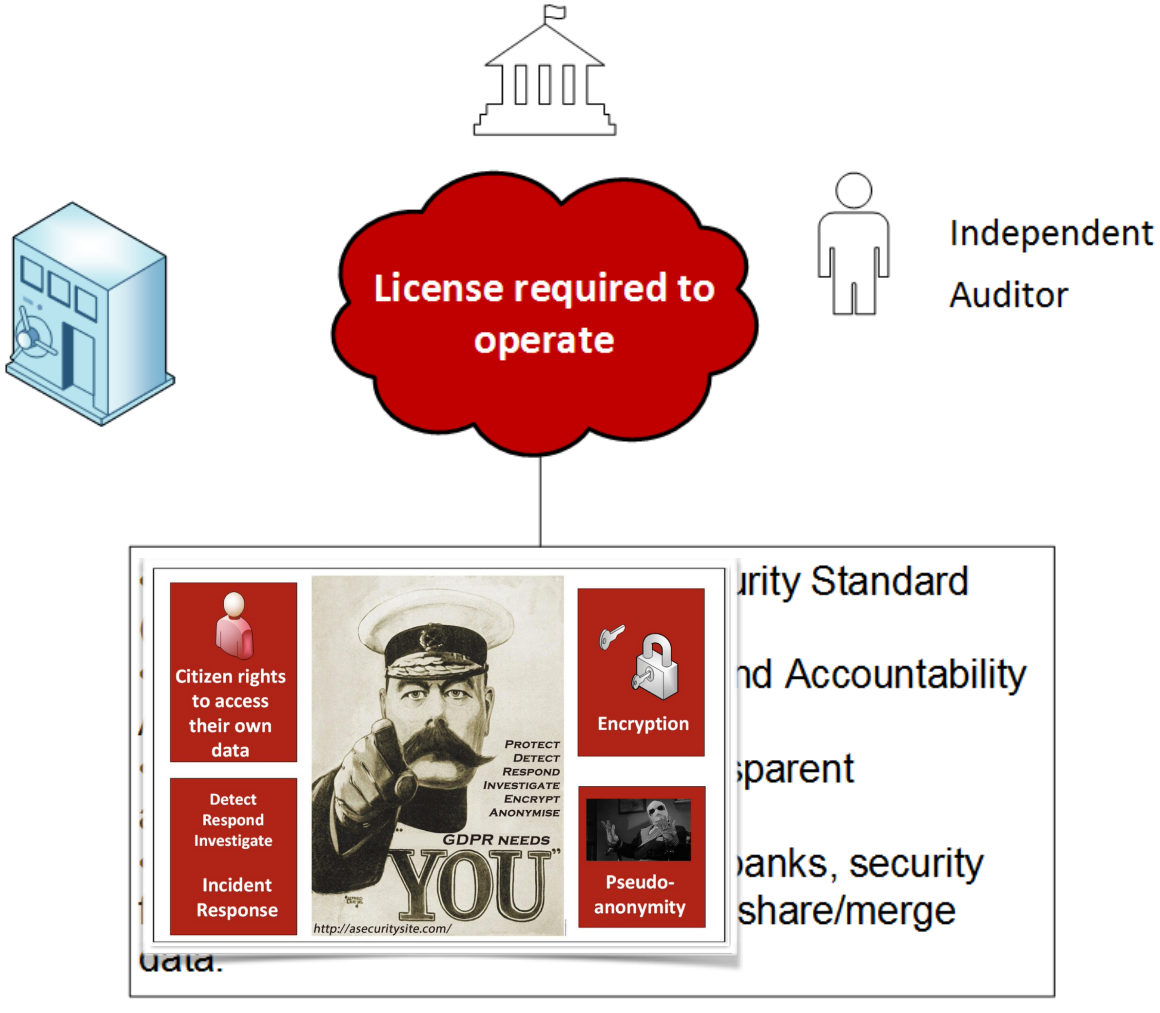




Audit Compliance



Audit Compliance



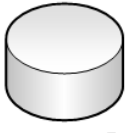
Surrogate identifiers

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

Normal currency

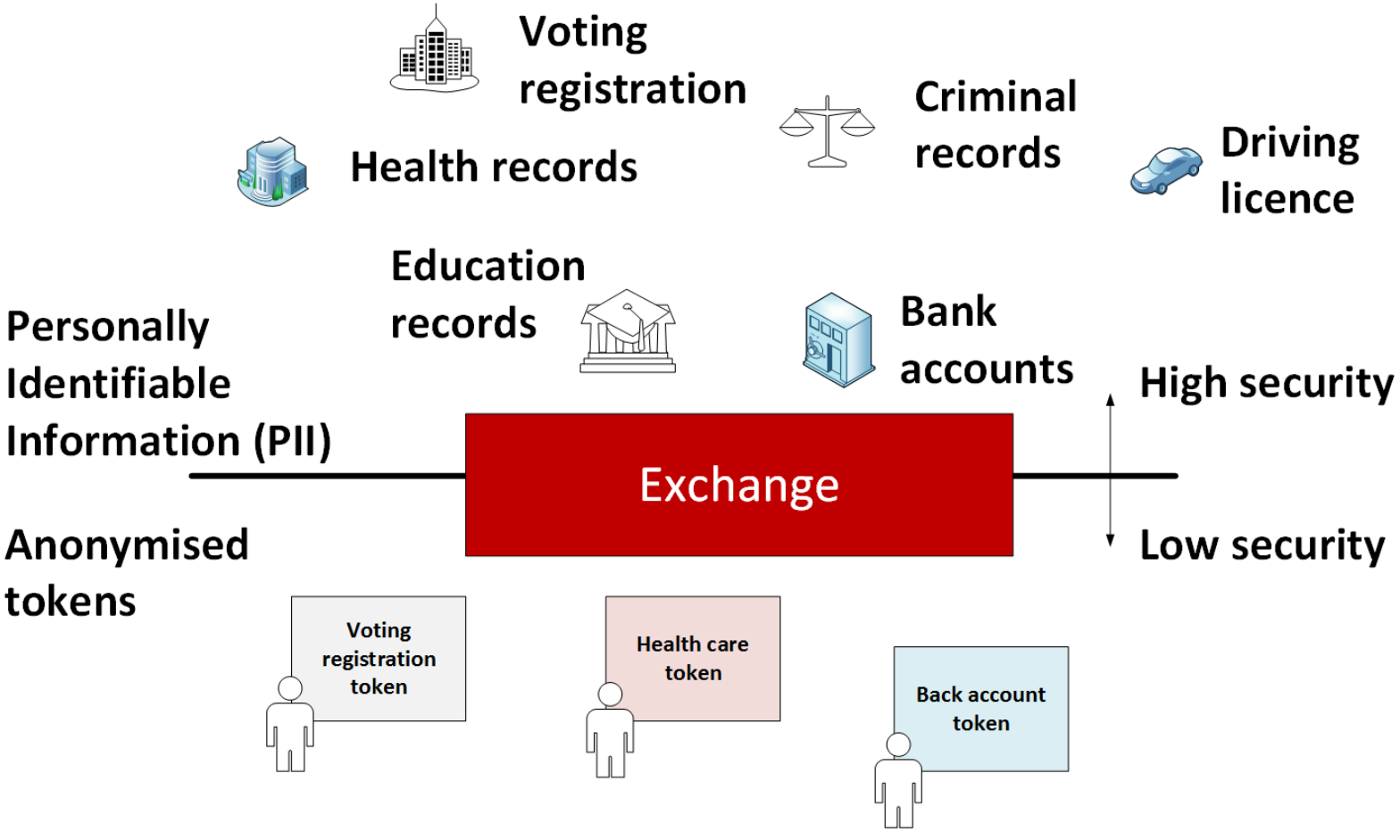
Exchange

Tokens

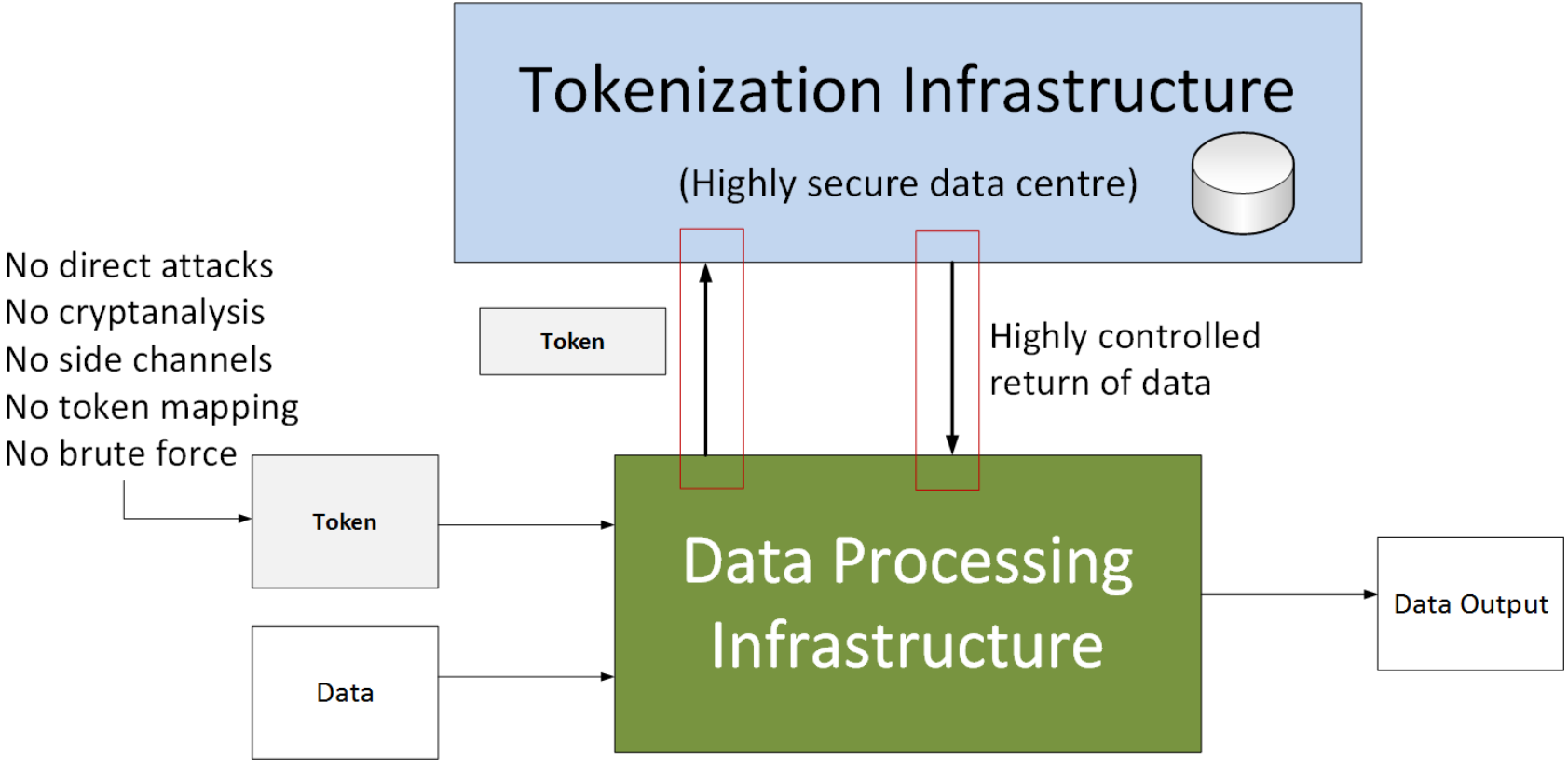
High security

Low security

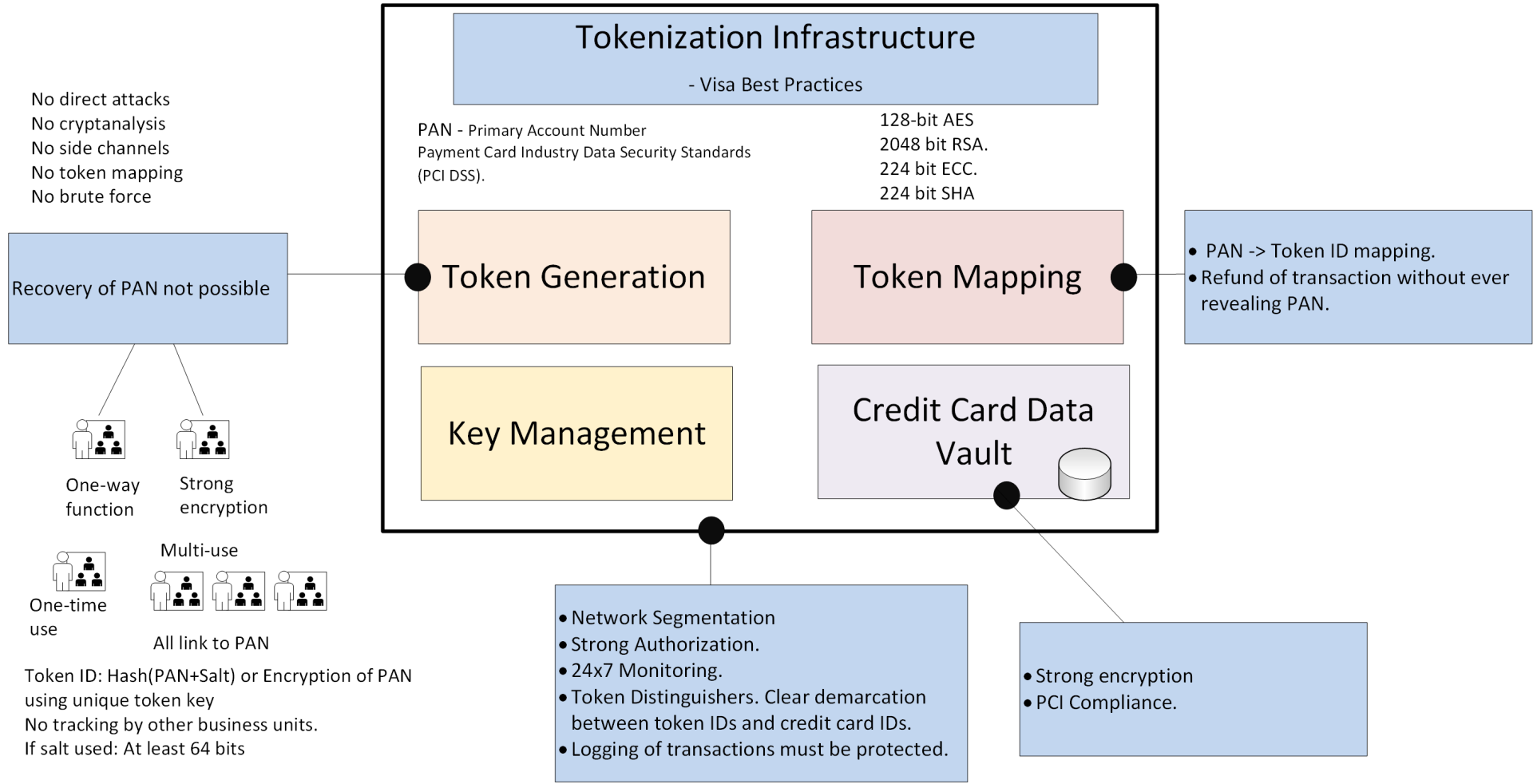
Tokenization with data



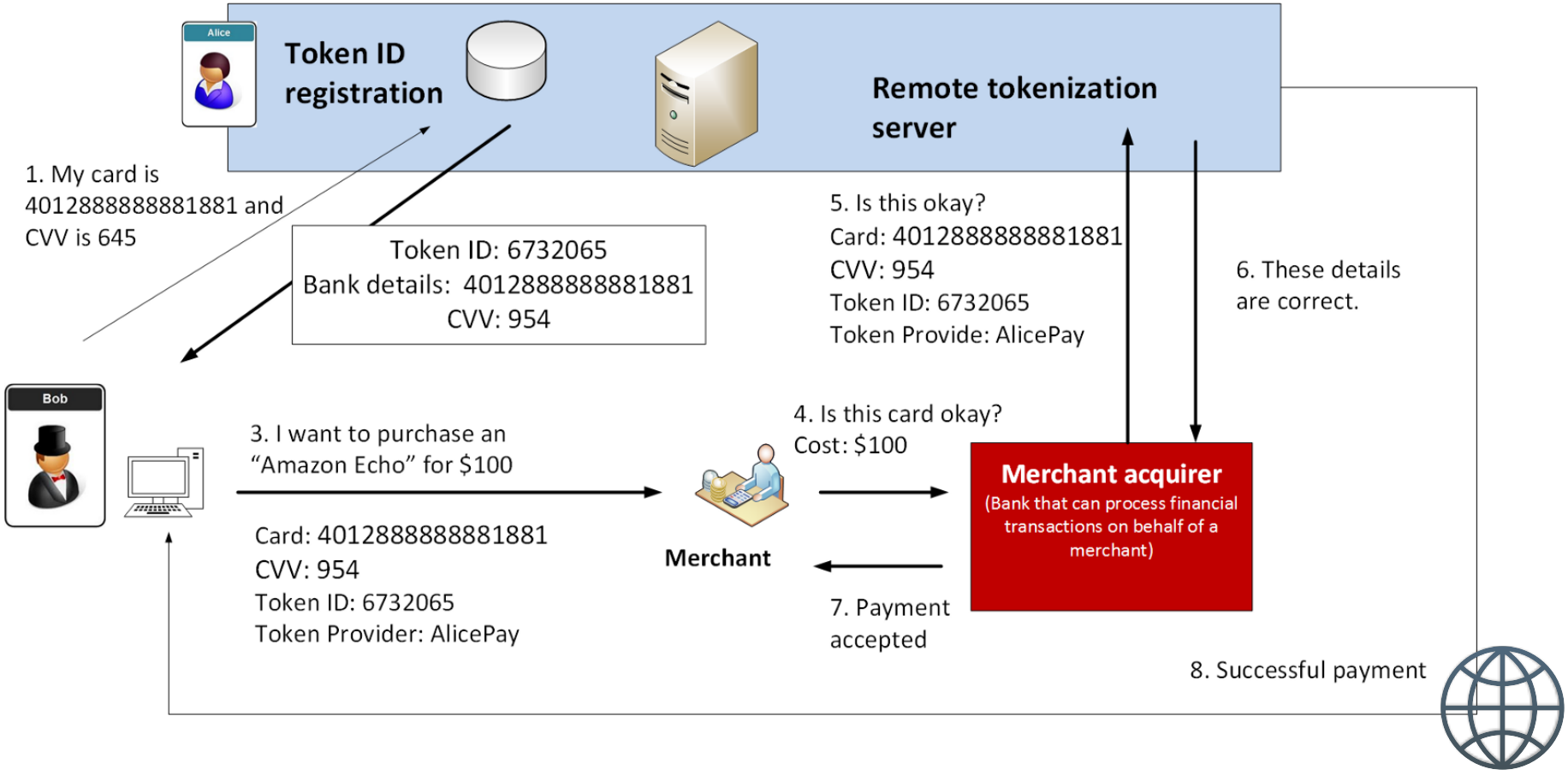
Tokenization with data



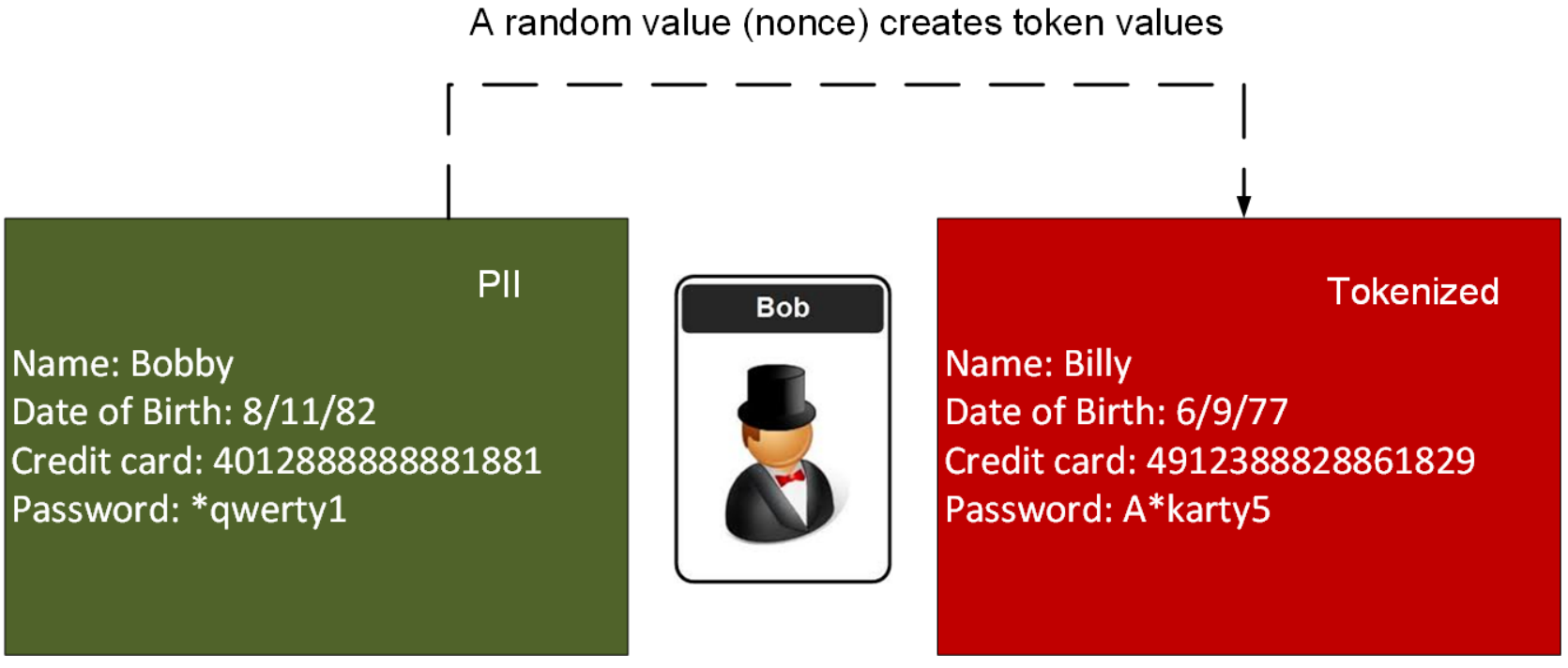
Visa Best Practice for Tokenization



Token Mapping



Token Mapping



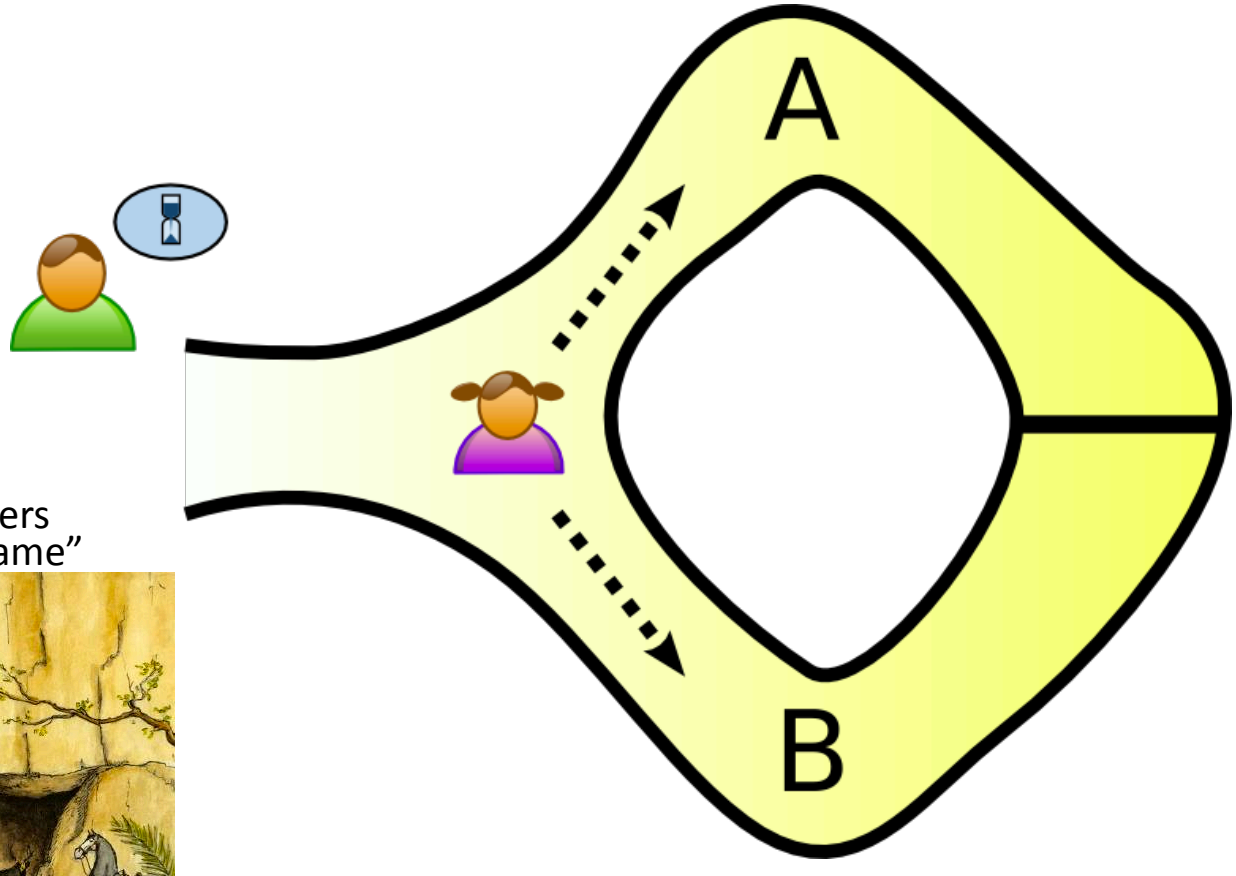
Next Generation Crypto

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

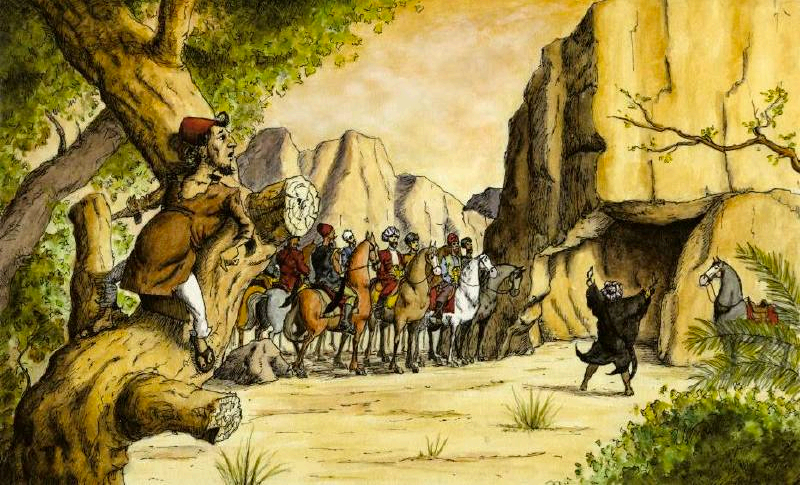
Prof Bill Buchanan OBE
http://asecuritysite.com/zero

Zero-knowledge Proof

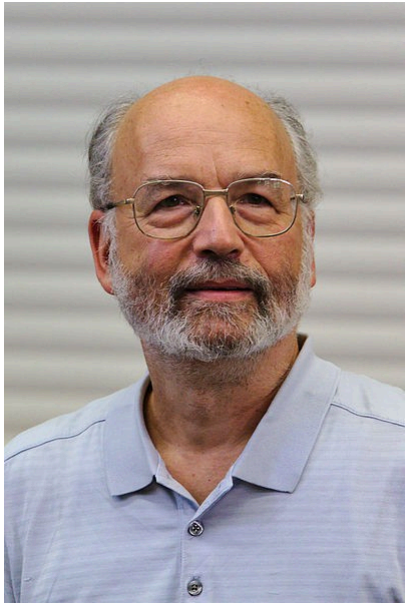
- Peggy is the prover.
- Victor is the verifier.



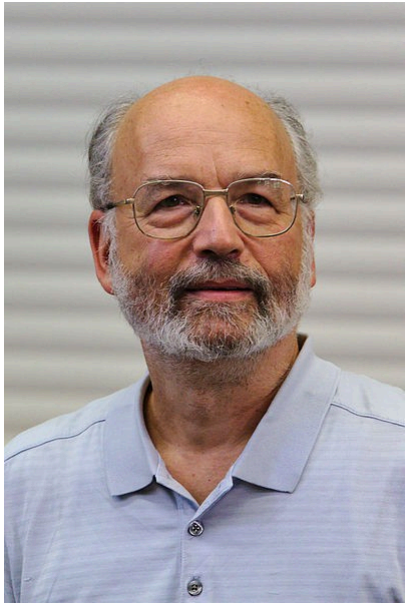
Ali Baba - poor woodcutter - discovers the secret of a thieves as “open sesame”



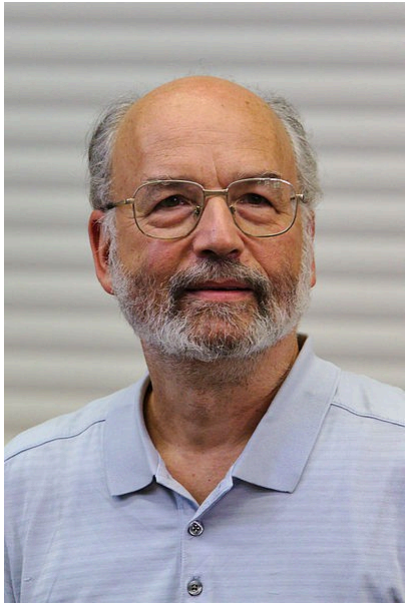
Zero Knowledge Proof: Fiat-Shamir

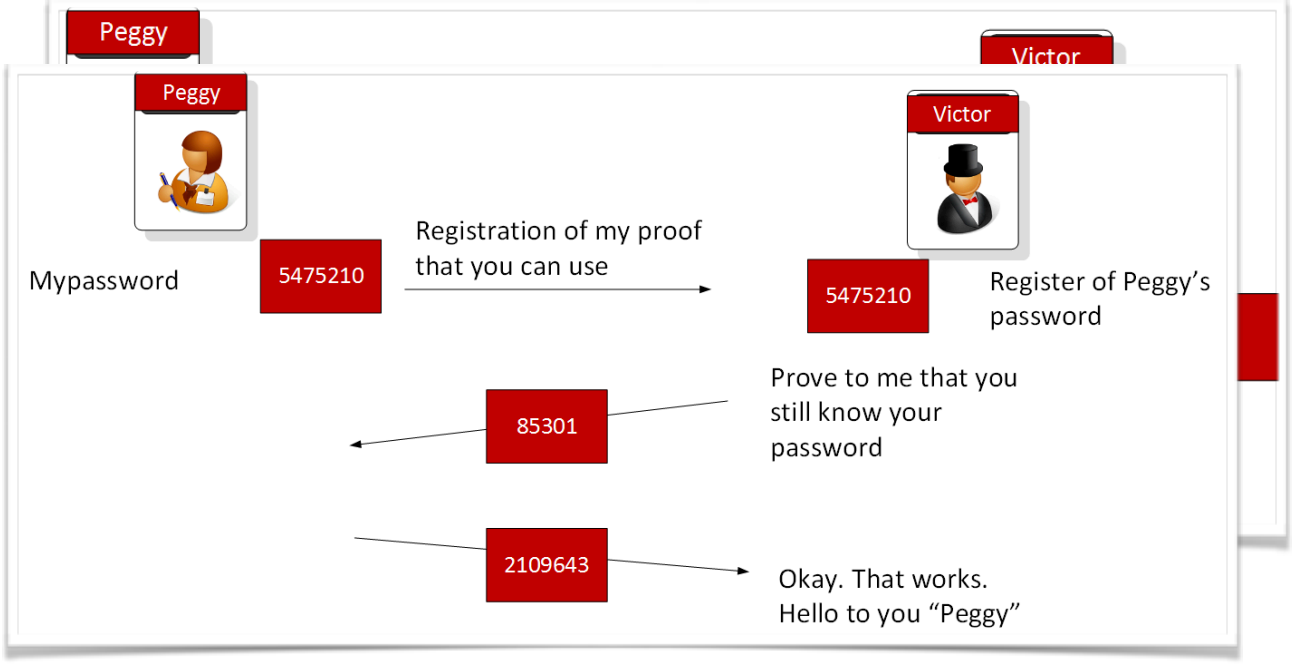


Zero Knowledge Proof: Fiat-Shamir

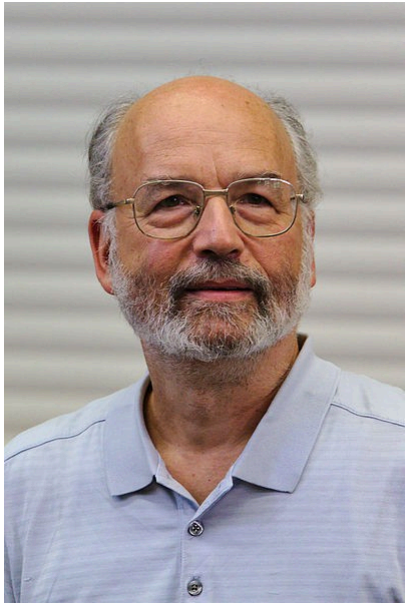


Zero Knowledge Proof: Fiat-Shamir





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 $t' = g^r y^c$

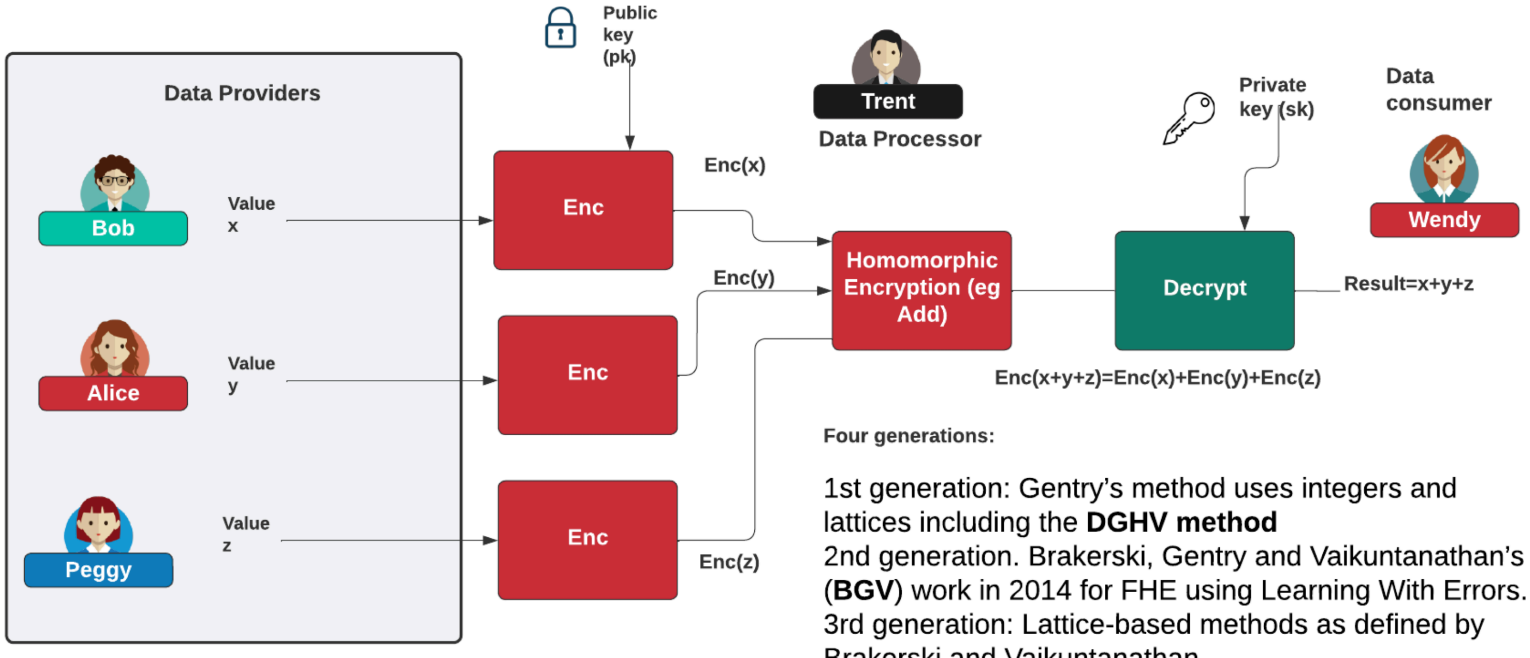
Recheck c = H(g,y,t')

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

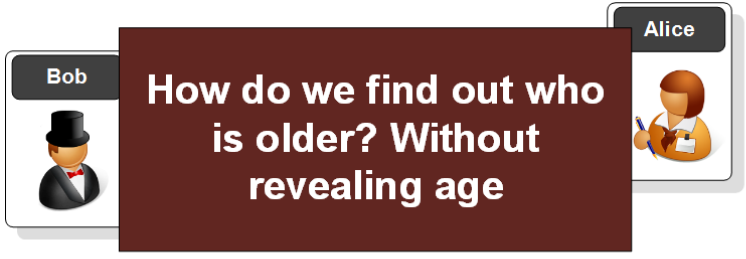
Homomorphic Encryption

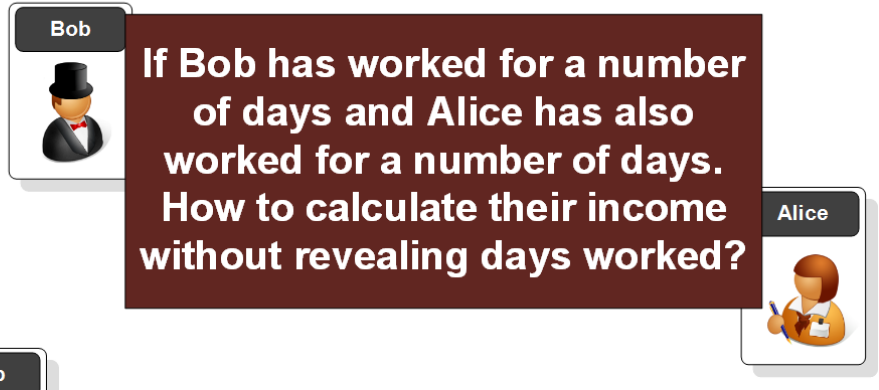


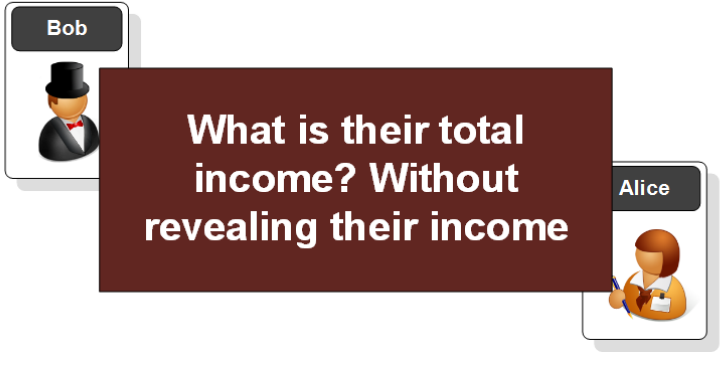
Four generations:

1st generation: Gentry's method uses integers and lattices including the **DGHV method**
2nd generation. Brakerski, Gentry and Vaikuntanathan's (**BGV**) work in 2014 for FHE using Learning With Errors.
3rd generation: Lattice-based methods as defined by Brakerski and Vaikuntanathan.
4th generation: **CKKS** (Cheon, Kim, Kim, Song) and which uses floating-point numbers.

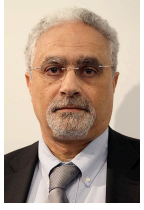
Homomorphic Encryption







Homomorphic Encryption (ElGamal - Multiply/Divide)



IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-31, NO. 4, JULY 1985 469

A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms

TAHER ELGAMAL, MEMBER, IEEE

Abstract—A new signature scheme is proposed, together with an implementation of the Diffie–Hellman key distribution scheme that achieves a public key cryptosystem. The security of both systems relies on the difficulty of computing discrete logarithms over finite fields.

I. Introduction

IN 1976, Diffie and Hellman [3] introduced the concept of public key cryptography. Since then, several attempts have been made to find practical public key systems (see, for example, [6], [7], [9]) depending on the difficulty of

Hence both A and B are able to compute K_{AB}. But, for an intruder, computing K_{AB} appears to be difficult. It is not yet proved that breaking the system is equivalent to computing discrete logarithms. For more details refer to [3].

In any of the cryptographic systems based on discrete logarithms, p must be chosen such that $p-1$ has at least one large prime factor. If $p-1$ has only small prime factors, then computing discrete logarithms is easy (see [8]).

Now suppose that A wants to send B a message m, where $0 \le m \le p-1$. First A chooses a number k uni-

$$Y = g^x \pmod{p}$$

$$a = g^k \pmod{p}$$

$$b = y^k M \pmod{p}$$

$$M = \frac{b}{a^x} \pmod{p}$$

$$a_1 = g^{k_1} \pmod{p}$$

$$a_2 = g^{k_2} \pmod{p}$$

$$b_1 = y^{k_1} M_1 \pmod{p}$$

$$b_2 = y^{k_2} M_2 \pmod{p}$$

$$a = a_1 \times a_2 = g^{k_1} \times g^{k_2} = g^{k_1+k_2} \pmod{p}$$

$$b = b_1 \times b_2 = Y^{k_1} M_1 \times Y^{k_2} M_2 = Y^{k_1+k_2} M_1 M_2 \pmod{p}$$

$$M = \frac{b}{a^x} = \frac{Y^{k_1+k_2} M_1 M_2}{(g^{(k_1+k_2)})^x} = \frac{Y^{k_1+k_2} M_1 M_2}{g^{(k_1+k_2)x}} = \frac{Y^{k_1+k_2} M_1 M_2}{(g^x)^{(k_1+k_2)}} = \frac{Y^{(k_1+k_2)} M_1 M_2}{Y^{(k_1+k_2)}} = M_1 M_2 \pmod{p}$$

ElGamal Multiply

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE transactions on information theory*, *31*(4), 469-472.

Homomorphic Encryption (Add/Subtract)

Public-Key Cryptosystems Based on Composite Degree Residuosity Classes

Pascal Paillier[1,2]

[1] GEMPLUS
Cryptography Department
34 Rue Guynemer, 92447 Issy-Les-Moulineaux
paillier@gemplus.com
[2] ENST
Computer Science Department
46, rue Barrault, 75634 Paris Cedex 13
paillier@inf.enst.fr

Abstract. This paper investigates a novel computational problem, namely the Composite Residuosity Class Problem, and its applications to public-key cryptography. We propose a new trapdoor mechanism and derive from this technique three encryption schemes : a trapdoor permutation and two homomorphic probabilistic encryption schemes computationally comparable to RSA. Our cryptosystems, based on usual modular arithmetics, are provably secure under appropriate assumptions in the standard model.

$$g \in \mathbb{Z}_{N^2}^*$$

$$\mu = (L(g^{\lambda} \pmod{n})^2))^{-1} \pmod{N}$$

$$c = g^m \cdot r^N \pmod{N^2}$$

$$m = L(c^{\lambda} \pmod{N})^2) \cdot \mu \pmod{N}$$

$$C_1 = g^{m_1} \cdot r_1^N \pmod{N^2}$$
$$C_2 = g^{m_2} \cdot r_2^N \pmod{N^2}$$

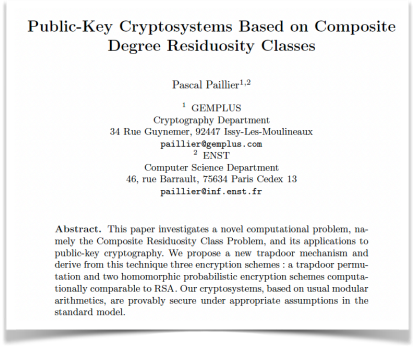
$$C_1 \cdot C_2 = g^{m_1} \cdot r_1^N \cdot g^{m_2} \cdot r_2^N \pmod{N^2}$$
$$C_1 \cdot C_2 = g^{m_1+m_2} \cdot r_1^N \cdot r_2^N \pmod{N^2}$$

$$N = pq$$
$$PHI = (p-1)(q-1)$$
$$\lambda = \operatorname{lcm}(p-1, q-1)$$

Paillier with Kryptology

Paillier, P. (1999, May). Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques* (pp. 223-238). Springer, Berlin, Heidelberg.

Four Generations of HE

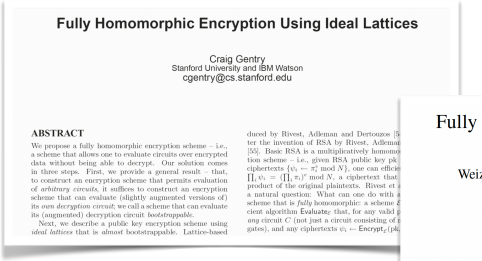


Public-Key Cryptosystems Based on Composite Degree Residuosity Classes

Pascal Paillier[1,2]

[1] GEMPLUS
Cryptography Department
34 Rue Guynemer, 92447 Issy-Les-Moulineaux
paillier@gemplus.com
[2] ENST
Computer Science Department
46, rue Barrault, 75634 Paris Cedex 13
paillier@inf.enst.fr

Abstract. This paper investigates a novel computational problem, namely the Composite Residuosity Class Problem, and its applications to public-key cryptography. We propose a new trapdoor mechanism and derive from this technique three encryption schemes : a trapdoor permutation and two homomorphic probabilistic encryption schemes computationally comparable to RSA. Our cryptosystems, based on usual modular arithmetics, are provably secure under appropriate assumptions in the standard model.



Fully Homomorphic Encryption Using Ideal Lattices

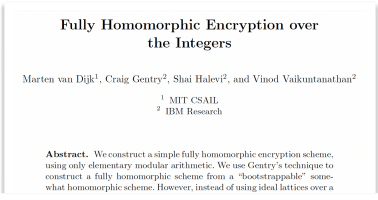
Craig Gentry
Stanford University and IBM Watson
cgentry@cs.stanford.edu

ABSTRACT

We propose a fully homomorphic encryption scheme – i.e., a scheme that allows one to evaluate circuits over encrypted data without being able to decrypt. Our solution comes in three steps. First, we provide a general result – that, to construct an encryption scheme that permits evaluation of *arbitrary circuits*, it suffices to construct an encryption scheme that can evaluate (slightly augmented versions of) its *own decryption circuit*; we call a scheme that can evaluate its (augmented) decryption circuit *bootstrappable*.

Next, we describe a public key encryption scheme using *ideal lattices* that is almost bootstrappable. Lattice-based

duced by Rivest, Adleman and Dertouzos [5] ter the invention of RSA by Rivest, Adleman [55]. Basic RSA is a multiplicatively homomo tion scheme – i.e., given RSA public key pk ciphertexts $\{c_i \leftarrow \pi_i^e \bmod N\}$, one can efficie $\prod_i c_i = (\prod_i \pi_i)^e \bmod N$, a ciphertext that product of the original plaintexts. Rivest et a natural question: What can one do with a scheme that is *fully* homomorphic: a scheme cient algorithm Evaluate$_{\mathcal{E}}$ that, for any valid *any* circuit C (not just a circuit consisting of gates), and any ciphertexts $\psi_i \leftarrow$ Encrypt$_{\mathcal{E}}(pk$

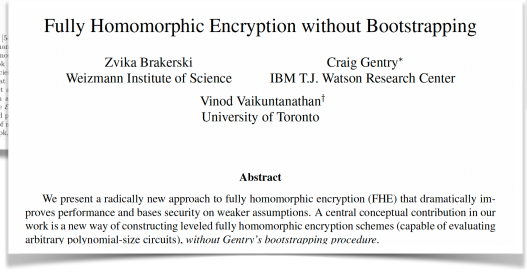


Fully Homomorphic Encryption over the Integers

Marten van Dijk[1], Craig Gentry[2], Shai Halevi[2], and Vinod Vaikuntanathan[2]

[1] MIT CSAIL
[2] IBM Research

Abstract. We construct a simple fully homomorphic encryption scheme, using only elementary modular arithmetic. We use Gentry's technique to construct a fully homomorphic scheme from a "bootstrappable" somewhat homomorphic scheme. However, instead of using ideal lattices over a



Fully Homomorphic Encryption without Bootstrapping

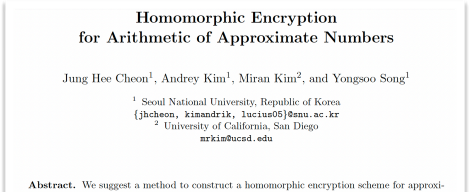
Zvika Brakerski
Weizmann Institute of Science

Craig Gentry*
IBM T.J. Watson Research Center

Vinod Vaikuntanathan†
University of Toronto

Abstract

We present a radically new approach to fully homomorphic encryption (FHE) that dramatically improves performance and bases security on weaker assumptions. A central conceptual contribution in our work is a new way of constructing leveled fully homomorphic encryption schemes (capable of evaluating arbitrary polynomial-size circuits), *without Gentry's bootstrapping procedure*.



Homomorphic Encryption for Arithmetic of Approximate Numbers

Jung Hee Cheon[1], Andrey Kim[1], Miran Kim[2], and Yongsoo Song[1]

[1] Seoul National University, Republic of Korea
{jhcheon, kimandrik, lucius05}@snu.ac.kr
[2] University of California, San Diego
mrkim@ucsd.edu

Abstract. We suggest a method to construct a homomorphic encryption scheme for approximate arithmetic. It supports an approximate addition and multiplication of encrypted messages, together with a new *rescaling* procedure for managing the magnitude of plaintext. This procedure truncates a ciphertext into a smaller modulus, which leads to rounding of plaintext. The main idea is to add a noise following significant figures which contain a main message. This noise is originally added to the plaintext for security, but considered to be a part of error occurring during approximate computations that is reduced along with plaintext by rescaling. As a result, our decryption structure outputs an approximate value of plaintext with a predetermined precision.

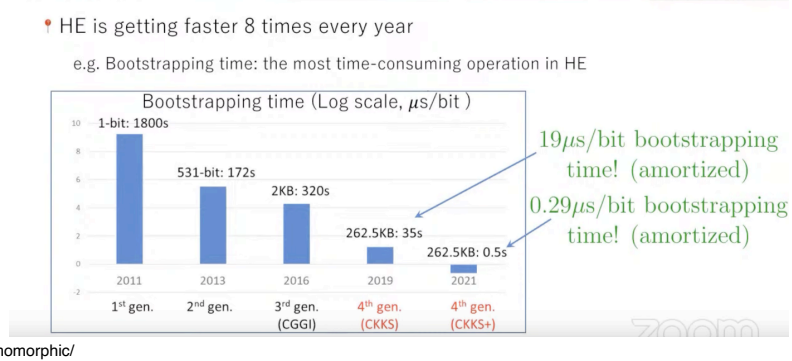
[1] Van Dijk, M., Gentry, C., Halevi, S., & Vaikuntanathan, V. (2010, May). Fully homomorphic encryption over the integers. In *Annual international conference on the theory and applications of cryptographic techniques* (pp. 24–43). Springer, Berlin, Heidelberg.

[2] Brakerski, Zvika, and Vinod Vaikuntanathan. "Efficient fully homomorphic encryption from (standard) LWE." *SIAM Journal on Computing* 43.2 (2014): 831–871.

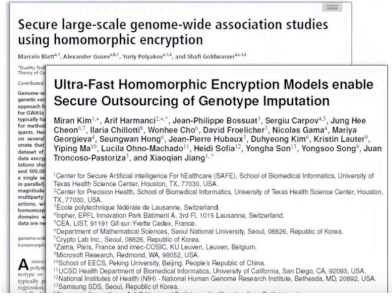
[3] Brakerski, Z., & Vaikuntanathan, V. (2014, January). Lattice-based FHE as secure as PKE. In *Proceedings of the 5th conference on Innovations in theoretical computer science* (pp. 1–12).

[4] Cheon, J. H., Kim, A., Kim, M., & Song, Y. (2017, December). Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security* (pp. 409–437). Springer, Cham.

- 1st generation: Gentry's method uses integers and lattices [1] including the **DGHV method**.
- 2nd generation. Brakerski, Gentry and Vaikuntanathan's (**BGV**) work in 2014 for FHE using Learning With Err
- 3rd generation: Lattice-based methods as defined by Brakerski and Vaikuntanathan [3].
- 4th generation: **CKKS** (Cheon, Kim, Kim, Song) and which uses floating-point numbers [4]. Here.



https://asecuritysite.com/homomorphic/



Secure large-scale genome-wide association studies using homomorphic encryption

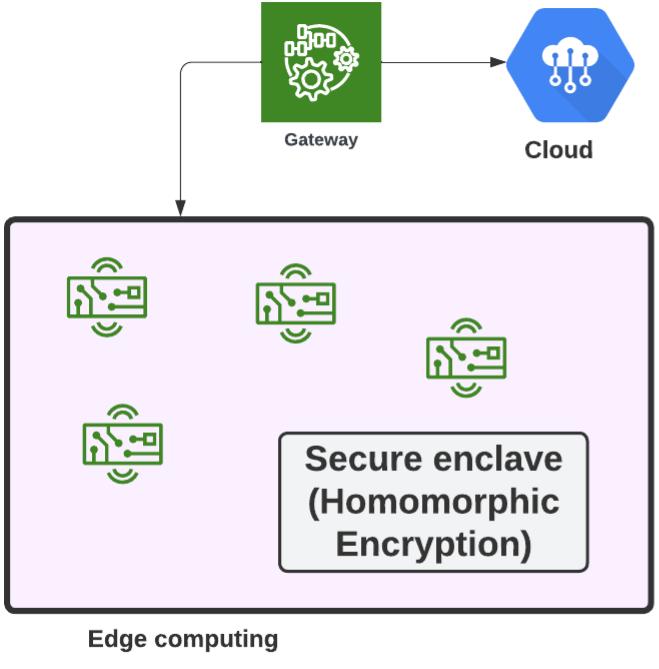
Ultra-Fast Homomorphic Encryption Models enable Secure Outsourcing of Genotype Imputation

2nd paper by 4 winning teams of 2019 iDASH Genomic Privacy Challenge.

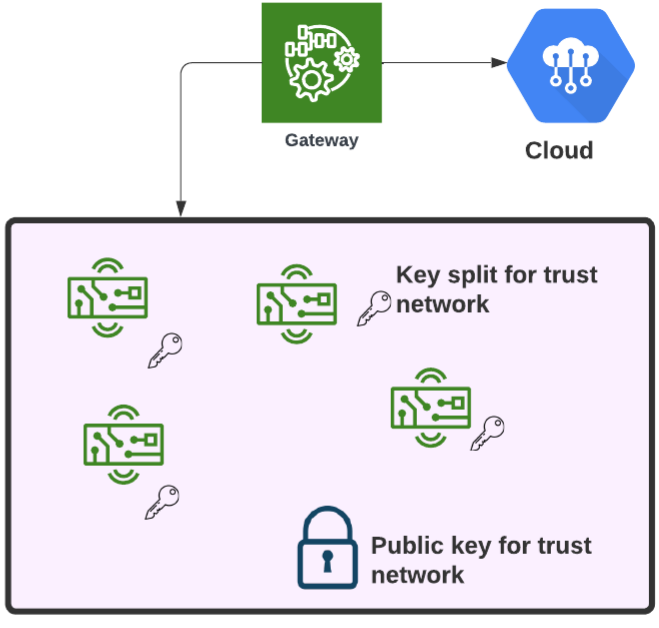
Algorithms: linear regression, logistic regression, and neural ne

< 25s evaluation of imputation model for 80K SNPs

Edge Computing/Trusted Environment



Edge computing



Distributed signing with
consensus

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

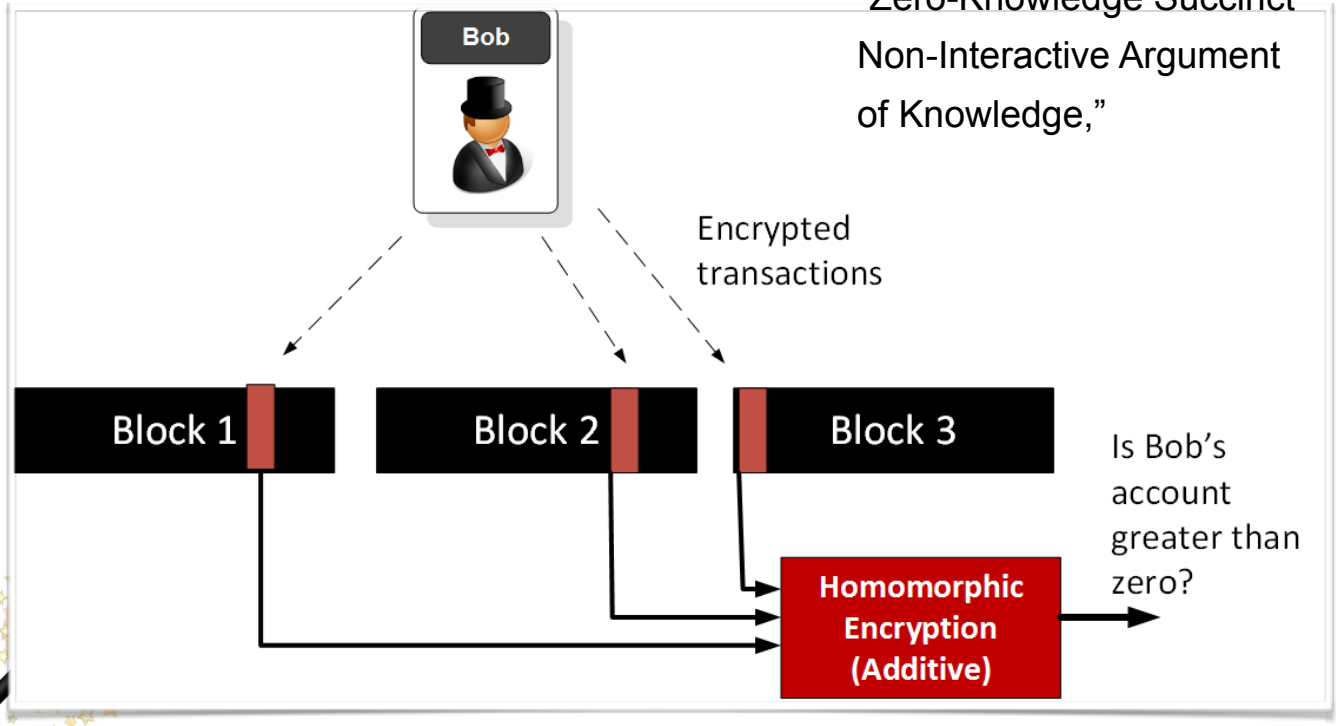
Humans that were at fault with a new Zcoin hack (Zcash) and which involved making transactions of £561,000 ($699,000), and with an associated profit of £349,000 ($435,000).

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,"



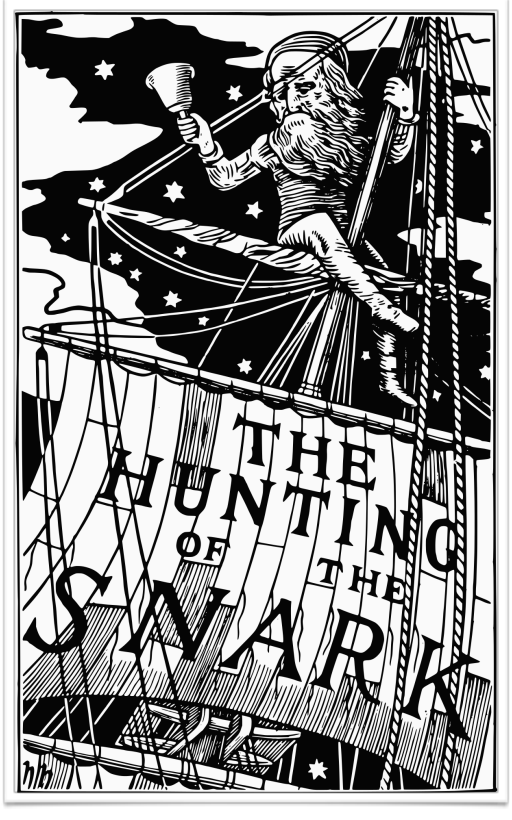
Pal Link

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

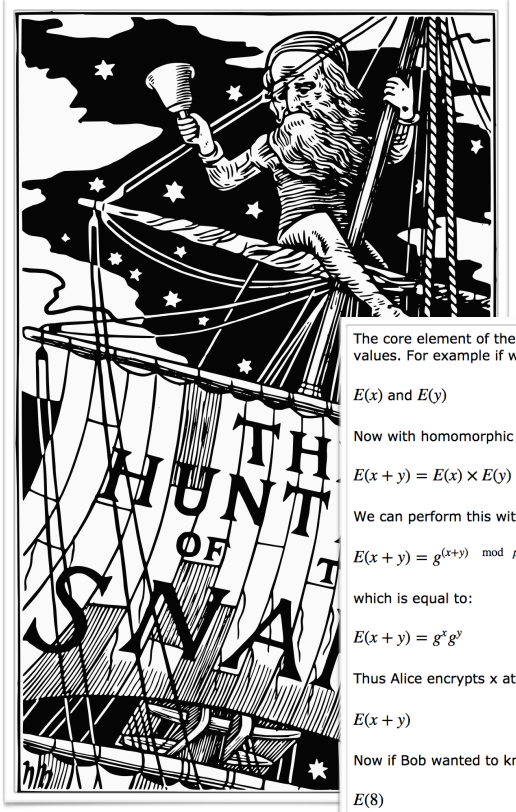
Bob doesn't want Alice to known the values that he is using and Alice doesn't want Bob to know the method she is using to compute a result

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:

$E(x+y)$

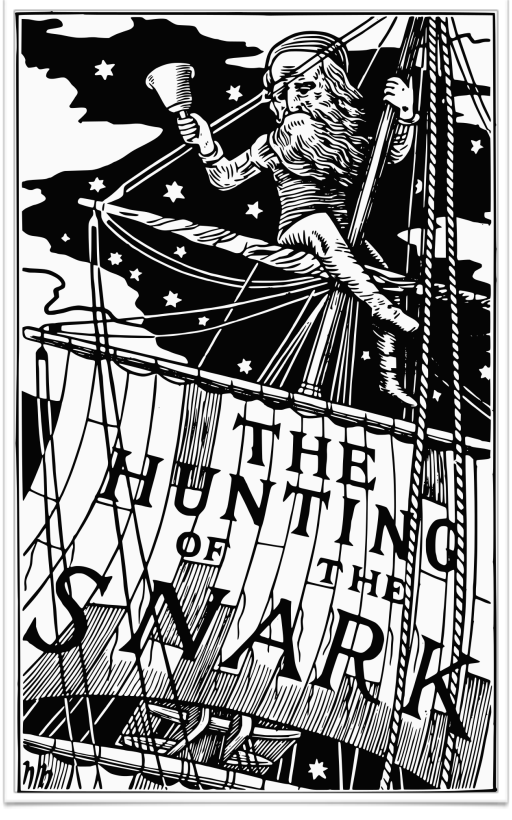
Now if Bob wanted to know if x plus y equals 8, Bob will then encrypt:

$E(8)$

Blind Evaluation

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

zkSNARK



Hidden Homomorphic

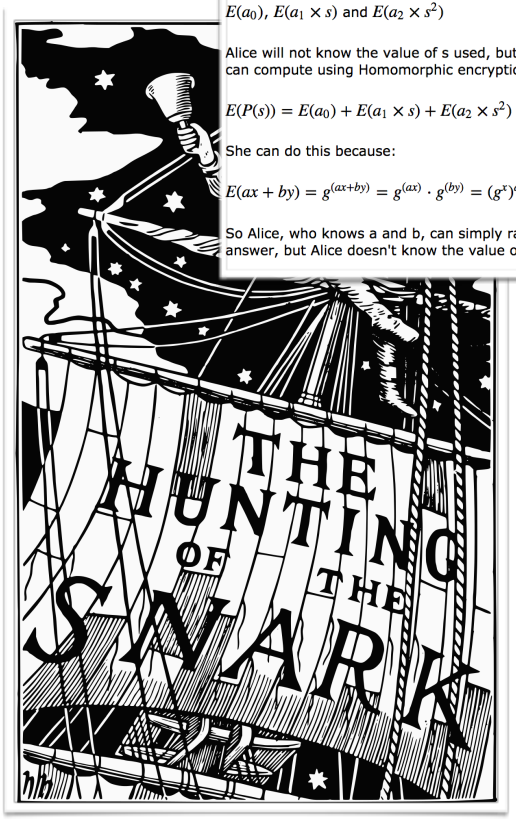
With HH we aim to prove that Alice knows two values (x, y) to equal a given answer (ans)

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 -

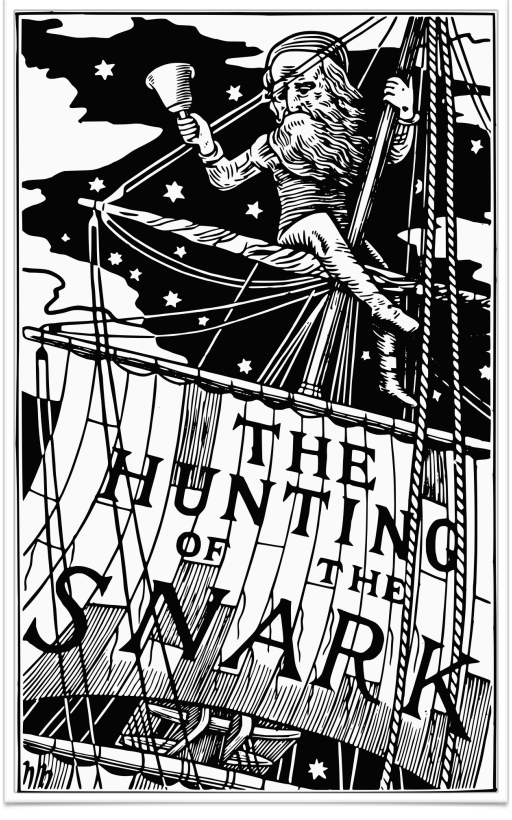
which is the lack of privacy in

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

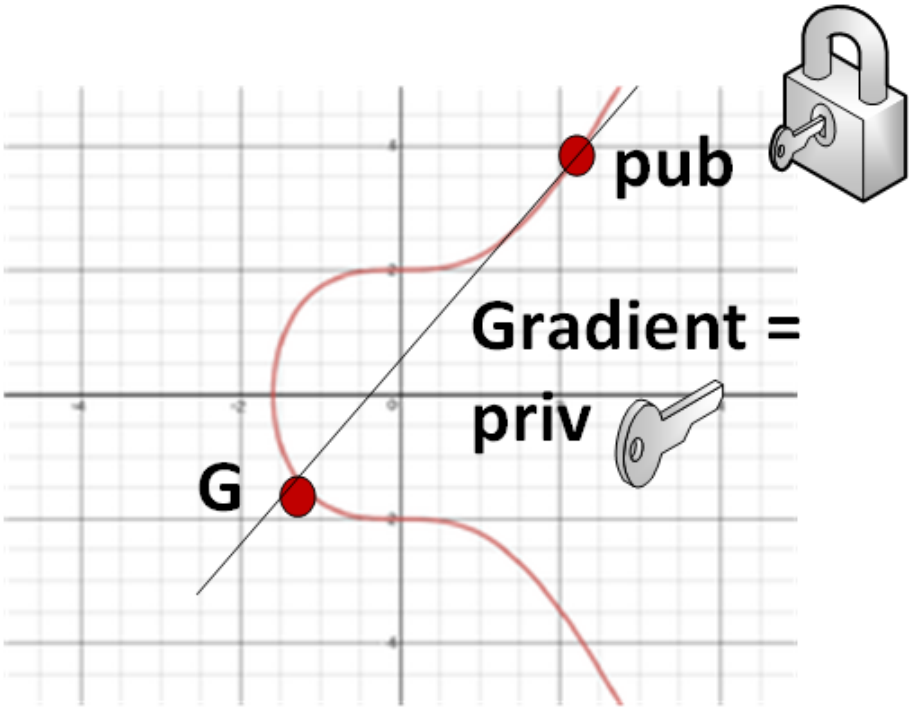
Bob doesn't want Alice to known the values that he is using and Alice doesn't want Bob to know the method she is using to compute a result

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

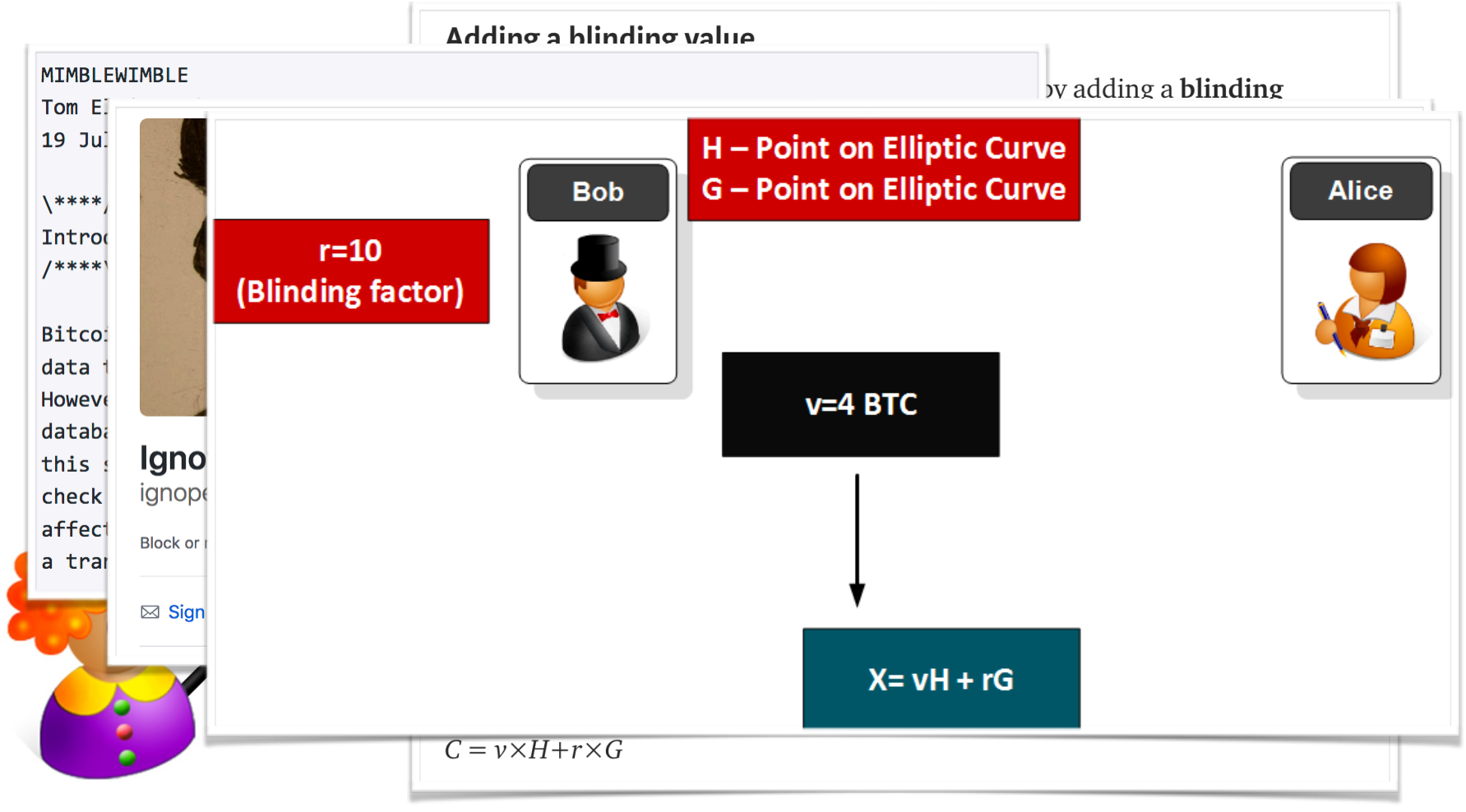
Adding a blinding value

…by adding a **blinding**

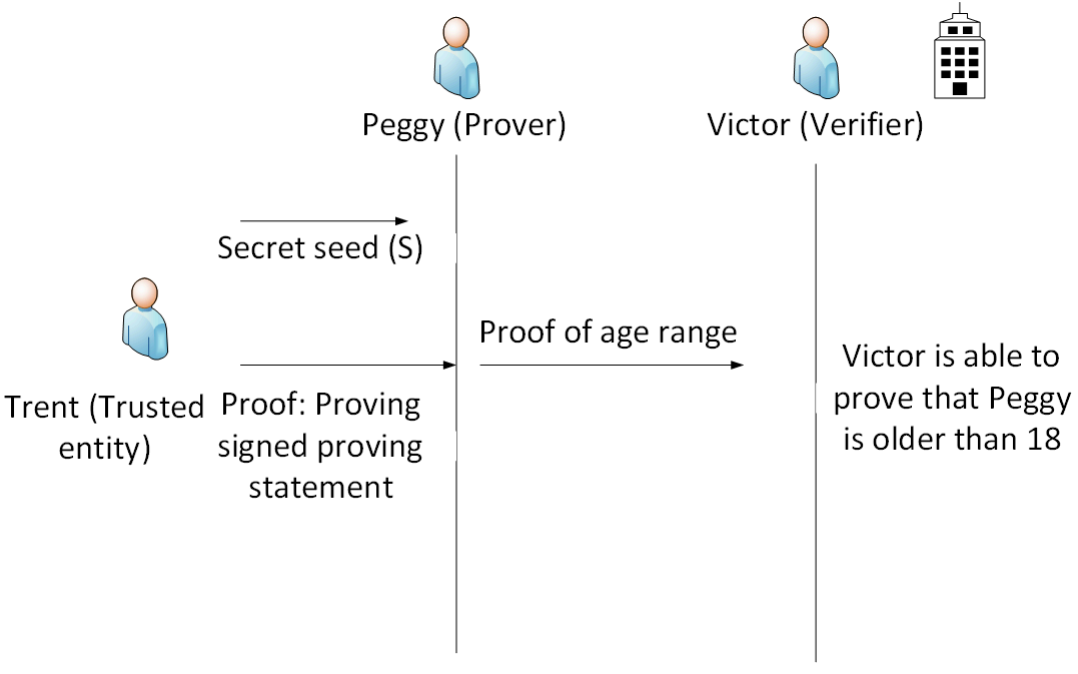
value is then (as defined as a Pedersen Commitment):

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

Pedersen Commitment and Mimblewimble



Range Proof

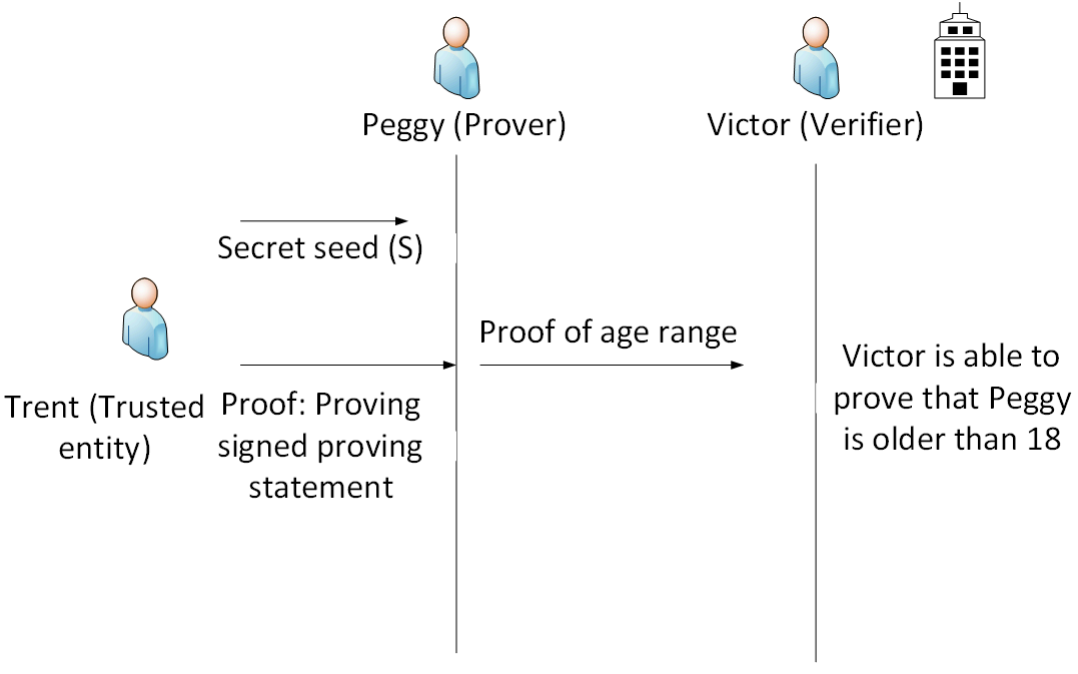

Peggy (Prover)
Victor (Verifier)
Secret seed (S)
Trent (Trusted entity)
Proof: Proving signed proving statement
Proof of age range
Victor is able to prove that Peggy is older than 18

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

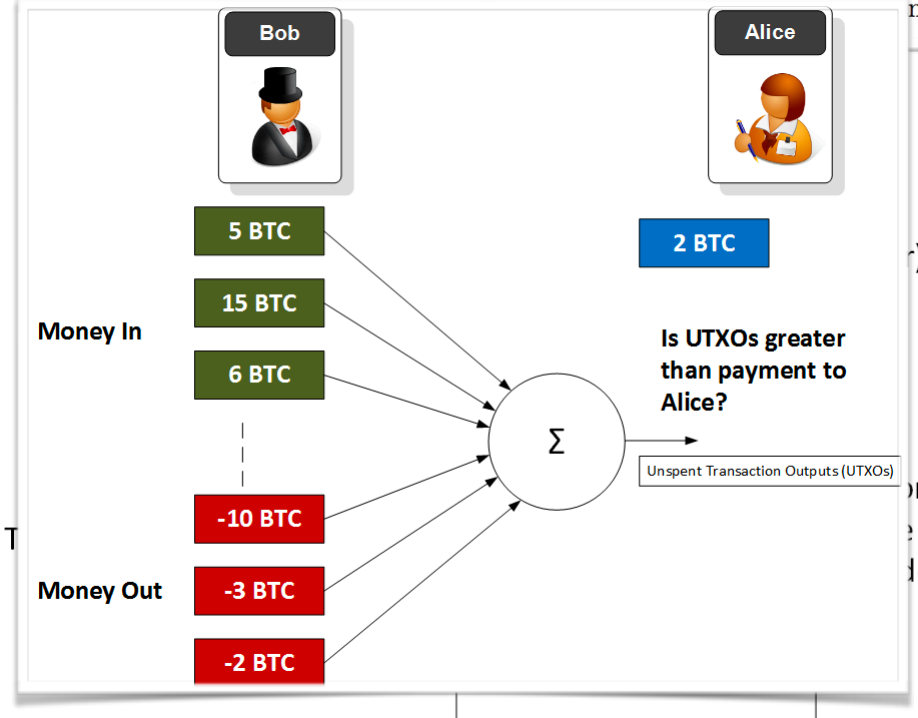


Range Proof

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



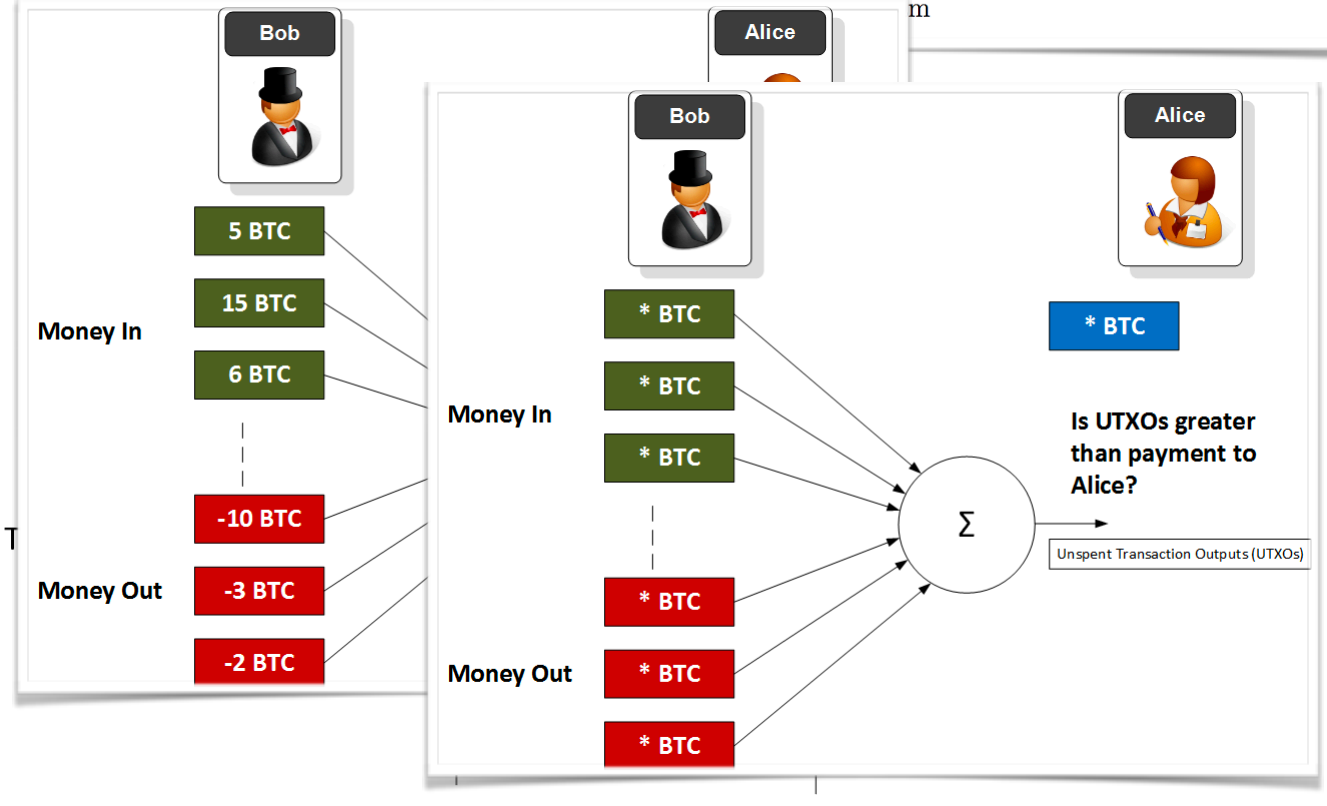
r is able to
that Peggy
der than 18

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.

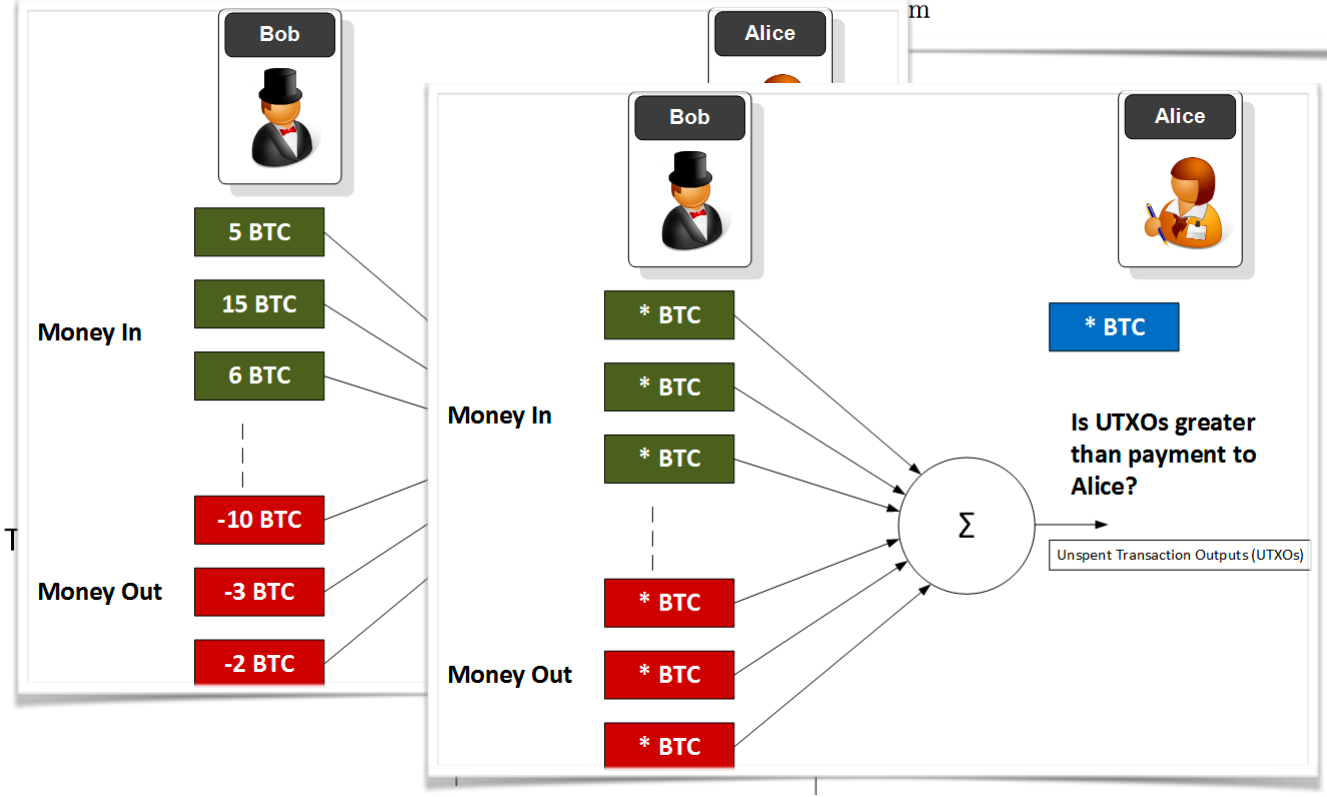
Range Proof

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

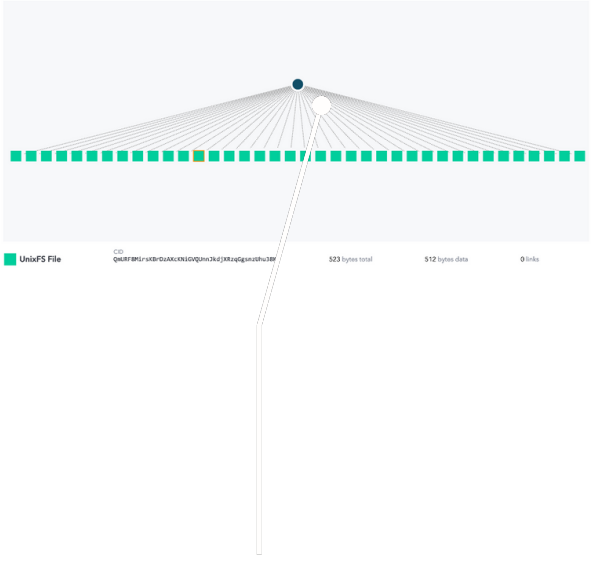
- Significant reduction in the size of the signature as opposed to other CT methods (such as zk-SNARKs and zk-STARKs).
- Significantly reduced transaction fees with shorter signatures.
- Supports MPC (Multiparty Computation) and where many parties can come together to create a single range proof, without 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