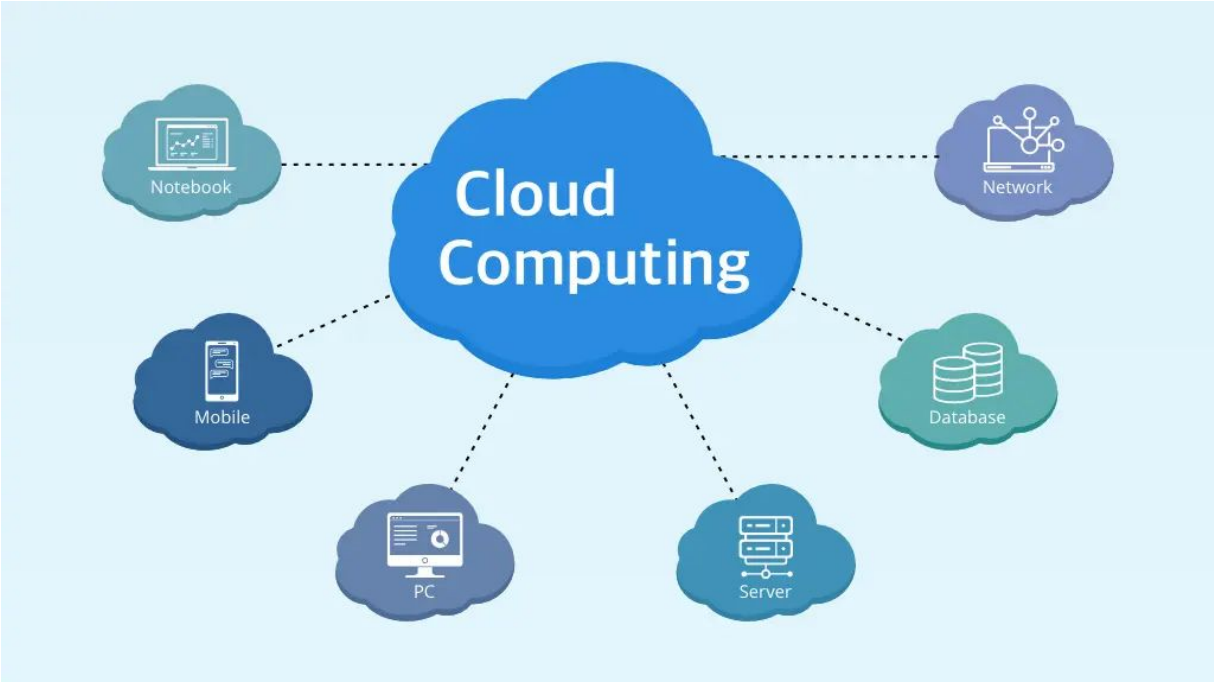
CSC 116
Homomorphic Encryption

Homomorphic Encryption is a type of encryption that allows computations to be performed directly on encrypted data **without decrypting it first**.

Why it's important:

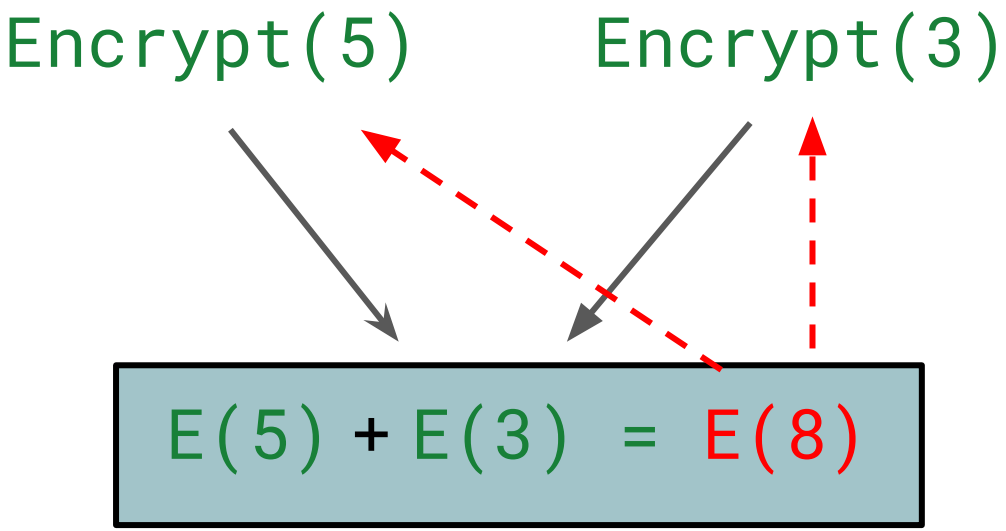
- Enables **privacy-preserving computation**.
- Useful in **cloud computing**, where a server can compute on encrypted data without learning the actual data.
- Vital for **secure data analysis**, **machine learning**, **finance**, and **healthcare** where sensitive data is involved.

Cloud data is not secure



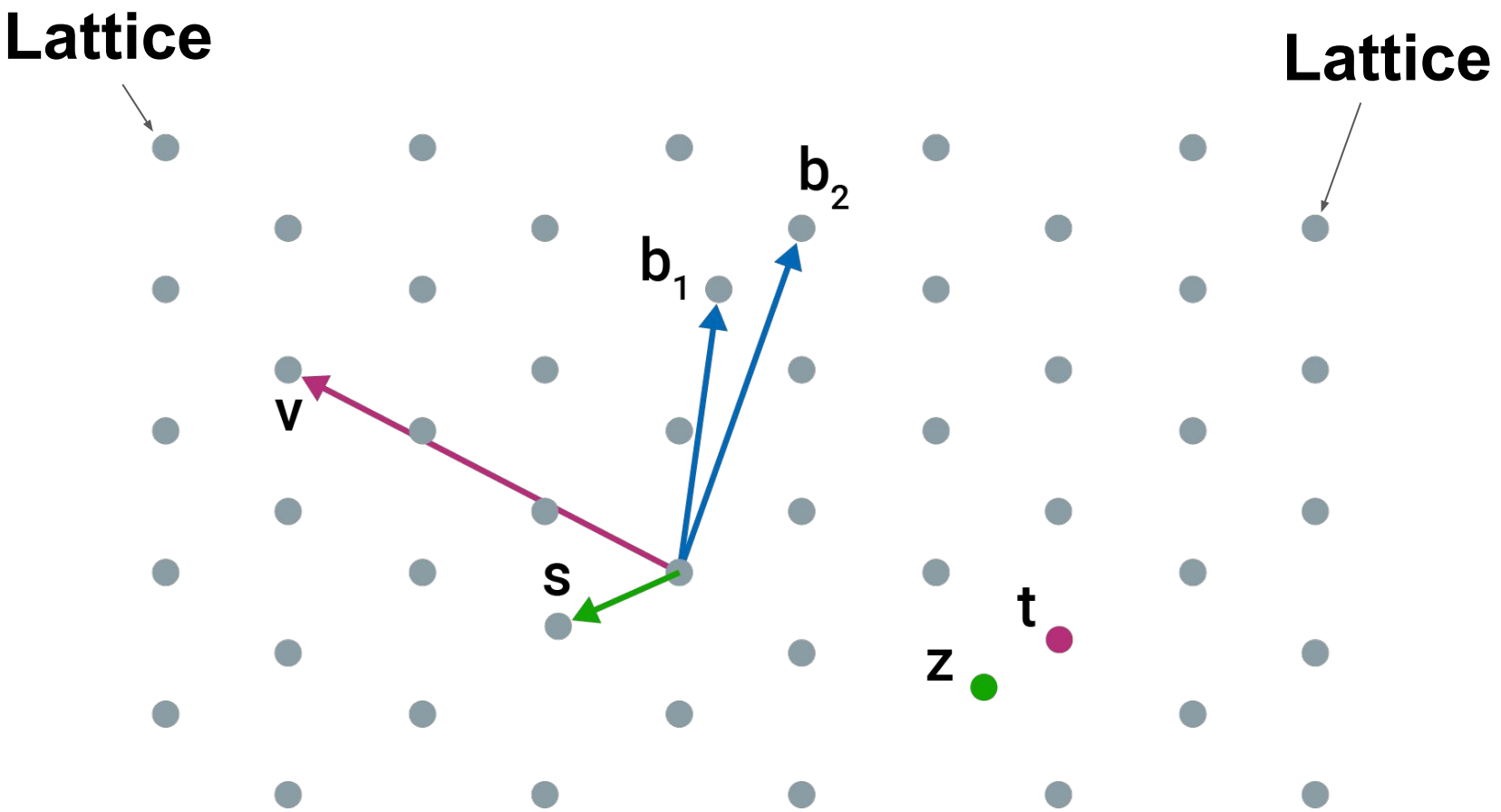
Example:

Imagine you encrypt the numbers 5 and 3. A server can **add the encrypted values**, and when you decrypt the result, you get 8 — **without the server ever knowing what 5 or 3 were**.


Encrypt(5)
Encrypt(3)
E(5) + E(3) = E(8)
Remote server

Lattice-based Cryptography

Lattice-based cryptography is a type of encryption that uses complex geometric structures called **lattices** to secure data. It is considered one of the most promising forms of **post-quantum cryptography** — meaning it's designed to be secure even against **quantum computers**.

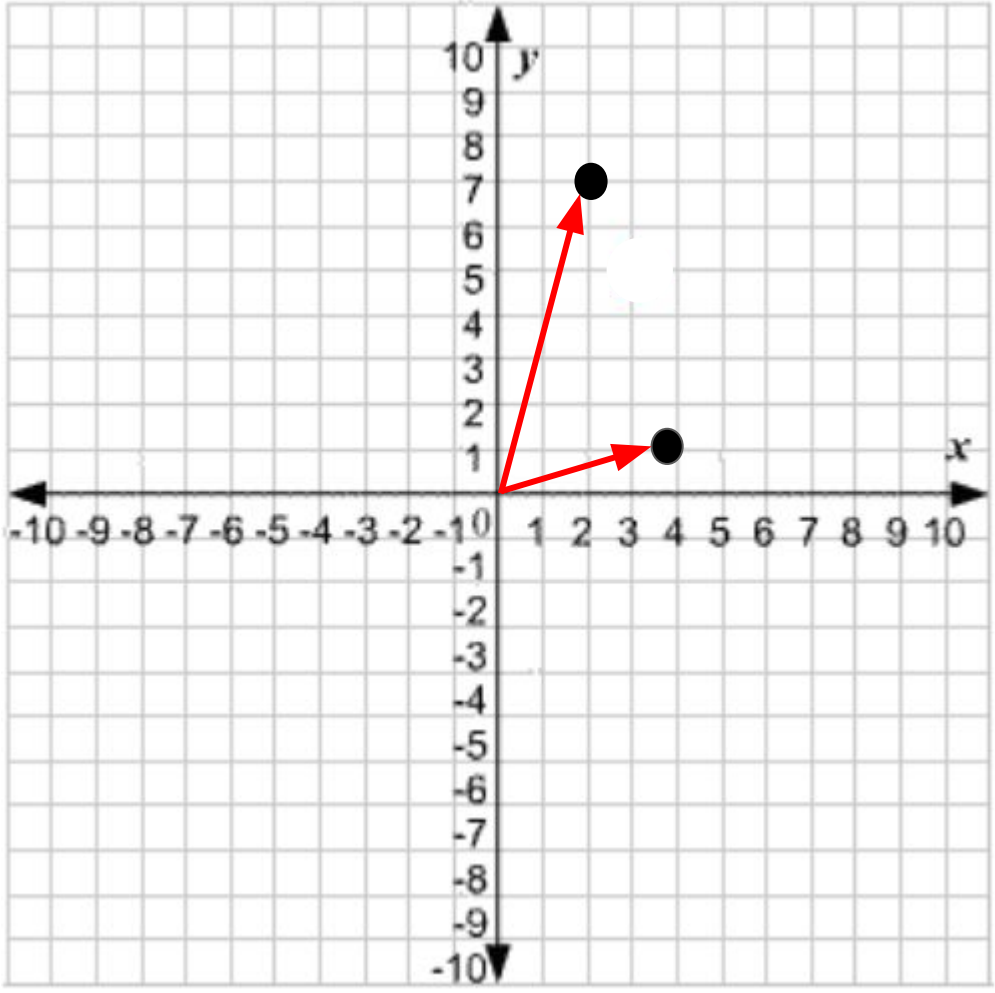

Lattice
Lattice
b2
b1
v
s
t
z

Private key = [3, 5]

S = a1 * s1 + a2* s2 + error (differetial privacy) mod 11

Random A = [a1, a2]

A1 = [2, 7] A1 = 2 * 3 + 7 * 5 + 1 = 42 mod 11 = 9
A2 = [4, 1] A2 = 4 * 3 + 1 * 5 - 1 = 16 mod 11 = 5


y
x
10
9
8
7
6
5
4
3
2
1
-10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0 1 2 3 4 5 6 7 8 9 10
-1
-2
-3
-4
-5
-6
-7
-8
-9
-10

A1 = [2, 7] **A1 = 2 * 3 + 7 * 5 + 1 = 42 mod 11 = 9**

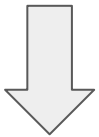
A2 = [4, 1] **A2 = 4 * 3 + 1 * 5 - 1 = 16 mod 11 = 5**

Pr Keys= [3, 5] **Mod = 11**

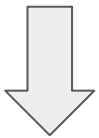
"The remainder after division."

Public keys= ([2, 7], 9), ([4, 1], 5)

Public keys= ([2, 7], 9), ([4, 1], 5)

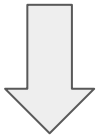


Public key= ([2+4 , 7+1], 9+5 mod 11)

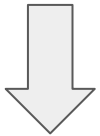


Public key= ([6 , 8], 3)

Public key= ([6 , 8], 3)

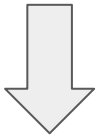


$6 \times 3 + 8 \times 5 = 18 + 40 = 58 \bmod 11 = 3$

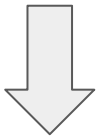


$3 - 3 = 0$

Public key= ([6 , 8], 8)



$6 \times 3 + 8 \times 5 = 18 + 40 = 58 \bmod 11 = 3$



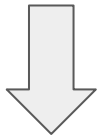
$8 - 3 = 5$

Good: 🍌
Not good: 🍎

Mod = 11 (0 - 10)

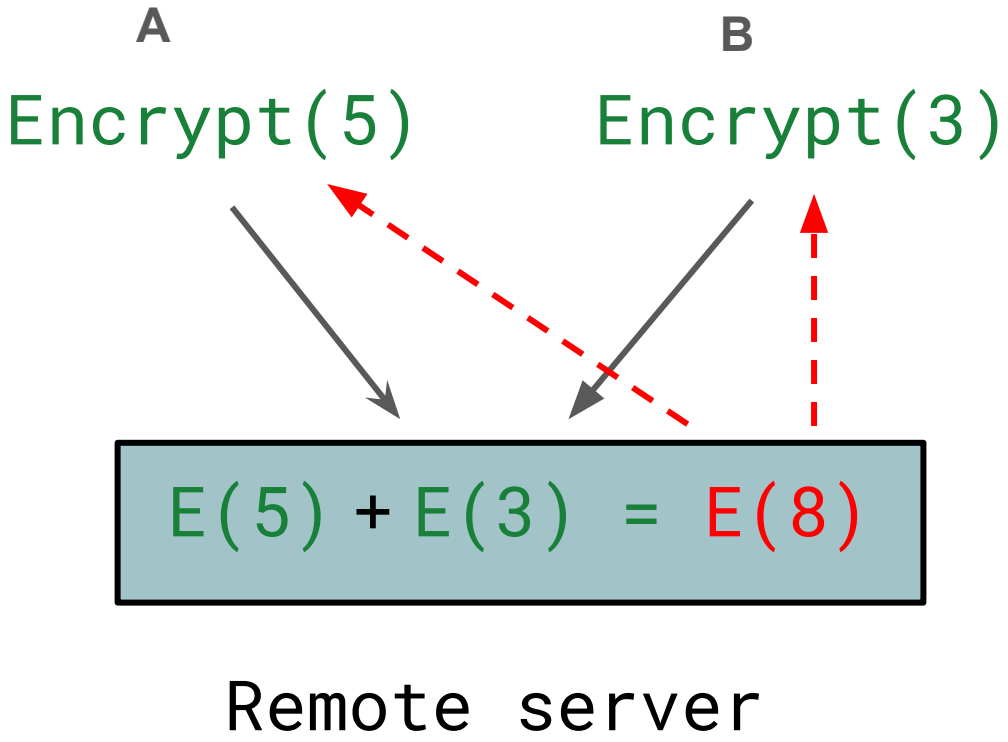
Number close to 0 means 0
Number close to 5 means 1

Only encrypt message for 0 and 1



(like 200) into binary: `11001000`

Why it can be used in HE?



0 = 0

+ 1 = 1

1

1 + 1 = 0

1 + 0 = 1

0 + 1 = 1

0 + 0 = 0

([6,8],3)

([6,8],8)

([6,8],3)+([6,8],8)=([12,16],11) mod 11

([12,16], 0) ⟵ **1**

$3 * 12 + 5 * 16 = 116 \bmod 11 = 6$

$6 - 0 = 6 \bmod 11 = 6$ is close to 5, so, it means 1

HE features:

Enables computation directly on encrypted data.

- No need to decrypt before processing.

Supports Arithmetic Operations

- Can perform **addition** and/or **multiplication** on ciphertexts.

Privacy-Preserving

- Sensitive data can be processed in the cloud **without revealing it**.

Noise Management Required

- Each operation adds **noise** to ciphertext.

HE limtations:

Only numerical data.

Slow computation

Limited operations, + - * /: Only good for Addition, Subtraction, Multiplication, Division.

Why hospitals need HE? (only numerical data)

- **Complies with HIPAA** (U.S. privacy law)
- Avoids patient data leaks when outsourcing to cloud or AI companies
- Enables collaboration across hospitals without revealing raw data

HE can absolutely be used in hospitals, especially for **privacy-preserving AI, analytics, and research collaboration**.
But it's still emerging in practice due to **performance and integration challenges**.

Real Parksinson Datasets

C	FD	FE	FF	FG	FH	FI	FJ	FK	FL	FM	FN	FO	FP	FQ	FR	FS	FT	FU	FV	FW	FX
E_e4	lowput_ex	DATSCAN_	DATSCAN_	con_cauda	ips_cauda	mean_cau	DATSCAN_	DATSCAN_	con_putan	ips_putam	mean_puta	con_striat	ips_striatu	mean_stri	Stage_par	Stage_sub	Stage_PDT	Stage_S	Stage_D	Stage_G	
1	0.422884	2.15	1.53	1.53	2.15	1.84	1.37	0.79	0.79	1.37	1.08	2.32	3.52	1.46	7	22	0	1	1		
1	0.48624	2	1.39	1.39	2	1.695	1.18	0.9	0.9	1.18	1.04	2.29	3.18	1.3675	5	29	0	1	1		
1	0.413708	1.79	1.43	1.43	1.79	1.61	0.89	0.76	0.76	0.89	0.825	2.19	2.68	1.2175	5	34	0	1	1		
1															4	42	0	1	1		
1	0.386278	1.89	1.12	1.12	1.89	1.505	1.11	0.7	0.7	1.11	0.905	1.82	3	1.205	5	42	0	1	1		
0	0.240906	1.19	1.43	1.19	1.43	1.31	0.48	0.58	0.48	0.58	0.53	1.67	2.01	0.92	7	24	0	1	1		
0	0.273404	1.3	1.32	1.3	1.32	1.31	0.64	0.54	0.64	0.54	0.59	1.94	1.86	0.95	7		1	1	1		
0															6		1	1	1		
0															9		1	1	1		
1	0.217016	1.21	1.46	1.21	1.46	1.335	0.45	0.78	0.45	0.78	0.615	1.66	2.24	0.975	6	15	0	1	1		
1	0.243108	1.32	1.54	1.32	1.54	1.43	0.5	0.64	0.5	0.64	0.57	1.82	2.18	1	7	16	1	1	1		
1															6	13	1	1	1		
0	0.321371	1.86	2.36	1.86	2.36	2.11	0.72	1.38	0.72	1.38	1.05	2.58	3.74	1.58	11	34	0	1	1		
0	0.341553	1.82	2.41	1.82	2.41	2.115	0.76	1.21	0.76	1.21	0.985	2.58	3.62	1.55	10	32	1	1	1		
0	0.280867	1.65	2.09	1.65	2.09	1.87	0.62	1.01	0.62	1.01	0.815	2.27	3.1	1.3425	17	30	1	1	1		
0															9	27	1	1	1		
2	0.17646	0.86	1.2	0.86	1.2	1.03	0.33	0.48	0.33	0.48	0.405	1.19	1.68	0.7175	2	20	0	1	1		
2	0.167254	0.93	1.23	0.93	1.23	1.08	0.31	0.4	0.31	0.4	0.355	1.24	1.63	0.7175	3	18	1	1	1		
2	0.152435	0.86	1.34	0.86	1.34	1.1	0.28	0.42	0.28	0.42	0.35	1.14	1.76	0.725	2	37	1	1	1		
2															3	32	1	1	1		
0	0.448237	1.91	2.5	1.91	2.5	2.205	0.94	1.28	0.94	1.28	1.11	2.85	3.78	1.6575	7	18	0	1	1		
0	0.389459	1.6	2.38	1.6	2.38	1.99	0.81	1.35	0.81	1.35	1.08	2.41	3.73	1.535	7	17	1	1	1		
0	0.459581	1.72	2.35	1.72	2.35	2.035	0.95	1.09	0.95	1.09	1.02	2.67	3.44	1.5275	7	21	1	1	1		
0															16	28	1	1	1		
1	0.359569	1.8	1.47	1.47	1.8	1.635	0.87	0.79	0.79	0.87	0.83	2.26	2.67	1.2325	1	20	0	1	1		
1	0.311614	2.06	1.59	1.59	2.06	1.825	0.82	0.68	0.68	0.82	0.75	2.27	2.88	1.2875	2	31	0	1	1		
1	0.258436	1.68	1.55	1.55	1.68	1.615	0.58	0.56	0.56	0.58	0.57	2.11	2.26	1.0925	3	15	1	1	1		
1															1	20	1	1	1		
0	0.357273	2.29	2.97	2.29	2.97	2.63	0.73	1.57	0.73	1.57	1.15	3.02	4.54	1.89	12	30	0		1		
0	0.261798	1.65	2.45	1.65	2.45	2.05	0.53	0.87	0.53	0.87	0.7	2.18	3.32	1.375	9	33	1		1		
0	0.24342	1.36	2.24	1.36	2.24	1.8	0.49	0.71	0.49	0.71	0.6	1.85	2.95	1.2	11	40	1		1		
0															9	49	1		1		

Python Libraries

Homomorphic Encryption (HE)

https://github.com/Lab41/PySEAL

Zero Knowledge Proof

https://github.com/sdiehl/zkp

Zero Knowledge Proof

https://github.com/sdiehl/zkp

Threshold signatures

https://cryptography.io/en/latest/

pip install cryptography

The most popular python library for encryption

https://pypi.org/project/cryptography/

Encryption/Decryption:

- Symmetric (e.g., AES)
- Asymmetric (e.g., RSA, ECC)

Hashing (e.g., SHA-256)

Digital Signatures (e.g., RSASSA-PSS)

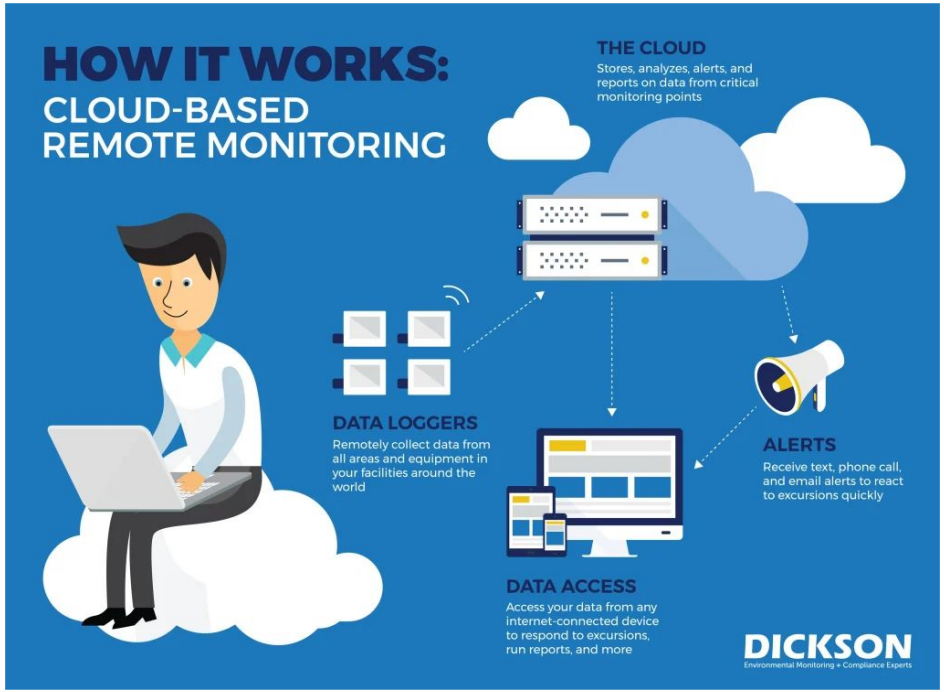
Certificates, PKI tools

Use case

Remote monitoring involves observing and tracking data from a distance, often using technology like sensors and networks, to gain insights into systems, equipment, or even people's health.

Doctor has Keys to decrypt remote patient datasets.

Patients' datasets are encypted by HE, send to cloud for computation and results will be sent to doctors.



Conclusions and Discussions: Questions