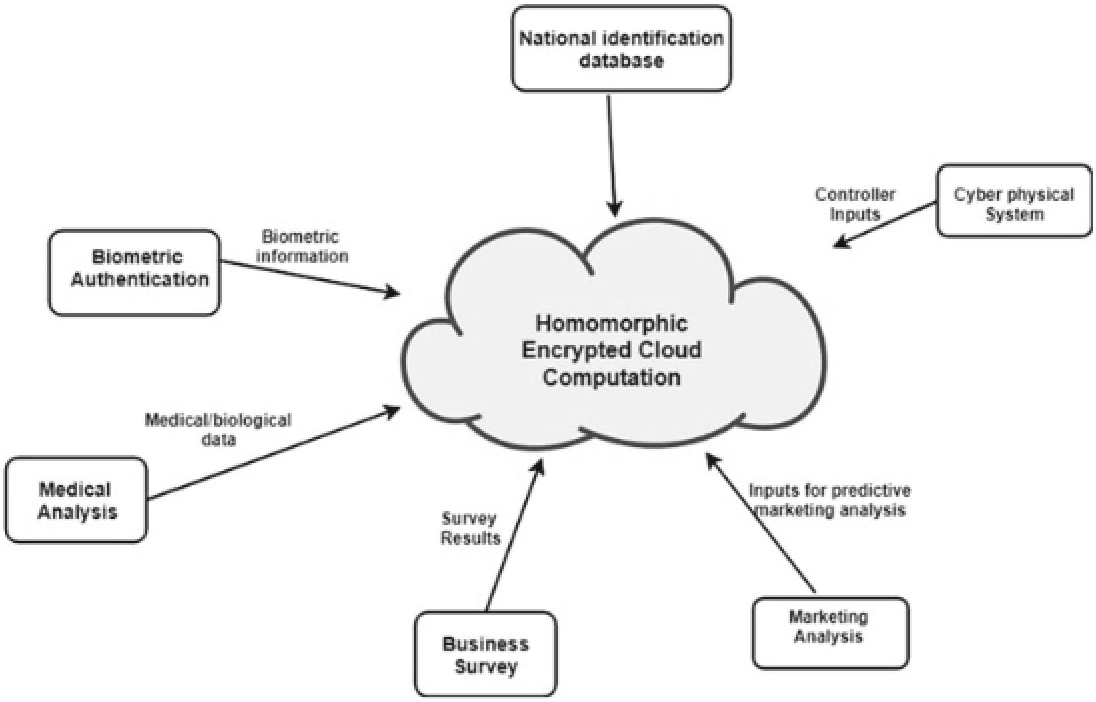
1. Homomorphic Encryption

Homomorphic Encryption

- https://www.youtube.com/watch?v=umqz7kKWxyw - Good talk

Usually homomorphic encryption is easier explained through applications. So let's go through some examples

Applications



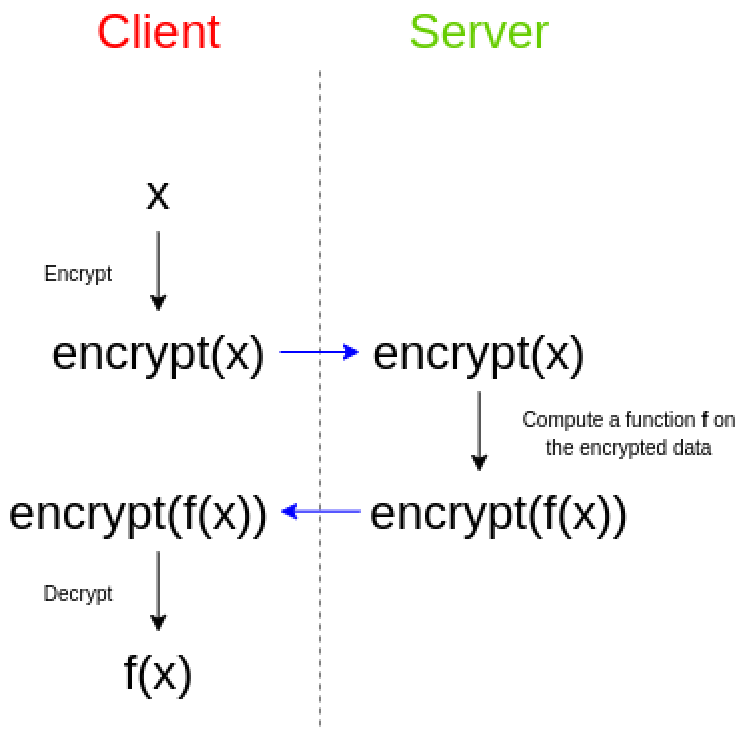
Outsourcing storage and computations

- Let A be a company that wants to store data in the cloud C
- A doesn't want the cloud provider C to see the sensitive data. Therefore he encrypts the information

Problem
A wants to use the information (do computations on it) without locally retrieving it and decrypting it (defeating the purpose of storing it in the cloud)
Solution

- Homomorphic encryption

- The cloud provider can process the information in the encrypted form



Private queries

- Let D be a database provider

Problem
The client A wants to retrieve a query without the database provider D learning which query it is
Solution
Homomorphic encryption lets us encrypt the index of the record

Limitations

- Encrypted output
- All inputs must be encrypted under the same key
- No integrity guarantees

Definitions

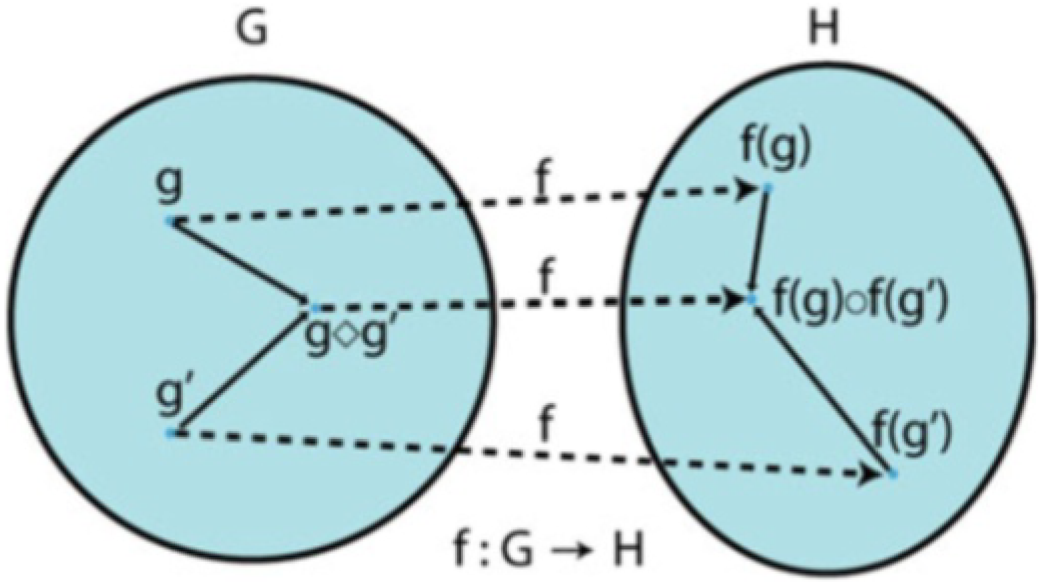
Homomorphism

Let

- G, G' be groups

> A Homomorphism is a map $f : G \to G'$ with the following property:
>
> $$f(xy) = f(x)f(y) \; \forall x, y, \in G$$

- Homomorphisms preserve structure



Example
$x \mapsto e^x$ is a homomorphism from the multiplicative to the additive group

Homomorphic encryption

> Let $(KeyGen, Encrypt, Decrypt, Evaluate)$ be a tuple of procedures ($(KeyGen, E, D, Eval)$)
> Let $C \in \mathcal{C}$ be a circuit where $\mathcal{C}$ is the permitted family of circuits
>
> - $(sk, pk) \leftarrow KeyGen(1^\lambda, 1^\tau)$ - is a *randomized* algorithm takes *security* parameter λ and a *functionality* parameter τ and outputs secret/public key pair
> - $c \leftarrow (E(pk, b))$ - E with a is a *randomized* algorithm that encrypts a bit b
> - $b \leftarrow (D(sk, c))$ - Decrypts the bit from the ciphertext
> - $\vec{c'} \leftarrow Eval(pk, C, \vec{c})$
>   - C is a circuit
>   - $\vec{c} = (c_1, ...c_t)$
>   - takes a vector of ciphertexts and outputs another vector of ciphertexts

Corectness

Correctly decrypt both fresh and evaluated ciphertexts

- $\forall C \in \mathcal{C},\ \forall b \in \{0, 1\}$
- $Pr[D(sk, E(pk, b)) = b] = 1$
- $Pr[D(sk, Eval(pk, C, E(pk, b))) = C(\vec{b})] = 1$

Properties

Security

- The security is the classic semantic security definition of indistinguishability. This is given by the (E, D) algorithms

Compactness

- A very important property that must be satisfied

> *Intuitive definition*
>
> - The size of the ciphertext does not grow with the complexity of the evaluated circuit
> - There is a polynomial f s.t $\forall\lambda$ (security parameter) the decryption algorithm can be expressed as a circuit of size at most $f(\lambda)$

Circuit privacy

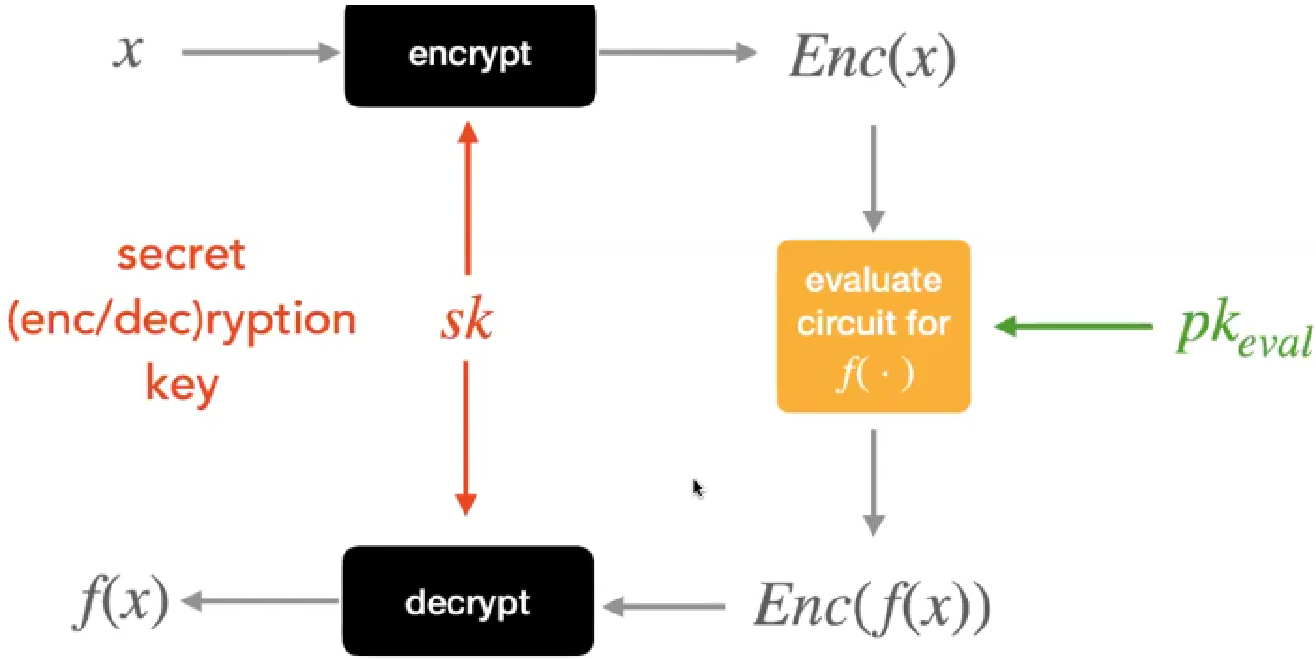
> The ciphertext does not reveal anything about the circuit that it evaluates .beyond the output value

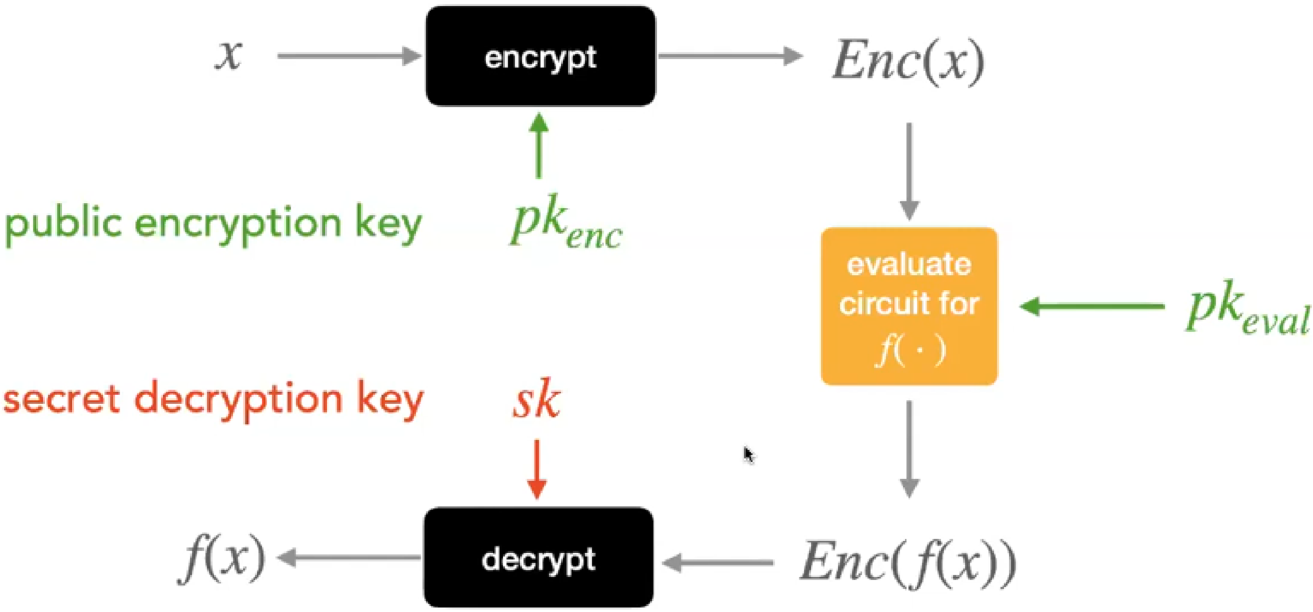
Note

- This can be extended to vectors of ciphertexts instead of ciphertext bits

Other structures:

- Equivalent constructions





Classification

- https://vitalik.ca/general/2020/07/20/homomorphic.html
  Partially homomorphic encryption

> Given $E(m1)$, and $E(m2)$ you can do limited operations (Addition, multiplication)

Somewhat homomorphic encryption

> Limited number of multiplications (Circuits of a maximum depth)
> Given $E(m_1), ..., E(m_n)$ you can compute $E(p(m_1, ...m_n))$ where p is a polynomial of a limited degree

Fully homomorphic encryption

> Unlimited multiplications and additions

Resources

- https://en.wikipedia.org/wiki/Homomorphic_encryption
  High level explanations
- https://www.youtube.com/watch?v=2TVqFGu1vhw
- https://vitalik.ca/general/2020/07/20/homomorphic.html