Key++

A Blockchain Based FHE Service

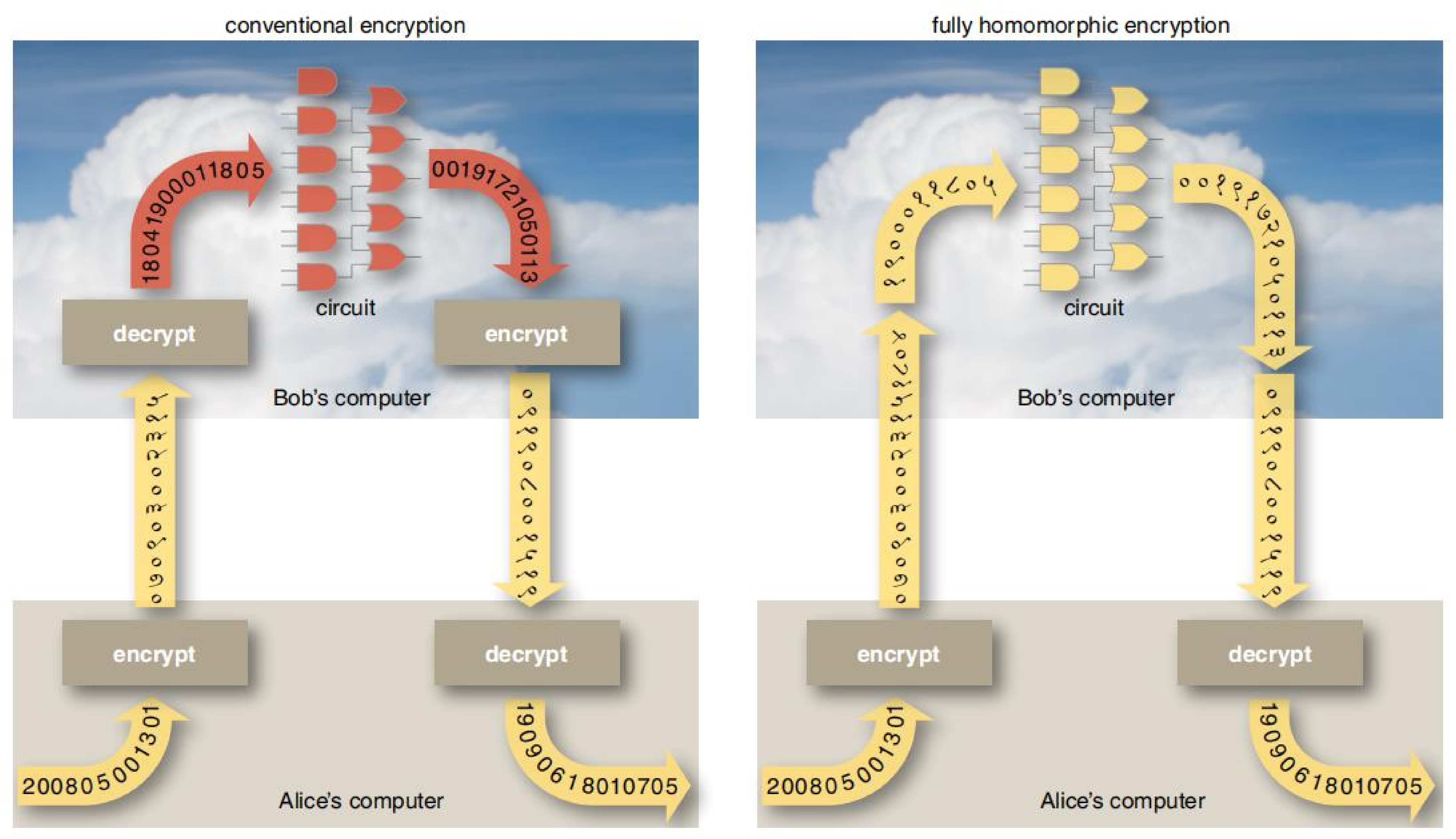
support@keyxx.com

www.keyxx.com

Why Homomorphic Encryption?

- When data is encrypted into ***Cipher*** form, no one can read without decryption. So cryptography should be the best way of privacy protecting.
- But the purpose of sharing data through multiple parties is to ***Leverage & Compute*** them, not just for backup.
- The fact that ciphers of conventional encryption scheme can not be computed makes cryptography hard to be used.
- However, Homomorphic Encryption (HE) can compute encrypted data directly, which gives a perfect solution of ***Utilizing & Protecting*** data simultaneously.
- That's why HE had been believed as the "***Holy Grill***" of cryptography since it was proposed in 1978 by Ron Rivest, etc.
- Today, tech gaints such as Google, IBM, Intel, Microsoft are working on this field. But the current HE techonology is still on its very early stage, which is ***Low Efficient*** for real business cases.

FHE re-defines the mode of Cloud Computing.

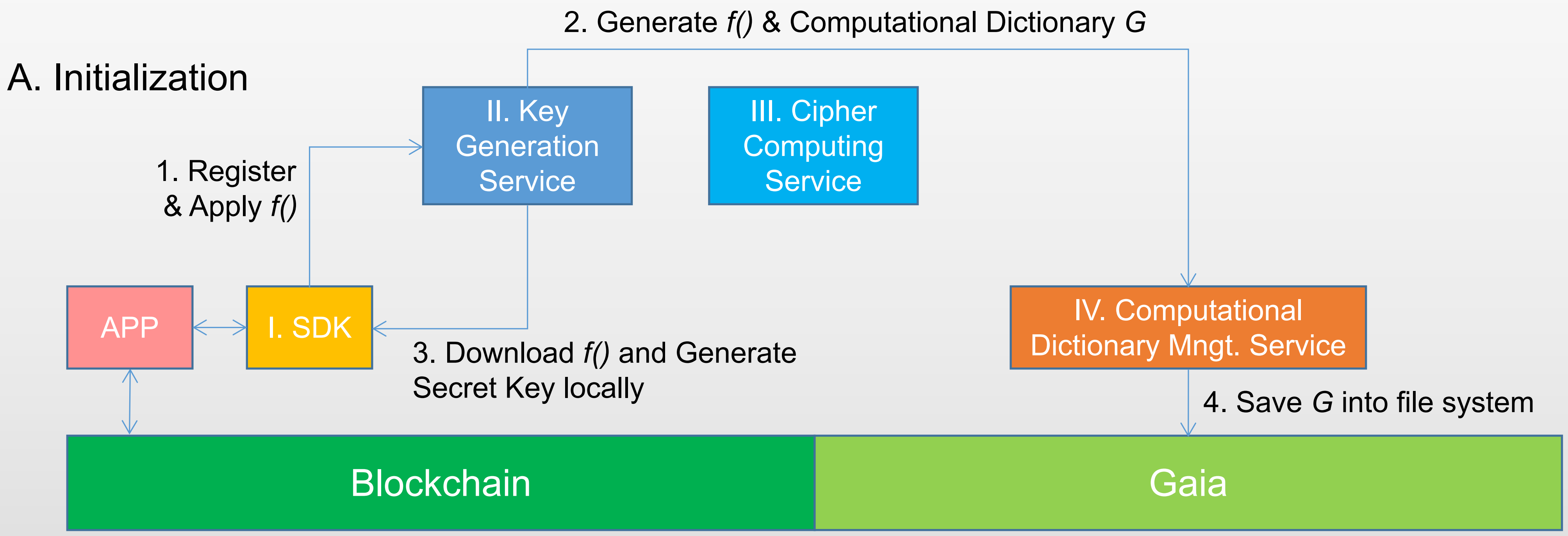


Key++ uses patented FHE technology

- A high performance FHE scheme named ShaftStop was proposed in 2015 by the team of this project.
- We've already got 2 authorized Chinese patents, and 5 others including one US patent are waiting to be authorized.
  - “POLYNOMIAL FULLY HOMOMORPHIC ENCRYPTION SYSTEM BASED ON COEFFICIENT MAPPING”, US15736648, 2017-12-14.
- ShaftStop uses coefficient mapping polynomial to archeive Homomorphism.
  - The principle formula is: $P=\sum_i a_i f(x_i)y_i$
  - A Classified function *f()* is introduced as a part of the secret key to enhance the security.
  - The concept of Computational Dictionary is proposed to compute encrypted data.
  - This scheme is extremely efficient.

Enpower every APP with FHE capacity

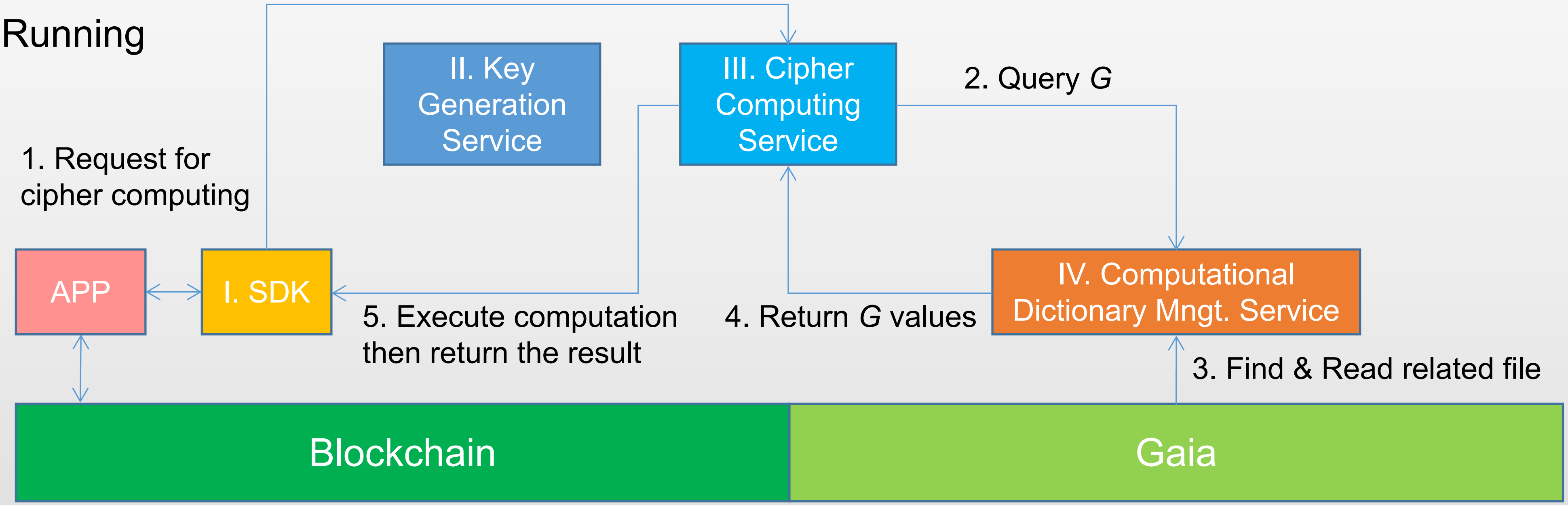
- Key++ is a blockchain based Layer2 solution of providing FHE service to the community. Anyone can use this service very easily to gain the power of FHE.
- It has four major parts: Developer SDK, Key Generation Service, Cipher Computing Service, and Computational Dictionary Management Service.


2. Generate *f()* & Computational Dictionary *G*
A. Initialization
II. Key Generation Service
III. Cipher Computing Service
1. Register & Apply *f()*
APP
I. SDK
3. Download *f()* and Generate Secret Key locally
IV. Computational Dictionary Mngt. Service
4. Save *G* into file system
Blockchain
Gaia

Enpower every APP with FHE capacity

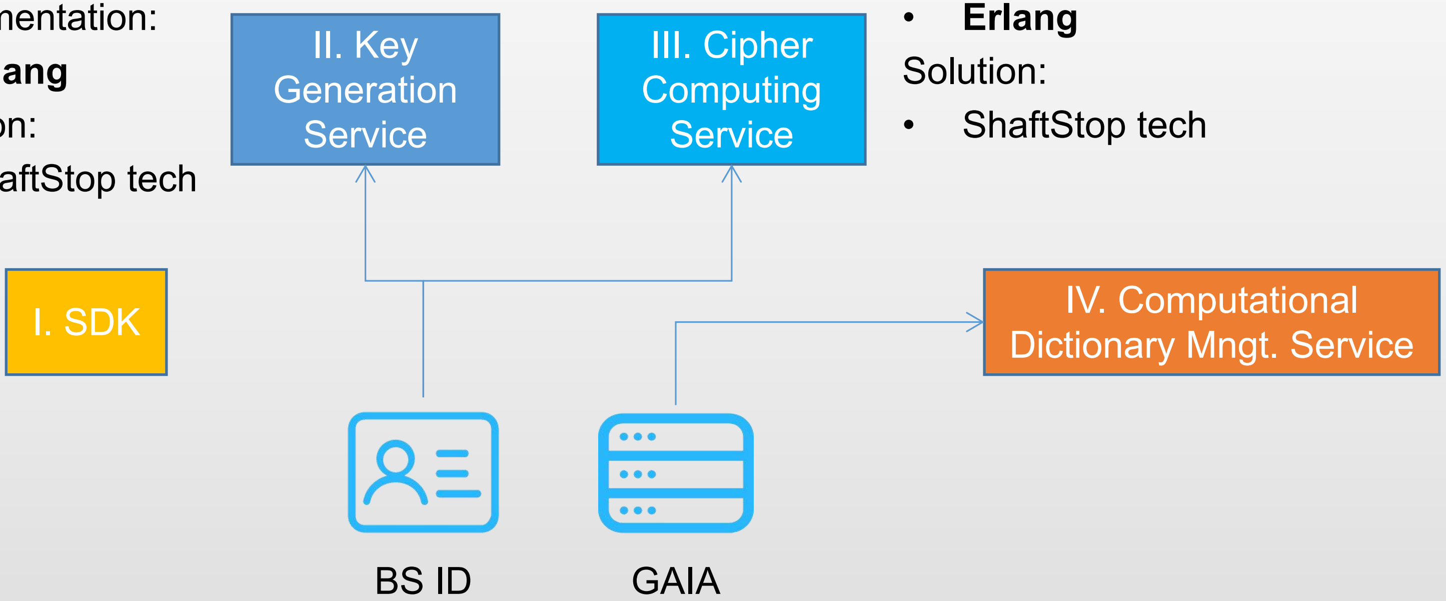
- Key++ is a blockchain based Layer2 solution of providing FHE service to the community. Anyone can use this service very easily to gain the power of FHE.
- It has four major parts: Developer SDK, Key Generation Service, Cipher Computing Service, and Computational Dictionary Management Service.

B. Running


II. Key Generation Service
III. Cipher Computing Service
2. Query *G*
1. Request for cipher computing
APP
I. SDK
5. Execute computation then return the result
4. Return *G* values
IV. Computational Dictionary Mngt. Service
3. Find & Read related file
Blockchain
Gaia

Conceptual Design of Key++

- In the initial stage, Key++ services are deployed on centralized servers or cloud based platforms, such as GAE, SAE, Pivotal or HEROKU.
- Part III and Part IV can be migrated to decentralized platform later.



II. Key Generation Service
III. Cipher Computing Service
I. SDK
IV. Computational Dictionary Mngt. Service
BS ID
GAIA

Functions:

- Generate *f()*
- Generate *G*

Implementation:

- **Erlang**

Solution:

- ShaftStop tech

Functions:

- Compute Ciphers

Implementation:

- **Erlang**

Solution:

- ShaftStop tech

Functions:

- Simple Key-Value DB Service
- Authorization

Implementation:

- **JAVA**

Solution:

- Split *G* into small files
- Maintain the index table
- Access file system, i.e. GAIA

Functions:

- Key Gen. & Mngt.
- Encrypt & Decrypt data
- Cipher Computing

Implementation:

- **C#**, **JAVA**, Python, Golang...

Business Model

- Key++ can be seen as a middleware, or one kind of PaaS providing FHE services.
- So the natural charging mode is pay-per-usage.
  - Generating *f()* and *G* requires a relatively high expense.
  - Computing ciphers charges low, but the request happens more frequently.
  - Users also need to pay for the storage space of *G* monthly.
- Because the cost structure of Key++ is also basing on the usage of the infrastructures, it is easy to sustain a reasonable GPR.
- The payment from users mainly covers the operational cost of the project. When the decentralized version came online, the cost of using Key++ could be dramatically decreased.
- Our purpose is to enpower the developers and promote the ecosystem.
- An open source version with basic functionalities 'Holycloak' is planed to be developed later.

End