



BoAT Framework SDK

Getting Started

aitos.io
Dec .2020
AITOS.19.70.100UM
2.0.2

BoAT Framework Overview

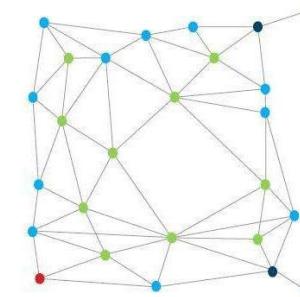
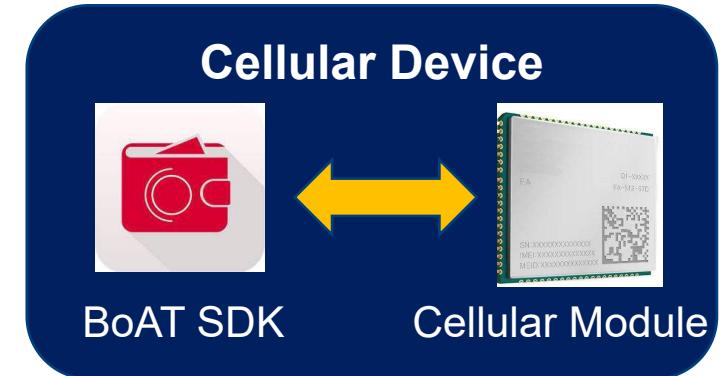
What is BoAT Framework

BoAT is a blockchain application framework for cellular modules.

BoAT runs as an SDK for IoT applications in cellular module. It could generate unique identity (cryptographic keypair) for the device, sign the IoT data with the key and invoke blockchain smart contract with the signed data. The blockchain stores the signed data in a decentralized way to ensure the data are tamper-resistant during the full lifecycle, which is essential for data value.

BoAT IoT SDK Features

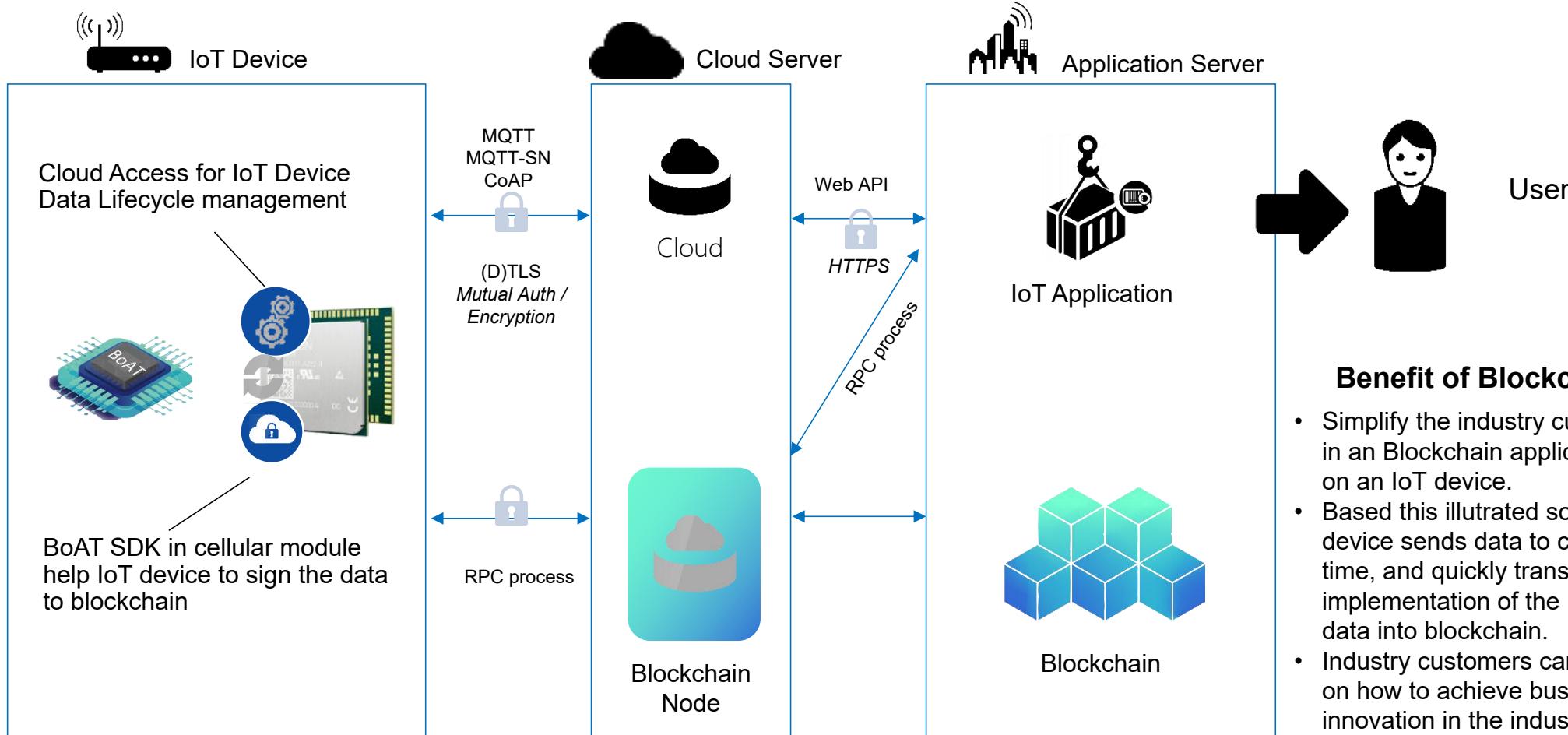
- ✓ Sign data
- ✓ Send blockchain transactions
- ✓ Invoke blockchain smart contract
- ✓ Manage blockchain cryptography keys in the IoT device
- ✓ C and Java implementation



Blockchain

BoAT: Evolution form IoT to IoT + Blockchain

IoT modules are easily retrofitted and upgraded into "Blockchain-Ready" modules.

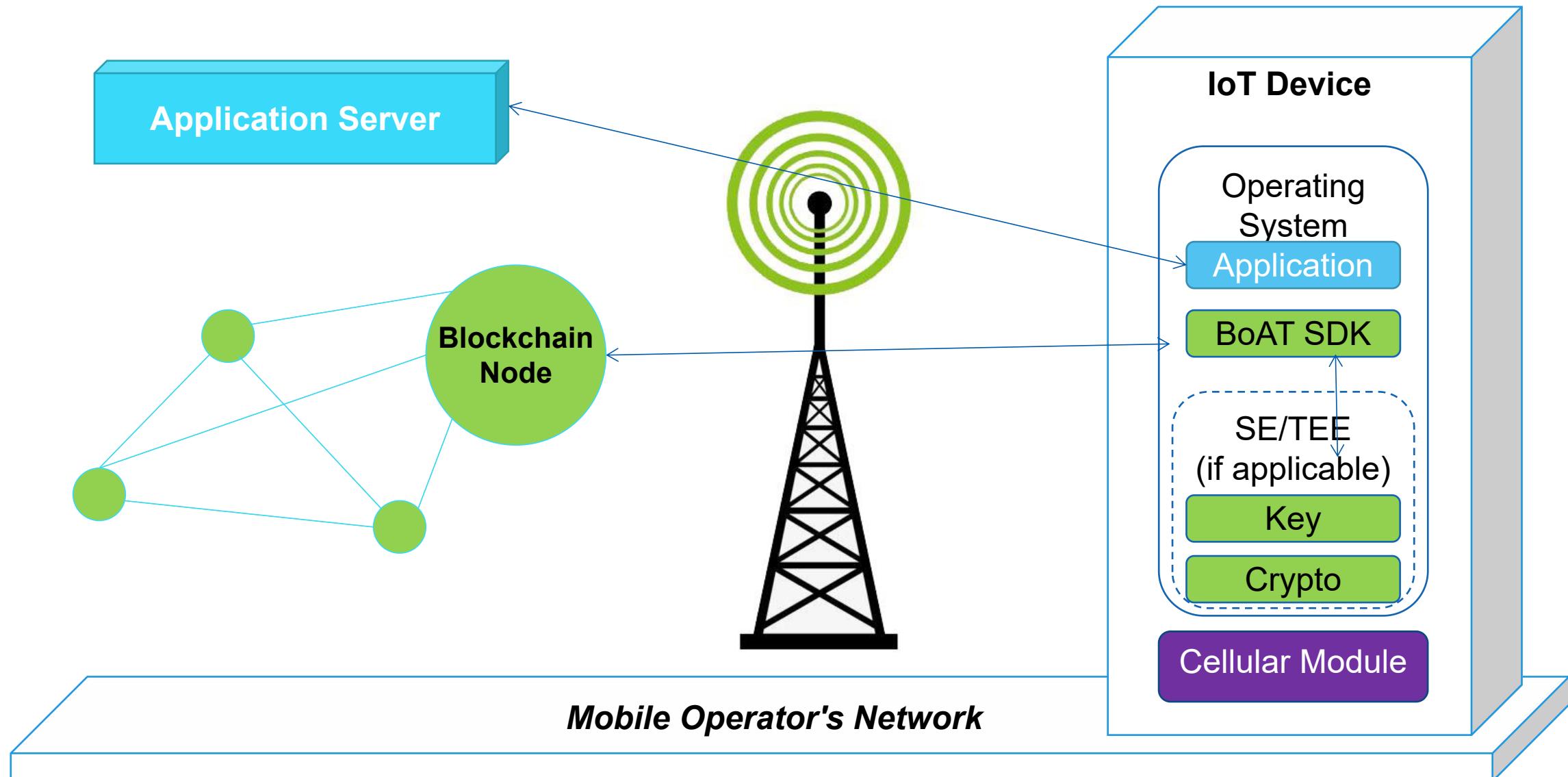


Benefit of Blockchain Module

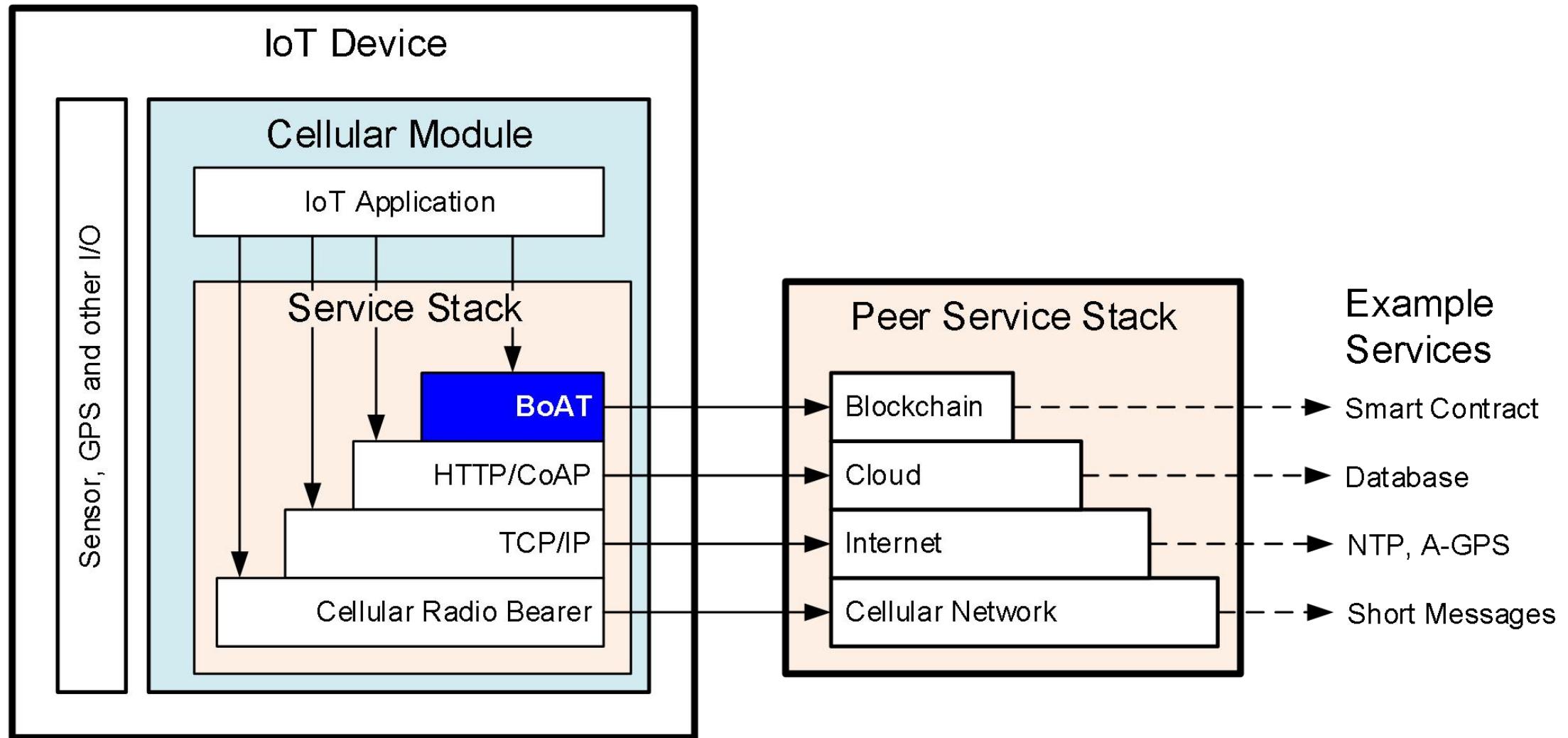
- Simplify the industry customer to engage in an Blockchain application pilot project on an IoT device.
- Based this illustrated solution, the IoT device sends data to cloud, at the same time, and quickly transforms the implementation of the IoT device sending data into blockchain.
- Industry customers can thus focus more on how to achieve business model innovation in the industry scenario with blockchain capabilities

BoAT Framework and

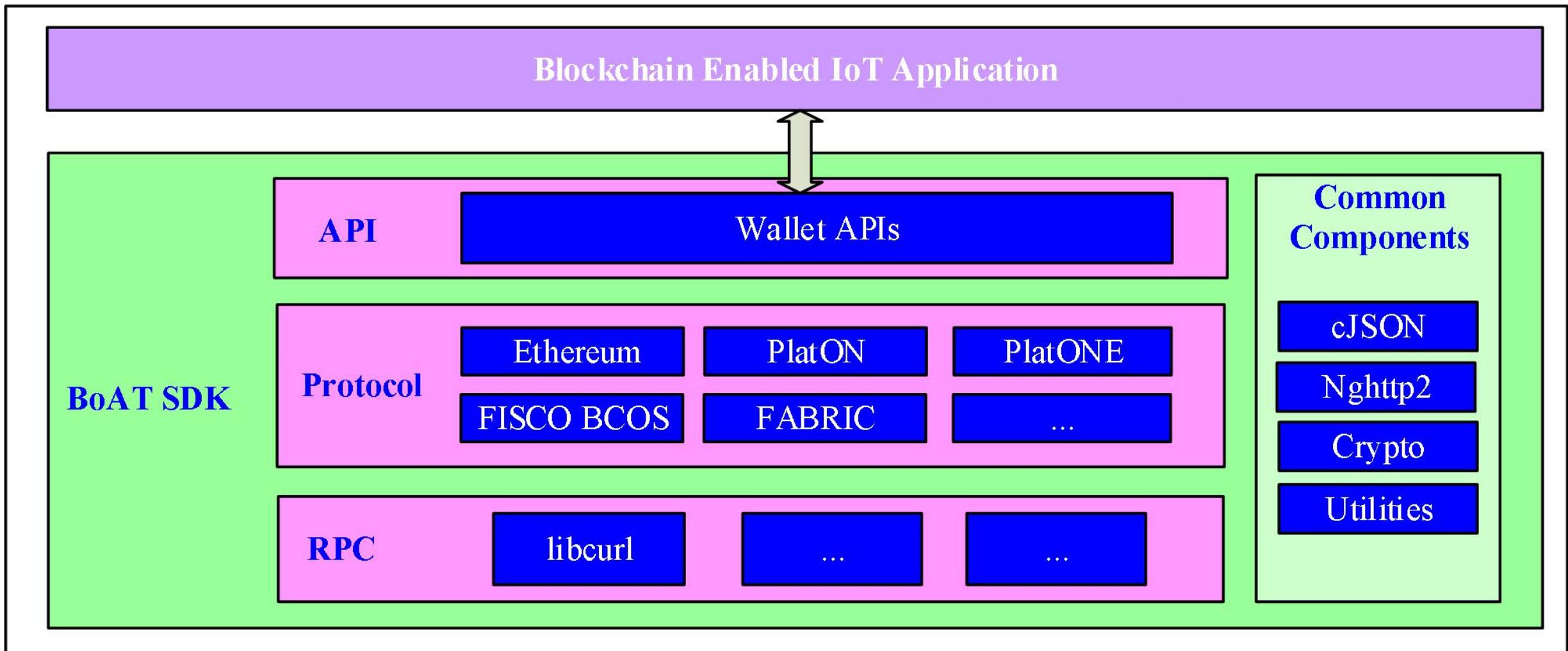
BoAT Elements



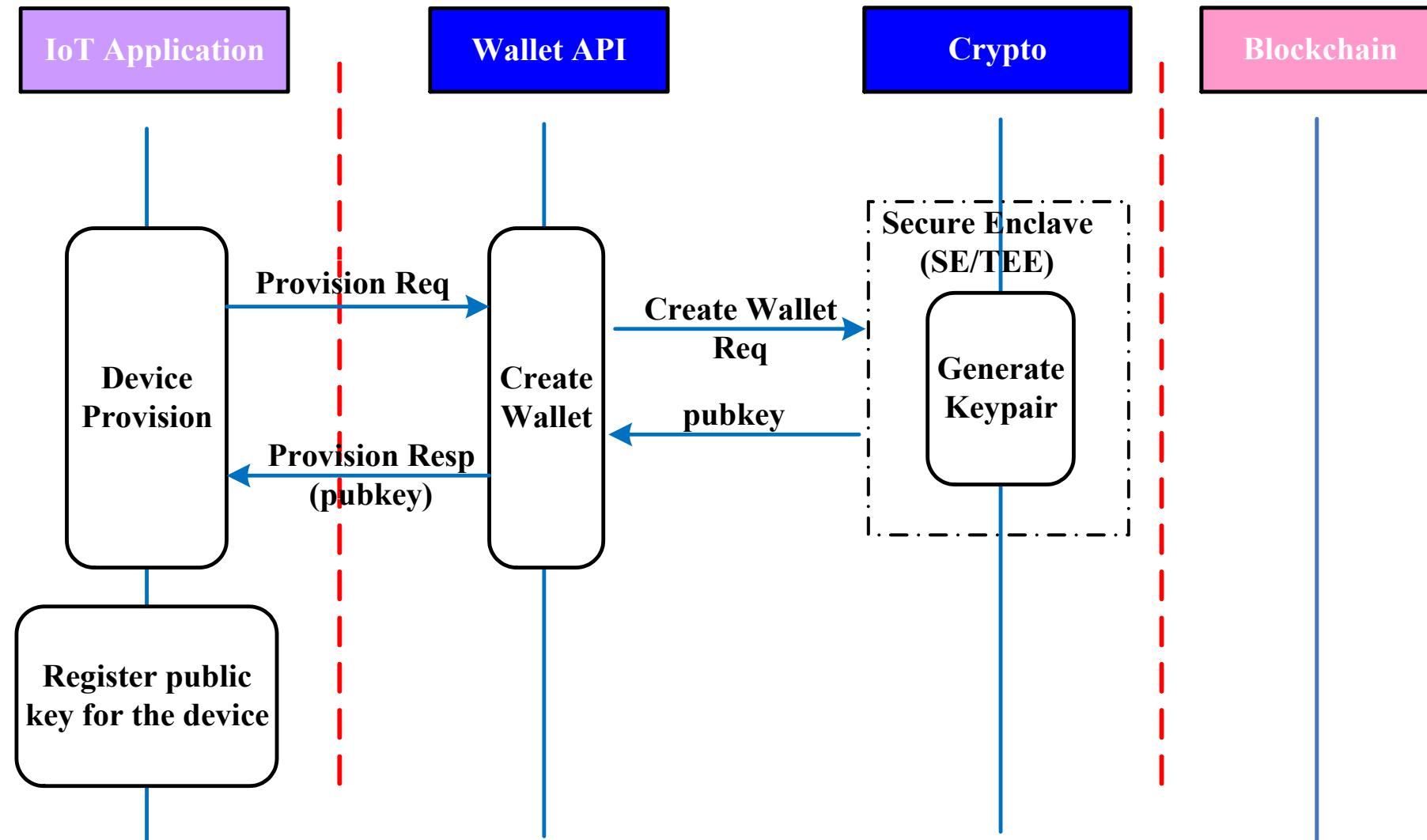
BoAT in System



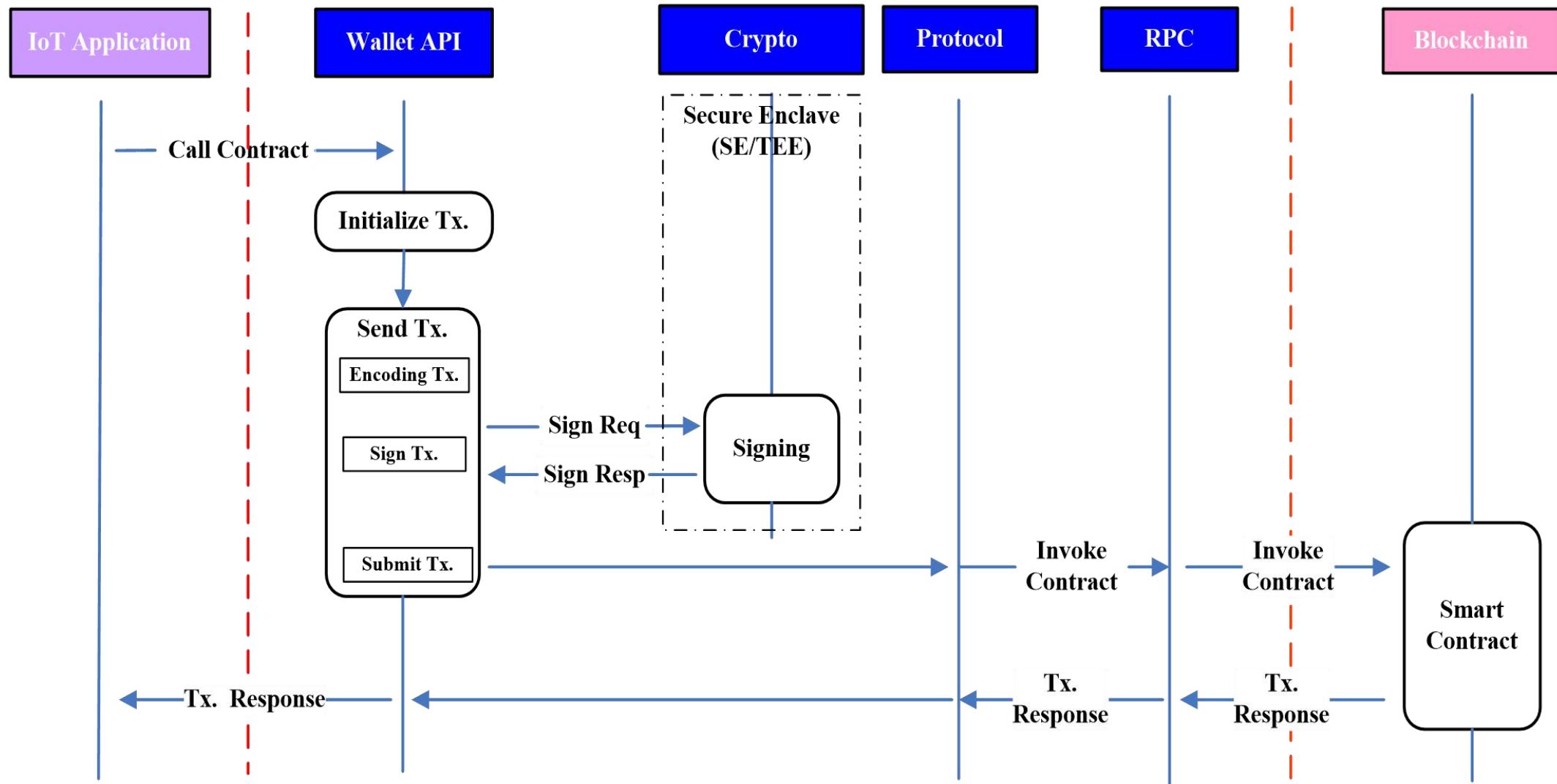
BoAT Software Architecture



Typical Procedure: Device Wallet Provision



Typical Procedure: Call Contract



BoAT IoT Framework SDK (C language)

Characterics

BoAT IoT Framework SDK is a C-language blockchain client SDK for Ethereum, PlatON, PlatONE, HyperLedger Fabric and FISCO BCOS.

Dependencies

BoAT is an embedded software for cellular modules and thus it depends on specific development environment for different types of cellular modules. Among them, linux-default version of BoAT IoT Framework SDK is tested with following dependencies. Lower version of those dependent softwares probably work well but are not tested.

Host OS - linux or Cygwin

Compiler - gcc 6.5.0

Cross Compiler - arm-oe-linux-gnueabi-gcc 4.9.2

GNU Make - 4.3

Python 2.7 (Python 3 is also compatible)

libcurl - 7.55.1

BoAT IoT Framework SDK Source Directory

```
<SDKRoot>
|
+---build           | Directory to store object and executable files
+---contract        | Contract ABI and generated C interface file
+---demo            | Boat SDK usage demo
+---docs            | API reference manual
+---hwdep           | Hardware dependency
|   \---default     | A default pure software version for development
+---include          | Header files for application to include
+---lib              | Lib files for application to link with
+---sdk              | SDK source
|   +---include      | Header files for SDK internal use
|   +---platform     | platform releated code
|   +---protocol     | Blockchain client protocol implementation
|   +---rlp           | RLP encoder
|   +---rpc           | Remote procedure call wrapper
|   +---third-party   | dependent third-party library
|   +---utilities     | Utility APIs
|   \---wallet         | SDK entry API implementation
+---tests            | Tests case
\---tools            | Tools for generating C interface from contract ABI
```

NOTE: ./build and ./lib are created in building

Run the Demo

To run demo you must have known the IP address of a blockchain node (Ethereum, PlatON, PlatONE, HyperLedger Fabric or FISCO BCOS) and migrate demo contract to the blockchain network.

Ethereum

This document takes Ethereum as an example to introduce the process of running the demo.

Other BlockChain

See the following links for details on how to set up blockchain nodes and deploy smart contract on them:

- PlatON : <https://www.platon.network>
- PlatONE : <https://platone.wxblockchain.com>
- Hyperledger Fabric : <https://hyperledger-fabric.readthedocs.io/en/release-1.4>
- FISCO BCOS : <https://fisco-bcos-documentation.readthedocs.io/en/latest>

Run the Demo

Blockchain Node Setup

In practice it's a good idea to install Truffle and Ganache, which are the popular Ethereum development suite and blockchain node simulator. Ethereum testnet such as Ropsten is also a good idea.

NOTE: Ganache 1.x or ganache cli cannot keep the "state" information once the process is terminated. If you're using such versions, you have to migrate the contract every time you run Ganache.

To install Truffle and Ganache, visit: <https://truffleframework.com>

Run the Demo

Contract Migration

You can deploy the solidity contracts with Truffle. See truffle documents for details. Demo contracts lie in <SDKRoot>/demo/demo_ethereum/demo_contract.

Modify the Key and Address

Modify the blockchain node URL (IP address and port), private key (or wallet index), recipient address / contract address in demo code to the value as you deployed.

Build

Step 1:

Extract boatiotsdk source to anywhere that you have write permission. Open a terminal window and cd into the directory where you have extracted boatiotsdk.

Run the Demo

Build (cont.)

Step 2:

Make sure dependency is satisfied.

Configure environment for cross-compiler if you are going to cross-compile boatiotsdk.

Step 3:

To build SDK library:

```
$make boatlibs
```

To build SDK demo:

```
$make demo
```

The generated demo locates at ./build/demo and libraries in ./lib.

Run the Demo

Run

To run the demo, open a terminal window, change working directory to *boatiotsdk* and execute:
`./build/demo/demo_ethereum_xxx`

The execution result will be printed in the terminal window.

Using BoAT IoT Framework SDK Library in Your Code

Contract C Interface Generation

Smart contract is the code running on the blockchain virtual machine. Smart contract runs like remote API calls. Though the programming language of smart contract is not C, it has defined ABI (Application Binary Interface). Remote call to the contract must follow the ABI.

However manually applying the rule of ABI is quite complex for embedded C programmers. BoAT IoT Framework SDK provides some tools to generate C interface codes from the ABI. The generated C API can be called from other part within the C project. Though not all contract ABIs can be converted to C interface due to lack of object-oriented programming capability, the tools could ease a lot of works.

The generation tools are written in Python and lie in `./tools`.

Copy the ABI json file generated by truffle or ctool during contract compilation, to the corresponding directory in `./contract`. The generation tool will be called against the ABI file during make. You can include generated head files (`./contract/generated`) in your C code to call the APIs.

Using BoAT IoT Framework SDK Library in Your Code

How to Call a Contract in Your C code

Take Ethereum as an example, following are the typical steps of calling a contract in C code:

1. Call BoatlotSdkInit() to initialize BoAT IoT Framework SDK.
2. Call BoatWalletCreate() with appropriate configuration to create a wallet.
3. Call BoatEthTxInit() with the wallet reference and other parameters to initialize a transaction object (even if it's a state-less call).
4. Call generated C interface API with the initialized transaction object and other arguments.
5. Check the return value of the C interface API. If it's not NULL, parse the string content as per contract prototype.
6. Call BoatlotSdkDeInit() to de-initialize BoAT IoT Framework SDK.

See demo codes for details.

To manually organize a contract call, refer to the generated C API codes. Note that the interface definition varies for different blockchain protocols.

Using BoAT IoT Framework SDK Library in Your Code

Configure Your Makefile and C code

To use BoAT IoT Framework SDK in your own code, please following these steps:

Step 1:

Place SDK source somewhere in your project and build SDK libraries.

Step 2:

Modify Makefile of your project:

Add include file search path: <SDKRoot>/include

Add to link options all library files in <SDKRoot>/lib in sequence:

 libboatwallet.a libboathwdep.a

Add to link options: -lcurl

Step 3:

Using python tool to generate contract C interface code from ABI and place the generated .c in your project (such that they could be compiled into your project).

Modify your application C code:

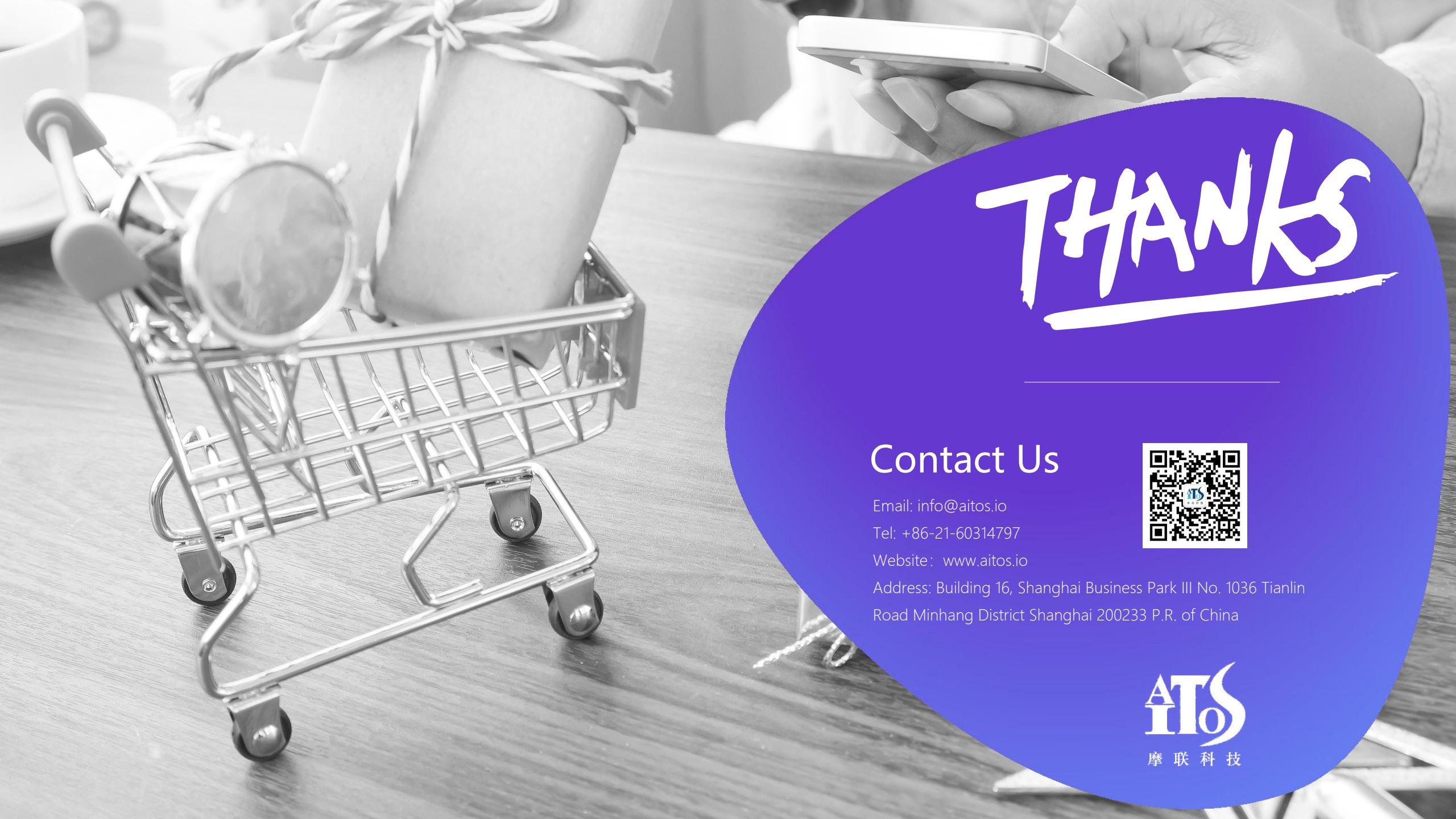
Add: #include "boatiotsdk.h"

Then follow instructions in “How to Call a Contract in Your C code”

BoAT IoT Framework SDK API

API	Description
BoatlotSdkInit	Initialize SDK
BoatlotSdkDelInit	De-initialize SDK
BoatWalletCreate	Create/Load a wallet
BoatWalletUnload	Unload a wallet
BoatWalletDelete	Delete a persistent wallet
BoatEthTxSend	Send a Ethereum transaction
BoatEthCallContractFunc	Call a Ethereum state-less contract function
BoatEthTransfer	Transfer value
BoatRandom	Generate random number
BoatHash	Compute hash
BoatSignature	Sign
...	...

See *BoAT IoT Framework SDK Reference Manual* for details.



THANKS

Contact Us

Email: info@aitos.io

Tel: +86-21-60314797

Website: www.aitos.io

Address: Building 16, Shanghai Business Park III No. 1036 Tianlin
Road Minhang District Shanghai 200233 P.R. of China

