AOSIO 개발 문서

아시다시피 AOS 는 EOSIO 를 기반으로 개발되었으며, 모든 EOSIO 문서와 사용은 AOS 에 적용됩니다. 개발자는 https://developers.eos.io/ 를 참조하여 더 많은 정보와 세부 사항을 참조할 수 있습니다.

1. 기초

1.1 AOSIO 설치

Aosio/aos 저장소의 소스코드 복제

```
git clone --recursive https://github.com/aosio/aos
```

```
cd aos
```

생성

```
./script/aosio_build.sh
```

설치

```
sudo ./script/aosio_install.sh
```

1.2 CDT 설치

Aosio.cdt 저장소의 소스코드 복제

```
git clone --recursive https://github.com/aosio/aosio.cdt
```

```
cd aosio.cdt
```

생성

```
./script/aosio_build.sh
```

설치

```
sudo ./script/aosio_install.sh
```

1.3 개발 지갑 생성

1.3.1 지갑 생성

```
claos wallet create --to-console
```

암호를 받게 되면 필요 시 이후 튜토리얼에서 사용할 수 있도록 잘 보관합니다.

1.3.2 지갑 열기

Kaosd 인스턴스 시작 시에는 지갑이 닫혀있습니다. 지갑을 오픈하려면 아래 명령을 실행하세요.

```
claos wallet open
```

아래 명령을 통해 지갑 리스트를 불러옵니다.

```
claos wallet list
```

1.3.3 지갑 잠금 해제

Kaosd 지갑이 열렸지만 여전히 잠금 상태입니다. 앞에서 받는 암호를 사용하여 잠금을 해제할 수 있습니다.

```
claos wallet unlock
```

시스템에 비밀번호를 입력하라는 메시지가 표시되면 귀하의 비밀번호를 이곳에 붙여넣기 한 후 enter 키를 눌러주세요.

1.3.4 지갑으로 암호 키 불러오기

시스템에서 일련의 개인 키를 생성합니다. Claos 에는 이를 위한 보조 기능이 있어, 다음 명령을 실행하면 됩니다.

```
claos wallet create_key
```

이후 새로운 개인 키와 개인 키에 상응하는 공개 키가 생성될 수 있습니다.

```
claos wallet import
```

시스템에 귀하의 개인 키 입력 메시지가 표시되면, 이전에 획득한 aosio 개발 키를 입력합니다.

1.4 kaosd 및 nodas 시작

1.4.1 kaosd 시작

```
kaosd &
```

1.4.2 nodaos 시작

```
nodaos -e -p aosio ₩
--plugin eosio::producer_plugin ₩
--plugin eosio::producer_api_plugin ₩
--plugin eosio::chain_api_plugin ₩
--plugin eosio::http_plugin ₩
--plugin eosio::history_plugin ₩
--plugin eosio::history_api_plugin ₩
--filter-on="*" ₩
--access-control-allow-origin='*' ₩
--contracts-console ₩
--http-validate-host=false ₩
--verbose-http-errors >> nodaos.log 2>&1 &
```

1.4.3 설치 상태 확인

Nodaos 의 블록 형성 여부를 확인합니다.

```
curl http://localhost:8888/v1/chain/get_info
```

높이 동일 여부를 확인할 수 있습니다.

1.5 테스트 계정 생성

Aosio 를 사용하여 alice, bob 이렇게 두 개의 테스트 계정을 만듭니다.

```
claos create account aosio alice
```

```
claos create account aosio bob
```

1.6 컨트랙트 컴파일

예를 들어 귀하의 컨트랙트 코드 문서가 hello.cpp 라면 귀하의 코드를 아래와 같이 wasm(WebAssembly) 텍스트 형식으로 컴파일해야 합니다.

```
eosio-cpp hello.cpp -o hello.wasm
```

1.7 컨트랙트 배치

예를 들어 귀하의 hello 컨트랙트를 alice 계정에 배치하고자 한다면 아래 작업을 수행해야 합니다.

```
claos set contract alice CONTRACTS_DIR/hello -p alice@active
```

1.8 컨트랙트 이행

claos push action alice hi '["bob"]' -p alice@active

1.9 인라인 작업 추가

인라인 작업 활성화를 위해 the aosio.code permission(코드 허가)를 컨트랙트 계정 활성화 권한에 추가해야 합니다.

```
claos set account permission alice active --add-code
```

2. 고급

EOSIO 관련 특징 외에, AOSIO 는 개발자가 다양하고 강력한 암호화 DAPP 을 개발하고 편리하게 사용할 수 있는 여러 종류의 intrinsic(인라인 함수)를 제공합니다.

2.1 정언_비밀번호_동일_증명

본 인라인 함수는 비밀번호 동일 증명을 제공하며 cipher_balance (비밀번호_잔액)와 이전에 제공된 eq_para(동일_파라미터) 및 공개 키 정보의 동일 여부를 확인할 수 있습니다.

포함

```
#include <eosio/crypto.hpp>
```

프로토 타입

```
void assert_cipher_equal_prove(const char* cipher_balance, size_t cb_len,
const char* eq_para, size_t eq_len, const char* pub_para, size_t pp_len);
```

파라미터

cipher_balance: cipher balance of original value

cb_len: length of cipher_balance

eq_para: equality parameter, cipher balance of value

eq_len: length of eq_para

pub_para: public key of sender

pp_len: length of pub_para

2.2 정언_비밀번호_유효_증명

본 인라인 함수는 비밀번호 유효 증명을 제공하여 계정 공개 키와 방탄 프로토콜을 검증할 수 있습니다.

포함

```
#include <eosio/crypto.hpp>
```

프로토 타입

```
void assert_cipher_valid_prove(const char* prove_para, size_t pv_len, const char* pub_para, size_t pp_len);
```

파라미터

```
prove_para: prove parameter
pv_len: length of prove_para
pub_para: public key parameter
pp_len: length of pub_para
```

2.3 정언_비밀번호_타원곡선 디지털 서명 알고리즘_서명

본 인라인 함수는 타원곡선 디지털 서명 알고리즘의 서명을 검증할 수 있습니다.

포함

```
#include <eosio/crypto.hpp>
```

프로토 타입

```
void assert_cipher_ecdsa_signature(const char* eq_para, size_t eq_len, const
char* from_pub_para, size_t fp_len, const char* to_pub_para, size_t tp_len,
const char* sig, size_t sig_len);
```

파라미터

eq_para: equality of prove paramter

eq_len: length of eq_para

from_pub_para: from account public key parameter

fp_len: length of from_pub_para

to_pub_para: to account public key parameter

tp_len: length of to_pub_para

sig: signature of the data

sig_len: length of sig

2.4 정언_비밀번호_암호화

본 인라인 함수는 검증 비밀번호 암호화 기능을 제공합니다. 암호문 및 암호화(공개 키, 데이터, 무작위) 일치 여부 확인에 사용할 수 있습니다.

포함

```
#include <eosio/crypto.hpp>
```

프로토 타입

void assert_cipher_encrypt(const char* random, size_t rand_len, unsigned int value, const char* pub_para, size_t pp_len, const char* ciphertext, size_t ct_len);

파라미터

random: random number

rand_len: length of random

value: a number that need to be encrypted

pub_para: public key

pp_len: length of pub_para

ciphertext:cipher text

ct_len: length of ciphertext

2.5 비밀번호 추가

본 인라인 함수는 비밀번호 추가 기능을 제공합니다.

포함

#include <eosio/crypto.hpp>

프로토 타입

eosio::cipher128 cipher_add(const char* cipher1, size_t len1, const char* cipher2, size_t len2);

파라미터

eosio::cipher128: a 128 char array

cipher1: first cipher string

len1: length of cipher1

cipher2: second cipher string

len2: length of cipher2

2.6 허가_암호키_취득

본 인라인 함수는 계정과 관련된 공개 키를 얻을 수 있습니다.

포함

#include <eosio/permission.hpp>

프로토 타입

int get_permission_key(name account, name permission, char* pub, size_t publen)

파라미터

account: account

permission: permission type of account, i.e, "active"or "owner"

pub: output char array

publen: length of pub

2.7 증명_방탄(bullet) 프로토콜

본 인라인 함수는 특정 값에 따라 방탄 프로토콜을 생성할 수 있습니다.

포함

#include <eosio/crypto.hpp>

프로토 타입

void prove_bulletproof(uint64_t value, const unsigned char* blind, size_t blindlen, unsigned char* proof, size_t plen)

파라미터

value: a unsigned int64

blind: a 32-byte unsigned char array, random number

blindlen: length of blind, should be 32

proof: output of bulletproof, 675-byte unsigned char array

plen: length of proof, should be 675

2.8 정언_검증_방탄 프로토콜

본 인라인 검증 협약의 경우, 검증 성공 시 통과되며 성공하지 못할 경우 Panic 오류가 발생합니다.

포함

#include <eosio/crypto.hpp>

프로토 타입

void assert_verify_bulletproof(const unsigned char* commit, size_t commitlen, const unsigned char* proof, size_t prooflen)

파라미터

commit: pedersen commit of a value, a 33-byte unsigned char array

commitlen: length of commit, should be 33

proof: a 675-byte unsigned char array

prooflen: length of proof

2.9 페더슨 약정

본 인라인 함수는 페더슨 약정을 생성할 수 있습니다.

포함

#include <eosio/crypto.hpp>

프로토 타입

void pedersen_commit(uint64_t value, const unsigned char* blind, size_t blindlen, unsigned char* commit, size_t commitlen)

파라미터

value: a unsigned int64

blind: a 32-byte unsigned char array, random number, should be same as prove_bulletproof's

blindlen: length of blind, should be 32

commit: output of pedersen commit, 33-byte unsigned char array

commitlen: length of commit, should be 33