Cryptography

Effective use of cryptography keeps your smart contract transactions secure on the Sui blockchain.

Move contracts in Sui support verifications for several on-chain signature schemes. Not all signatures supported in on-chain verification are supported as user signature verification. See Signatures for valid signature schemes for transaction authorization.

Go to Sui On-Chain Signatures Verification in Move .

A zero-knowledge proof is a method by which a party, known as the prover, can confirm the truthfulness of a claim without disclosing any information about the underlying data. For instance, it's possible for the prover to demonstrate they have solved a sudoku puzzle without showing the actual solution. Groth16 is one such proof you can use in your smart contracts.

Go to Groth16 .

A cryptographic hash function is a widely used cryptographic primitive that maps an arbitrary length input to a fixed length output, the hash value. The hash function is designed to be a one-way function, which means that it is infeasible to invert the function to find the input data from a given hash value, and to be collision resistant, which means that it is infeasible to find two different inputs that map to the same hash value. Use available hashing functions to provide security to your smart contracts.

Go to Hashing .

Use ECVRFs to generate a random number and provide proof that the number used a secret key for generation. The public key corresponding to the secret key verifies the proof, so you can use it as a random number generator that generates outputs that anyone can verify. Applications that need verifiable randomness on chain can also benefit from its use.

Go to ECVRF .

Sui allows you to mix and match key schemes in a single multisig account. For example, you can pick a single Ed25519 mnemonic-based key and two ECDSA secp256r1 keys to create a multisig account that always requires the Ed25519 key, but also one of the ECDSA secp256r1 keys to sign. You could use this structure for mobile secure enclave stored keys as two-factor authentication.

Go to Multisig .

Signature verification

Move contracts in Sui support verifications for several on-chain signature schemes. Not all signatures supported in on-chain verification are supported as user signature verification. See Signatures for valid signature schemes for transaction authorization.

Go to Sui On-Chain Signatures Verification in Move .

A zero-knowledge proof is a method by which a party, known as the prover, can confirm the truthfulness of a claim without disclosing any information about the underlying data. For instance, it's possible for the prover to demonstrate they have solved a sudoku puzzle without showing the actual solution. Groth16 is one such proof you can use in your smart contracts.

Go to Groth16 .

A cryptographic hash function is a widely used cryptographic primitive that maps an arbitrary length input to a fixed length output, the hash value. The hash function is designed to be a one-way function, which means that it is infeasible to invert the function to find the input data from a given hash value, and to be collision resistant, which means that it is infeasible to find two different inputs that map to the same hash value. Use available hashing functions to provide security to your smart contracts.

Go to Hashing .

Use ECVRFs to generate a random number and provide proof that the number used a secret key for generation. The public key corresponding to the secret key verifies the proof, so you can use it as a random number generator that generates outputs that anyone can verify. Applications that need verifiable randomness on chain can also benefit from its use.

Go to ECVRF .

Sui allows you to mix and match key schemes in a single multisig account. For example, you can pick a single Ed25519 mnemonic-based key and two ECDSA secp256r1 keys to create a multisig account that always requires the Ed25519 key, but also one of the ECDSA secp256r1 keys to sign. You could use this structure for mobile secure enclave stored keys as two-factor authentication.

Go to Multisig .

Groth16

A zero-knowledge proof is a method by which a party, known as the prover, can confirm the truthfulness of a claim without disclosing any information about the underlying data. For instance, it's possible for the prover to demonstrate they have solved a sudoku puzzle without showing the actual solution. Groth16 is one such proof you can use in your smart contracts.

Go to Groth16 .

A cryptographic hash function is a widely used cryptographic primitive that maps an arbitrary length input to a fixed length output, the hash value. The hash function is designed to be a one-way function, which means that it is infeasible to invert the function to find the input data from a given hash value, and to be collision resistant, which means that it is infeasible to find two different inputs that map to the same hash value. Use available hashing functions to provide security to your smart contracts.

Go to Hashing .

Use ECVRFs to generate a random number and provide proof that the number used a secret key for generation. The public key corresponding to the secret key verifies the proof, so you can use it as a random number generator that generates outputs that anyone can verify. Applications that need verifiable randomness on chain can also benefit from its use.

Go to ECVRF .

Sui allows you to mix and match key schemes in a single multisig account. For example, you can pick a single Ed25519 mnemonic-based key and two ECDSA secp256r1 keys to create a multisig account that always requires the Ed25519 key, but also one of the ECDSA secp256r1 keys to sign. You could use this structure for mobile secure enclave stored keys as two-factor authentication.

Go to Multisig .

Hashing

A cryptographic hash function is a widely used cryptographic primitive that maps an arbitrary length input to a fixed length output, the hash value. The hash function is designed to be a one-way function, which means that it is infeasible to invert the function to find the input data from a given hash value, and to be collision resistant, which means that it is infeasible to find two different inputs that map to the same hash value. Use available hashing functions to provide security to your smart contracts.

Go to Hashing .

Use ECVRFs to generate a random number and provide proof that the number used a secret key for generation. The public key corresponding to the secret key verifies the proof, so you can use it as a random number generator that generates outputs that anyone can verify. Applications that need verifiable randomness on chain can also benefit from its use.

Go to ECVRF .

Sui allows you to mix and match key schemes in a single multisig account. For example, you can pick a single Ed25519 mnemonic-based key and two ECDSA secp256r1 keys to create a multisig account that always requires the Ed25519 key, but also one of the ECDSA secp256r1 keys to sign. You could use this structure for mobile secure enclave stored keys as two-factor authentication.

Go to Multisig .

Elliptic Curve Verifiable Random Function (ECVRF)

Use ECVRFs to generate a random number and provide proof that the number used a secret key for generation. The public key corresponding to the secret key verifies the proof, so you can use it as a random number generator that generates outputs that anyone can verify. Applications that need verifiable randomness on chain can also benefit from its use.

Go to ECVRF .

Sui allows you to mix and match key schemes in a single multisig account. For example, you can pick a single Ed25519 mnemonic-based key and two ECDSA secp256r1 keys to create a multisig account that always requires the Ed25519 key, but also one of the ECDSA secp256r1 keys to sign. You could use this structure for mobile secure enclave stored keys as two-factor authentication.

Go to Multisig .

Multisig

Sui allows you to mix and match key schemes in a single multisig account. For example, you can pick a single Ed25519 mnemonic-based key and two ECDSA secp256r1 keys to create a multisig account that always requires the Ed25519 key, but also one of the ECDSA secp256r1 keys to sign. You could use this structure for mobile secure enclave stored keys as two-factor authentication.

Go to Multisig .

Related links