# Algorand Key Specification

September 6, 2019

**Abstract**

This document specifies the list of keys and their capabilities in Algorand.

## Contents



# 1 Overview

An algorand node uses two kind of cryptographic keys:

- *participation keys*, a set of keys used for authentication, i.e. identify an account. Algorand uses a two-level ephemeral signature scheme that ensures forward security, which will be detailed in next section.
- *selection credential*, key used for proving membership of selection.

A vote is valid only if both the participation keys and the selection credential are valid.

# 2 Participation Keys

## 2.1 Algorand's Two-level Ephemeral Signature Scheme for Authentication

An ephemeral subkey is a key pair that produces one-time signature for messages. It must be deleted after use to ensure forward security. Algorand's ephemeral

subkeys uses Ed25519 public-key signature system.

Algorand uses a two-level ephemeral signature scheme. Instead of signing voting messages directly, an algorand account uses her master key to sign an intermediate ephemeral sub-key. This intemediate ephemeral sub-key signs a batch of leaf level ephemeral sub-keys. Hence, each intermediate ephemeral sub-key is associate with a batch number (Batch), and each leaf ephemeral sub-key is associate with a batch number (of its parent key) and an offset (Offset, denotes its offset in current batch). A voting message is signed hierarchically: the master key $\rightarrow$ batch sub-key $\rightarrow$ leaf sub-key $\rightarrow$ voting message (more details in next sub-section: One-time Signature).

After usage, an Algorand node deletes the outdated ephemeral subkeys. Algorand allows users to set the number of rounds that a ephemeral sub-key can be used repeatedly, KeyDilution. For example, the default KeyDilution value of the current consensus protocol (V17) is $10,000$. An algorand account can change her KeyDilution via key registration transactions (see the ledger specification).

## 2.2 One-time Signature

OneTimeSignatureSubkeyBatchID identifies an intermediate level ephemeral sub-key of a batch. OneTimeSignatureSubkeyBatchID is signed by the master key. It has the following fields:

- *SubKey Public key* SubKeyPK, the public key of this sub-key.
- *Batch* Batch, batch number of this sub-key.

OneTimeSignatureSubkeyOffsetID identifies an leaf level ephemeral sub-key. OneTimeSignatureSubkeyOffsetID is signed with a batch sub-key. It has the following fields:

- *SubKey Public key* SubKeyPK, the public key of this sub-key.
- *Batch* Batch, batch number of this sub-key.
- *Offset* Offset, offset of this sub-key in current batch.

Finally, OneTimeSignature is a cryptographic signature used in voting messages between algorand users. It contains the following fields:

- *Signature* Sig, a signature of message under PK
- *Public Key* PK, the public key of the message signer, PK is part of a leaf level ephemeral subkey.
- *Old Style Signature* PKSigOld, **deprecated** field. It is still in the message only for compability reason.
- *Public Key 2* PK2, the public key of the current batch.
- *Public Key 1 Signature* PK1Sig, a signature of OneTimeSignatureSubkeyOffsetID under PK2.

- *Public Key 2 Signature* PK2Sig, a signature of OneTimeSignatureSubkeyBatchID under the master key.

# 3 Selection Credential

To check the validity of a voting message, its selection credential needs to be verfied. Algorand uses Verifiable Random Function (VRF) to generate selection credentials (more details in crypto specification).

More specifically, an unverified vote (unauthenticatedVote) has the following fields:

- *Row Vote* R, an inner struct contains Sender, Round, Period, Step, and Proposal.
- *Unverified Credential* Cred, unverified selection credential. Cred contains a single field Proof, which is a VRF proof.
- *Signature* Sig, one-time signature of the vote.

Once receiving an unverified vote (unauthenticatedVote) from the network, an Algorand node verifies its selection credential by checking the validity of the VRF Proof (in Cred), the committee membership parameters that it is conditioned on, and the voter's voting stake. If verified, the result of this verification is wrapped in a Credential struct, containing the following fields:

- *Unverifed Credential* UnauthenticatedCredential, the unverified credential from input.
- *Weight* Weight, the weight of the vote.
- *VRF Output* VrfOut, the cached output of VRF verification.
- *Domain Separation Enabled* DomainSeparationEnabled, Domain separation flag, now must be true by the protocol.
- *Hashable* Hashable, the original credential

And this verified credential is wrapped in a Vote struct with *Row Vote* (R), *Verifid Credential* (Credential), and *Signature* (Sig).