

Workgroup: Network Working Group  
Internet-Draft:  
draft-hallambaker-threshold-sigs  
Published: 9 March 2020  
Intended Status: Informational  
Expires: 10 September 2020  
Authors: P. M. Hallam-Baker  
ThresholdSecrets.com

## Threshold Signatures in Elliptic Curves

### Abstract

A Threshold signature scheme is described. The signatures created are computationally indistinguishable from those produced using the Ed25519 and Ed448 curves as specified in RFC8032 except in that they are non-deterministic. Threshold signatures are a form of digital signature whose creation requires two or more parties to interact but does not disclose the number or identities of the parties involved.

<https://mailarchive.ietf.org/arch/browse/cfrg/>Discussion of this draft should take place on the CFRG mailing list ([cfrg@irtf.org](mailto:cfrg@irtf.org)), which is archived at .

### Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 10 September 2020.

### Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

## **Table of Contents**

- [1. Introduction](#)
  - [1.1. Applications](#)
    - [1.1.1. HSM Binding](#)
    - [1.1.2. Code Signing](#)
    - [1.1.3. Signing by Redundant Services](#)
- [2. Definitions](#)
  - [2.1. Requirements Language](#)
  - [2.2. Defined Terms](#)
  - [2.3. Related Specifications](#)
  - [2.4. Implementation Status](#)
- [3. Principles](#)
  - [3.1. Direct shared threshold signature](#)
  - [3.2. Shamir shared threshold signature](#)
  - [3.3. Stateless computation of final share](#)
    - [3.3.1. Side channel resistance](#)
  - [3.4. Security Analysis](#)
    - [3.4.1. Calculation of r values](#)
    - [3.4.2. Replay Attack](#)
    - [3.4.3. Malicious Contribution Attack](#)
    - [3.4.4. Rogue Key Attack](#)
- [4. Ed2519 Signature](#)
- [5. Ed448 Signature](#)

- [6. Test Vectors](#)
  - [6.1. Direct Threshold Signature Ed25519](#)
  - [6.2. Direct Threshold Signature Ed448](#)
  - [6.3. Shamir Threshold Signature Ed25519](#)
  - [6.4. Shamir Threshold Signature Ed448](#)
- [7. Security Considerations](#)
  - [7.1. Rogue Key attack](#)
  - [7.2. Disclosure or reuse of the value r](#)
  - [7.3. Resource exhaustion attack](#)
  - [7.4. Signature Uniqueness](#)
- [8. IANA Considerations](#)
- [9. Acknowledgements](#)
- [10. Normative References](#)
- [11. Informative References](#)

## **1. Introduction**

Threshold encryption and key generation provide compelling advantages over single private key approaches because splitting the private key permits the use of that key to be divided between two or more roles.

All existing digital signatures allow the signer role to be divided between multiple parties by attaching multiple signatures to the signed document. This approach, known as multi-signatures, is distinguished from a threshold signature scheme in that the identity and roles of the individual signers is exposed. In a threshold signature scheme, the creation of a single signature requires the participation of multiple signers and the signature itself does not reveal the means by which it was constructed.

Rather than considering multi-signatures or threshold signatures to be inherently superior, it is more useful to regard both as two points on a continuum of choices:

**Multi-signatures** Multiple digital signatures on the same document.  
Multi-signatures are simple to create and provide the verifier

with more information but require the acceptance criteria to be specified independently of the signature itself. This requires that the application logic or PKI provide some means of describing the criteria to be applied.

**Multi-party key release** A single signature created using a single private key stored in an encrypted form whose use requires participation of multiple key decryption shares.

**Threshold signatures** A single signature created using multiple signature key shares. Signature creation may be subject to complex criteria such as requiring an  $(n,t)$  quorum of signers but these criteria are fixed at the time the signature is created

**Aggregate Signatures** A single signature created using multiple signature key shares such that validation of the aggregate signature serves to validate the participation of each of the individual signers.

This document builds on the approach described in [[draft-hallambaker-threshold](#)] to define a scheme that creates threshold signatures that are computationally indistinguishable from those produced according to the algorithm specified in [[RFC8032](#)]. The scheme does not support the creation of aggregate signatures.

The approach used is based on that developed in FROST [[Komlo](#)]. This document describes the signature scheme itself. The techniques used to generate keys are described separately in [[draft-hallambaker-threshold](#)].

As in the base document, we first describe signature generation for the case that  $n = t$  using 'direct' coefficients, that is the secret scalar is the sum of the secret shares. We then show how the scheme is modified using Shamir secret sharing [[Shamir79](#)] and Lagrange coefficients for the case that  $n > t$ .

## 1.1. Applications

Threshold signatures have application in any situation where it is desired to have finer grain control of signing operations without this control structure being visible to external applications. It is of particular interest in situations where legacy applications do not support multi-signatures.

### 1.1.1. HSM Binding

Hardware Security Modules (HSMs) prevent accidental disclosures of signature keys by binding private keys to a hardware device from which it cannot be extracted without substantial effort. This provides effective mitigation of the chief causes of key disclosure

but requires the signer to rely on the trustworthiness of a device that represents a black box they have no means of auditing.

Threshold signatures allow the signer to take advantage of the key binding control provided by an HSM without trusting it. The HSM only contributes one of the key shares used to create the signature. The other is provided by the application code (or possibly an additional HSM).

### **1.1.2. Code Signing**

Code signing is an important security control used to enable rapid detection of malware by demonstrating the source of authorized code distributions but places a critical reliance on the security of the signer's private key. Inadvertent disclosure of code signing keys is commonplace as they are typically stored in a form that allows them to be used in automatic build processes. Popular source code repositories are regularly scanned by attackers seeking to discover private signature keys and passwords embedded in scripts.

Threshold signatures allow the code signing operation to be divided between a developer key and an HSM held locally or by a signature service. The threshold shares required to create the signature can be mapped onto the process roles and personnel responsible for authorizing code release. This last concern might be of particular advantage in open source projects where the concentration of control embodied in a single code signing key has proved to be difficult to reconcile with community principles.

### **1.1.3. Signing by Redundant Services**

Redundancy is as desirable for trusted services as for any other service. But in the case that multiple hosts are tasked with compiling a data set and signing the result, there is a risk of different hosts obtaining a different view of the data set due to timing or other concerns. This presents the risk of the hosts signing inconsistent views of the data set.

Use of threshold signatures allows the criteria for agreeing on the data set to be signed to be mapped directly onto the requirement for creating a signature. So if there are three hosts and two must agree to create a signature, three signature shares are created and with a threshold of two.

## **2. Definitions**

This section presents the related specifications and standard, the terms that are used as terms of art within the documents and the terms used as requirements language.

## 2.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [\[RFC2119\]](#).

## 2.2. Defined Terms

See [\[draft-hallambaker-threshold\]](#).

## 2.3. Related Specifications

This document extends the approach described in [\[draft-hallambaker-threshold\]](#) to support threshold signatures. The deterministic mechanism described in specification [\[draft-hallambaker-mesh-udf\]](#) is used to generate the private keys used in the test vectors.

## 2.4. Implementation Status

The implementation status of the reference code base is described in the companion document [\[draft-hallambaker-mesh-developer\]](#).

## 3. Principles

The threshold signatures created according to the algorithms described in this document are compatible with but not identical to the signatures created according to the scheme described in [\[RFC8032\]](#). In particular:

- \*The signature verification algorithm is unchanged.

- \*The unanimous threshold scheme produces values of  $R$  and  $S$  that are deterministic but different from the values that would be obtained by using the aggregate private key to sign the same document.

- \*The deterministic quorate threshold scheme produces values of  $R$  and  $S$  that are deterministic for a given set of signers but will change for a different set of signers or if the aggregate private key was used to sign the same document.

- \*?The non-deterministic quorate threshold scheme produces values of  $R$  and  $S$  that will be different each time the document is signed.

Recall that a digital signature as specified by [\[RFC8032\]](#) consists of a pair of values  $S$ ,  $R$  calculated as follows:

$$R = r.B$$

$$S = r + k.s \bmod L$$

Where  $B$  is the base point of the elliptic curve.

$r$  is an unique, unpredictable integer value such that  $0 < r < L$

$k$  is the result of applying a message digest function determined by the curve (Ed25519, Ed448) to a set of parameters known to the verifier which include the values  $R$ ,  $A$  and  $PH(M)$ .

$A$  is the public key of the signer,  $A = s.B$

$PH(M)$  is the prehash function of the message value.

$s$  is the secret scalar value

$L$  is the order of the elliptic curve group.

To verify the signature, the verifier checks that:

$$S.B = R + k.A$$

This equality must hold for a valid signature since:

$$\begin{aligned} S.B &= (r + k.s).B \\ &= r.B + k.(s.B) \\ &= R + k.A \end{aligned}$$

The value  $r$  plays a critical role in the signature scheme as it serves to prevent disclosure of the secret scalar. If the value  $r$  is known,  $s$  can be calculated as  $s = (S-r).k^{-1} \bmod L$ . It is therefore essential that the value  $r$  be unguessable.

Furthermore, if the same value of  $r$  is used to sign two different documents, this results two signatures with the same value  $R$  and different values of  $k$  and  $S$ . Thus

$$S_1 = r + k_1.s \bmod L$$

$$S_2 = r + k_2.s \bmod L$$

$$s = (S_1 - S_2)(k_1 - k_2)^{-1} \bmod L$$

The method of constructing  $r$  **MUST** ensure that it is unique and unguessable.

### 3.1. Direct shared threshold signature

A threshold signature  $R, S$  is constructed by summing a set of signature contributions from two or more signers. For the case that the composite private key is the sum of the key shares ( $n = t$ ), each signer  $i$  provides a contribution as follows:

$$A_i = s_i.B$$

$$R_i = r_i.B$$

$$S_i = r_i + k.s_i \text{ mod } L$$

Where  $s_i$  and  $r_i$  are the secret scalar and unguessable value for the individual signer.

The contributions of signers  $\{1, 2, \dots, n\}$  are then combined as follows:

$$R = R_1 + R_2 + \dots + R_n$$

$$S = S_1 + S_2 + \dots + S_n$$

$$A = s.B$$

$$\text{Where } s = (s_1 + s_2 + \dots + s_n) \text{ mod } L$$

The threshold signature is verified in the same manner as before:

$$S.B = R + k.A$$

Substituting for  $S.B$  we get:

$$= (S_1 + S_2 + \dots + S_n).B$$

$$= S_1.B + S_2.B + \dots + S_n.B$$

$$= (r_1 + k.s_1).B + (r_2 + k.s_2).B + \dots + (r_n + k.s_n).B$$

$$= (r_1.B + k.s_1.B) + (r_2.B + k.s_2.B) + \dots + (r_n.B + k.s_n.B)$$

$$= (R_1 + k.A_1) + (R_2 + k.A_2) + \dots + (R_n + k.A_n)$$

Substituting for  $R + k.A$  we get:

$$= R_1 + R_2 + \dots + R_n + k.(A_1 + A_2 + \dots + A_n)$$

$$= R_1 + R_2 + \dots + R_n + k.A_1 + k.A_2 + \dots + k.A_n$$

$$= (R_1 + k.A_1) + (R_2 + k.A_2) + \dots + (R_n + k.A_n)$$



As expected, the operation of threshold signature makes use of the same approach as threshold key generation and threshold decryption as described in [[draft-hallambaker-threshold](#)]. As with threshold decryption it is not necessary for each key share holder to have a public key corresponding to their key share. All that is required is that the sum of the secret scalar values used in calculation of the signature modulo the group order be the value of the aggregate secret scalar corresponding to the aggregate secret key.

While verification of [[RFC8032](#)] signatures is unchanged, the use of threshold signatures requires a different approach to signing. In particular, the fact that the value  $k$  is bound to the value  $R$  means that the participants in the threshold signature scheme must agree on the value  $R$  before the value  $k$  can be calculated. Since  $k$  is required to calculate the signature contributions  $S_i$  can be calculated, it is thus necessary to calculate the values  $R_i$  and  $S_i$  in separate phases. The process of using a threshold signature to sign a document thus has the following stages orchestrated by a dealer as follows:

0. The dealer determines the values  $F$ ,  $C$  and  $PH(M)$  as specified in [[RFC8032](#)] and transmits them to the signers  $\{1, 2, \dots n\}$ .
1. Each signer generates a random value  $r_i$  such that  $1 \leq r_i \leq L$ , calculates the value  $R_i = r_i \cdot B$  and returns  $R_i$  to the dealer .
2. The dealer calculates the value  $R = R_1 + R_2 + \dots + R_n$  and transmits  $R$  and  $A$  to the signers  $\{1, 2, \dots n\}$ .
3. Each signer uses the supplied data to determine the value  $k$  and hence  $S_i = r_i + k \cdot s_i \bmod L$  and transmits it to the dealer .
4. The dealer calculates the value  $S = S_1 + S_2 + \dots + S_n$  and verifies that the resulting signature  $R, S$  verifies according to the mechanism specified in [[RFC8032](#)]. If the signature is correct, the dealer publishes it. Otherwise, the dealer **MAY** identify the signer(s) that provided incorrect contributions by verifying the values  $R_i$  and  $S_i$  for each.

For clarity, the dealer role is presented here as being implemented by a single party.

### 3.2. Shamir shared threshold signature

To construct a threshold signature using shares created using Shamir Secret Sharing, each private key value  $s_i$  is multiplied by the Lagrange coefficient  $l_i$  corresponding to the set of shares used to construct the signature:

$$A_i = s_i l_i \cdot B$$

$$R_i = r_i.B$$

$$S_i = ri + klisi \bmod L$$

It is convenient to combine the derivation of  $S_i$  for the additive and Shamir shared threshold signatures by introducing a key multiplier coefficient  $c_i$ :

$$S_i = ri + kcisi \bmod L$$

Where  $c_i = 1$  for the additive shared threshold signature

$c_i = l_i$  for the Shamir shared threshold signature

### 3.3. Stateless computation of final share

One of the chief drawbacks to the algorithm described above is that it requires signers to perform two steps with state carried over from the first to the second to avoid reuse of the value  $r_i$ . This raises particular concern for implementations such as signature services or HSMS where maintaining state imposes a significant cost.

Fortunately, it is possible to modify the algorithm so that the final signer does not need to maintain state between steps:

0. All the signers except the final signer  $F$  generate their value  $r_i$  and submit the corresponding value  $R_i$  to the dealer
1. Dealer calculates the value  $R - R_F$  and sends it to the final signer together with the all the other parameters required to calculate  $k$  and the final signer's key multiplier coefficient  $c_F$ .
2. The final signer generates its value  $r_F$
3. The final signer calculates the value  $R_F$  from which the values  $R$  and  $k$  can now be determined.
4. The final signer calculates its key share contribution  $S_F = rF + kcFsF \bmod L$ .
5. The final signer returns the values  $S_F$  and  $R$  to the dealer.
6. The dealer reports the value  $R$  to the other signers and continues the signature process as before.

While this approach to stateless computation of the signature contributions is limited to the final share, this is sufficient to cover the overwhelming majority of real-world applications where  $n = t = 2$ .

Note that the final signer **MAY** calculate its value  $r_F$  deterministically provided that the parameters  $R - R_F$  and  $c_F$  are used in its determination. Other signers **MUST NOT** use a deterministic means of generating their value  $r_i$  since the information known to them at the time this parameter is generated is not sufficient to fix the value of  $R$ .

### 3.3.1. Side channel resistance

The use of Kocher side channel resistance as described in [[draft-hallambaker-threshold](#)] entails randomly splitting the private key into two shares and performing the private key operation separately on each share to avoid repeated operations using the same private key value at the cost of performing each operation twice.

This additional overhead **MAY** be eliminated when threshold approaches are used by applying blinding factors whose sum is zero to each of the threshold shares.

For example, if generation of the threshold signature is divided between an application program A and an HSM B using the final share approach to avoid maintaining state in the HSM, we might generate a blinding factor thus:

0. A generates a random nonce  $n_A$  and sends it to B with the other parameters required to generate the signature.
1. B generates a random nonce  $n_B$
2. B calculates the blinding factor  $x$  by calculating  $H(n_A, n_B)$  *where  $H$  is a strong cryptographic digest function and converting the result to an integer in the range  $1 \times L$ .*
3. B calculates the signature parameters as before except that the threshold signature contribution is now  $S_B = r_B + k(c_B s_B + x) \bmod L$ .
4. B returns the nonce  $n_B$  to A with the other parameters.
5. A calculates the blinding factor  $x$  using the same approach as B
6. A calculates the signature parameters as before except that the threshold signature contribution is now  $S_A = r_A + k(c_A s_A - x) \bmod L$ .

This approach **MAY** be extended to the case that  $t > 2$  by substituting a Key Derivation Function (e.g. [[RFC5860](#)]) for the digest function.

### 3.4. Security Analysis

We consider a successful breach of the threshold signature scheme to be any attack that allows the attacker to create a valid signature for any message without the participation of the required threshold of signers.

Potential breaches include:

- \*Disclosure of the signature key or signature key share.
- \*Modification of signature data relating to message  $M$  to allow creation of a signature for message  $M'$ .
- \*Ability of one of the signers to choose the value of the aggregate public key.
- \*Access control attacks inducing a signer to create a signature contribution that was not properly authenticated or authorized.

We regard attacks on the access control channel to be out of scope for the threshold signature algorithm, though they are certainly a concern for any system in which a threshold signature algorithm is employed.

We do not consider the ability of a signer to cause creation of an invalid signature to represent a breach.

#### 3.4.1. Calculation of $r$ values

The method of constructing the values  $r_i$  **MUST** ensure that each is unique and unguessable both to external parties, the signers and the dealer. The deterministic method specified in [\[RFC8032\]](#) cannot be applied to generation of the values  $r_i$  as it allows the dealer to cause signers to reveal their key shares by requesting multiple signature contributions for the same message but with different values of  $k$ . In particular, requesting signature contributions for the same message:

With different Lagrange coefficients.

With a false value of  $R$

To avoid these attacks, the value  $r_i$  is generated using a secure random number generator. This approach requires the signer to ensure that values are never reused requiring that the signing API maintain state between the first and second rounds of the algorithm.

While there are many approaches to deterministic generation of  $r_i$  that appear to be sound, closer inspection has demonstrated these to be vulnerable to rogue key and rogue contribution attacks.

#### **3.4.2. Replay Attack**

The most serious concern in the implementation of any Schnorr type signature scheme is the need to ensure that the value  $r_i$  is never revealed to any other party and is never used to create signatures for two different values of  $k.s_i$ .

Ensuring this does not occur imposes significant design constraints as creating a correct signature contribution requires that the signer use the same value of  $r_i$  to construct its value or  $R_i$  and  $S_i$ .

For example, a HSM device may be required to perform multiple signature operations simultaneously. Since the storage capabilities of an HSM device are typically constrained, it is tempting to attempt to avoid the need to track the value of  $r_i$  within the device itself using an appropriately authenticated and encrypted opaque state token. Such mechanisms provide the HSM with the value of  $r_i$  but do not and cannot provide protection against a replay attack in which the same state token is presented with a request to sign different values of  $k$ .

#### **3.4.3. Malicious Contribution Attack**

In a malicious contribution attack, one or more parties present a signature contribution that does not meet the criteria  $R_i = r_i.B$  and  $S_i = r_i + ks_i$ .

Such an attack is not considered to be a breach as it merely causes the signature process to fail.

#### **3.4.4. Rogue Key Attack**

A threshold signature scheme that allows the participants to 'bring their own key' may be vulnerable to a rogue key attack in which a signer is able to select the value of the aggregate public signature key by selecting a malicious public signature key value.

The scheme described in this document is a threshold signature scheme and does not support this feature. Consequently, this attack is not relevant. It is described here for illustrative purposes only.

This particular attack only applies when the individual signers create their own signature shares. It is not a concern when the signature shares are created by splitting a master signature private key.

Consider the case where the aggregate public key signature is calculated from the sum of public signature key share values presented by the signers:

$$A = A_1 + A_2 + \dots + A_n$$

If the public key values are presented in turn, the last signer presenting their key share can force the selection of any value of  $A$  that they choose by selecting  $A_n = A_m - (A_1 + A_2 + \dots + A_{n-1})$

The attacker can thus gain control of the aggregate signature key by choosing  $A_m = s_m.B$  where  $s_m$  is a secret scalar known only to the attacker. But does so at the cost of not knowing the value  $s_n$  and so the signer cannot participate in the signature protocol.

This attack allows the attacker and the attacker alone to create signatures which are validated under the aggregate signature key.

The attack is a consequence of the mistaken assumption that a signature created under the signature key  $A_1 + A_2 + \dots + A_n$  provides evidence of the individual participation of the corresponding key holders without separate validation of the aggregate key.

Enabling the use of threshold signature techniques by ad-hoc groups of signers using their existing signature keys as signature key shares presents serious technical challenges that are outside the scope of this specification.

#### 4. Ed2519 Signature

The means by which threshold shares are created is described in [[draft-hallambaker-threshold](#)].

The dealer selects the signers who are to construct the signature. Each signer then computes the value  $R_i$ :

0. Randomly generate an integer  $r_i$  such that  $1 \leq r_i \leq L$ .
1. Compute the point  $R_i = r_i B$ . For efficiency, do this by first reducing  $r_i$  modulo  $L$ , the group order of  $B$ . Let the string  $R_i$  be the encoding of this point.
2. Transmit the value  $R_i$  to the dealer
3. At some later point, the dealer **MAY** complete the signature by returning the values  $F$ ,  $C$ ,  $A$  and  $R$  as specified in [[RFC8032](#)] together with the key multiplier coefficient  $c_i$ . The signers **MAY** then complete their signature contributions:

4. Compute  $\text{SHA512}(\text{dom2}(F, C) \parallel R \parallel A \parallel \text{PH}(M))$ , and interpret the 64-octet digest as a little-endian integer  $k$ .
5. Compute  $S_i = (r_i + kc_iS_i) \bmod L$ . For efficiency, again reduce  $k$  modulo  $L$  first.
6. Return the values  $R_i, S_i$  to the dealer .

The dealer then completes the signature by:

0. Computing the composite value  $S = S_1 + S_2 + \dots + S_n$
1. Verifying that the signature  $R, S$  is valid.
2. Publishing the signature.

## 5. Ed448 Signature

The means by which threshold shares are created is described in [[draft-hallambaker-threshold](#)].

The dealer selects the signers who are to construct the signature. Each signer then computes the value  $R_i$ :

0. Randomly generate an integer  $r_i$  such that  $1 \leq r_i \leq L$ .
1. Compute the point  $R_i = r_iB$ . For efficiency, do this by first reducing  $r_i$  modulo  $L$ , the group order of  $B$ . Let the string  $R_i$  be the encoding of this point.

Transmit the value  $R_i$  to the dealer

0. At some later point, the dealer **MAY** complete the signature by returning the values  $F, C, A$  and  $R$  as specified in [[RFC8032](#)] together with the key multiplier coefficient  $c_i$ . The signers **MAY** then complete the signature contributions:
1. Compute  $\text{SHAKE256}(\text{dom4}(F, C) \parallel R \parallel A \parallel \text{PH}(M), 114)$ , and interpret the 114-octet digest as a little-endian integer  $k$ .
2. Compute  $S_i = (r_i + kc_iS_i) \bmod L$ . For efficiency, again reduce  $k$  modulo  $L$  first.
3. Return the values  $R_i, S_i$  to the dealer.

The dealer then completes the signature by:

0. Computing the composite value  $S = S_1 + S_2 + \dots + S_n$
1. Verifying that the signature  $R, S$  is valid.

2. Publishing the signature.

## 6. Test Vectors

### 6.1. Direct Threshold Signature Ed25519

The signers are Alice and Bob's Threshold Signature Service 'Bob'.  
Each creates a key pair:

ED25519Alice's Key (ED25519)

UDF: ZAAA-GTSI-GXED-255X-XALI-CEXS-XKEY

Scalar: 56271244081186130980636545017945156580516101894352492  
459594967614223399428880

Encoded Private

33 40 0E 22 D8 67 17 F4 8A 9F 6A 46 61 B4 0E AD

8C D0 DD C3 79 CD 85 BD 95 5C 90 B9 6C CB 8C 23

X: 11116793672970427161790264469280294507189044728140547954071022  
7976454124042406369344932655633664630560242213431409139866940  
284702002648469365756492647970

Y: 61655404171611396573021808119108664749574235125343680206454285  
6299141386615046548323087409388548650272224487089895079970526  
0143544115364878870129761259200

Encoded Public

E2 AB 8F 37 62 C8 7B F9 E9 BC 59 0C 2E 99 A5 58

0C C3 19 D5 CD DA 53 DF 3E C1 F0 C0 FE D3 55 5E

ED25519Bob's Key (ED25519)

UDF: ZAAA-GTSI-G2ED-255X-XBOB-XS XK-EY

Scalar: 54940772670153459146152925564198105262971485730889818  
986727608573229799020168

Encoded Private

68 9A 68 92 8A 06 17 84 35 3C B7 08 F8 56 00 3F

BA 31 8C 42 B0 42 FE 2D 18 F2 7F AB CD 10 49 F1

X: 14271495069349838216379540196263140964032393512903842206168182  
5518850827098876289800868735522232908519794251130907125878675  
6343411484065706313568410880015

Y: 28094328948004112428189466223757440886388684291254605355859923  
6240968229706795825282419594219442074647093851302547452470435  
9438513477629978601366725015573

Encoded Public

32 E5 8D 5E 66 B2 F9 E9 14 79 08 71 96 3B 9A 75

A2 31 59 4B 8E ED 18 EF BD FF 11 D4 47 2A 8C F4

The composite Signature Key  $A = A_a + A_b$



Aggregate Key = Alice + Bob ( )

UDF: TBS

Scalar: 26569330913556569171916721364983482306308422345436973  
56293312113171384684213

Encoded Private

B5 CE 0E B3 9C CF 18 99 CF 8D 4C BB AE 81 79 1F  
CE 13 AA 3E 63 59 5B AC 8D 2C EB A4 55 C5 DF 05

X: 67872685043898469012456949773240814121645904736114813455820339  
8688906486811443744733724675994181258029547346985079901494367  
752381127781166234556148580090

Y: 36481740058369645484420180976004932062085375941522344052907594  
0118552792158551197107484892204562290802810655253510302448455  
4128548992118101415797909250954

Encoded Public

29 65 63 86 4F FB 10 8D BA 7A 0A 68 04 6D 00 DA  
9B 1D C3 A4 AF BA 95 B4 5D 27 B4 35 00 2F DF 32

To sign the text "This is a test", Alice first generates her value  $r$  and multiplies it by the base point to obtain the value  $R_a$ :

Alice:

$r_a =$  47498736868184230171598682949552850028048129926454476046383202  
22101432831360

$R_a =$

37 4D BC B2 FC 26 43 69 DA B7 DF 78 F7 05 63 1D  
C4 96 FB FB D6 CC 6A 9E 92 ED E7 35 93 44 B8 14

Alice passes her value  $R_a$  to Bob along with the other parameters required to calculate  $i$ . Bob then calculates his value  $R_a$  and multiplies it by the base point to obtain the value  $R_b$ :

Bob:

$r_b =$  52485056662860498129500187267044856221580824276836907703860095  
4226327269184

$R_b =$

AB A1 A8 28 11 6E FC A0 E9 8B CF 53 A2 4C AF F8  
66 59 69 4B 2F 37 93 77 92 3A 70 95 33 DE 5B 54

Bob can now calculate the composite value  $R = R_a + R_b$  and thus the value  $k$ .

```
R =
  5B 68 76 8C  CA 23 E6 84  36 92 76 F1  9E FF 08 8F
  0A 16 A9 55  E3 96 9C 84  36 28 89 DB  06 16 43 44
k = 2386567530061545624821866403866306492890802298065266303263411446
    38069188344
```

Bob calculates his signature scalar contribution and returns the value to Alice:

```
Bob:
S_b = 28944981827334439860137457106479900995796126061421325490732808
      17601718529025
```

Alice can now calculate her signature scalar contribution and thus the signature scalar S.

```
Alice:
S_a = 42757354096062105044904298703340896803929542552166792901005683
      43184517117786
S = 7170233592339654490504175580982079779972566861358811839173849160
    786235646811
```

Alice checks to see that the signature verifies:

```
S.B = R + kA =
X: 40361643976948744971992641065784582623361456745859092512349846
   75625733276535
Y: 41192378513331957483774498173296750751876504986403840969777850
   360997558579289
```

## 6.2. Direct Threshold Signature Ed448

The signers are Alice and Bob's Threshold Signature Service 'Bob'. Each creates a key pair:

ED448Alice's Key (ED448)

UDF: ZAAA-ITSI-GXED-44XA-LICE-XSXK-EY

Scalar: 63495803583658817688110446314786076976347236361354035  
5597788771064742993095132758589292255654895141583596922516472  
738879360490167934280

Encoded Private

A0 53 4C 93 3C 34 00 76 AE 5D B5 4A C2 71 5F 43  
E1 D6 63 2C 5C 56 53 C8 98 A0 8F 03 FF F5 22 96  
91 45 8C 2B CF E3 FD 7E 2A 9E 0B D6 F4 CC 66 61  
43 62 72 7B 34 B4 79 92

X: 24743197509267833262111449556527285120868167712209919570838426  
3466168536901525943558973091346360088759980994772668117646359  
614426660579

Y: 21342899120576770537664462049685258390853729788303428349051130  
8752175233505795318243164692156369495328007220135137156078814  
081547431302

Encoded Public

0A 3B F3 27 E7 E1 67 63 2C 59 E2 1C D1 84 C7 83  
E8 1E D1 68 9F 32 A1 16 99 00 5C DA 29 B9 6C 08  
E4 15 57 7E E5 63 C2 32 08 23 41 68 5F 49 1F FF  
BC 4D CD 3A 4E A6 85 49 00

ED448Bob's Key (ED448)

UDF: ZAAA-ITSI-G2ED-44XB-OBXS-XKEY

Scalar: 72649803773199751564998543891898904839718409312910780  
0262041941160989643727331987658132182181970054245587322070535  
846720571414845714224

Encoded Private

BC 53 B4 74 3E A7 A7 FA 9F 05 9A BC 8C 22 26 15  
A1 4E BB 10 0E B5 59 6B DE 9C 1B E9 F2 3C 65 42  
E7 B4 47 18 60 AC 18 A6 D2 78 B8 BC CE F5 F4 28  
B2 3A FF 08 61 EF 6B 7C

X: 58235851934808640621920816872959059172692411187640950432203039  
8116748997750134460231406698091317008063030408798536634284207  
667468558264

Y: 34390767697909283892495761259186538632120422458392131201372282  
6056455656591826216381185634080685718154852726725624178995827  
091591132128

Encoded Public

93 63 5A 45 2D 4C 94 32 45 23 CD E2 A8 46 E4 78  
A0 80 59 DA 36 CB 6B 0C 06 64 6F BE 51 AB C0 BF  
1E DB A8 3F 2B 3B 80 0F BF 00 E6 78 DD E0 83 E9  
AC 20 02 55 87 07 39 38 00

The composite Signature Key  $A = A_a + A_b$

Aggregate Key = Alice + Bob ( )

UDF: TBS

Scalar: 89488306051273634069773238262841883041784075539841550  
3672228636597106090916876462340541507950185640860121886233669  
49466515613996100051

Encoded Private

D3 29 DD AB F6 0D 99 8B 75 65 B8 06 36 C9 3A 2C  
D4 08 C3 9B 7C F9 77 8C 68 29 0E 3D 5D C7 3E 00  
92 8B DC AE 26 FB 16 39 CD 25 1B 23 4A 5A 05 61  
1D 5C C4 70 0A C9 84 1F

X: 17985659098670117617173315763082238685735647626871251468000984  
2080317111091696183607307614171726960576308774975742249260532  
199160570999

Y: 31506323224859159594386181995639405170623657273945727288760063  
1624406694682617334725040181287905351066763414658543828623841  
509161975864

Encoded Public

9B 3E DF 49 55 40 9F 7B EA 0B AA 40 B7 3D 15 82  
60 9F 7C 40 CF 67 DE 56 56 0D 03 87 63 3B 15 F2  
45 33 FE 48 BD 2D A0 A2 8B CC 74 DA 94 0F 39 00  
AC 39 CB 0A 9F A4 EB B0 00

To sign the text "This is a test", Alice first generates her value  $r$  and multiplies it by the base point to obtain the value  $R_A$ :

Alice:

$r_a$  = 68686103432614286085087961100697733989779035076023642515818127  
98858367044290553184740969706908768491109520568381137474912463623  
0094998

$R_a$  =

84 D1 68 50 1E BB 9D 40 17 52 C5 D2 96 41 DE FC  
22 D9 04 02 82 27 1B 32 CD 1D 8C 53 85 AB 40 79  
F5 00 C2 4C 89 A6 71 0A 39 08 A6 98 4C 2A FE FD  
B9 F7 66 FC E9 F2 20 B9 80

Alice passes her value  $R_A$  to Bob along with the other parameters required to calculate  $i$ . Bob then calculates his value  $R_A$  and multiplies it by the base point to obtain the value  $R_b$ :

Bob:

```
r_b = 76096557183484521564781984661044762292570684233728343619181213
      42816465520852868086783053490696058232580309131924136177891398775
      9613091
```

R\_b =

```
F3 60 25 C5 13 CE AA A9 B2 7E 9D 05 5B B1 F5 6B
4F 5B A5 EC 49 3F 39 A4 B5 C3 3C 7F 80 17 44 5B
02 83 FD 12 06 A2 DF E8 C6 46 1D 2A 6C 8E 75 70
02 53 71 3F 95 C2 71 A8 00
```

Bob can now calculate the composite value  $R = R_a + R_b$  and thus the value  $k$ .

R =

```
25 F2 BA FD D1 BB 3F 38 7B 4F 26 63 47 9A 78 81
4F CB 1F 82 8D F9 84 D4 34 96 E1 5A 4A 52 46 2C
13 EF 67 37 E0 61 13 9F B2 1F 7E C5 B9 56 7F B6
CA 88 D7 0B CC BC 96 C5 00
```

```
k = 1280456290623708148598382410642762981433750669722496871781111755
     77910741520269875312421999840786826452062625868870120134096187323
     572465
```

Bob calculates his signature scalar contribution and returns the value to Alice:

Bob:

```
S_b = 13468720239873188337021798218509333541110080817756031277945581
      40664081463680336638329881442786750747115016847691428033880365372
      48950911
```

Alice can now calculate her signature scalar contribution and thus the signature scalar  $S$ .

Alice:

```
S_a = 11878333834477344997625309702086496132606331016796170754366862
      41133668758758146277756495968716749041239572075324760597254347275
      98536495
S = 7176085966960361070914012723395716314875377817369250525275188838
    46290182823087058924419858586576028721485985925272008086975089878
    37627
```

Alice checks to see that the signature verifies:

$S.B = R + kA =$

X: 16291004486439090147776612791895755846785064919509905636229908  
328194692798396  
Y: 46841527536088411435030533441004631079380726947860208328057908  
29635735834402

### 6.3. Shamir Threshold Signature Ed25519

The administrator creates the composite key pair

ED25519Aggregate Key (ED25519)

UDF: ZAAA-GTSI-GQED-255X-XAGG-REGA-TEXK-EY

Scalar: 39348647608109113656999806950437958090469802387424444  
589375066079861075223816

Encoded Private

37 39 5E 7A 8B A5 A0 19 46 4B 58 22 EA 24 A5 71  
45 2C 2A AC 7A 3E FB CA CE 3F D4 12 9A BA EB 70

X: 14198837758377867455716504277518729070915183249890461230792115  
9904969716778427995951234766002164511738587575257530388758374  
7824906047250057721855068523970

Y: 20211025649802071998810413948266748565975140520947927724517956  
2067625505077751598018629551746824533726709810990193455662385  
6152736116303441031851305458040

Encoded Public

6E 13 79 B4 39 DA 97 9C 5A 34 CE 79 CD 1B 50 DF  
A0 76 AD 49 81 6D 52 59 A4 2C DB CE 44 FF 3E F5

Three key shares are required for Alice, Bob and Carol with a threshold of two. The parameters of the Shamir Secret Sharing polynomial are:

$a_0 = 3934864760810911365699980695043795809046980238742444458937506607$   
 $9861075223816$

$a_1 = 4447543804183276703367140352869026105461853047473514285943821414$   
 $241349264676$

The key share values for the participants are

xa = 1  
ya = 3741579482988170765278279250490187507889572786185132393071818643  
89698982558

xb = 2  
yb = 4821701752482093779894968277918044856250810326092027525251003278  
631048247234

xc = 3  
yc = 2032239979333108269288922067744076720855547014185634205192873754  
586943260921

Alice and Carol are selected to sign the message "This is another test"

The Lagrange coefficients are:

la = 3618502788666131106986593281521497120428558179689953803000975469  
142727125496  
lc = 3618502788666131106986593281521497120428558179689953803000975469  
142727125494

Alice and Carol select their values ra, rc

ra = 5377248352669516780549162073457874087773506582785629831437934867  
461127075879

Ra =  
2F 0E 76 87 F3 98 84 41 03 29 5F 5A 84 7D 62 34  
A3 9B 64 8C 34 11 99 F1 D5 A9 63 0E 8E 3D 6C C9

rc = 4232465617601402201287633784079511069602658258584839658166776648  
89765437047

Rc =  
53 7C 5B 73 DE 10 64 82 3C CF CF 19 5F E8 83 24  
A0 14 A3 F2 FF 7E 4D 0D 42 42 8E B3 03 66 49 04

The composite value  $R = R_a + R_c$

R =  
5E BA 21 F2 87 4E C8 4E B8 4B E9 5C 1E 3A B2 67  
B8 D0 E3 B3 98 C8 DB E0 E6 50 35 8D 47 9B C1 E1

The value  $k$  is

$k = 31647328596154272207177743411362878636270581278391351236822318404$   
 $2871610589$

The values  $R$  and  $k$  (or the document to be signed) and the Lagrange coefficients are passed to Alice and Carol who use them to calculate their secret scalar values:

$s_a = 5612369224482256147917418875735281261834359179277698589607727965$   
 $84548473837;$   
 $s_c = 2602382798999576972342132247649458760000784672597136700404538591$   
 $849255495034$

The signature contributions can now be calculated:

$S_a = 6873614337238287985974631938614575791975512646788133685349814447$   
 $956390424246$   
 $S_c = 6734965213168676845758127805874268408589146808656626354387236455$   
 $007092488600$

The dealer calculates the composite value  $S = S_a + S_b$

$S = 6371573973074702617759573181445849959707543096064852433735099964$   
 $678028661857$

The dealer checks to see that the signature verifies:

$S.B = R + kA =$   
 $X: 50976148565709102066754003035356376378566475226930883450768310$   
 $75625019906058$   
 $Y: 74629887867177957343114045996705864971381351024041289421281109$   
 $60605559633693$

#### 6.4. Shamir Threshold Signature Ed448

The administrator creates the composite key pair



ED448Aggregate Key (ED448)

UDF: ZAAA-ITSI-GQED-44XA-GGRE-GATE-XKEY

Scalar: 50890460656419721531273587958284096015810982760541575  
4207268050539683337837216003977228732536078674802149039736292  
653681850024283019712

Encoded Private

78 22 7E 3B 89 95 80 5D 04 19 DC 27 F1 7F 9B E4  
86 2B 0B DD 55 64 EE 04 19 49 4D DE B9 04 3B 9E  
8B 7D DC EC EC 8F DD 1D E7 88 86 FD 11 FD 78 EF  
1A 8B 84 8F 77 00 73 65

X: 44109173355278142669484438370724914685176368933547176239809629  
7503768465595321590690311221269514682222687386378631457535068  
446135118173

Y: 53219402718535721212460981200104434180077825188675868294070079  
5084662920552823356888138706016038637934794839496624474125511  
419755284720

Encoded Public

43 61 20 A0 B1 DF AA BD 6B 55 00 97 A3 BE CB B8  
09 57 20 88 16 69 E4 B9 E1 7E 9C 13 C0 41 5B CB  
4D 3E E4 99 2E 2D 48 89 1C C0 FB 26 58 C2 DD 5C  
C1 DC 17 82 D7 A0 43 EE 80

Three key shares are required for Alice, Bob and Carol with a threshold of two. The parameters of the Shamir Secret Sharing polynomial are:

a0 = 5089046065641972153127358795828409601581098276054157542072680505  
39683337837216003977228732536078674802149039736292653681850024283  
019712

a1 = 3934291191212650584059669841203896874838912563882890929786754329  
01065050103930671683138406271792350963735688973370364434513108789  
92904

The key share values for the participants are

xa = 1  
ya = 3118475254618553241339722078876528141267932728756118294017944444  
35183096299031399695530728818078200859172750635470321098006758306  
3279

xb = 2  
yb = 4246138716674505908193642049091549688965705836758502759188548773  
44583359733833811652691479153600171049652964036917396544313784620  
56183

xc = 3  
yc = 8180429907887156492253311890295446563804618400641393688975303102  
45648409837764483335829885425392522013388653010287760978826893410  
49087

Alice and Carol are selected to sign the message "This is another  
test"

The Lagrange coefficients are:

la = 9085484053695086131866547598600056679420517008591475753518627489  
75730019807697928580978776458461879816551468545458311523868779298  
24891  
lc = 9085484053695086131866547598600056679420517008591475753518627489  
75730019807697928580978776458461879816551468545458311523868779298  
24889

Alice and Carol select their values ra, rc

ra = 1610705867304076392921723072340177274788014288395871190735371844  
45304821479265411365593663429368194644596000340187951474891649791  
38278

Ra =  
CF 97 83 F4 16 2D EF A1 85 95 8F 65 68 BC 59 56  
94 9F 51 61 A7 F9 95 27 81 E6 83 7F 0A E6 16 C0  
50 8D 8A D5 B6 47 3C 64 E0 34 1B AB F3 BF 89 96  
41 5D A8 29 06 04 F9 D3 00

rc = 1513721922038322665004883340036457345550315134561588659560214983  
90578815745979201541151452164958584639864068264575123975657731927  
908051

Rc =  
7A 5A E9 4E 9B E9 9E 66 C1 E2 CE 3A B5 CD 73 FA  
4C C7 9E E9 C3 59 37 46 11 55 62 6A DC 69 8D 7F  
63 20 CF B0 30 6D 77 5E B4 5F 09 B3 8A ED C6 36  
64 8F 6B 70 02 DE AB 45 00

The composite value  $R = R_a + R_c$

R =

```
D0 51 EC 22 5C 8A 25 9E E7 B6 0B 1E 26 54 0F 51
4C 65 1C B5 24 B1 89 91 FA 63 32 39 89 89 24 9B
03 3E E6 83 1A 74 FA 79 0D E5 90 D7 C2 7C 3B FF
22 D5 B2 7F B7 14 C8 F0 80
```

The value k is

```
k = 11205778835971090763459400118127617086725238194171737531994098751
92020469089993225609677938158558606382862723600565685005808743888
1311
```

The values R and k (or the document to be signed) and the Lagrange coefficients are passed to Alice and Carol who use them to calculate their secret scalar values:

```
sa = 9553255341887869118067505910431535900610706917904893497621319156
41007484252552638535308385781173609945427381140778859688569793044
19808;
sc = 4995269099751507885739891653452333397518207808270778909030975938
52905814888815686913063833745765618809857142040314431034455332593
00346
```

The signature contributions can now be calculated:

```
Sa = 1288724802012784524546710951030879127642885816514372129177355341
52388852607084448690299630591486640562077067211721300969711457389
734384
Sc = 9245427182572705582371845728508254342708120847149751739704864321
05098990035212645834079206651328626851538428963929848345208375317
71245
```

The dealer calculates the composite value  $S = S_a + S_b$

```
S = 3961707095310378564105860041616932260295944995110521524441162756
77527476490661275575117959649271272839206163990226234994585390618
55850
```

The dealer checks to see that the signature verifies:

$S.B = R + kA =$

X: 89820587157297640914878146420868301420919058721019003413415111  
40349053172466

Y: 29679650828270530327946891692133348611591938177520550739646498  
568569373309365

## 7. Security Considerations

All the security considerations of [[RFC7748](#)], [[RFC8032](#)] and [[draft-hallambaker-threshold](#)] apply and are hereby incorporated by reference.

### 7.1. Rogue Key attack

The rogue key attack described in [[draft-hallambaker-threshold](#)] is of particular concern to generation of threshold signatures.

If  $A$  and  $B$  are public keys, the intrinsic degree of trust in the composite keypair  $A + B$  is that of the lesser of  $A$  and  $B$ .

### 7.2. Disclosure or reuse of the value $r$

As in any Schnorr signature scheme, compromise of the value  $r$  results in compromise of the private key. The base signature specification [[RFC8032](#)] describes a deterministic construction of  $r$  that ensures confidentiality and uniqueness for a given value of  $k$ .

As described above, this approach is not applicable to the generation of values of  $r_i$  to compute threshold signature contributions. Accordingly the requirements of [[RFC4086](#)] regarding requirements for randomness **MUST** be observed.

Implementations **MUST NOT** use a deterministic generation of the value  $r_i$  for any threshold contribution except for calculating the final contribution when all the other parameters required to calculate  $k$  are known.

### 7.3. Resource exhaustion attack

Implementation of the general two stage signing algorithm requires that signers track generation and use of the values  $r_i$  to avoid reuse for different values of  $R_i$ . Implementations **MUST** ensure that exhaustion of this resource by one party does not cause other parties to be denied service.

## 7.4. Signature Uniqueness

Signatures generated in strict conformance with [RFC8032] are guaranteed to be unique such that signing the same document with the same key will always result in the same signature value.

The signature modes described in this document are computationally indistinguishable from those created in accordance with [RFC8032] but are not unique.

Implementations **MUST** not use threshold signatures in applications where signature values are used in place of cryptographic digests as unique content identifiers.

## 8. IANA Considerations

This document requires no IANA actions.

## 9. Acknowledgements

[TBS]

## 10. Normative References

### [draft-hallambaker-mesh-udf]

Hallam-Baker, P., "Mathematical Mesh 3.0 Part II: Uniform Data Fingerprint.", Work in Progress, Internet-Draft, draft-hallambaker-mesh-udf-08, 6 January 2020, <<https://tools.ietf.org/html/draft-hallambaker-mesh-udf-08>>.

### [draft-hallambaker-threshold]

Hallam-Baker, P., "Threshold Key Generation and Decryption in Ed25519 and Ed448", Work in Progress, Internet-Draft, draft-hallambaker-threshold-00, 5 January 2020, <<https://tools.ietf.org/html/draft-hallambaker-threshold-00>>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/rfc/rfc2119>>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/rfc/rfc4086>>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <<https://www.rfc-editor.org/rfc/rfc7748>>.

**[RFC8032]**

Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <<https://www.rfc-editor.org/rfc/rfc8032>>.

**11. Informative References**

**[draft-hallambaker-mesh-developer]**

Hallam-Baker, P., "Mathematical Mesh: Reference Implementation", Work in Progress, Internet-Draft, draft-hallambaker-mesh-developer-09, 23 October 2019, <<https://tools.ietf.org/html/draft-hallambaker-mesh-developer-09>>.

**[Komlo]**

Komlo, C. and I. Goldberg, "FROST: Flexible Round-Optimized Schnorr Threshold Signatures", 2020.

**[RFC5860]**

Vigoureux, M., Ward, D., and M. Betts, "Requirements for Operations, Administration, and Maintenance (OAM) in MPLS Transport Networks", RFC 5860, DOI 10.17487/RFC5860, May 2010, <<https://www.rfc-editor.org/rfc/rfc5860>>.

**[Shamir79]**

Shamir, A., "How to share a secret.", 1979.