One Round Threshold ECDSA without Roll-Call

ALEXANDRE BOUEZ
Alexandre.Bouez@irt-systemx.fr

KALPANA SINGH
Kalpana.Singh@irt-systemx.fr

Contents

I. Prerequisites

1. Participants

We propose a protocol for threshold ECDSA using Shamir's (t, n) secret sharing.

- A group of n shareholders (players) $\mathrm{P}_1, \dots, \mathrm{P}_n$ each old an equivalent share of the secret key. Any $(t+1)$ subset of this group should be able to issue a signature, given that the pre-signing phase was correctly performed in advance.
- The subset of players participating in the generation of the signature (signatories) is $S \subset [1..n]$, $|S| = t + 1$.

We make some assumptions about the players:

- $\forall i \in [1..n]$, every player P_i has a secret key sK_i and the associated public key pK_i is at least known by all other players.
- Players have knowledge of:
  - The $\{G, g\}$ cyclic group used by the ECDSA signature scheme.
  - An additively homomorphic scheme E used hereafter.

2. Key generation

The participants share a **unique secret key** in the form of a (t, n)-Shamir secret x such that:

- $\forall i \in [1..n]$, P_i holds the share x_i.
- The values g^{x_i} is public and there exists a **unique public key** $y = g^x$ associated to the secret key x.
- It can be transformed into a (t, t+1)-additive secret for the subset $S \subset [1..n]$ with $|S| = t + 1$ by using the appropriate $\lambda_{i,S}$ Lagrangian coefficient such that:

$$w_i = (\lambda_{i,S})(x_i) \quad \text{and} \quad x = \sum_{i \in S} w_i$$

Protocol: TODO

3. Multiplicative-to-Additive (MtA) share conversion

We propose a basic implementation of an MtA share conversion protocol. This is not a new implementation and is based off of preexisting work [3, 4]. The following protocol includes options verification steps (zero-knowledge proofs).

Let us consider two additive secrets

$$u = \sum_{i \in \mathrm{I}} u_i \text{ and } v = \sum_{i \in \mathrm{I}} v_i$$

$$\text{where } \mathrm{I} = [1..n], n \in Z_q.$$

Given $(i, j) \in [1..n]^2$ and $i \neq j$, players P_i and P_j execute the MtA protocol for their respective shares of x and y:

- TODO

II. Protocol with late Roll-Call

The following protocol allows for a pre-signing that is independent from the subset of signatories S. This subset must however be known before the signing of the message.

1. Pre-signing

Pre-signing is comprised of multiple protocols requiring different number of participants. The first protocol must be completed before any other, the last two are independent and can be completed in any order/simultaneously.

Players can engage in **multiple simultaneous pre-signing phases** in order to **significantly decrease the number of interactions** across multiple signatures.

a. Signing secret generation

The signing secret k is generated between all players much the same way as the private key x (see I.2), but is different for every signature and is therefore part of pre-signing.

It takes the form of a (t, n)-Shamir secret k such that:

- $\forall i \in [1..n], \mathrm{P}_i$ holds the share k_i.
- The value g^{k_i} is public.
- It can be transformed into a (t, t+1)-additive secret for the subset $S \subset [1..n]$ with $|S| = t+1$ by using the appropriate $\lambda_{i,S}$ Lagrangian coefficient such that:

$$l_i = (\lambda_{i,S})(k_i) \quad \text{and} \quad k = \sum_{i \in S} l_i$$

The value u should **never be compiled**, the shares of u should **never be published**. This value should be changed for every signature. Revealing it once the signature was compiled would reveal the secret key x.

b. MtA between all players

Every pair of players $(\mathrm{P}_i, \mathrm{P}_j)$ performs an MtA protocol (see I.3) with shares k_i and x_i. Their respective resulting additive shares are $a_{i\to j}$ and $b_{i\to j}$.[1]

- **Step 0.** Player P_i broadcasts:

$$E_{pK_i}(x_i)$$

  - Player P_j chooses a random nonce $u_{i\to j}$. His additive share is $b_{i\to j} = -u_{i\to j}$.

- **Step 1.** P_j sends to P_i:

$$E_{pK_i}(x_i k_j + u_{i\to j})$$

  - Player P_i decommits his own share $a_{i\to j} = x_i k_j + u_{i\to j}$.

c. Computing r within a subset S'

A subset of players $S' \subset [1..n], \;\; |S'| = t+1$ compute and broadcast the public value $r = H'(g^{k^{-1}})$. This subset can be different from the subset of signatories invoked in the signing protocol.

[1] The direction notation $i \to j$ for this phase is defined by who initiated the MtA share conversion.

We propose an implementation of this protocol. This is not a new implementation and is based off of preexisting work [3].

- **Step 1.** Player P_i chooses $\gamma_i \in_R Z_q$, computes $[C_i, D_i] = \mathrm{Com}(g^{\gamma_i})$ and broadcasts C_i.
- **Step 2.** Now that the subset S' is identified, parties can compute:

$$l'_i = (\lambda_{i,S'})(k_i)$$

$$w'_i = (\lambda_{i,S'})(x_i)$$

Note that:

$$k\gamma = \sum_{i,j\in S'} l'_i\gamma_j \mod q$$

Every pair of players performs **one** MtA protocol (see I.3) between k and γ such that every P_i ends up with a (t+1) additive share of $\delta = \sum_{i\in S'} \delta_i = k\gamma$.

- **Step 3.** Every player P_i broadcasts δ_i and computes $\delta^{-1} = k^{-1}\gamma^{-1}$.
- **Step 4.** Every player P_i broadcasts D_i, which allows them to compute and broadcast r and R where:

$$r = \mathrm{H}'(\mathrm{R}) \text{ and } \mathrm{R} = \left(\prod_{i\in S'} g^{\gamma_i}\right)^{\delta^{-1}} = g^{\gamma k^{-1}\gamma^{-1}} = g^{k^{-1}}$$

2. Signing

- **Step 0.** A player calls for the signing of a message *m* **by the subset S** using the signing secret ID_k.
- **Step 1.** Once (*t*+1) volunteers have been identified, they take on the role of signatory and we have now identified $S \subset [1..n]$. They can compute the appropriate Lagrangian coefficients $\lambda_{*,S}$.

For the following equations,

- $\forall i \in [1..n]$, we define $u_{i\to i} = 0$ and $a_{i\to i} + b_{i\to i} = x_i k_i$.
- $\forall i, j \in [1..n]$,
  - $\alpha_{i\to j} = (\lambda_{i,S}\lambda_{j,S})(a_{i\to j})$,
  - $\beta_{i\to j} = (\lambda_{i,S}\lambda_{j,S})(b_{i\to j})$.

Every signatory P_i can now broadcast his part of the signature:

$$s_i = l_i m + r\sum_{j\in S}(\alpha_{i\to j} + \beta_{j\to i}) = \lambda_{i,S}\left[k_i m + r\sum_{j\in S}\lambda_{j,S}\left(a_{i\to j} + b_{j\to i}\right)\right]$$

From which every signatory can deduce:

$$s = \sum_{i\in S} s_i = km + r\sum_{i,j\in S}(\alpha_{i\to j} + \beta_{j\to i}) = k(m + rx) \text{ and verify } (s, r).$$

III. Protocol without Roll-Call

The following protocol allows for a pre-signing and signing that are completely independent from the subset of signatories S. Knowledge of the subset is only required when compiling the final signature.

This protocol is very similar to the previous one, therefore we only describe the modified sections.

1. Pre-signing

b. Modified MtA between all players

For the modified version of this section, we use Shamir secrets instead of nonces and broadcast the result. Every player P_i generates locally two new (t, n) secrets v_i and u_i with shares $v_{i\to j}$ and $u_{i\to j}$.

Given a subset $S \subset [1..n]$ with $|S| = t+1$ and the appropriate Lagrangian coefficients $\lambda_{i,S}$, we define $\mu_{i\to j} = (\lambda_{j,S})(u_{i\to j})$ and $\eta_{i\to j} = (\lambda_{j,S})(v_{i\to j})$. Therefore, $u_i = \sum_{j\in S} \mu_{i\to j}$ and $v_i = \sum_{j\in S} \eta_{i\to j}$.[2]

Every pair of players $(\mathrm{P}_i, \mathrm{P}_j)$ performs a modified version of the MtA protocol (see I.3) with shares k_i and x_i.

- **Step 0.** Player P_i broadcasts:

$$E_{pK_i}(x_i)$$

- **Step 1.** Player P_j sends to P_i:

$$E_{pK_i}(x_i k_j + u_{j\to i})$$

- **Step 2.** Player P_i can now broadcast:

$$a_{i\to j} = (x_i k_j + u_{j\to i}) + v_{i\to j}$$

2. Signing

- **Step 0.** A player calls for the signing of a message *m* **by any subset** using the signing secret ID_k.
- **Step 1.** Every signatory P_i can now broadcast his part of the signature:

$$s_i = l_i m - v_i - u_i \quad \text{and} \quad a_{i\to i} = x_i k_i + u_{i\to i} + v_{i\to i}$$

Once (t+1) parties have been volunteered as signatories, we have now identified $S \subset [1..n]$ and can compute the appropriate Lagrangian coefficients $\lambda_{*,S}$.

Any party (signatory or other) as well as any automatized system listening in on broadcasts can now compute and verify the signature (s, r):

$$s = \sum_{i\in S} \lambda_{i,S}.s_i + r\sum_{i,j\in S} \lambda_{i,S}.\lambda_{j,S}.a_{i\to j}$$

$$s = km + rkx + \sum_{i,j\in S} (\lambda_{j,S}.\mu_{j\to i} + \lambda_{i,S}.\eta_{i\to j}) - \sum_{i\in S} \lambda_{i,S}.\eta_i - \sum_{j\in S} \lambda_{j,S}.\mu_j$$

$$s = km + rkx$$

[2] The direction notation $i \to j$ for these secrets is defined by who created them secret.

IV. Comments

1. Complexity

Using interactions as a reference due to their prevalence and because user interactions are the most time-consuming step, for a (t, n) implementation, we have:

<table>
<tr><th colspan="2"></th><th>Late Roll-Call</th><th>No Roll-Call</th><th>[3] Simplified</th></tr>
<tr><td rowspan="3">Pre-signing</td><td>Signing secret</td><td rowspan="3">TODO</td><td rowspan="3">TODO</td><td rowspan="3">TODO</td></tr>
<tr><td>MtA with n</td></tr>
<tr><td>MtA with (t+1)</td></tr>
<tr><td colspan="2">Signing</td><td>TODO</td><td>TODO</td><td>TODO</td></tr>
<tr><td colspan="2">Full protocol</td><td>TODO</td><td>TODO</td><td>TODO</td></tr>
</table>

2. Possible vulnerabilities

Some modifications to the original protocol may lead to vulnerabilities, but most changes have a similar security level to previously used tools:

- The modified signing secret k has the same security level as any (t, n) Shamir secret, including the private key x. It should be treated with the same level of care (data protection, confidentiality…).
- This protocol requires that players broadcast $E_{pK_i}(x_i)$ in . Breaking the homomorphic encryption would allow any
-

3. Future research

This article is a first attempt at resolving a previously undiscussed limitation found in non-interactive threshold ECDSA protocols thus far.

There are a few research directions that can be looked into to improve the proposed protocols:

- **Time complexity:** Reducing the number of interactions during the pre-signing phase.
- **Security:** Reducing the number of sensitive values that have to be shared and/or saved (secret shares and homomorphic encryptions of shares).

V. References

[1] Elliptic Curve Digital Signature Algorithm, **Wikipedia**. Last edited 6 Jan 2021 (accessed 10 Mar 2021).

[2] A Survey of ECDSA Threshold Signing, J.-P. Aumasson, A. Hamelink & O. Shlomovits, **IACR**. Last edited 10 Nov 2020 (accessed 10 Mar 2021).

[3] One Round Threshold ECDSA with Identifiable Abort, R. Gennaro & S. Goldfeder, **IACR**. Last edited 11 May 2020 (accessed 10 Mar 2021).

[4] Fast multiparty threshold ECDSA with fast trustless setup, R. Gennaro & S. Goldfeder, **IACR**. Last edited 4 Feb 2019 (accessed 24 Mar 2021).