Group Action based Threshold Schemes

DISCLAIMER: the slides are only meant to stimulate the discussion, not to be a precise description of the state of the art.

Giacomo Borin

5 June 2025

Legend

- **[-]** bad thing
- **[+]** good thing
- **[?]** open question
- **[D]** additional comment from the discussion in the group

Context

- Threshold schemes allow a group of users to jointly perform a cryptographic operation.
- Cryptographic Group Actions ($\star : \mathcal{G} \times X \to X$) provide a natural way to define these schemes.
  - Different schemes already proposed in the literature.
  - Recent work $\underline{\text{greatly improves}}$ the efficiency of group action computations[1].
- We will submit these schemes to the NIST Call for Multi-Party Threshold Schemes[2], under Class S:
  - $\underline{\text{S1 (Signing)}}$, S2 (PKE ?), S4 (KeyGen ?).
- Today we would like to briefly present on overview of the idea and discuss which choices are more reasonable. **[+]** All comments, critiques and remarks are welcome.

[1] Dartois et al., PEGASIS: Practical Effective Class Group Action using 4-Dimensional Isogenies.

[2] Brandao and Peralta, Nist first call for multi-party threshold schemes.

NIST guidlines

Nist first call for multi-party threshold schemes : 1413–1417

Class S considers cryptographic schemes not standardized by NIST …
A submission shall be motivated by a serious intention of proposing for technical consideration a scheme that, besides being secure and practical, is believed to have a high potential for adoption in the real world.

[?] Killer features:

- Different quantum assumption;
- Possible parameters trade-offs;
- Only for PKE: compactness;
- Other ideas?

[D] We still struggle to find a killer feature, we should hear from practitioners

Design choices

Group Actions

Consider a group action $\star : \mathcal{G} \times X \to X$:

- Using CSI-FiSh[3] we may assume $\mathcal{G} = \mathbb{Z}/N\mathbb{Z}$;
- In general, e.g. for PEGASIS we can only assume that $\mathcal{G}$ is abelian (N is unknown);
- We do not plan to consider non-abelian groups; **[D]** there will also be some work on non-abelian groups, that will probably lead to a separate but parallel submission.
- We want to define protocols only using the group action framework, making the submission modular
- we still need to provide a "practical" instantiation of the group action.
    - **[D]** The PEGASIS team is working on improving the implementation and we should encourage theme to do so.
    - **[D]** Maybe we will also have a new CSI-FiSh instantiation for higher security levels.

[3] Beullens, Kleinjung, and Vercauteren, "CSI-FiSh: efficient isogeny based signatures through class group computations".

MPC and Sharing

We need that, for any set of T users, each party $\mathcal{P}_i$ can compute a share $\mathsf{g}[i]$ of the secret key:

$$\mathsf{x}_0 \xrightarrow{\mathsf{g}[1]} \mathsf{x}[1] \xrightarrow{\mathsf{g}[2]} \dots \xrightarrow{\mathsf{g}[T]} \mathsf{x}[T] = \mathsf{y}$$

Secret Sharing	**Group Type**	**Efficient**	**Easy Proofs**
Linear	Cyclic	Yes	Not really (PVP[4])
Integer Sharings	Abelian	Yes	Not really (?)
Replicated	Any*	No	Yes
Vandermonde[5]	Any	Yes*	Yes

[4] Campos and Muth, "On Actively Secure Fine-Grained Access Structures from Isogeny Assumptions".

[5] Desmedt, Di Crescenzo, and Burmester, "Multiplicative Non-abelian Sharing Schemes and their Application to Threshold Cryptography".

Signatures

Centralized Signature

Algorithm 1 Sign(y, g, msg)

```
1:  // Commitment Generation
2:  for $i = 1, \dots, \lambda$ do
3:      $\tilde{g}[j] \leftarrow_\$ \mathcal{G}$
4:      $x[j] \leftarrow \tilde{g}[j] \star x_0$
5:  // Challenge Computation
6:  $salt \leftarrow_\$ \{0,1\}^{2\lambda}$
7:  $ch = H(x[:], salt, msg)$
8:  // Response Computation
9:  for $j = 1$ to $\lambda$ do
10:     $resp[j] = \tilde{g}[j] g^{-ch[j]}$
11: return ch, resp[:], salt.
```

Algorithm 2 KeyGen(y, g)

```
1: $sk = g \leftarrow_\$ \mathcal{G}$
2: $vk = y \leftarrow g \star x_0$
3: return sk, vk
```

Algorithm 3 Verify(y, msg, sig)

```
1: Parse ch, resp[:], salt $\leftarrow$ sig
2: $y_0 = x_0, y_1 = y$
3: for $j = 1$ to $\lambda$ do
4:     $x[j] \leftarrow resp[j] \star y_{ch[j]}$
5: return $ch = H(x[:], salt, msg)$
```

Signature Sizes

For a security level of 128 bits, and a delay of 2^{16} additional hashing operations, we can use the following parameters[6]:

# pks	512 bits		1024 bits		2048 bits	
1	64 B	4.48 KB	128 B	8.92 KB	256 B	17.80 KB
8	512 B	1840 B	1024 B	3.55 KB	2.00 KB	7.05 KB
256	16.00 KB	880 B	32.00 KB	1712 B	64.00 KB	3.30 KB
512	32.00 KB	816 B	64.00 KB	1584 B	128.00 KB	3.05 KB
1024	64.00 KB	752 B	128.00 KB	1456 B	256.00 KB	2.80 KB

[D] The sizes are not optimal, but at least we can have trade-offs

[6] The first rows list the size of the underlying field in bits, the first column is the number of public keys used in the signature. Then the even columns are the signature sizes, the odd columns are the public key sizes (relative to the prime size and number of public keys).

Distributed Signature I

Algorithm 4 $\mathsf{T.SignCommit}_i(\mathsf{xsh}[:i,:])$

```
1: $\mathsf{salt}_i \leftarrow_\$ \{0,1\}^{2\lambda}$
2: $\mathsf{c}_{\mathsf{salt},i} \leftarrow \mathsf{COM}(\mathsf{salt}_i)$
3: **for** $j = 1$ to $\lambda$ **do**
4:     $\tilde{\mathsf{g}}[i,j] \leftarrow_\$ \mathcal{G}$;
5:     $\mathsf{xsh}[i,j] = \tilde{\mathsf{g}}[i,j] \star \mathsf{xsh}[i-1,j]$;
6: $\mathsf{pm}_0[i] \leftarrow \mathsf{c}_{\mathsf{salt},i}, \mathsf{xsh}[i,:]$
7: **return** $\mathsf{pm}_0[i]$
```

- [-] Round-robin protocol: the protocol need to be sequential, i.e. T online rounds required.
- [+] Independent of the message to be signed.
- [+] No NIZK, only a commitment to salt_i (the randomness).
- [?] What is the best communication model?

Distributed Signature II

Algorithm 5 T.SignRand$_i$(pm$_0$[:], msg)

```
1: Store $x[:] \leftarrow xsh[T, :]$.
2: Store msg.
3: $pm_1[i] \leftarrow salt_i$
4: return $pm_1[i]$.
```

- The randomness is opened once both x[:] and msgare known.
- Necessary to avoid concurrent attacks.
- **[?]** Are there better ways to do this? E.g. using strategies like RingTail or FROST? Twists?

Distributed Signature III

Algorithm 6 $\mathtt{T}.\mathsf{SignFinal}_i(\mathsf{pm}_1[:])$

```
1: for $i = 1$ to $T$ do
2:     Parse $\mathsf{salt}_i \leftarrow \mathsf{pm}_1[i]$.
3:     if $\mathsf{COM}(\mathsf{salt}_i) \neq \mathsf{c}_{\mathsf{salt},i}$ then
4:         return $\perp$                                    ▷ Invalid salt for $\mathcal{P}_i$
5: $\mathsf{salt} \leftarrow \sum_{i=1}^{T} \mathsf{salt}_i$.
6: $\mathsf{ch} \leftarrow \mathsf{H}(\mathsf{x}[:] \| \mathsf{salt} \| \mathsf{msg})$
7: for $j = 1$ to $\lambda$ do
8:     $\mathsf{rsh}[i, j] \leftarrow \tilde{\mathsf{g}}[i, j]\mathsf{g}[i]^{-\mathsf{ch}[j]}$
9: return $\mathsf{pm}_2[i] := \mathsf{rsh}[i, :]$.
```

Distributed Signature IV

Algorithm 7 T.SignCombine(rsh$[:, :]$)

```
1: for $j = 1$ to $\lambda$ do
2:     resp$[i] \leftarrow \sum_{i=1}^{T}$ rsh$[i, j]$;
3: sig $=$ ch, resp$[:]$, salt
4: return sig.
```

- The public messages can be combined to get a final signature.
- Given the message the signature is computed with 2 online rounds (independently of the number of users).
- The total is $T + 2$ online rounds.
- The correctness of the pre-signatures can be verified using shared public keys, with similar techniques to the ones used in FROST[7].

[7] Komlo and Goldberg, "FROST: Flexible Round-Optimized Schnorr Threshold Signatures".

Security – Assumptions

We are willing to make the following assumptions:

- One-way EGA (old: Vectorization Problem hardness)
- Weak Pseudorandom EGA (old: Decisional Parallelization Problem), if needed for KeyGen.
- Use the Random Oracle Model.
- Use (some flavor of) the Algebraic Action Model / Explicit Isogeny Model[8] (or other flavours of it).
- **[?]** Interactive version of One-way EGA?

The **quantum security** of One-way EGA is still a matter of discussion!!

[8] Orsini and Zanotto, "Simple Two-Message OT in the Explicit Isogeny Model".

Security – Properties

Nist first call for multi-party threshold schemes : 1125–1150

- We need to **prove** unforgeability against static active adversaries of the signature.
- Adaptive security would require more complicated techniques and an interactive version of the One-way EGA.
- Recovery mechanisms:
  - Identifiable Aborts, (almost) *a la* FROST[a];
  - **[?]** What about robustness?

[a] Komlo and Goldberg, "FROST: Flexible Round-Optimized Schnorr Threshold Signatures".

[?] Should we do UC security?

PKE

Distributing CSIDH NIKE

- CSIDH Encaps/Decaps both requires to evaluate a **[+]** single group action.
- **[+]** No FS transform is needed since we have efficient public-key validation.
- Relatively easy to distribute (still requires a **[-]** round-robin).
- **[?]** Use cases?

Distributed Key Generation

Didn't expected to arrive here, but we can discuss it.
[D] Not required by NIST, but it is a nice feature to have.
[D] It strongly depends on the sharing scheme used.
[D] But, there is already some research on it and some results from lattices can be applied.

References I

Beullens, Ward, Thorsten Kleinjung, and Frederik Vercauteren. "CSI-FiSh: efficient isogeny based signatures through class group computations". In: International conference on the theory and application of cryptology and information security. Springer. 2019, pp. 227–247.

Brandao, L and Rene Peralta. Nist first call for multi-party threshold schemes. 2023. DOI: `https://doi.org/10.6028/NIST.IR.8214C.ipd`.

Campos, Fabio and Philipp Muth. "On Actively Secure Fine-Grained Access Structures from Isogeny Assumptions". In: Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022. Ed. by Jung Hee Cheon and Thomas Johansson. Springer, Cham, Sept. 2022, pp. 375–398. DOI: `10.1007/978-3-031-17234-2_18`.

Dartois, Pierrick et al. PEGASIS: Practical Effective Class Group Action using 4-Dimensional Isogenies. Cryptology ePrint Archive, Paper 2025/401. 2025. URL: `https://eprint.iacr.org/2025/401`.

References II

Desmedt, Yvo, Giovanni Di Crescenzo, and Mike Burmester. "Multiplicative Non-abelian Sharing Schemes and their Application to Threshold Cryptography". In: ASIACRYPT'94. Ed. by Josef Pieprzyk and Reihaneh Safavi-Naini. Vol. 917. LNCS. Springer, Berlin, Heidelberg, 1995, pp. 21–32. DOI: 10.1007/BFb0000421.

Komlo, Chelsea and Ian Goldberg. "FROST: Flexible Round-Optimized Schnorr Threshold Signatures". In: SAC 2020. Ed. by Orr Dunkelman, Michael J. Jacobson Jr., and Colin O'Flynn. Vol. 12804. LNCS. Springer, Cham, Oct. 2020, pp. 34–65. DOI: 10.1007/978-3-030-81652-0_2.

Orsini, Emmanuela and Riccardo Zanotto. "Simple Two-Message OT in the Explicit Isogeny Model". In: CiC 1.1 (2024), p. 15. DOI: 10.62056/a39qgy4e-.