GATOR: Group Action AdapTOR Signatures via MPC-in-the-Head

Nico Döttling[1], Manar Mohamed[2], and Riccardo Zanotto[1,2]

[1]CISPA, Saarbrücken, Germany
[2]Saarland University, Saarbrücken, Germany

May 26, 2026

Abstract

Adaptor signatures are a foundational fairness primitive for blockchain applications. They enhance blockchain functionality by enabling applications such as atomic swaps, payment channels, and other fair-exchange protocols. At a high level, they allow a buyer to produce a *pre-signature* tied to a public statement, which a seller holding a corresponding witness can *adapt* into a valid signature. Once this signature is posted on-chain, the seller obtains payment, while the buyer can *extract* the witness from the finalized signature.

Existing practical adaptor signature constructions are predominantly tied to discrete-logarithm-based signatures, such as ECDSA and Schnorr, reflecting their widespread use in current blockchain deployments. However, the threat of Shor's algorithm and the broader transition toward post-quantum cryptography raise the question of whether adaptor functionality can also be realized efficiently for post-quantum signature schemes.

In this work, we answer this question for a broad class of signatures based on cryptographic group actions. Building on efficient MPC-in-the-Head-style group-action signatures, we obtain adaptor functionality through small modifications, yielding a general framework for adaptor signatures from arbitrary group actions. In particular, our construction supports selling group-action discrete logarithms, which in certain parameter regimes correspond to the secret keys of the underlying signature scheme.

We further discuss concrete instantiations from several group-action families, including those underlying schemes such as CSI-FiSh, LESS, MEDS, and ALTEQ, and present a modular proof-of-concept implementation. We obtain a $\sim$25 kB pre-signature with $\sim$100 ms pre-signing time for LESS, MEDS and ALTEQ, while a $\sim$4 kB pre-signature with $\sim$3.6 s pre-signing time for CSI-FiSh.

1 Introduction

Digital signatures are a central tool for any form of online interaction requiring authenticity, providing authentication to parties in an untrusted network. In particular, signature schemes are critical for establishing authenticity of transactions in blockchains. The landscape of blockchains is quickly evolving; first-generation blockchains such as Bitcoin primarily allow users to append transactions with fixed senders, receivers, and amounts, newer systems such as Ethereum support fully fledged updates to a state machine, enabling expressive *smart contracts*. This highlights the necessity of deploying cryptographic primitives that enable complex behavior and interaction on-chain, to achieve guarantees such as integrity and fairness.

Adaptor signatures have recently been introduced as a cryptographic tool to move complex interactions off-chain while retaining on-chain integrity and fairness guarantees. Informally, an adaptor signature scheme allows a *buyer*, holding secret keys of a signature scheme Σ, to generate a *pre-signature* $\tilde{\sigma}$ tied to a public statement stmt of a certain NP relation Rel. A party holding a witness wit for this statement, usually called the *seller*, can *adapt* the pre-signature $\tilde{\sigma}$ into a valid signature σ; once the adapted signature is published on the blockchain, the buyer can *extract* the witness from it. This mechanism turns signatures into a lightweight witness-exchange primitive, not requiring any complex on-chain behaviour, and thus underlies applications such as atomic swaps, payment channels, scriptless scripts, and fair-exchange protocols.

Most practical adaptor signatures exploit the algebraic structure of discrete-logarithm-based signatures, most notably Schnorr signatures. Standard Schnorr-based adaptor signatures admit the following very easy construction [25]: let $Y = g^y$ be a statement-witness pair for the (finite field) discrete logarithm relation, and $Z = g^z$ a public key for the Schnorr signature scheme, with secret key z. A valid Schnorr signature for a message m consists of $R = g^r$, and $s = r + cz$, for $c = \mathsf{H}(Z, m, R)$. The buyer can instead generate a pre-signature for the statement Y by using $\tilde{c} = \mathsf{H}(Z, m, R \cdot Y)$ and $\tilde{s} = r + \tilde{c} \cdot z$. It is easy to see that anyone holding the witness y can adapt this pre-signature into the a valid signature for commitment $R \cdot Y$ using $s = y + \tilde{s}$; the buyer will also be able to extract the witness simply via $y = s - \tilde{s}$.

This gives an especially clean form of adaptor functionality: the adaptor relation is the same discrete-logarithm relation that also underlies the signature scheme Σ itself. In other words, Schnorr adaptor signatures naturally support the "secret-key/public-key" language

$$\mathcal{L}_{\mathsf{skpk}} = \{\, \mathsf{pk} : \exists \mathsf{sk} \text{ with } (\mathsf{sk}, \mathsf{pk}) \in \mathsf{SUPP}(\Sigma.\mathsf{KGen}(1^\lambda)) \,\}.$$

For many applications, however, the adaptor relation need only be some hard relation whose witness is useful to exchange; it does not have to coincide with the secret-key relation of the underlying signature scheme. Still, supporting $\mathcal{L}_{\mathsf{skpk}}$ is a desirable feature, since it recovers the native functionality enjoyed by Schnorr adaptor signatures.

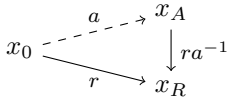
This algebraic picture becomes much less clear in the post-quantum setting. The discrete-logarithm structure used by Schnorr and ECDSA is vulnerable to Shor's algorithm [36], motivating the transition toward post-quantum signatures in settings such as blockchain systems, public-key infrastructures, and standardized protocols. A natural question is therefore:

> Can we construct practical post-quantum adaptor signatures with functionality comparable to Schnorr adaptor signatures?

In this work we study this question for signatures based on cryptographic group actions. A group action $\mathcal{G} = (G, X, \star, x_0)$ can be viewed as a post-quantum analogue of exponentiation: a secret group element $g \in G$ acts on a public set element $x \in X$, producing $g \star x$. The corresponding group-action discrete-logarithm problem asks, given $x = g \star x_0$, to recover such a group element g. Group actions capture several post-quantum signature candidates and constructions, including isogeny-based schemes such as CSI-FiSh [5] and SeaSign [20], as well as isomorphism- and equivalence-based schemes such as LESS [6], MEDS [15], and ALTEQ [39].

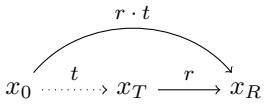
The main barrier in translating the Schnorr approach to the group action setting is that in the classical setting, the "private key" space has *two operations*, addition and multiplication, so that $r + c \cdot z$ has a perfectly valid meaning, and also the "public key" space has *one operation*, multiplication, allowing us to compute $R \cdot Y$. While this structure allows for many constructions, it is also the reason why Schor's algorithm can recover discrete logarithms in quantum polynomial time. A group action instead deliberately allows only for *one* operation in the "private key" space G, and *no* operations on the "public key" space X. This limits the expressiveness of the group action setting.

Nonetheless, we can still quite easily construct identification schemes assuming hardness of the group action discrete logarithm problem. Namely, given a public key $x_A = a \star x_0$, with secret $a \in G$, we can commit to $x_R = r \star x_0$ with a randomly sampled $r \leftarrow\!\$\, G$; then for a binary challenge $c \in \{0, 1\}$ the prover responds either with r or $r \cdot a^{-1}$, which can be verified by checking that they give a valid link between x_0 and x_R or between x_A and x_R. By repeating this protocol λ many times we achieve security. This is the core idea at the heart of most group action-based signature schemes, such as CSI-FiSh, LESS, MEDS, and ALTEQ.



However, the lack of structure on the set X makes it apparently difficult to construct adaptor signatures, since it is impossible to "compose" set elements for embedding statements into the signature, as the Schnorr construction does with $R \cdot Y$. The IAS (Isogeny-based Adaptor Signature) scheme [37] constructs adaptor signatures from the CSIDH group action [12] that are able to sell CSIDH dlogs, but it uses a (specific purpose) NIZK for the language of "DDH" tuples $(x_0, x_1, g \star x_0, g \star x_1)$, which incurs a significant efficiency loss.

The recent ORCAS (One-Round Cheating Adaptor Signature) scheme [11] overcomes this obstruction through a *cheating* mechanism. In one repetition of the basic group action identification protocol, instead of committing to x_R from the base point x_0, the signer commits from x_T, where $x_T = t \star x_0$ is the adaptor statement.



Notice that whenever the corresponding challenge for that round is "connect x_R to x_0" the honest buyer can not respond because they do not know the path from x_0 to x_T. However, a seller holding the witness t can answer this challenge with $r \cdot t$ and obtain a valid signature. This is a valid construction for a generic group action, even though the authors mainly advertise it for CSI-FiSh.

Applying this idea directly to basic group-action signatures leads to a signature-size barrier. The resulting signatures contain roughly λ opened group elements, and hence have size about

$$\lambda \cdot \ell_G,$$

where ℓ_G is the size of a group element. This is acceptable for actions with small group elements, such as the isogeny-based CSI-FiSh, but becomes prohibitively expensive for actions whose group elements are large matrices, as in LESS, MEDS, and ALTEQ.

Existing practical group action signatures avoid this cost by using pseudorandomness, opening compact seeds rather than expanded group elements. More concretely, the secret committing element r is generated from a small seed, and when the challenge is "return r", then the scheme actually hands out the small seed; moreover, the scheme heavily optimizes towards having many challenges of this type, and less of the $a^{-1}r$ type, which result in paying the full ℓ_G cost. A direct application of the ORCAS strategy is incompatible with this seed-opening optimization: after adaptation, the response would be $r \cdot t$, but the verifier expects a short seed expanding to the opened group element. Finding such a seed is computationally infeasible in general. This leads to the central question of our work:

> Can one obtain group action adaptor signatures with size below the $\lambda \cdot \ell_G$ barrier?

We answer this question affirmatively. Our construction starts from efficient MPC-in-the-Head-style group action signatures, in which a signing secret is shared across virtual parties and most shares are generated pseudorandomly from a puncturable PRF. We show how to incorporate the one-round cheating idea into this compact transcript structure. The resulting adaptor signatures support the group action discrete logarithm relation $\mathsf{Rel}_{\mathcal{G}}$, defined by

$$\mathsf{Rel}_{\mathcal{G}} = \{(x, g) \in X \times G : x = g \star x_0\}.$$

In particular, in parameter regimes where the public key of the underlying signature scheme is a single group-action instance, this relation coincides with the signing secret-key/public-key relation. Thus our framework supports selling signing secret keys from $\mathcal{L}_{\mathsf{skpk}}$ in certain regimes, while also applying to the more general setting where adaptor signatures are used merely as hard-relation witness-exchange primitives.

1.1 Our Contributions

Our contributions are as follows.

- **A generic group action adaptor signature framework.** We construct adaptor signatures from arbitrary cryptographic group actions, with security based on the corresponding group action discrete logarithm problem. The construction is not tied to any specific action.
- **Beating the $\lambda \cdot \ell_G$ barrier.** We combine the one-round cheating idea for group action adaptor signatures with MPC-in-the-Head signatures using puncturable PRFs. This preserves the compactness benefits of modern group action signatures, avoiding the need to reveal λ full group elements.
- **Selling relation.** Our adaptor relation is the group action discrete logarithm relation $\mathsf{Rel}_{\mathcal{G}}$. In parameter regimes where public keys are single group-action instances, this coincides with the secret-key/public-key relation of the underlying signature scheme $\mathcal{L}_{\mathsf{skpk}}$, recovering the native functionality of Schnorr adaptor signatures.

- **Security analysis.** We prove pre-signature correctness, perfect adaptability, simple extractability, and strong canonical signing. The latter implies unlinkability and witness hiding.
- **Concrete instantiations and implementation.** We discuss instantiations from several group-action families, including those underlying CSI-FiSh, LESS, MEDS, and ALTEQ, and provide a modular proof-of-concept implementation illustrating the resulting size and runtime tradeoffs.

1.2 Related Works

We discuss the prior work most relevant to our construction: group-action-based signatures, adaptor signatures and generic constructions, post-quantum adaptor signatures, and the closest group-action-based adaptor signatures.

Group-action-based signatures. Group actions have become a unifying abstraction for several post-quantum signature schemes. Isogeny-based examples include signatures derived from commutative isogeny group actions, such as SeaSign [20] and CSI-FiSh [5] from CSIDH [13]. Beyond isogenies, group actions also arise from equivalence and isomorphism problems, leading to code- and matrix-code-based schemes such as LESS [6, 4, 3], MEDS [15, 14], and related constructions such as ALTEQ [39, 7]. More generic approaches consider signatures from abstract group actions and isomorphism problems, often through zero-knowledge or MPC-in-the-Head techniques [30]. Our work belongs to this last line: rather than adapting a single concrete signature scheme, we use the transcript structure of a generic group-action MPC-in-the-Head signature and bind it to an external group-action instance representing the witness to be exchanged.

Adaptor signatures and witness exchange. Adaptor signatures bind a pre-signature to a hard-relation instance so that any valid completion reveals a corresponding witness. They are therefore well suited to blockchain-style witness exchange, where a seller commits to delivering a witness through a pre-signature and a buyer extracts it from the final signature. This mechanism underlies applications such as scriptless scripts, payment channels, and blockchain protocols based on adaptor signatures [40, 2, 38]. Here, the traded witness is a group-action element, and the adaptor relation matches the group-action relation underlying the base signature.

Generic constructions of adaptor signatures. Adaptor signatures have also been studied as a general primitive, with stronger security definitions and generic transformations from broader cryptographic tools [18, 25, 31]. Universal constructions from black-box multi-party computation are also known [16]. These works clarify the general feasibility and security of adaptor signatures, while direct constructions remain useful for showing how the adaptor relation can be embedded into a specific signature structure.

Post-quantum adaptor signatures. Post-quantum adaptor signatures have been proposed from several assumptions. Esgin, Ersoy, and Erkin construct lattice-based adaptor signatures for payment channel networks [23]. Other works build adaptor signatures from isogeny-based group actions, including IAS-based constructions and adaptor variants of SQISign [37, 29, 33]. These works demonstrate the feasibility of post-quantum witness exchange, but are tailored to specific algebraic settings and underlying signature schemes, with the group-action-based constructions relying on commutative actions.

Group action-based adaptor schemes. ORCAS [11] is the closest work to ours. It introduces a one-round cheating technique for obtaining adaptor signatures from group-action identification protocols: one commitment is generated from the adaptor statement rather than from the base point, and a holder of the corresponding witness can later turn it into an honest response. Our construction uses the same high-level cheating principle, but applies it inside an MPC-in-the-Head signature with puncturable pseudorandom generation of the virtual-party shares. This distinction is crucial for actions with large group elements: applying the ORCAS approach directly leads to signatures of size roughly $\lambda \cdot \ell_G$, whereas our goal is to preserve the compactness of MPC-in-the-Head group-action signatures. Indeed, their main focus was CSI-FiSh, which has natively small group elements, and does not need pseudorandom expansion to achieve competitive signature sizes. In short, the comparison point is as follows. Compared with ORCAS, our construction is preferable when the underlying group-action signature already derives many hidden group elements pseudorandomly, so that revealing λ full group elements would destroy the compactness benefits of the signature; this is also the regime where, for $s = 1$, our adaptor relation matches the native

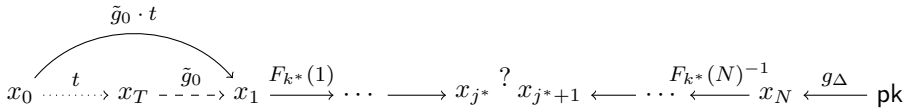


Figure 1: The cheating round of our adaptor signature construction

secret-key/public-key relation of the base signature. Conversely, when group elements are already short and one only needs the one-round cheating transformation itself, ORCAS remains the simpler approach.

1.3 Technical Overview

Our main goal is to construct adaptor signatures for the $\mathsf{Rel}_{\mathcal{G}}$ relation of a group action $\mathcal{G} = (G, X, \star, x_0)$. To achieve this, we adopt the MPC-in-the-Head approach to construct group action-based signatures. More concretely, the framework proposed in [30] consists in sharing the secret key group element between N "virtual parties"

$$\mathsf{sk} = g_1 \cdot g_2 \cdots g_N,$$

and construct a chain of objects as follows.

$$x_0 \xrightarrow{g_1} x_1 \xrightarrow{g_2} \cdots \xrightarrow{g_{N-1}} x_{N-1} \xrightarrow{g_N} \mathsf{pk}$$

A challenge index $j^* \in \{1, \ldots, N\}$ is then derived via Fiat-Shamir on the set elements $x_1, \ldots, x_{N-1}$. The response consists in opening all the g_j's except g_{j^*}. The verifier can now reconstruct all the elements in the chain, starting from both x_0 using $g_1, \ldots, g_{j^*-1}$ in order, and from pk using $g_N^{-1}, \ldots, g_{j^*+1}^{-1}$ in order. The soundness error of this protocol is $1/N$, thus we repeat r many times to achieve security, with $1/N^r \leq 2^{-\lambda}$.
In this unoptimized signature scheme, the final signature contains $(N-1) \cdot r = \lambda \cdot \frac{N-1}{\log N}$ group elements, which is still bigger than λ for $N \geq 2$. To reduce this communication cost, the scheme uses a *puncturable PRF* F_k to derive the shares pseudorandomly instead of sampling them randomly and explicitly sending them all. In particular, the scheme derives $g_j \leftarrow F_k(j)$, and computes an offset

$$g_\Delta = g_N \cdot g_{N-1} \cdots g_1 \cdot \mathsf{sk}^{-1}$$

in order to maintain a valid sharing of the secret key. In this way, for each round, the response consists of the punctured key k^* at index j^*, and this offset g_Δ. The total signature size then becomes approximately

$$r \cdot (\ell_G + \lambda \cdot \log N) = \lambda^2 + 2r \cdot \lambda,$$

assuming the GGM [27] construction of the puncturable PRF, whose punctured keys correspond to nodes on the co-path of the punctured index, each of size around λ. Whenever group elements have size $\ell_G \geq \frac{\lambda^2}{\lambda - 2r}$, this construction achieves a smaller signature size than the naive construction.

Our adaptor signature scheme builds on this framework, and in particular uses a variant of this signature as its base signature scheme. In particular, we use a first step g_0, which is still generated uniformly random, and we do not "puncture" the chain there. Concretely, the response of each round consists of g_0, g_Δ and the punctured key, with the property that g_0 is uniformly random,

$$g_\Delta = F_k(N) \cdots F_k(1) \cdot g_0 \cdot \mathsf{sk}^{-1},$$

and the verifier checks by computing the action from both sides that the commitments correspond.
We now describe how to extend this construction to achieve the adaptor functionality. For pre-signing, given a statement $x_T = t \star x_0$, we employ the cheating strategy of ORCAS to select a random round and start our chain from x_T instead of x_0. In particular, we sample a random $\tilde{g}_0$, a PPRF key k, a random g_Δ, and a random index $j^* \in \{1, \ldots, N\}$. We compute the first step $x_1 = \tilde{g}_0 \star x_T$ using the random $\tilde{g}_0$ on the "fake" start point x_T, and then begin using $F_k(j)$ to extend the chain, up until x_{j^*}. Notice that the buyer *can not* connect x_T to pk, unless they know the witness t, which is what they are buying. Thus they compute the other half of the chain starting from pk, then $x_N = g_\Delta \star \mathsf{pk}$, and then proceed with $F_k(j)^{-1}$. This whole process is describe in Figure 1.

The buyer now computes the challenge, given the cheating round chain and the honestly generated ones, *hoping* that the hash results in the j^* challenge, which is the only one that the buyer can now answer. If that's not the case, the buyer resamples the cheating round. The expected number of resamples is 1.58, and each full resample costs $2N + 2$ group actions in compute time. The resulting pre-signature is composed by the initial step $\tilde{g}_0$, the offset g_Δ, and the punctured key k^*, for all the rounds.
When receiving a pre-signature $\tilde{\sigma}$, the only place where it does not verify as a valid signature is when verifying that $\tilde{g}_0 \star x_0 = x_1$ in the cheating round. However, the seller who knows the witness t to $x_T = t \star x_0$, can compose $g_0 = \tilde{g}_0 \cdot t$ and create now a valid link from x_0 to x_1, since

$$(\tilde{g}_0 \cdot t) \star x_0 = \tilde{g}_0 \star (t \star x_0) = \tilde{g}_0 \star x_T = x_1,$$

as shown also in Figure 1.
Extraction from an adapted signature is also then immediate: the buyer knows $\tilde{g}_0$, and sees $\tilde{g}_0 \cdot t$ in the adapted signature, thus can easily extract the witness t.

In this work we show that this construction satisfies the usual notions of extractability (i.e. that a malicious seller can not provide signatures from which the buyer can not extract a valid witness) and adaptability (i.e. if a pre-verification algorithm succeeds, then any witness will lead to a valid adapted signature). Moreover, we show that our construction achieves a strong variant of canonical signing, which in turn implies unlinkability (i.e. that adapted signatures are indistinguishable from standard signatures even when the statement is public) and witness hiding (i.e. the adapted signature does not reveal anything about the witness).

2 Preliminaries

Notation. We denote the security parameter by $\lambda \in \mathbb{N}$ and write 1^λ for its unary representation. We write PPT and DPT for probabilistic and deterministic polynomial-time algorithms, respectively. For a finite set S, we write $x \leftarrow\!\$\ S$ to denote that x is sampled uniformly at random from S. We denote by $\mathsf{negl}(\lambda)$ an arbitrary negligible function in the security parameter.

2.1 NP Languages and Hard Relations

We recall the notions of NP relations and hard relations, and fix notation that we will use throughout the paper.

Definition 1 (NP relation and language)**.** A relation $\mathsf{Rel} \subseteq \{0,1\}^* \times \{0,1\}^*$ is an *NP relation* if there exists a polynomial p and a DPT algorithm V_{Rel} such that:

1. for every $(\mathsf{stmt}, \mathsf{wit}) \in \mathsf{Rel}$, it holds that $|\mathsf{wit}| \leq p(|\mathsf{stmt}|)$; and
2. for every pair $(\mathsf{stmt}, \mathsf{wit})$,

$$(\mathsf{stmt}, \mathsf{wit}) \in \mathsf{Rel} \quad \Longleftrightarrow \quad V_{\mathsf{Rel}}(\mathsf{stmt}, \mathsf{wit}) = 1.$$

The NP language associated with Rel is

$$\mathcal{L}_{\mathsf{Rel}} \ := \ \{\mathsf{stmt} \in \{0,1\}^* \mid \exists\, \mathsf{wit} \in \{0,1\}^* \text{ s.t. } (\mathsf{stmt}, \mathsf{wit}) \in \mathsf{Rel}\}.$$

We write $\mathsf{Rel}(\mathsf{stmt}, \mathsf{wit}) = 1$ as shorthand for $V_{\mathsf{Rel}}(\mathsf{stmt}, \mathsf{wit}) = 1$, i.e., we identify Rel with its verifier whenever this is clear from context.

Hard relations. We say that a relation is *hard* if it is computationally infeasible to find a witness given only the statement. Formally, we use the following definition.

Definition 2 (Hard relation)**.** A relation Rel is *hard* if there exists a PPT sampling algorithm $\mathsf{RGen}_{\mathsf{Rel}}(1^\lambda)$ that outputs pairs $(\mathsf{stmt}, \mathsf{wit})$ such that $\mathsf{Rel}(\mathsf{stmt}, \mathsf{wit}) = 1$, and for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that

$$\Pr\left[\mathsf{Rel}(\mathsf{stmt}, \mathsf{wit}') = 1 \,\middle|\, \begin{matrix}(\mathsf{stmt}, \mathsf{wit}) \leftarrow \mathsf{RGen}_{\mathsf{Rel}}(1^\lambda) \\ \mathsf{wit}' \leftarrow \mathcal{A}(\mathsf{stmt})\end{matrix}\right] \ \leq \ \mathsf{negl}(\lambda).$$

The probability is taken over the randomness of $\mathsf{RGen}_{\mathsf{Rel}}$ and $\mathcal{A}$.

2.2 Cryptographic Group Actions

Group actions are a mathematical framework that builds on group theory. Their use for cryptography was first envisioned by Couveignes [17], and Rostovtsev and Stolbunov [34], who built key exchange protocols from commutative group actions, and gave an instantiation via the isogeny-based action on ordinary curves. A more modern and comprehensive treatment was given by Alamati et al. [1], where they generically construct signatures, oblivious transfer and other primitives from cryptographic group actions.

Definition 3 (Effective Group Action)**.** Let G_λ be a family of groups and X be a set. The *action* of G on X is denoted by a binary operation

$$\star : G \times X \to X$$

that satisfies the following two properties

- $e \star x = x$ for any $x \in X$, where $e \in G$ is the identity.
- $g_2 \star (g_1 \star x) = (g_2 \cdot g_1) \star x$ for all $x \in X$ and $g_1, g_2 \in G$, where $g_2 \cdot g_1$ denotes the group operation.

An *effective group action* is a family $\mathcal{G}_\lambda = (G_\lambda, X_\lambda, \star, x_0)$, where G_λ acts on X_λ, and there are PPT algorithms for all the following operations:

- Group operations, i.e. operation, inverse, sampling, equality and membership testing.
- Set operations, i.e. equality and membership testing.
- The action operation itself $g \star x$ for any $g \in G_\lambda$ and $x \in X_\lambda$.

Finally, there is an *origin* point $x_0 \in X_\lambda$.

Definition 4. Given a cryptographic group action $\mathcal{G} = (G, X, \star, x_0)$, the *group action discrete logarithm* problem is, given a set element $x = g \star x_0$ for some uniformly random $g \leftarrow\!\$\ G$, to compute a group element g' such that $x = g' \star x_0$.

In particular, we can define the relation $\mathsf{Rel}_\mathcal{G}$, with associated NP language $\mathcal{L}_\mathcal{G}$, for statements $x \in X$ and witnesses $g \in G$:

$$\mathsf{Rel}_\mathcal{G} = \{(x, g) \in X \times G : x = g \star x_0\}.$$

For a cryptographic group action, the relation $\mathsf{Rel}_\mathcal{G}$ has a very simple RGen algorithm given by sampling a uniform $g \leftarrow\!\$\ G$, using the PPT sampler from the definition of being a cryptographic group action. The assumption that discrete logarithm is hard in $\mathcal{G}$ is then also saying that $\mathsf{Rel}_\mathcal{G}$ is a hard relation.

We require that we are also able to sample deterministically from the group, given an initial seed, in a way that is still close to uniform. We formalize the notion in the following.

Definition 5 (Pseudorandom sampler)**.** Let $\{G_\lambda\}_{\lambda \in \mathbb{N}}$ be a family of finite groups, and let $\Gamma_\lambda : \{0,1\}^{m(\lambda)} \to G_\lambda$ be a (family of) DPT algorithm. We say that Γ is a *pseudorandom sampler* for $\{G_\lambda\}$ if for every PPT distinguisher $\mathcal{D} : G \to \{0,1\}$ it holds that

$$\left| \Pr_{s \leftarrow\!\$ \{0,1\}^{m(\lambda)}} \left[\mathcal{D}(\Gamma_\lambda(s)) = 1\right] - \Pr_{g \leftarrow\!\$ G_\lambda} \left[\mathcal{D}(g) = 1\right] \right| \leq \mathsf{negl}(\lambda).$$

Equivalently, the distributions $\Gamma(U_m)$ and U_G are indistinguishable, where U_m denotes the uniform distribution over $\{0,1\}^m$ and U_G denotes the uniform distribution over G.

When clear from the context, we drop the dependency on λ.

2.3 Sigma protocols

Let Rel be an NP relation with associated language $\mathcal{L}_\mathsf{Rel}$. A *Sigma protocol* for Rel is a three-move protocol $\Sigma = (\mathsf{P}_1, \mathsf{V}, \mathsf{P}_2, \mathsf{Verify}_\Sigma)$ between a prover and a verifier. On common input stmt and prover private input wit such that $\mathsf{Rel}(\mathsf{stmt}, \mathsf{wit}) = 1$, the protocol proceeds as follows:

$$\mathsf{com} \leftarrow \mathsf{P}_1(\mathsf{stmt}, \mathsf{wit}), \qquad \mathsf{chal} \leftarrow \mathsf{V}(\mathsf{stmt}, \mathsf{com}), \qquad \mathsf{rsp} \leftarrow \mathsf{P}_2(\mathsf{stmt}, \mathsf{wit}, \mathsf{com}, \mathsf{chal}).$$

The verifier then accepts or rejects by computing

$$\mathsf{Verify}_\Sigma(\mathsf{stmt}, \mathsf{com}, \mathsf{chal}, \mathsf{rsp}) \in \{0,1\}.$$

We call $\mathsf{View}_\Sigma(\mathsf{stmt}, \mathsf{wit}) = (\mathsf{com}, \mathsf{chal}, \mathsf{rsp})$ a transcript of the protocol.
We say that the protocol is *public-coin* if the verifier generates the challenge chal by uniformly sampling in some challenge space.
A Sigma protocol satisfies *completeness* if, for every $(\mathsf{stmt}, \mathsf{wit}) \in \mathsf{Rel}$, an honest execution between $\mathsf{P}(\mathsf{stmt}, \mathsf{wit})$ and $\mathsf{V}(\mathsf{stmt})$ is accepted with probability 1, except possibly with negligible probability.

Definition 6 (Honest-verifier zero-knowledge)**.** A Sigma protocol is *honest-verifier zero-knowledge* if there exists a PPT simulator Sim such that, for every $(\mathsf{stmt}, \mathsf{wit}) \in \mathsf{Rel}$, the distribution of real accepting transcripts generated by an honest execution of the protocol on input $(\mathsf{stmt}, \mathsf{wit})$ is indistinguishable from the distribution

$$(\mathsf{com}, \mathsf{chal}, \mathsf{rsp}) \leftarrow \mathsf{Sim}(\mathsf{stmt}).$$

That is,

$$\{\mathsf{View}_\Sigma(\mathsf{stmt}, \mathsf{wit})\}_\lambda \approx \{\mathsf{Sim}(\mathsf{stmt})\}_\lambda,$$

where $\approx$ denotes the relevant notion of indistinguishability, e.g., perfect, statistical, or computational indistinguishability.

Definition 7 (2-special soundness)**.** A Sigma protocol has *2-special soundness* if there exists a DPT extractor Ext such that, for every statement stmt, given any two accepting transcripts

$$(\mathsf{com}, \mathsf{chal}, \mathsf{rsp}) \quad \text{and} \quad (\mathsf{com}, \mathsf{chal}', \mathsf{rsp}')$$

with the same first message com and distinct challenges $\mathsf{chal} \neq \mathsf{chal}'$, the extractor outputs a witness

$$\mathsf{wit} \leftarrow \mathsf{Ext}(\mathsf{stmt}, \mathsf{com}, \mathsf{chal}, \mathsf{rsp}, \mathsf{chal}', \mathsf{rsp}')$$

such that

$$\mathsf{Rel}(\mathsf{stmt}, \mathsf{wit}) = 1.$$

2.4 Digital Signatures

We recall the syntax and properties of digital signatures [21, 28].

Definition 8 (Digital signature)**.** A *digital signature* scheme $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ for messages of length ℓ_m is defined by the following algorithms:

- $\underline{(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda).}$ The *key generation* is a PPT algorithm that on input a security parameter 1^λ, outputs a private-public key pair $(\mathsf{sk}, \mathsf{vk})$.
- $\underline{\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m).}$ The *signing* algorithm is a PPT algorithm that on input a private key sk and a message $m \in \{0,1\}^{\ell_m}$, outputs a signature σ.
- $\underline{b \leftarrow \mathsf{Verify}(\mathsf{vk}, m, \sigma).}$ The *verification* algorithm is a DPT algorithm that on input a public key vk, a message $m \in \{0,1\}^{\ell_m}$ and a signature σ, outputs a bit b.

We define the correctness of signatures in the usual way:

Definition 9 (Correctness)**.** A digital signature scheme Σ is *correct* if for any $m \in \{0,1\}^{\ell_m}$ it holds, that for all $\lambda \in \mathbb{N}$ and all $(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$:

$$\Pr[\mathsf{Verify}(\mathsf{vk}, m, \mathsf{Sign}(\mathsf{sk}, m)) = 1] = 1.$$

We use the standard notion of EUF-CMA for signatures.

Definition 10 (EUF-CMA security)**.** A signature scheme Σ is existential unforgeable under a chosen message attack, or EUF-CMA secure, if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that for all $\lambda \in \mathbb{N}$ it holds that $\Pr[\mathsf{EUF\text{-}CMA}_{\mathcal{A},\Sigma}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$, where the experiment $\mathsf{EUF\text{-}CMA}_{\mathcal{A},\Sigma}$ is defined in Figure 2 and the probability is taken over all random choices of all randomized algorithms.

$\mathsf{EUF\text{-}CMA}_{\mathcal{A},\Sigma}(\lambda)$	$\mathcal{O}_{\mathsf{Sign}}(\mathsf{sk}, m)$
$M := \emptyset$	$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m)$
$(\mathsf{sk}, \mathsf{vk}) \leftarrow \mathsf{KGen}(1^\lambda)$	$M = M \cup \{m\}$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{vk})$	**return** σ
assert $m^* \notin M$	
return $\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*)$	

Figure 2: The EUF-CMA game for digital signatures

2.5 Adaptor Signatures

We recall here the standard notion of adaptor signatures, following the formalization proposed by Dai, Okamoto, and Yamamoto [18].

Definition 11. An *adaptor signature scheme* $\mathsf{AS}_{\Sigma,\mathsf{Rel}}$ w.r.t. a signature scheme $\Sigma = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ and a hard relation Rel consists of a tuple of four algorithms $\mathsf{AS}_{\Sigma,\mathsf{Rel}} = (\mathsf{pSign}, \mathsf{pVerify}, \mathsf{Adapt}, \mathsf{Extract})$ defined as:

- $\tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, \mathsf{stmt})$: the *pre-signing* algorithm is a PPT algorithm that on input a secret key sk, message m, and statement stmt, outputs a pre-signature $\tilde{\sigma}$.
- $b \leftarrow \mathsf{pVerify}(\mathsf{vk}, m, \tilde{\sigma}, \mathsf{stmt})$: the *pre-verification* algorithm is a DPT algorithm that on input a verification key vk, message m, pre-signature $\tilde{\sigma}$, and statement stmt, outputs a bit b.
- $\sigma \leftarrow \mathsf{Adapt}(\mathsf{vk}, \tilde{\sigma}, \mathsf{wit})$: the *adapting* algorithm is a DPT algorithm that on input a verification key vk, pre-signature $\tilde{\sigma}$, and a witness wit for the statement stmt, outputs an adapted signature σ.
- $\mathsf{wit} \leftarrow \mathsf{Extract}(\mathsf{vk}, \tilde{\sigma}, \sigma, \mathsf{stmt})$: the *extracting* algorithm is a DPT algorithm that on input a verification key vk, a pre-signature $\tilde{\sigma}$, a signature σ and a statement stmt, outputs a witness wit, or $\perp$.

We often omit the subscript from $\mathsf{AS}_{\Sigma,\mathsf{Rel}}$ writing AS to improve readability if the signature scheme Σ and the hard relation Rel are clear from the context. The correctness definitions look as follows:

Definition 12 (Pre-signature correctness)**.** An adaptor signature AS satisfies *pre-signature correctness*, if for all $\lambda \in \mathbb{N}$ and $m \in \{0,1\}^{\ell_m}$:

$$\Pr\left[\begin{array}{c|l} \mathsf{pVerify}(\mathsf{vk}, m, \tilde{\sigma}, \mathsf{stmt}) = 1 \wedge & (\mathsf{sk}, \mathsf{vk}) \leftarrow \Sigma.\mathsf{KGen}(1^\lambda), \\ \Sigma.\mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1 \wedge & (\mathsf{stmt}, \mathsf{wit}) \leftarrow \mathsf{RGen}_{\mathsf{Rel}}(1^\lambda), \\ (\mathsf{stmt}, \mathsf{wit}') \in \mathsf{Rel} & \tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, \mathsf{stmt}), \\ & \sigma \leftarrow \mathsf{Adapt}(\mathsf{vk}, \tilde{\sigma}, \mathsf{wit}), \\ & \mathsf{wit}' := \mathsf{Extract}(\mathsf{vk}, \tilde{\sigma}, \sigma, \mathsf{stmt}) \end{array}\right] = 1.$$

Correctness models an honest execution of the adaptor signature protocol, it requires that adapting an honestly generated pre-signature with the correct witness yields a valid signature. Additionally, correctness requires that the witness can be extracted from both the pre-signature and the signature.

We also recall the canonical signing property: given only a finalized signature σ, it is impossible to distinguish that from a "fresh" signature of the base scheme. We follow [11] in defining a new *strong canonical signing* property, where indistinguishability holds even when the $(\mathsf{stmt}, \mathsf{wit})$ pair used to pre-sign and adapt is known to the adversary.

Definition 13 (Strong canonical signing)**.** Let $\mathsf{AS}_{\Sigma,\mathsf{Rel}}$ be an adaptor signature scheme with respect to a base signature scheme Σ and a relation Rel.
For every valid statement-witness pair $(\mathsf{stmt}, \mathsf{wit}) \in \mathsf{Rel}$, define the canonical signing algorithm $\mathsf{Sign}^{\mathsf{can}}$ by

$$\mathsf{Sign}^{\mathsf{can}}(\mathsf{sk}, \mathsf{vk}, m, \mathsf{stmt}, \mathsf{wit}): \quad \tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, \mathsf{stmt}),\ \sigma \leftarrow \mathsf{Adapt}(\mathsf{vk}, \tilde{\sigma}, \mathsf{wit}).$$

We say that AS has *strong canonical signing* with respect to Σ if for every $\lambda \in \mathbb{N}$, every key pair $(\mathsf{sk}, \mathsf{vk})$ in the support of $\Sigma.\mathsf{KGen}(1^\lambda)$, every message m, and every valid statement-witness pair $(\mathsf{stmt}, \mathsf{wit}) \in \mathsf{Rel}$, the following two distributions are computationally indistinguishable:

$$\{\mathsf{Sign}(\mathsf{sk}, m)\} \overset{c}{\approx} \{\mathsf{Sign}^{\mathsf{can}}(\mathsf{sk}, \mathsf{vk}, m, \mathsf{stmt}, \mathsf{wit})\}.$$

Schemes that achieve this notion also achieve the standard notion of canonical signing, Moreover, [11] notice that strong canonical signing implies unlinkability, which in turn implies witness hiding, as discussed in [31].

Security against malicious seller: we want to protect buyers from malicious sellers that attempt to transform a pre-signature into a valid signature while preventing witness extraction. An adaptor signature scheme is fully extractable if such a malicious seller fails except with a negligible probability. Assuming Rel satisfies one-wayness, full extractability is implied by a simpler notion, called *simple extractability* [18]. *Full extractability* in turn generically implies the classical notions of unforgeability and extractability notions of [2].

Definition 14 (Simple extractability)**.** An adaptor signature scheme AS is extractable, if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that for every $\lambda \in \mathbb{N}$ it holds that

$$\Pr[\mathsf{Ext}_{\mathcal{A},\mathsf{AS}}(\lambda) = 1] \leq \mathsf{negl}(\lambda) \ ,$$

where the experiment $\mathsf{Ext}_{\mathcal{A},\mathsf{AS}}$ is described in Figure 3, and the probability is taken over the random choices of all probabilistic algorithms. In the simple extractability game, $\mathcal{A}$ is given access to a pSign oracle and it wins the game if it can come up with a valid forgery (m^*, σ^*) such that the extraction fails against every pre-signature $\tilde{\sigma}$ previously obtained on m^*. The game is defined in Figure 3.

$\mathsf{Ext}_{\mathcal{A},\mathsf{AS}}(\lambda)$

1 : $\mathcal{T} \leftarrow \emptyset$
2 : $(\mathsf{sk}, \mathsf{vk}) \leftarrow \Sigma.\mathsf{KGen}(1^\lambda)$
3 : $(m^*, \sigma^*) \leftarrow \mathcal{A}(\mathsf{vk})^{\mathsf{pSign}(\mathsf{sk},\cdot,\cdot)}$
4 : **assert** $\Sigma.\mathsf{Verify}(\mathsf{vk}, m^*, \sigma^*)$
5 : **for** $(\mathsf{stmt}, \tilde{\sigma}) \in \mathcal{T}[m^*]$ **do**
6 : $\quad \widehat{\mathsf{wit}} \leftarrow \mathsf{AS}.\mathsf{Extract}(\mathsf{vk}, \tilde{\sigma}, \sigma^*, \mathsf{stmt})$
7 : $\quad$ **if** $(\mathsf{stmt}, \widehat{\mathsf{wit}}) \in \mathsf{Rel}$ **then**
8 : $\quad\quad$ **return** 0
9 : **return** 1

$\mathsf{pSign}(\mathsf{sk}, m, \mathsf{stmt})$

1 : $\tilde{\sigma} \leftarrow \mathsf{AS}.\mathsf{pSign}(\mathsf{sk}, m, \mathsf{stmt})$
2 : $\mathcal{T}[m] \leftarrow \mathcal{T}[m] \cup \{(\mathsf{stmt}, \tilde{\sigma})\}$
3 : **return** $\tilde{\sigma}$

Figure 3: The simple extractability game $\mathsf{Ext}_{\mathcal{A},\mathsf{AS}}(\lambda)$.

Security against a malicious buyer: the notion of *adaptability* guarantees that if a seller pre-verifies a pre-signature as valid and knows a corresponding witness to the underlying statement, then it should be able to adapt the pre-signature into a full valid signature.

Definition 15 (Pre-signature adaptability)**.** An adaptor signature scheme AS satisfies *pre-signature adaptability* if for every PPT adversary $\mathcal{A}$ there exists a negligible function negl such that for every $\lambda \in \mathbb{N}$ it holds that:

$$\Pr\big[\mathsf{Adaptability}_{\mathcal{A},\mathsf{AS}}(\lambda) = 1\big] \leq \mathsf{negl}(\lambda) \ ,$$

where the experiment $\mathsf{Adaptability}_{\mathcal{A},\mathsf{AS}}$ is described in Figure 4, and the probability is taken over the random choices of all probabilistic algorithms.
Moreover, we say that the scheme has *perfect* pre-signature adaptability if for all (even unbounded) adversaries $\mathcal{A}$ we have that: $\Pr\big[\mathsf{Adaptability}_{\mathcal{A},\mathsf{AS}}(\lambda) = 1\big] = 0$.

2.6 Puncturable PRFs

We recall the definition of puncturable PRFs, following the syntax of [35].

Definition 16 (PPRF)**.** A *puncturable PRF* $\mathsf{PPRF} = (\mathsf{KGen}, \mathsf{Eval}, \mathsf{Puncture}, \mathsf{pEval})$ with inputs in $\{0,1\}^n$ and outputs in $\{0,1\}^m$ is defined by the following algorithms:

- $K \leftarrow \mathsf{KGen}(1^\lambda)$: the *key generation* is a PPT algorithm that on input a security parameter 1^λ, outputs a secret key K.

$\mathsf{Adaptability}_{\mathcal{A},\mathsf{AS}}(\lambda)$

1 : $(\mathsf{sk}, \mathsf{vk}) \leftarrow \Sigma.\mathsf{KGen}(1^\lambda)$
2 : $(m, \tilde{\sigma}, (\mathsf{stmt}, \mathsf{wit})) \leftarrow \mathcal{A}(\mathsf{sk})$
3 : **assert** $(\mathsf{stmt}, \mathsf{wit}) \in \mathsf{Rel} \wedge \mathsf{pVerify}(\mathsf{vk}, m, \tilde{\sigma}, \mathsf{stmt}) = 1$
4 : **return** $1 - \Sigma.\mathsf{Verify}(\mathsf{vk}, m, \mathsf{Adapt}(\mathsf{vk}, \tilde{\sigma}, \mathsf{wit}))$

Figure 4: The adaptability game $\mathsf{Adaptability}_{\mathcal{A},\mathsf{AS}}(\lambda)$.

- $y \leftarrow \mathsf{Eval}(K, x)$: the *evaluation* algorithm is a DPT algorithm that takes as input a secret key K and a message $x \in \{0,1\}^n$, and outputs a value $y \in \{0,1\}^m$.
- $K^* \leftarrow \mathsf{Puncture}(K, x^*)$: the *puncturing* algorithm is a DPT algorithm that takes as input a secret key K and a specific point $x^* \in \{0,1\}^n$, and outputs a punctured key K^*.
- $y \leftarrow \mathsf{pEval}(K^*, x)$: the *punctured evaluation* algorithm is a DPT algorithm that takes as input a punctured key K^* and a message $x \in \{0,1\}^n$, and outputs a value $y \in \{0,1\}^m$.

When clear from the context, we use $F_K(x)$ as a shorthand for $\mathsf{PPRF.Eval}(K, x)$, and adopt the convention that $\mathsf{pEval}(\mathsf{Puncture}(K, x^*), x^*) = \perp$.

The required properties are the following:

Definition 17. A puncturable PRF PPRF is *correct* if for all honestly generated keys $K \leftarrow \mathsf{KGen}(1^\lambda)$ and all points x^* we have that

$$\mathsf{pEval}(K^*, x) = F_K(x) \; \forall x \neq x^*,$$

where $K^* \leftarrow \mathsf{Puncture}(K, x^*)$.

Definition 18. A puncturable PRF PPRF is *pseudorandom* if for every PPT adversary $\mathcal{A}$ and all points x^* we have that

$$|\Pr[\mathcal{A}(x^*, K^*, F_K(x^*)) = 1] - \Pr[\mathcal{A}(x^*, K^*, y^*) = 1]| \leq \mathsf{negl}(\lambda)$$

where $K \leftarrow \mathsf{KGen}(1^\lambda)$, $K^* \leftarrow \mathsf{Puncture}(K, x^*)$ and y^* is sampled uniformly from $\{0,1\}^m$.

Theorem 1 ([26, 8])**.** *If one-way functions exist, then PPRFs exist for all choices of input and output lengths n, m.*

When used for correlated generation of randomness, such as in MPCitH protocols, variants of PPRF have been studied, such as the multi-instance PPRFs from [10].

3 Generic Template for Group Action Adaptor Signatures

We describe here our main construction of an adaptor signature scheme that sells group action discrete logarithms, based on the hardness of the same discrete logarithm relation. More concretely, let $\mathcal{G} = (G, X, \star, x_0)$ be a group action, and $\mathsf{Rel}_\mathcal{G}$ be the corresponding discrete logarithm relation

$$\{(x, g) \in X \times G : x = g \star x_0\}.$$

We first present a base signature scheme $\Sigma_\mathcal{G}$, whose security is based on the hardness of the discrete logarithm in $\mathcal{G}$. In particular, we follow the MPCitH paradigm for constructing signatures [30, 10] from cryptographic group actions.
Then, we construct an adaptor signature scheme $\mathsf{AS}_{\Sigma_\mathcal{G}, \mathsf{Rel}_\mathcal{G}}$, whose underlying signature scheme is $\Sigma_\mathcal{G}$ and that allows for selling witnesses of $\mathsf{Rel}_\mathcal{G}$, i.e. discrete logarithms.
In Section 3.2 we describe the Sigma protocol for proving knowledge of a group action discrete logarithm. In Section 3.2 we describe our basic signature scheme $\Sigma_\mathcal{G}$. In Section 3.3 we give the construction of the adaptor signature scheme $\mathsf{AS}_{\Sigma_\mathcal{G}, \mathsf{Rel}_\mathcal{G}}$ and prove the relative security properties.
The main parameters of our constructions are N, the number of simulated parties in the MPCitH protocol, and r, the number of parallel repetition of the basic protocol we need to achieve security.
A main ingredient in this whole section is a puncturable PRF PPRF as in Definition 16.

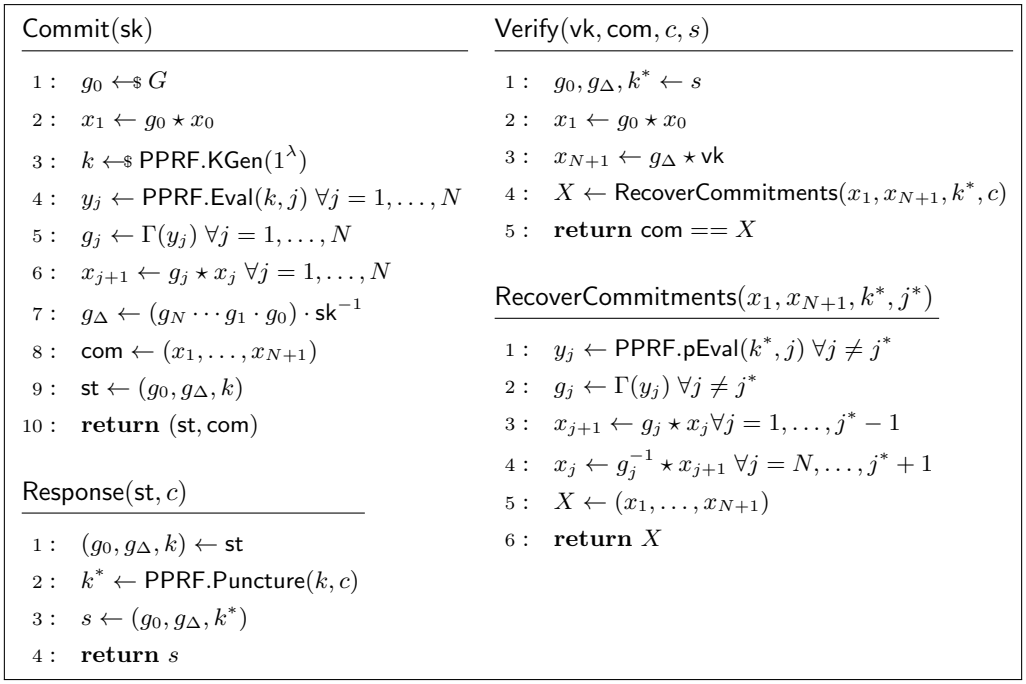


Figure 5: MPCitH Sigma protocol for $\mathsf{Rel}_\mathcal{G}$

3.1 Sigma Protocol for Group Action Discrete Logarithm

We describe here a Sigma protocol for $\mathsf{Rel}_\mathcal{G}$, i.e. a scheme to prove knowledge of a group element $\mathsf{sk} \in G$ such that $\mathsf{vk} = \mathsf{sk} \star x_0$, where x_0 is the baseline point of the group action. This is only a minor modification from Joux's MPCitH protocol [30]. In Figure 5 we describe the basic protocol with N parties and a single round.
We denote by $\Gamma : \{0,1\}^\lambda \to G$ a pseudorandom sampler as in Definition 5.

Theorem 2 ([30]). *The sigma protocol in Figure 5 has HVZK and 2-special soundness*

We sketch the proofs, since we will need them for the proofs of the adaptor signature construction.

Lemma 3 (Computational HVZK). *Assume that* PPRF *has puncturing pseudorandomness, and that* Γ *is a pseudorandom sampler. Then the sigma protocol of Figure 5 is computationally honest-verifier zero-knowledge.*
More precisely, if $\varepsilon_{\mathsf{PPRF}}$ *denotes the advantage against the puncturing pseudorandomness of the PPRF, and* $\varepsilon_{\mathsf{prsamp}}$ *denotes the advantage against the pseudorandom sampler, then the HVZK advantage is at most* $\varepsilon_{\mathsf{PPRF}} + \varepsilon_{\mathsf{prsamp}}$.

Proof. We describe a simulator Sim for the transcript, conditioned on a fixed challenge $j^* \in [N]$.
On input (vk, j^*), the simulator samples

$$g_0 \leftarrow\$ G, \qquad g_\Delta \leftarrow\$ G, \qquad k \leftarrow\$ \mathsf{PPRF.KGen}(1^\lambda),$$

sets

$$k^* := \mathsf{PPRF.Puncture}(k, j^*), \qquad x_1 := g_0 \star x_0, \qquad x_{N+1} := g_\Delta \star \mathsf{vk},$$

and reconstructs the commitment chain as

$$X := \mathsf{RecoverCommitments}(x_1, x_{N+1}, k^*, j^*).$$

It then outputs the pair of commitment X and response s

$$(X, s), \qquad \text{where} \qquad s = (g_0, g_\Delta, k^*).$$

We compare three distributions over one-round transcripts:

- H_0, the real accepting transcript conditioned on challenge j^*;

- H_1, in which we sample a uniform y^* and use $g_{j^*} = \Gamma(y^*)$ to compute g_Δ;
- H_2, in which we sample a uniformly random $g_{j^*} \leftarrow\!\!\$\, G$;
- H_3, the simulator output above.

In particular, in hybrids H_1 and H_2 we sample $g_0 \leftarrow\!\!\$\, G, k \leftarrow\!\!\$\, \mathsf{PPRF.KGen}(1^\lambda)$, and then compute the puncturing key $k^* := \mathsf{PPRF.Puncture}(k, j^*)$ and all the values $g_j := \Gamma(\mathsf{PPRF.pEval}(k^*, j))$ for all $j \neq j^*$. Then, in H_1 we set $g_{j^*} = \Gamma(y)$ for a uniformly random bit-string y, while in H_2 we set $g_{j^*} \leftarrow\!\!\$\, G$ at random. We then form the chain

$$x_1 := g_0 \star x_0, \qquad x_{i+1} := g_i \star x_i \text{ for } i = 1, \ldots, N,$$

and finally compute the offset

$$g_\Delta := (g_N \cdots g_{j^*+1} \cdot g_{j^*} \cdot g_{j^*-1} \cdots g_1 \cdot g_0) \cdot \mathsf{sk}^{-1},$$

and output

$$(X, s), \qquad X = (x_1, \ldots, x_{N+1}), \qquad s = (g_0, g_\Delta, k^*).$$

We now prove that $H_0 \overset{\text{c}}{\approx} H_1 \overset{\text{c}}{\approx} H_2 \overset{\text{s}}{\approx} H_3$.
For $H_0 \overset{\text{c}}{\approx} H_1$, observe that the only difference is whether the missing value at index j^* is the real punctured value $\mathsf{PPRF.Eval}(k, j^*)$ or a uniform random string y^*. Thus any distinguisher between H_0 and H_1 immediately gives a distinguisher for the puncturing pseudorandomness of the PPRF: given a pseudorandomness challenge (j^*, k^*, y), where y is either the real missing value or uniform, the reduction uses k^* to compute all values g_j for $j \neq j^*$, plugs in y at the missing position, builds the chain exactly as above, computes

$$g_\Delta = (g_N \cdots g_{j^*+1} \cdot \Gamma(y) \cdot g_{j^*-1} \cdots g_1 \cdot g_0) \cdot \mathsf{sk}^{-1},$$

and outputs the resulting transcript. If $y = \mathsf{PPRF.Eval}(k, j^*)$, this is exactly H_0; if y is uniform, this is exactly H_1. Hence

$$|\Pr[\mathcal{D}(H_0) = 1] - \Pr[\mathcal{D}(H_1) = 1]| \leq \varepsilon_{\mathsf{PPRF}}.$$

For $H_1 \overset{\text{c}}{\approx} H_2$, the only difference is if g_{j^*} is the image through Γ of a uniformly random string y^*, or is itself uniformly random in G. Thus,

$$|\Pr[\mathcal{D}(H_1) = 1] - \Pr[\mathcal{D}(H_2) = 1]| \leq \varepsilon_{\mathsf{prsamp}}.$$

Finally, we prove that $H_2 \overset{\text{s}}{\approx} H_3$. In H_2, we compute g_Δ as

$$g_\Delta = (g_N \cdots g_{j^*+1}) \cdot u \cdot (g_{j^*-1} \cdots g_1 \cdot g_0) \cdot \mathsf{sk}^{-1},$$

for a uniform random $u \in G$. While in H_3, we simply sample g_Δ uniformly from G. However, multiplication by fixed group elements is a permutation of G, so also in H_2 the value g_Δ is uniformly random, because we have a bijection $u \mapsto g_\Delta$, for fixed k^*, g_0, and sk. Thus the transcripts (X, s), for $s = (g_0, g_\Delta, k^*)$ have the same distribution in H_2 and H_3, i.e. g_Δ is uniform and independent of the secret key sk.

□

Lemma 4 (Two-special soundness)**.** *The Sigma protocol of Figure 5 is* 2*-special sound.*

Proof. Let

$$(X, c, s), \quad (X, c', s')$$

be two accepting transcripts with the same commitment but with distinct challenges $c \neq c'$. Suppose that $c > c'$ and write the two accepting responses as

$$s = (g_0, g_\Delta, k^*) \qquad \text{and} \qquad s' = (g'_0, g'_\Delta, k^{*\prime}).$$

Since both transcripts are accepting and have the same commitment chain $X = (x_1, \ldots, x_{N+1})$, we can write down the equations

$$(g_{c-1} \cdots g_1 \cdot g_0) \star x_0 = x_c \quad \text{and} \quad (g'_N \cdots g'_c) \star x_c = x_{N+1} = g'_\Delta \star \mathsf{vk}$$

If we define

$$\widehat{\mathsf{sk}} := (g'_\Delta)^{-1} \cdot g'_N \cdots g'_c \cdot g_{c-1} \cdots g_1 \cdot g_0,$$

we obtain

$$\mathsf{vk} = \widehat{\mathsf{sk}} \star x_0.$$

Thus $\widehat{\mathsf{sk}}$ is a witness for the public key, proving 2-special soundness. □

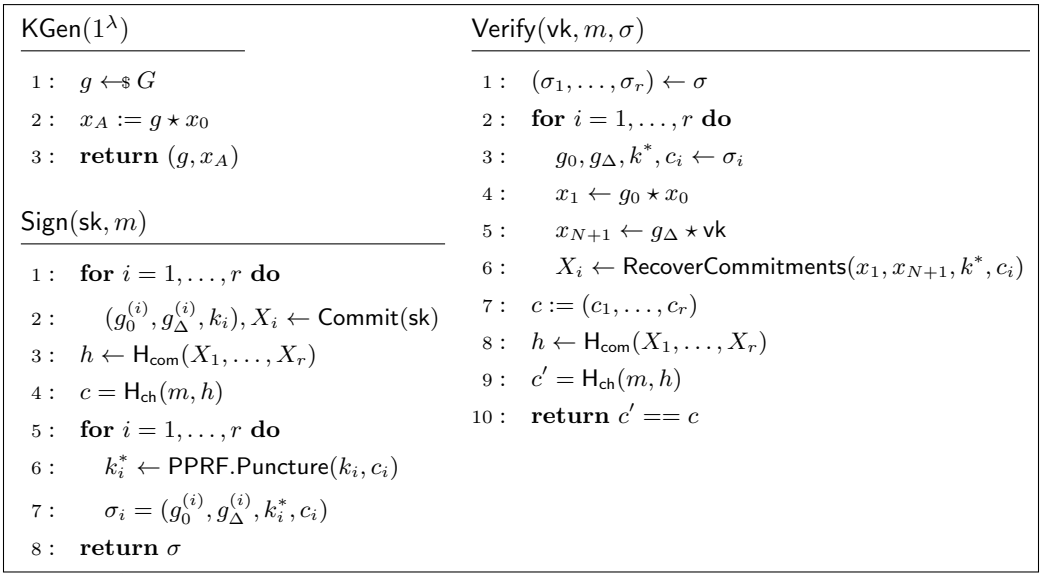


Figure 6: The signature scheme $\Sigma_{\mathcal{G}}$

3.2 Base MPCitH Signature Scheme

We describe the full signature scheme $\Sigma_{\mathcal{G}}$ in Figure 6, where we employ parallel repetitions of the base Sigma protocol and the Fiat-Shamir heuristic [24] to achieve an unforgeable signature scheme. In particular, we repeat for r many rounds, and choose r such that the challenge space becomes exponentially big, i.e. $N^r \geq 2^\lambda$.
We employ two hash functions, modeled as random oracles,

$$\mathsf{H}_{\mathsf{com}} : \{0,1\}^* \to \{0,1\}^\lambda, \quad \text{and} \quad \mathsf{H}_{\mathsf{ch}} : \{0,1\}^* \to [N]^r.$$

We use $\mathsf{H}_{\mathsf{com}}$ to obtain $\mathsf{H}_{\mathsf{com}}(X)$ as a more compact commitment rather than X. Since the collision probability is negligible, we still achieve 2-special soundness. The hash function H_{ch} is used to generate the challenge, and the signature scheme is constructed by applying Fiat-Shamir using H_{ch} as a hash function.

Theorem 5 (adapted from [30])**.** *Assume that the group action discrete logarithm problem is hard, that the PPRF is puncturing-secure, that Γ is pseudorandom, that $\mathsf{H}_{\mathsf{com}}, \mathsf{H}_{\mathsf{ch}}$ are random oracles, and that $N^r \geq 2^\lambda$. Then the signature scheme of Figure 6 is EUF-CMA secure.*

Proof. The protocol of Figure 6 is the compressed MPCitH protocol of Joux [30], with two minor changes. First, we expose an additional uniformly random group element g_0 and thus work from the randomized base point $x_1 = g_0 \star x_0$. This does not affect the structure of the verification equation, since

$$x_{N+1} = g_\Delta \star \mathsf{vk} = \left((g_N \cdots g_1) \cdot g_0 \cdot \mathsf{sk}^{-1}\right) \star (\mathsf{sk} \star x_0) = (g_N \cdots g_1) \star x_1.$$

Hence, conditioned on g_0, the verifier reconstructs exactly the same punctured chain as in Joux's protocol.

Second, we compress the commitment tuple by setting $h = \mathsf{H}_{\mathsf{com}}(X_1, \ldots, X_r)$ before deriving the Fiat-Shamir challenge $c = \mathsf{H}_{\mathsf{ch}}(m, h)$. This is the same global hash-commitment step used in the compressed protocol of [30]; the only additional bad event is that two distinct commitment tuples collide under $\mathsf{H}_{\mathsf{com}}$, which occurs with negligible probability since $\mathsf{H}_{\mathsf{com}}$ is modeled as a random oracle.

By Theorems 3 and 4, our underlying identification protocol satisfies the same HVZK and special-soundness properties required by the Fiat-Shamir analysis in [30]. It therefore follows by the same argument that the Fiat-Shamir transform of parallel repetition yields an EUF-CMA secure signature scheme. □

3.3 Our Adaptor Signature Scheme

We now give a construction and security analysis of our adaptor signature scheme $\mathsf{AS}_{\Sigma_{\mathcal{G}}, \mathsf{Rel}_{\mathcal{G}}}$. The base signature scheme $\Sigma_{\mathcal{G}} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify})$ is defined in Section 3.2. We define the adaptor algorithms $\mathsf{pSign}, \mathsf{pVerify}, \mathsf{Adapt}$, and $\mathsf{Extract}$ in Figure 7.

$\mathsf{Cheat}(\mathsf{vk}, x_T, j^*)$

$g_0 \leftarrow\!\!\$\ G$
$x_1 \leftarrow g_0 \star x_T$
$g_\Delta \leftarrow\!\!\$\ G$
$x_{N+1} \leftarrow g_\Delta \star \mathsf{vk}$
$k \leftarrow\!\!\$\ \mathsf{PPRF.KGen}(1^\lambda)$
$y_j \leftarrow \mathsf{PPRF.Eval}(k, j) \quad \forall j = 1, \ldots, N$
$g_j \leftarrow \Gamma(y_j) \quad \forall j = 1, \ldots, N$
$x_{j+1} \leftarrow g_j \star x_j \quad \forall j = 1, \ldots, j^* - 1$
$x_j \leftarrow g_j^{-1} \star x_{j+1} \quad \forall j = N, \ldots, j^* + 1$
$X \leftarrow (x_1, \ldots, x_{N+1})$
return (g_0, g_Δ, k, X)

$\mathsf{pSign}(\mathsf{sk}, m, x_T)$

$i^* \leftarrow\!\!\$\ [1, r], j^* \leftarrow\!\!\$\ [1, N]$
$\mathsf{vk} \leftarrow \mathsf{sk} \star x_0$
for $i = 1, \ldots, r$ **do**
  if $i == i^*$ **then**
    continue
  else
    $g_0^{(i)}, g_\Delta^{(i)}, k_i, X_i \leftarrow \mathsf{Commit}(\mathsf{sk})$
$c = [0, \ldots, 0]$
while $c_{i^*} \neq j^*$ **do**
  $j^* \leftarrow\!\!\$\ [1, N]$
  $g_0^{(i^*)}, g_\Delta^{(i^*)}, k_{i^*}, X_{i^*} \leftarrow \mathsf{Cheat}(\mathsf{vk}, x_T, j^*)$
  $c = \mathsf{H}_{\mathsf{ch}}(m, \mathsf{H}_{\mathsf{com}}(X_1, \ldots, X_r))$
for $i = 1, \ldots, r$ **do**
  $k_i^* \leftarrow \mathsf{PPRF.Puncture}(k_i, c_i)$
  $\tilde{\sigma}_i = (g_0^{(i)}, g_\Delta^{(i)}, k_i^*, c_i)$
return $\tilde{\sigma} = (i^*, \tilde{\sigma}_1, \ldots, \tilde{\sigma}_r)$

$\mathsf{pVerify}(\mathsf{vk}, m, \tilde{\sigma}, x_T)$

$(i^*, \tilde{\sigma}_1, \ldots, \tilde{\sigma}_r) \leftarrow \tilde{\sigma}$
for $i = 1, \ldots, r$ **do**
  $g_0, g_\Delta, k^*, c_i \leftarrow \tilde{\sigma}_i$
  if $i == i^*$ **then**
    $x_1 \leftarrow g_0 \star x_T$
  else
    $x_1 \leftarrow g_0 \star x_0$
  $x_{N+1} \leftarrow g_\Delta \star \mathsf{vk}$
  $X_i \leftarrow \mathsf{RecoverCommitments}(x_1, x_{N+1}, k^*, c_i)$
$c := (c_1, \ldots, c_r)$
$c' = \mathsf{H}_{\mathsf{ch}}(m, \mathsf{H}_{\mathsf{com}}(X_1, \ldots, X_r))$
return $c' == c$

$\mathsf{Adapt}(\mathsf{vk}, \tilde{\sigma}, t)$

$(i^*, \tilde{\sigma}_1, \ldots, \tilde{\sigma}_r) \leftarrow \tilde{\sigma}$
$\sigma_i = \tilde{\sigma}_i \forall i = 1, \ldots, r$
$(g_0^{(i^*)}, g_\Delta^{(i^*)}, k_{i^*}^*, c_{i^*}) \leftarrow \tilde{\sigma}_{i^*}$
$g' \leftarrow g_0^{(i^*)} \cdot t$
$\sigma_{i^*} = (g', g_\Delta^{(i^*)}, k_{i^*}^*, c_{i^*})$
return $\sigma = (\sigma_1, \ldots, \sigma_r)$

$\mathsf{Extract}(\mathsf{vk}, \tilde{\sigma}, \sigma, x_T)$

$(i^*, \tilde{\sigma}_1, \ldots, \tilde{\sigma}_r) \leftarrow \tilde{\sigma}$
$(\sigma_1, \ldots, \sigma_r) \leftarrow \sigma$
$(\tilde{g}_0, _, _, _) \leftarrow \tilde{\sigma}_{i^*}$
$(g_0, _, _, _) \leftarrow \sigma_{i^*}$
return $\tilde{g}_0^{-1} \cdot g_0$

Figure 7: The adaptor signature scheme $\mathsf{AS}_{\Sigma_\mathcal{G}, \mathsf{Rel}_\mathcal{G}}$

Notation. We denote statement-witness pairs of $\mathsf{Rel}_{\mathcal{G}}$ with (x_T, t), where $x_T = t \star x_0$.
For a tentative signature $\sigma = (\sigma_1, \ldots, \sigma_r)$, let

$$\mathsf{Com}(\sigma) := (X_1, \ldots, X_r)$$

denote the tuple of commitment chains reconstructed by the verification algorithm, namely

$$X_i \leftarrow \mathsf{RecoverCommitments}(x_1, x_{N+1}, k_i^*, c_i),$$

where for each round $\sigma_i = (g_0^{(i)}, g_\Delta^{(i)}, k_i^*, c_i)$ one sets

$$x_1 := g_0^{(i)} \star x_0, \qquad x_{N+1} := g_\Delta^{(i)} \star \mathsf{vk}.$$

Likewise, for a pre-signature $\tilde{\sigma} = (i^*, \tilde{\sigma}_1, \ldots, \tilde{\sigma}_r)$ and a statement x_T, let

$$\mathsf{pCom}(\tilde{\sigma}, x_T) := (X_1, \ldots, X_r)$$

denote the tuple of commitment chains reconstructed by the pre-verification algorithm, namely with

$$x_1 := \begin{cases} g_0^{(i)} \star x_T & \text{if } i = i^*, \\ g_0^{(i)} \star x_0 & \text{if } i \neq i^*, \end{cases} \qquad x_{N+1} := g_\Delta^{(i)} \star \mathsf{vk}.$$

where $i \in [1, r]$ denotes the round index, and i^* denotes the index of the cheating round.

Proposition 6 (Correctness)**.** *The adaptor signature scheme of Figure 7 satisfies pre-signature correctness.*

Proof. Let $(\mathsf{sk}, \mathsf{vk}) \leftarrow \Sigma.\mathsf{KGen}(1^\lambda)$, let $(x_T, t) \in \mathsf{Rel}$, and let

$$\tilde{\sigma} \leftarrow \mathsf{pSign}(\mathsf{sk}, m, x_T).$$

We first show that $\mathsf{pVerify}(\mathsf{vk}, m, \tilde{\sigma}, x_T) = 1$. By definition of $\mathsf{pVerify}, \mathsf{Commit}$, and Cheat, it holds that the x_1 and x_{N+1} values reconstructed by $\mathsf{pVerify}$ are exactly the same generated by pSign: in the cheating round by definition, and for Commit by completeness of the Sigma protocol. Now, by correctness of the PPRF we also get that the y_j and g_j values are the same in every round, thus

$$\mathsf{pCom}(\tilde{\sigma}, x_T)$$

is reconstructed correctly, and the recomputed challenge vector equals the one contained in $\tilde{\sigma}$. Therefore $\mathsf{pVerify}(\mathsf{vk}, m, \tilde{\sigma}, x_T) = 1$.
Now let

$$\sigma \leftarrow \mathsf{Adapt}(\mathsf{vk}, \tilde{\sigma}, t).$$

By definition of the Adapt algorithm, only the cheating round is modified:

$$\tilde{\sigma}_{i^*} = (\tilde{g}_0, \tilde{g}_\Delta, \tilde{k}^*, j^*) \qquad \rightsquigarrow \qquad \sigma_{i^*} = (g_0, \tilde{g}_\Delta, \tilde{k}^*, j^*)$$

where

$$g_0 := \tilde{g}_0 \cdot t.$$

Since $x_T = t \star x_0$, the Verify algorithm executed on σ reconstructs the first node of the cheating round as:

$$g_0 \star x_0 = (\tilde{g}_0 \cdot t) \star x_0 = \tilde{g}_0 \star (t \star x_0) = \tilde{g}_0 \star x_T.$$

Thus the special round of σ reconstructs exactly the same commitment chain as the special round of $\tilde{\sigma}$ under pre-verification. All other rounds are unchanged.
Therefore

$$\mathsf{Com}(\sigma) = \mathsf{pCom}(\tilde{\sigma}, x_T),$$

so the planted Fiat-Shamir challenge is the correct one for the cheating round and $\Sigma.\mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1$.
Finally, the extraction algorithm outputs:

$$\mathsf{Extract}(\mathsf{vk}, \tilde{\sigma}, \sigma, x_T) = \tilde{g}_0^{-1} \cdot g_0 = \tilde{g}_0^{-1} \cdot (\tilde{g}_0 \cdot t) = t.$$

Since $(x_T, t) \in \mathsf{Rel}$, correctness follows. □

Proposition 7 (Perfect adaptability)**.** *The adaptor signature scheme of Figure 7 is perfectly adaptable.*

Proof. Let $(x_T, t) \in \mathsf{Rel}$ and let $\tilde{\sigma}$ be any valid pre-signature bound to statement x_T for a message m under a public key vk, i.e.

$$\mathsf{pVerify}(\mathsf{vk}, m, \tilde{\sigma}, x_T) = 1.$$

Denote now by $(X_1, \ldots, X_r) = \mathsf{pCom}(\tilde{\sigma}, x_T)$ the commitments reconstructed by $\mathsf{pVerify}$.
Let $\sigma = \mathsf{Adapt}(\mathsf{vk}, \tilde{\sigma}, t)$ be the adapted signature. Notice now that, if $x_T = t \star x_0$, then Adapt substitutes the $\tilde{g}_0^{(i^*)}$ from the pre-signature $\tilde{\sigma}$ with $g_0^{(i^*)} = \tilde{g}_0^{(i^*)} \cdot t$ in the final signature σ. Thus, we have that

$$\mathsf{Com}(\sigma) = \mathsf{pCom}(\tilde{\sigma}, x_T),$$

since in the cheating round i^* the Verify algorithm computes $g_0^{(i^*)} \star x_0 = \tilde{g}_0^{(i^*)} \star (t \star x_0) = \tilde{g}_0^{(i^*)} \star x_T$, which is exactly the reconstructed commitment by Adapt.
Since $\mathsf{pVerify}(\mathsf{vk}, m, \tilde{\sigma}, x_T) = 1$, if $c = (c_1, \ldots, c_r)$ is the challenge vector contained in $\tilde{\sigma}$, then

$$c = \mathsf{H}_{\mathsf{ch}}(m, \mathsf{H}_{\mathsf{com}}(\mathsf{pCom}(\tilde{\sigma}, x_T))).$$

Given that the commitments are the same, and that the challenges are the same, we have also $\mathsf{H}_{\mathsf{ch}}(m, \mathsf{H}_{\mathsf{com}}(\mathsf{Com}(\sigma))) = c$, thus

$$\mathsf{Verify}(\mathsf{vk}, m, \sigma) = 1.$$

□

Proposition 8 (Strong canonical signing)**.** *Assume that the Sigma protocol of Figure 5 is computational honest-verifier zero-knowledge. Then the adaptor signature scheme of Figure 7 has strong canonical signing.*

Proof. Fix $\lambda \in \mathbb{N}$, a key pair $(\mathsf{sk}, \mathsf{vk})$ in the support of $\mathsf{KGen}(1^\lambda)$, a message m, and a valid statement-witness pair $(x_T, t) \in \mathsf{Rel}$. Thus

$$\mathsf{vk} = \mathsf{sk} \star x_0 \qquad \text{and} \qquad x_T = t \star x_0.$$

We show that

$$\mathsf{Sign}(\mathsf{sk}, m) \overset{\mathrm{c}}{\approx} \mathsf{Sign}^{\mathsf{can}}(\mathsf{sk}, \mathsf{vk}, m, x_T, t),$$

where

$$\mathsf{Sign}^{\mathsf{can}}(\mathsf{sk}, \mathsf{vk}, m, x_T, t) := \mathsf{Adapt}(\mathsf{vk}, \mathsf{pSign}(\mathsf{sk}, m, x_T), t).$$

We use the following hybrids.

Hybrid $\mathcal{H}_0$. This is the ordinary signing distribution

$$\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, m).$$

Equivalently, in the random-oracle model, we use the following programmed Fiat–Shamir view of signing. First sample a challenge vector

$$c = (c_1, \ldots, c_r) \leftarrow\!\!\$\ [N]^r.$$

For every round $i \in [r]$, generate an honest accepting transcript of the Sigma protocol on statement vk, conditioned on challenge c_i:

$$(\mathsf{st}_i, X_i) \leftarrow \mathsf{Commit}(\mathsf{sk}), \qquad s_i \leftarrow \mathsf{Response}(\mathsf{st}_i, c_i).$$

We program the random oracle so that

$$\mathsf{H}_{\mathsf{ch}}(m, \mathsf{H}_{\mathsf{com}}(X_1, \ldots, X_r)) := c,$$

and output

$$\sigma = \big((s_1, c_1), \ldots, (s_r, c_r)\big).$$

This programmed view is distributed as ordinary signing, except for the standard negligible random-oracle programming bad event.

Hybrid $\mathcal{H}_1$. This hybrid is identical to the programmed signing experiment above, except that we sample an index

$$i^* \leftarrow\!\$\, [r]$$

and generate the i^*-th round using the HVZK simulator, conditioned on the challenge c_{i^*}:

$$(X_{i^*}, s_{i^*}) \leftarrow \mathsf{Sim}(\mathsf{vk}, c_{i^*}).$$

All other rounds $i \neq i^*$ are generated honestly as in $\mathcal{H}_0$.
By computational HVZK of the Sigma protocol,

$$\mathcal{H}_0 \overset{\text{c}}{\approx} \mathcal{H}_1.$$

Indeed, any distinguisher between these two hybrids immediately gives a distinguisher between a real accepting transcript and a simulated accepting transcript of the Sigma protocol on statement vk, conditioned on the same challenge. The reduction embeds its HVZK challenge transcript in the randomly chosen round i^* and generates all other rounds honestly.

Hybrid $\mathcal{H}_2$. This is the distribution

$$\sigma \leftarrow \mathsf{Sign}^{\mathsf{can}}(\mathsf{sk}, \mathsf{vk}, m, x_T, t).$$

We claim that

$$\mathcal{H}_1 \equiv \mathcal{H}_2.$$

Indeed, consider the special round i^* of the canonical signing procedure. Before adaptation, the pre-signing algorithm runs the cheating-round generation procedure with respect to x_T. It samples

$$\tilde{g}_0 \leftarrow\!\$\, G, \qquad g_\Delta \leftarrow\!\$\, G, \qquad k \leftarrow \mathsf{PPRF.KGen}(1^\lambda),$$

punctures at the challenge index c_{i^*}, and reconstructs the commitment chain from

$$\tilde{g}_0 \star x_T \qquad \text{and} \qquad g_\Delta \star \mathsf{vk}.$$

After adaptation, the first group element is replaced by

$$g_0 := \tilde{g}_0 \cdot t.$$

Since t is fixed, the map

$$\tilde{g}_0 \mapsto \tilde{g}_0 \cdot t$$

is a permutation of G. Hence g_0 is uniformly distributed in G.
Moreover, since $x_T = t \star x_0$, ordinary verification of the adapted signature reconstructs the first node as

$$g_0 \star x_0 = (\tilde{g}_0 \cdot t) \star x_0 = \tilde{g}_0 \star (t \star x_0) = \tilde{g}_0 \star x_T.$$

Thus the adapted cheating round reconstructs exactly the same commitment chain as the pre-signature round.
Therefore, after adaptation, the special round consists of a uniform g_0, a uniform g_Δ, a punctured key k^* at challenge c_{i^*}, and the commitment chain reconstructed from

$$g_0 \star x_0 \qquad \text{and} \qquad g_\Delta \star \mathsf{vk}.$$

This is precisely the distribution of the HVZK simulator $\mathsf{Sim}(\mathsf{vk}, c_{i^*})$ used in $\mathcal{H}_1$.
All rounds $i \neq i^*$ are generated honestly and are left unchanged by adaptation. The index i^* is uniform in both experiments, and the Fiat–Shamir challenge vector is distributed in the same way. Hence

$$\mathcal{H}_1 \equiv \mathcal{H}_2.$$

Combining the hybrids gives

$$\mathsf{Sign}(\mathsf{sk}, m) \overset{\text{c}}{\approx} \mathsf{Sign}^{\mathsf{can}}(\mathsf{sk}, \mathsf{vk}, m, x_T, t).$$

□

Recall that, as [11] points out, strong canonical signing directly implies unlinkability, which in turns implies also witness hiding.

Proposition 9 (Simple extractability)**.** *Assume that the group action discrete logarithm problem is hard, and that the Sigma protocol of Figure 5 has computational HVZK. Then, in the programmable random-oracle model, the adaptor signature scheme of Figure 7 is simply extractable.*

Proof. Let $\mathcal{A}$ be a PPT adversary against the simple extractability game $\mathsf{Ext}_{\mathcal{A},\mathsf{AS}}(\lambda)$ of Figure 3 for the adaptor signature scheme $\mathsf{AS}_{\Sigma_{\mathcal{G}},\mathsf{Rel}_{\mathcal{G}}}$. Recall that

$$\mathsf{H}_{\mathsf{com}} : \{0,1\}^* \to \{0,1\}^{\lambda} \qquad \text{and} \qquad \mathsf{H}_{\mathsf{ch}} : \{0,1\}^* \to [N]^r$$

are modeled as independent random oracles. Let Q_{pS} be the number of pre-signing queries and let $Q_{\mathsf{H}_{\mathsf{ch}}}$ be the number of H_{ch} queries made by $\mathcal{A}$.

Game $\mathcal{H}_0$. This is the real simple extractability experiment $\mathsf{Ext}_{\mathcal{A},\mathsf{AS}}(\lambda)$. Pre-signing queries are answered using the genuine algorithm $\mathsf{pSign}(\mathsf{sk},\cdot,\cdot)$. Let W_0 denote the event that $\mathcal{A}$ wins in $\mathcal{H}_0$.

Game $\mathcal{H}_1$. This game answers pre-signing queries by first sampling a random challenge c, and then programming the random oracle H_{ch} to answer c on the commitment.
In particular, on a pre-signing query (m, x_T), the oracle samples

$$i^* \leftarrow\!\$\ [r], \qquad c = (c_1, \ldots, c_r) \leftarrow\!\$\ [N]^r.$$

For every round $i \neq i^*$, it honestly generates a Sigma-protocol transcript on statement vk, conditioned on challenge c_i:

$$(\mathsf{st}_i, X_i) \leftarrow \mathsf{Commit}(\mathsf{sk}), \qquad s_i \leftarrow \mathsf{Response}(\mathsf{st}_i, c_i).$$

For the cheating round $i = i^*$, it runs the $\mathsf{Cheat}(\mathsf{vk}, x_T, c_{i^*})$ algorithm, obtains the values $(g_0, g_\Delta, k, X_{i^*})$, punctures k at position c_{i^*} to get a punctured key $k^* \leftarrow \mathsf{PPRF.Puncture}(k, c_{i^*})$, defines s_{i^*} as (g_0, g_Δ, k^*) and uses

$$(X_{i^*}, s_{i^*})$$

for the commitment-response pair of the cheating round. It then computes

$$h := \mathsf{H}_{\mathsf{com}}(X_1, \ldots, X_r)$$

and programs

$$\mathsf{H}_{\mathsf{ch}}(m, h) := c.$$

Finally, it returns

$$\tilde{\sigma} = (i^*, (s_1, c_1), \ldots, (s_r, c_r)),$$

and stores $(x_T, \tilde{\sigma})$ in $\mathcal{T}[m]$.
Let $\mathsf{Bad}_{\mathsf{prog}}$ be the event that, during this simulation, the oracle tries to program $\mathsf{H}_{\mathsf{ch}}(m, h)$ at a point that has already been queried or assigned a different value. Let W_1 be the event that $\mathcal{A}$ wins in $\mathcal{H}_1$.
Conditioned on $\neg\mathsf{Bad}_{\mathsf{prog}}$, the programmed view is distributed as the real rejection-sampling view of pSign. Hence

$$\Pr[W_0] = \Pr[W_1 \mid \neg\mathsf{Bad}_{\mathsf{prog}}].$$

Game $\mathcal{H}_2$. This game is identical to $\mathcal{H}_1$, except that for each pre-signing query (m, x_T) and for every non-cheating round $i \neq i^*$, the honest transcript conditioned on c_i is replaced by the HVZK simulator:

$$(X_i, s_i) \leftarrow \mathsf{Sim}(\mathsf{vk}, c_i).$$

The cheating round $i = i^*$ is still generated by executing $\mathsf{Cheat}(\mathsf{vk}, x_T, c_{i^*})$, and the oracle still programs $\mathsf{H}_{\mathsf{ch}}(m, h) := c$ as in $\mathcal{H}_1$.
Let W_2 be the event that $\mathcal{A}$ wins in $\mathcal{H}_2$.
By a standard hybrid over the pre-signing queries and over the $r-1$ non-special rounds in each query, computational HVZK gives

$$|\Pr[W_1 \mid \neg\mathsf{Bad}_{\mathsf{prog}}] - \Pr[W_2 \mid \neg\mathsf{Bad}_{\mathsf{prog}}]| \leq Q_{\mathsf{pS}} \cdot (r-1) \cdot \varepsilon_{\mathsf{HVZK}}.$$

Indeed, in each hybrid the reduction embeds its HVZK challenge transcript in one non-cheating round and generates all other rounds as in the game.

We now analyze $\mathcal{H}_2$. Suppose that $\mathcal{A}$ outputs (m^*, σ^*), and let

$$\mathsf{Com}(\sigma^*) = (X_1^*, \ldots, X_r^*)$$

be the commitment tuple reconstructed from σ^*.
We define two complementary events.

- E_1 is the event that there exists a pair $(x_T, \tilde{\sigma}) \in \mathcal{T}[m^*]$ such that

$$\mathsf{pCom}(\tilde{\sigma}, x_T) = \mathsf{Com}(\sigma^*).$$

- E_2 is the event that for every pair $(x_T, \tilde{\sigma}) \in \mathcal{T}[m^*]$,

$$\mathsf{pCom}(\tilde{\sigma}, x_T) \neq \mathsf{Com}(\sigma^*).$$

Event E_1: forgery re-uses commitments from a pre-signing query. Suppose that $(x_T, \tilde{\sigma}) \in \mathcal{T}[m^*]$ gives matching commitments with σ^*. Write

$$\tilde{\sigma} = (i^*, (s_1, c_1), \ldots, (s_r, c_r)).$$

Let $s_{i^*} = (\tilde{g}_0, \tilde{g}_\Delta, \tilde{k}^*)$ be the response in the special round of $\tilde{\sigma}$, and let $s_{i^*}^* = (g_0, g_\Delta, k^*)$ be the corresponding response in σ^*.
Since the commitment tuples are equal, the first node of round i^* is equal in both reconstructions. Therefore

$$g_0 \star x_0 = \tilde{g}_0 \star x_T.$$

Define $\hat{t} := \tilde{g}_0^{-1} \cdot g_0$. Then

$$\hat{t} \star x_0 = \tilde{g}_0^{-1} \star (g_0 \star x_0) = \tilde{g}_0^{-1} \star (\tilde{g}_0 \star x_T) = x_T.$$

Hence $(x_T, \hat{t}) \in \mathsf{Rel}$. By definition of the extraction algorithm,

$$\mathsf{Extract}(\mathsf{vk}, \tilde{\sigma}, \sigma^*, x_T) = \tilde{g}_0^{-1} \cdot g_0 = \hat{t}.$$

Thus extraction succeeds for this queried pre-signature, and the adversary does not win the simple extractability game. Consequently,

$$\Pr[W_2 \wedge E_1] = 0.$$

Event E_2: forgery uses new commitments. We now consider the case where no pre-signing query on m^* has the same reconstructed commitment tuple as σ^*. We define two additional bad events in $\mathcal{H}_2$.

- $\mathsf{Bad}_{\mathsf{col}}$ is the event that two distinct commitment tuples considered during the experiment have the same value under $\mathsf{H}_{\mathsf{com}}$.
- $\mathsf{Bad}_{\mathsf{sig}}$ is the event that $\mathcal{A}$ outputs a valid signature σ^* with reconstructed commitment tuple $(X_1^*, \ldots, X_r^*)$, where

$$h^* = \mathsf{H}_{\mathsf{com}}(X_1^*, \ldots, X_r^*),$$

  without having queried $\mathsf{H}_{\mathsf{ch}}(m^*, h^*)$ before outputting σ^*.

Let

$$\mathsf{Good} := \neg\mathsf{Bad}_{\mathsf{prog}} \wedge \neg\mathsf{Bad}_{\mathsf{col}} \wedge \neg\mathsf{Bad}_{\mathsf{sig}}.$$

Assume that $W_2 \wedge E_2 \wedge \mathsf{Good}$ occurs. Since σ^* is valid, if $c^* = (c_1^*, \ldots, c_r^*)$ is the challenge vector contained in σ^*, then

$$h^* = \mathsf{H}_{\mathsf{com}}(X_1^*, \ldots, X_r^*), \qquad c^* = \mathsf{H}_{\mathsf{ch}}(m^*, h^*).$$

Since $\neg\mathsf{Bad}_{\mathsf{sig}}$ holds, the adversary queried $\mathsf{H}_{\mathsf{ch}}(m^*, h^*)$ before outputting σ^*. Since $\neg\mathsf{Bad}_{\mathsf{col}}$ holds, the value h^* uniquely determines the commitment tuple $(X_1^*, \ldots, X_r^*)$ among all commitment tuples considered in the experiment.
Let

$$q := \Pr[W_2 \wedge E_2 \wedge \mathsf{Good}].$$

We build a reduction $\mathcal{B}$ against the group action discrete logarithm problem. On input a discrete-logarithm instance x_B, the reduction sets $\mathsf{vk} := x_B$ and runs $\mathcal{A}$ in $\mathcal{H}_2$, answering pre-signing queries using the simulated oracle described above.
When the event defining q occurs, the reduction rewinds $\mathcal{A}$ to the query $\mathsf{H}_{\mathsf{ch}}(m^*, h^*)$, keeps the adversary's coins and all previous random-oracle answers fixed, and answers this query with an independently sampled challenge vector

$$c^{**} \leftarrow\$ [N]^r.$$

By the forking lemma, with probability at least

$$q\left(\frac{q}{Q_{\mathsf{H}_{\mathsf{ch}}}} - \frac{1}{N^r}\right),$$

the rewound adversary outputs a second valid signature σ^{**} on the same message m^* and with the same reconstructed commitment tuple, but with a different challenge vector $c^{**} \neq c^*$.
Choose an index $i \in [r]$ such that $c_i^* \neq c_i^{**}$. The i-th round of σ^* and the i-th round of σ^{**} are two accepting transcripts of the underlying Sigma protocol on statement vk, with the same commitment X_i^* and distinct challenges c_i^* and c_i^{**}. Therefore, by 2-special soundness, the reduction extracts a group element $\widehat{\mathsf{sk}} \in G$ such that

$$\mathsf{vk} = \widehat{\mathsf{sk}} \star x_0.$$

Since $\mathsf{vk} = x_B$, this solves the group action discrete logarithm problem.
It follows that

$$\varepsilon_{\mathsf{dlog}} \geq q\left(\frac{q}{Q_{\mathsf{H}_{\mathsf{ch}}}} - \frac{1}{N^r}\right).$$

It follows that

$$q \leq \sqrt{Q_{\mathsf{H}_{\mathsf{ch}}} \cdot \varepsilon_{\mathsf{dlog}}} + \frac{Q_{\mathsf{H}_{\mathsf{ch}}}}{N^r}.$$

We now combine the bounds. Since E_1 and E_2 are complementary and $\Pr[W_2 \wedge E_1] = 0$, we have

$$\begin{aligned}\Pr[W_2 \mid \neg\mathsf{Bad}_{\mathsf{prog}}] &= \Pr[W_2 \wedge E_2 \mid \neg\mathsf{Bad}_{\mathsf{prog}}] \\ &\leq \frac{q}{\Pr[\neg\mathsf{Bad}_{\mathsf{prog}}]} + \Pr[\mathsf{Bad}_{\mathsf{col}} \vee \mathsf{Bad}_{\mathsf{sig}} \mid \neg\mathsf{Bad}_{\mathsf{prog}}].\end{aligned}$$

We now bound the bad events. The event $\mathsf{Bad}_{\mathsf{prog}}$ occurs only if, during a programmed pre-signing query, the adversary has already queried or forced the point (m, h) that the oracle wants to program, where $h = \mathsf{H}_{\mathsf{com}}(X_1, \ldots, X_r)$ comes from a freshly generated commitment tuple. This happens with negligible probability. The event $\mathsf{Bad}_{\mathsf{col}}$ is a collision event for the random oracle $\mathsf{H}_{\mathsf{com}}$ over polynomially many commitment tuples, and is negligible. Finally, $\mathsf{Bad}_{\mathsf{sig}}$ occurs only if the adversary outputs a valid signature without querying $\mathsf{H}_{\mathsf{ch}}(m^*, h^*)$. In that case the challenge value $\mathsf{H}_{\mathsf{ch}}(m^*, h^*)$ is uniform in $[N]^r$ and independent of the adversary's output, so the probability that it equals the challenge vector contained in σ^* is at most $1/N^r$.
Thus,

$$\Pr[W_2 \mid \neg\mathsf{Bad}_{\mathsf{prog}}] \leq \sqrt{Q_{\mathsf{H}_{\mathsf{ch}}} \cdot \varepsilon_{\mathsf{dlog}}} + \frac{Q_{\mathsf{H}_{\mathsf{ch}}} + 1}{N^r} + \mathsf{negl}(\lambda)$$

Finally, we have that

$$\begin{aligned}\Pr[\mathsf{Ext}_{\mathcal{A},\mathsf{AS}}(\lambda) = 1] &= \Pr[W_0] \\ &= \Pr[W_1 \mid \neg\mathsf{Bad}_{\mathsf{prog}}] \\ &\leq \Pr[W_2 \mid \neg\mathsf{Bad}_{\mathsf{prog}}] + Q_{\mathsf{pS}} \cdot (r-1) \cdot \varepsilon_{\mathsf{HVZK}} \\ &\leq \sqrt{Q_{\mathsf{H}_{\mathsf{ch}}} \cdot \varepsilon_{\mathsf{dlog}}} + \frac{Q_{\mathsf{H}_{\mathsf{ch}}} + 1}{N^r} + Q_{\mathsf{pS}} \cdot (r-1) \cdot \varepsilon_{\mathsf{HVZK}} + \mathsf{negl}(\lambda).\end{aligned}$$

Under the group action discrete logarithm assumption, the HVZK property, and the choice of parameters satisfying $N^{-r} \leq 2^{-\lambda}$, the right-hand side is negligible. Therefore the adaptor signature scheme is simply extractable. □

3.4 Optimizations

We discuss some potential changes to the protocol and proof for better results.

3.4.1 Punctured PRFs

RO-based PPRF We can use a random-oracle based puncturable PRF, as described by Joux [30], to achieve "online extraction" in the Sigma protocol. In particular, by making the adversary derive the group elements they are using to generate the commitments X through a random oracle, the reduction is able to read them, and (partially) reconstruct the PPRF tree, and in particular partial information on the chains linking the different x_i's in the commitments. In some sense, in this case we can use the random oracle for a similar extraction property to the one guaranteed by *algebraic models* such as [22, 32].

In particular, in the proof of extractability, we could actually achieve

$$\Pr[W_2 \mid \neg \mathsf{Bad}_{\mathsf{prog}}] \leq \varepsilon_{\mathsf{dlog}} + \frac{Q_{\mathsf{H}_{\mathsf{ch}}}}{N^r}.$$

We refer to [30] for details on this random oracle construction of a PPRF, and the relative EUF-CMA proof without the quadratic loss.

Multi-instance PPRF In the proof of extractability, the reduction loses a factor of $Q \cdot r \cdot \varepsilon_{\mathsf{PPRF}}$. Using the *multi-instance* PPRF from [10, Theorem 11] we can obtain a practically faster PPRF (since AES in implemented in hardware), while having around the same security loss of number of rounds times independent usages of the PRF.
The analysis is in the ideal cipher model, so we retain the possibility of reading the queries that the adversary makes to construct the PPRF tree.

3.4.2 Multiple keys

In our construction, we can achieve soundness either by increasing the number of parties N, or the number or rounds r, in a way such that $r \log N \geq \lambda$. A classical way to increase the challenge space even more is to use a *vector* of public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_s$ for some parameter s; some protocols, and in particular most of the group action-based ones such as CSI-FiSh, LESS, MEDS, and ALTEQ, allow to generate a single commitment, and then the challenge selects the key on which to respond. This increases the challenge space to $(Ns)^r$, thus allowing the protocol designer to decide on trade-offs between signing efficiency and a larger public key size.

On the adaptor signature side, this is perfectly sound is we want to sell witness of the relation $\mathsf{Rel}_{\mathcal{G}}$ of discrete logarithms. In particular, all the proofs still work with minor modifications; the only non-trivial one is extractability, where we need to reduce to the "multi-target" group action inversion problem. We can thus achieve adaptor functionality with respect to this $\Sigma_{\mathcal{G}}^{(s)}$ signature scheme with respect to the basic relation $\mathsf{Rel}_{\mathcal{G}}$.

However, when using multiple public keys, the language $\mathcal{L}_{\mathsf{skpk}}$ of valid private-public key pairs of the underlying signature scheme changes, and it is not automatic anymore to be able to sell witnesses of this language. We could still achieve the same functionality by using multiple adaptor signatures for the s many components, but embedding a whole secret key inside a single adapted signature gets more challenging.

In particular, trying to extend our approach would lead to choosing s many rounds in which to cheat, and the probability of getting the correct challenge decreases exponentially from $1/N$ to $1/N^s$, thus making pre-signing even slower. We leave the study of an optimized version for efficiently selling witnesses of $\mathcal{L}_{\mathsf{skpk}}$ using $\Sigma_{\mathcal{G}}^{(s)}$ itself in the case of large values of s as an interesting future research direction.

4 Performance Analysis

We recall that the parameters of our scheme are N, the number of parties in the MPCitH protocol, r, the number of parallel repetitions, and s, the number of public keys. We denote throughout the section the size of group elements by ℓ_G, and the size of set elements as ℓ_X. We count the number of action operations as a proxy for running time.

4.1 Analysis of the base signature scheme

We discuss the resulting $\mathsf{KGen}, \mathsf{Sign}$, and Verify algorithms of $\Sigma_{\mathcal{G}}$.

Our parameters are bound to the soundness equation

$$r \cdot (\log N + \log s) \geq \lambda.$$

4.1.1 KeyGen

The key generation algorithm samples s many group elements g_i, and computes all the actions $\mathsf{pk}_i = g_i \star x_0$.

- Runtime: s many group actions.
- Size of secret key: $s \cdot \ell_G$.
- Size of public key: $s \cdot \ell_X$.

4.1.2 Sign

The signing algorithm repeats the following for r many rounds: it expands the PPRF key into N many group elements, samples g_0, and computes g_Δ; then it computes all the elements $x_1, \ldots, x_{N+1}$. It hashes all the commitments from all the rounds, gets a challenge, punctures the PPRF keys and returns (g_0, g_Δ, k^*) for all rounds.

- Runtime: $r \cdot (N + 1)$ many group actions.
- Size of signature: $r \cdot (2\ell_G + \ell_{\mathsf{PPRF}}(N))$

We denote by $\ell_{\mathsf{PPRF}}(N)$ the size of a punctured key of a puncturable PRF with range $\{1, \ldots, N\}$. Following the GGM construction, we can estimate $\ell_{\mathsf{PPRF}}(N) = \lambda \cdot \log N$. Signature size thus becomes

$$2r \cdot \ell_G + \lambda \cdot r \log N = \lambda^2 + r \cdot (2\ell_G - \lambda \log s).$$

We notice here an interesting phenomenon: when $\log s \geq 2\frac{\ell_G}{\lambda}$, the coefficient of r becomes negative, so taking large r *decreases* signature size, while in the other case we want to decrease r as much as possible. Usually, we are in the regime $2\ell_G \gg \lambda \log s$ for feasible choices of s, except for CSI-FiSh, where the extremely small group elements make it so that $s \geq 3$ is enough. Thus, for CSI-Fish and many public keys, the best choice is selecting $N = 2$.

4.1.3 Verify

The verify algorithm re-computes all the actions needed to construct the chains from the punctured PRF. In particular, it has a runtime of

$$r \cdot (N + 1)$$

many group actions, since we are skipping g_{j^*}, but we need to act with g_Δ.

4.2 Analysis of the adaptor algorithm

We now analyze the pSign, pVerify, Adapt, Extract algorithms of our construction.

4.2.1 PreVerify

This algorithm is identical to the Verify algorithm, except when starting from x_T instead of x_0 in the cheating round. Thus its runtime is of $r \cdot (N + 1)$ many group actions.

4.2.2 Adapt

The adapt algorithm consists of a single group operation, which has very small cost compared to the action cost, and returns a normal signature.

4.2.3 Extract

The extract algorithm runs a group inversion, and a group operation; both are negligible. It returns a group element of size ℓ_G.

4.2.4 PreSign

The pre-signing algorithm runs $r-1$ honest rounds, and a single cheating round. The pre-signature size is the same of signature size, up to adding the index, which occupies very little space.
We then analyze the runtime. For the $(r-1)$ honest rounds, it computes $N+1$ group actions each. In the cheating round, we employ the following strategy: first compute all the forward and backward chains $X^{(f)}$ and $X^{(b)}$ from a single PPRF seed k, i.e.

$$X_1^{(f)} = g_1 \star x_T,\ X_j^{(f)} = g_j \star X_{j-1}^{(f)}, \quad X_{N+1}^{(b)} = g_\Delta \star \mathsf{pk},\ X_j^{(b)} = g_j^{-1} \star X_{j+1}^{(b)}.$$

Then it simply re-hashes the chains cut-and-pasted at different indices j until the wanted puncturing index is found

$$\mathsf{H}\left(X_1^{(f)}, \ldots, X_j^{(f)}, X_{j+1}^{(b)}, X_N^{(b)}\right) = j.$$

This is a randomized process, that we can model as finding fixed points of a random function having domain and range $[N]$. This happens with probability

$$\left(1 - \frac{1}{N}\right)^N \approx 1 - e^{-1} \approx 0.63$$

for large N, meaning that we need to re-sample the cheating round (i.e. the PRF key, and recompute the actions) on average $\frac{1}{1-e^{-1}} \approx 1.58$ times. Since in each round we compute $2N+2$ actions, the expected number of actions is $\approx 3.16N + 3$. The expected total number of actions in pre-signing is thus

$$(r + 2.16)N + r + 2.$$

5 Instantiations and Experiments

We instantiate our adaptor signature scheme with the group actions underlying CSI-FiSh, LESS, MEDS and ALTEQ. In this section, we briefly recall these actions, following [9, 19], and report benchmarks for our implementation of the resulting adaptor signature schemes.

Our implementation, written in Python and C++, is agnostic to the underlying group action and supports a modular plug-in interface. Starting from the reference implementations of CSI-FiSh, LESS, MEDS, and ALTEQ, we isolated and abstracted the group-action functionality, and implemented wrappers that allow each action to be used within our generic adaptor-signature framework.
The code is available at `https://github.com/drago-96/GATOR`.
We ran the experiments for MEDS and ALTEQ on an Apple M1 Max chip. For LESS and CSI-FiSh, whose reference implementations include x86 assembly optimizations, we used an Intel Xeon Platinum Basic 8581C CPU @ 2.30GHz.

Figure 8 provides an overview of the experimental results. Each point corresponds to a parameter choice (s, N, r); specifically, we consider $N = 2^1, \ldots, 2^{15}$, and choose the corresponding value of r so that $r \cdot (\log N + \log s) \geq \lambda$. The color of each point indicates the underlying group action, while the public-key size, which is proportional to the parameter s, is represented by the size of the marker.

We also report the full benchmark data for the algorithms $\mathsf{KGen}, \mathsf{Sign}, \mathsf{Verify}, \mathsf{pSign}, \mathsf{pVerify}, \mathsf{Adapt}$, and $\mathsf{Extract}$, including running times, public-key sizes, and pre-signature and signature sizes. All sizes are reported in kB, and all timings are reported in ms.
All data points in the tables and figures are averages over 10 runs.

5.1 CSI-FiSh

CSI-FiSh is based on the CSIDH class-group action. Let p be the CSIDH prime and let $\mathcal{O} = \mathbb{Z}[\pi] \subseteq \mathbb{Q}(\sqrt{-p})$ be the relevant imaginary quadratic order. The *group* is $G_{\text{CSI-FiSh}} = \mathrm{Cl}(\mathcal{O})$ the ideal class group of $\mathcal{O}$. The *set* $X_{\text{CSI-FiSh}}$ is the set of $\mathbb{F}_p$-isomorphism classes of supersingular elliptic curves with $\mathbb{F}_p$-rational endomorphism ring $\mathcal{O}$. In CSIDH notation, these curves are represented by Montgomery coefficients $E_A : y^2 = x^3 + Ax^2 + x$, for $A \in \mathbb{F}_p$. Thus one obtains the action

$$\mathrm{Cl}(\mathcal{O}) \times X_{\text{CSI-FiSh}} \longrightarrow X_{\text{CSI-FiSh}}, \qquad ([\mathfrak{a}], E_A) \longmapsto [\mathfrak{a}] \star E_A.$$

For the CSIDH-512 instance used in CSI-FiSh, the class group structure is known and the class group is cyclic. If $N = |\mathrm{Cl}(\mathcal{O})|$, then one may identify $\mathrm{Cl}(\mathcal{O}) \cong \mathbb{Z}/N\mathbb{Z}$ and write the action as $[a] \star E_A$, for $a \in \mathbb{Z}/N\mathbb{Z}$. The *origin* of this action is E_0, the elliptic curve $y^2 = x^3 + x$.

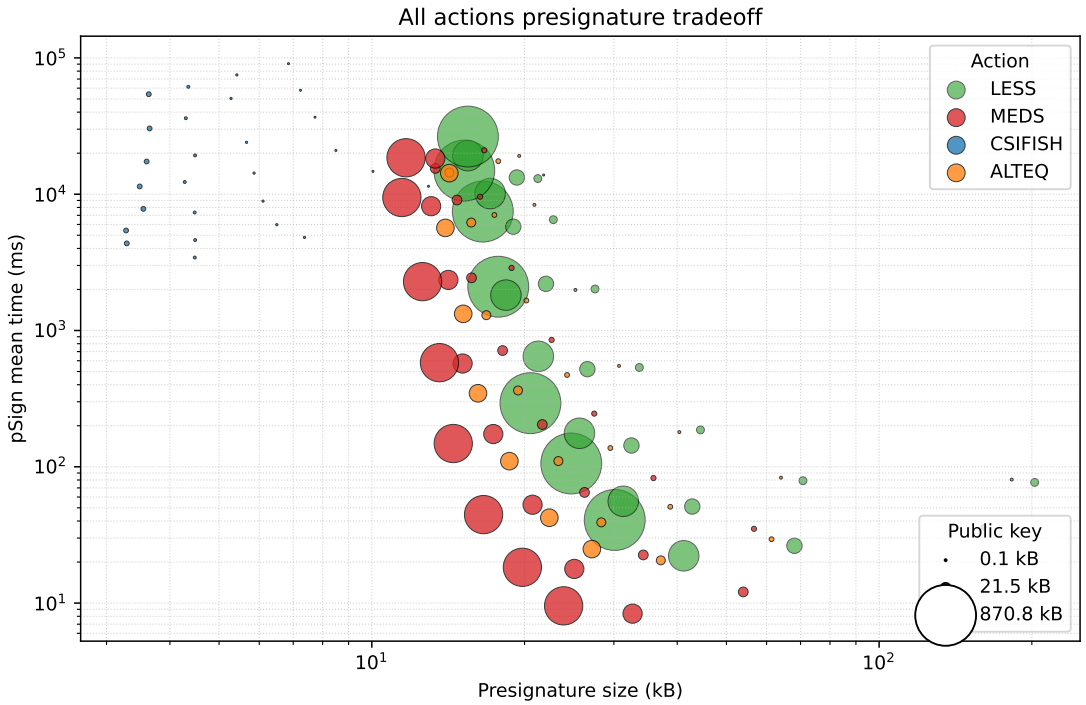


Figure 8: Benchmark results of our implementation with different group actions

Table 1: CSI-FiSh benchmarks

s	N	R	sk	pk	pSig	Sig	KeyGen	pSign	pVer	Adapt	Ext	Sign	Ver
1	8	43	0.0527	0.083	10.1	10	35.7	14372	13768	0.0275	0.0051	13783	13739
2	8	32	0.0928	0.1533	7.5	7.49	72.2	10916	10235	0.0255	0.0049	10305	10242
4	8	26	0.1729	0.2939	6.1	6.09	142	9040	8325	0.0254	0.005	8450	8324
8	32	16	0.333	0.5752	4.78	4.77	284	21447	18833	0.0238	0.0051	19112	18823
16	16	16	0.6533	1.14	4.28	4.27	570	11493	9691	0.0243	0.0052	10276	9695
32	2	22	1.29	2.26	3.8	3.79	1140	3671	2360	0.0254	0.0055	3507	2358
64	2	19	2.58	4.51	3.29	3.28	2277	4550	2033	0.0233	0.0054	4311	2026

The underlying hard problem is the Group Action Inversion Problem for the CSIDH action: given $E_0, E_1 = [a] \star E_0$, find a class-group element $a \in \mathbb{Z}/N\mathbb{Z}$ such that

$$E_1 = [a] \star E_0.$$

The concrete CSI-FiSh instantiation uses the CSIDH-512 parameter set. This has $p \approx 2^{512}$, uses 74 small primes in the CSIDH action, and in the original CSIDH sampling uses exponent vectors in $[-5, 5]^{74}$. For CSI-FiSh, the computed class group has order $N \approx 2^{257.136}$. Thus a canonical group element can be represented as an integer modulo N, so

$$\ell_G = \lceil \log_2 N \rceil \approx 258 \text{ bits} \approx 33 \text{ bytes}.$$

A set element is an isomorphism class of a CSIDH curve, represented by one Montgomery coefficient $A \in \mathbb{F}_p$. Hence

$$\ell_X = \lceil \log_2 p \rceil = 512 \text{ bits} = 64 \text{ bytes}.$$

5.1.1 Experimental results

We report a full benchmark on selected CSI-FiSh parameters in Table 1. As already noticed before, it is interesting to see that for bigger values of s, the signing time drops, since the coefficient of r becomes negative.

Table 2: LESS benchmarks.

s	N	R	sk	pk	pSig	Sig	KeyGen	pSign	pVer	Adapt	Ext	Sign	Ver
1	16	32	0.7588	13.6	53.7	53.7	0.1737	118	103	0.0176	0.004	108	102
2	32	22	1.5	27.2	37.6	37.6	0.3444	170	137	0.0174	0.0042	147	136
4	64	16	3	54.4	27.9	27.9	0.684	271	200	0.0191	0.0042	207	199
8	32	16	5.98	109	27.4	27.4	1.35	152	101	0.0162	0.0041	106	101
16	16	16	12	218	26.9	26.9	2.7	96.7	53	0.0115	0.0034	57.4	52.6
32	32	13	23.9	435	22.3	22.2	5.37	232	85.3	0.0141	0.0039	93.5	84.7
64	16	13	47.8	871	21.8	21.8	10.7	173	49.8	0.0132	0.0038	61.5	49.4

5.2 LESS

LESS is based on the action of the monomial group on linear codes. The *group* is

$$G_{\mathrm{LESS}} = \{DP_\pi : D \in (\mathbb{F}_q^\times)^n,\ \pi \in S_n\} \cong (\mathbb{F}_q^\times)^n \rtimes S_n$$

the monomial group, i.e. matrices where exactly one element in each row and column is non-zero. The *set* is

$$X_{\mathrm{LESS}} = \mathrm{Gr}_k(\mathbb{F}_q^n),$$

whose elements are linear $[n, k]$-codes over $\mathbb{F}_q$. We use the right-action convention

$$\mathcal{C} \cdot Q = \{cQ : c \in \mathcal{C}\}, \qquad \mathcal{C} \in \mathrm{Gr}_k(\mathbb{F}_q^n), \quad Q \in \mathsf{Mon}_n(q).$$

In the concrete scheme, codes are represented by generator matrices in standard or reduced-row echelon form. Thus the action is implemented as

$$H \star Q = \mathsf{SF}(HQ),$$

where SF denotes the chosen systematic/canonical representative of the row space, and H is any generator matix of the code $\mathcal{C}$.
The underlying hard problem is the Linear Equivalence Problem: given two generator matrices $G_0, G_1 \in \mathbb{F}_q^{k\times n}$, find a monomial matrix $Q \in \mathsf{Mon}_n(q)$ such that

$$\mathrm{Row}(G_1) = \mathrm{Row}(G_0Q), \quad \text{or equivalently} \quad G_1 = \mathsf{SF}(G_0Q).$$

For the NIST category-I LESS parameter sets used in the specifications,

$$q = 127, \qquad n = 252, \qquad k = 126.$$

A full monomial element can be represented using

$$n\big(\lceil \log_2 n \rceil + \lceil \log_2(q-1) \rceil\big)$$

bits. With the information-set optimisation used in the LESS accounting, the transmitted response group element has size

$$\ell_G = k\big(\lceil \log_2 n \rceil + \lceil \log_2(q-1) \rceil\big) = 126(8+7) \approx 237 \text{ bytes}.$$

A systematic generator matrix for a code has $k(n-k)$ free field elements, so

$$\ell_X = k(n-k)\lceil \log_2 q \rceil = 126 \cdot 126 \cdot 7 \approx 13892 \text{ bytes}.$$

5.2.1 Experimental results

We report a full benchmark on selected LESS parameters in Table 2. We observe that our group element serialization returns 756 bytes, instead of the expected 237, since we do not us the information-set optimisation.

Table 3: MEDS benchmarks

s	N	R	sk	pk	pSig	Sig	KeyGen	pSign	pVer	Adapt	Ext	Sign	Ver
1	32	26	0.595	5.38	35.9	35.9	0.078	84.2	70.6	0.021	0.014	78.4	68.8
2	128	16	1.18	10.7	23.1	23.1	0.153	214	173	0.025	0.018	193	172
4	64	16	2.34	21.5	22.6	22.6	0.297	109	85.4	0.02	0.11	95.8	85.6
8	32	16	4.67	43	22.1	22.1	0.593	54.2	42.6	0.016	0.014	48.5	42.7
16	64	13	9.33	85.9	18.4	18.4	1.17	92.6	69.2	0.018	0.015	78.8	69.4
32	64	12	18.6	172	17	17	2.34	90.3	64.2	0.018	0.015	73.6	64.1
64	128	10	37.3	344	14.5	14.5	4.71	170	107	0.039	0.024	123	107

5.3 MEDS

MEDS is based on matrix-code equivalence. Consider the vector space $V = \mathbb{F}_q^{n\times m}$. The *set* is

$$X_{\text{MEDS}} = \text{Gr}_k(V),$$

the set of k-dimensional matrix codes in V.
The acting *group* is

$$G_{\text{MEDS}} = GL_n(q) \times GL_m(q).$$

At the matrix level, an element $(A, B) \in GL_n(q) \times GL_m(q)$ maps $M \longmapsto AMB$. The induced action on matrix codes is $\mathcal{C} \cdot (A, B) = \{AMB : M \in \mathcal{C}\}$. After vectorising $n \times m$ matrices, a matrix code is represented by a generator matrix $H \in \mathbb{F}_q^{k\times nm}$. With the MEDS convention, the implemented action is

$$(A, B) \star H = \mathsf{SF}\big(H(A^T \otimes B)\big),$$

where SF chooses the standard representative of the row space.
The underlying hard problem is Matrix Code Equivalence. Given two bases

$$C_1, \ldots, C_k, \qquad D_1, \ldots, D_k \in \mathbb{F}_q^{n\times m},$$

find $A \in GL_n(q)$, $B \in GL_m(q)$, and a basis-change matrix $T = (t_{ij}) \in GL_k(q)$ such that

$$D_i = \sum_{j=1}^{k} t_{ij} AC_jB \qquad \text{for all } i = 1, \ldots, k.$$

For the category-I MEDS parameter sets,

$$q = 4093, \qquad n = m = k = 14.$$

A group element $(A, B) \in GL_n(q) \times GL_m(q)$ is represented by two matrices, hence

$$\ell_G = (n^2 + m^2)\lceil \log_2 q \rceil = (14^2 + 14^2) \cdot 12 = 588 \text{ bytes.}$$

A raw standard-form generator matrix for a k-dimensional subspace of $\mathbb{F}_q^{nm}$ has $k(nm - k)$ free field elements, giving

$$\ell_X^{\text{raw}} = k(nm - k)\lceil \log_2 q \rceil = 14(14 \cdot 14 - 14) \cdot 12 = 3822 \text{ bytes.}$$

The MEDS specification uses an additional public-key compression, reducing the concrete category-I set-element accounting to approximately

$$\ell_X^{\text{spec}} = 3297 \text{ bytes.}$$

5.3.1 Experimental results

We report a full benchmark on selected MEDS parameters in Table 3. We observe that our serialized group elements take 784 bytes instead of the spec's 588, since the chosen representation in systematic form could potentially not exist, because we fall outside the chart of the Grassmanian. Using the simple RREF always works, even though we pay in bigger serialized group elements.

Table 4: ALTEQ benchmarks

s	N	R	sk	pk	pSig	Sig	KeyGen	pSign	pVer	Adapt	Ext	Sign	Ver
1	32	26	0.6807	1.14	40.4	40.4	0.2395	189	300	0.0975	0.0046	179	307
2	32	22	1.35	2.26	34.2	34.2	0.4689	165	254	0.0953	0.0046	152	260
4	64	16	2.68	4.51	25.4	25.4	0.9172	238	362	0.1221	0.005	220	374
8	32	16	5.36	9.01	24.9	24.9	1.81	123	182	0.0669	0.0046	111	188
16	64	13	10.7	18	20.6	20.6	3.62	205	293	0.1578	0.0049	182	305
32	32	13	21.4	36	20.2	20.2	7.2	106	147	0.0622	0.0067	95.2	153
64	128	10	42.8	72	16.2	16.2	14.4	334	438	0.1467	0.0046	287	462

5.4 ALTEQ

ALTEQ is based on equivalence of alternating trilinear forms. The *set* is

$$X_{\mathrm{ALTEQ}} = \mathrm{ATF}(n, q) = \bigwedge^{3}(\mathbb{F}_q^n)^*,$$

the set of all alternating trilinear maps $\varphi : \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q$. The acting *group* is

$$G_{\mathrm{ALTEQ}} = GL_n(q).$$

Using the convention from the specifications, $A \in GL_n(q)$ acts by change of basis:

$$\varphi \star A = \varphi \circ A, \quad \text{where} \quad (\varphi \circ A)(x, y, z) = \varphi(A^T x, A^T y, A^T z).$$

The underlying hard problem is Alternating Trilinear Form Equivalence. Given two equivalent alternating trilinear forms $\varphi, \psi \in \mathrm{ATF}(n, q)$, find $A \in GL_n(q)$ such that $\psi = \varphi \cdot A$.
For the category-I ALTEQ parameter sets,

$$q = 2^{32} - 5, \qquad n = 13.$$

A group element $A \in GL_n(q)$ is represented by an $n \times n$ matrix, so

$$\ell_G = n^2 \lceil \log_2 q \rceil = 13^2 \cdot 32 = 676 \text{ bytes}.$$

An alternating trilinear form is determined by its coefficients on basis wedges $e_i^* \wedge e_j^* \wedge e_k^*$, $1 \le i < j < k \le n$, so

$$\ell_X = \binom{n}{3} \lceil \log_2 q \rceil = 286 \cdot 32 = 1144 \text{ bytes}.$$

5.4.1 Experimental results

We report a full benchmark on selected ALTEQ parameters in Table 4.

6 Conclusions

Our work demonstrates practical efficiency of group action-based adaptor signatures; in particular, our construction does not incur in significative increases in signing time or signature size with respect to the baseline protocols LESS, MEDS, and ALTEQ. For CSI-FiSh, given the natively small group elements, the puncturing technique of the MPCitH approach is instead less effective.

We stress that we create a signature scheme $\Sigma_{\mathcal{G}}$ and enhance it with adaptor functionality for the group action discrete logarithm $\mathsf{Rel}_{\mathcal{G}}$. Using parameters where $s = 1$, this is actually the secret-public key relation $\mathcal{L}_{\mathsf{skpk}}$ of the scheme $\Sigma_{\mathcal{G}}$ itself. Achieving an efficient construction where we can sell witnesses for the $\mathcal{L}_{\mathsf{skpk}}$ language even in parameter regimes where $s > 1$ is left as an interesting future work.

References

[1] Alamati, N., De Feo, L., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 411–439. Springer, Cham (Dec 2020). 10.1007/978-3-030-64834-3_14

[2] Aumayr, L., Ersoy, O., Erwig, A., Faust, S., Hostáková, K., Maffei, M., Moreno-Sanchez, P., Riahi, S.: Generalized channels from limited blockchain scripts and adaptor signatures. In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part II. LNCS, vol. 13091, pp. 635–664. Springer, Cham (Dec 2021). 10.1007/978-3-030-92075-3_22

[3] Baldi, M., Barenghi, A., Battagliola, M., Bitzer, S., Gianvecchio, M., Karl, P., Manganiello, F., Pavoni, A., Pelosi, G., Pintore, F., Santini, P., Schupp, J., Signorini, E., Slaughter, F., Wachter-Zeh, A., Weger, V.: LESS: Linear equivalence signature scheme (2025), https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-2/spec-files/less-spec-round2-web.pdf, nIST Additional Digital Signatures Round 2 submission

[4] Barenghi, A., Biasse, J.F., Persichetti, E., Santini, P.: LESS-FM: Fine-tuning signatures from the code equivalence problem. In: Cheon, J.H., Tillich, J.P. (eds.) PQCrypto 2021. pp. 23–43. Springer, Cham (2021). 10.1007/978-3-030-81293-5_2

[5] Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: Efficient isogeny based signatures through class group computations. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 227–247. Springer, Cham (Dec 2019). 10.1007/978-3-030-34578-5_9

[6] Biasse, J.F., Micheli, G., Persichetti, E., Santini, P.: LESS is more: Code-based signatures without syndromes. In: Nitaj, A., Youssef, A.M. (eds.) AFRICACRYPT 20. LNCS, vol. 12174, pp. 45–65. Springer, Cham (Jul 2020). 10.1007/978-3-030-51938-4_3

[7] Bläser, M., Duong, D.H., Narayanan, A.K., Plantard, T., Qiao, Y., Sipasseuth, A., Tang, G.: The ALTEQ Signature Scheme: Algorithm Specifications and Supporting Documentation. NIST Post-Quantum Additional Digital Signature Schemes Submission (2023), https://pqcalteq.github.io/ALTEQ_spec_2024.03.05.pdf, document version 2024-03-05.2

[8] Boneh, D., Waters, B.: Constrained pseudorandom functions and their applications. In: Sako, K., Sarkar, P. (eds.) ASIACRYPT 2013, Part II. LNCS, vol. 8270, pp. 280–300. Springer, Berlin, Heidelberg (Dec 2013). 10.1007/978-3-642-42045-0_15

[9] Borin, G., Persichetti, E., Santini, P., Pintore, F., Reijnders, K.: A guide to the design of digital signatures based on cryptographic group actions. Cryptology ePrint Archive, Report 2023/718 (2023), https://eprint.iacr.org/2023/718

[10] Bui, D., Carozza, E., Couteau, G., Goudarzi, D., Joux, A.: Faster signatures from MPC-in-the-head. In: Chung, K.M., Sasaki, Y. (eds.) ASIACRYPT 2024, Part I. LNCS, vol. 15484, pp. 396–428. Springer, Singapore (Dec 2024). 10.1007/978-981-96-0875-1_13

[11] Cartor, R., Daly, N., Gaggero, G., LeGrow, J.T., Sanguineti, A., Sconza, S.: Post-quantum adaptor signatures with strong security from cryptographic group actions. In: Post-Quantum Cryptography: 17th International Workshop, PQCrypto 2026, Saint-Malo, France, April 14–16, 2026, Proceedings, Part I. p. 370–405. Springer-Verlag, Berlin, Heidelberg (2026). 10.1007/978-3-032-22695-2_13

[12] Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. Cryptology ePrint Archive, Report 2018/383 (2018), https://eprint.iacr.org/2018/383

[13] Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 395–427. Springer, Cham (Dec 2018). 10.1007/978-3-030-03332-3_15

[14] Chou, T., Niederhagen, R., Persichetti, E., Ran, L., Randrianarisoa, T.H., Reijnders, K., Samardjiska, S., Trimoska, M.: MEDS: Matrix equivalence digital signature (2023), https://www.meds-pqc.org/spec/MEDS-2023-07-26.pdf, nIST Additional Digital Signatures Round 1 submission

[15] Chou, T., Niederhagen, R., Persichetti, E., Randrianarisoa, T.H., Reijnders, K., Samardjiska, S., Trimoska, M.: Take your MEDS: Digital signatures from matrix code equivalence. In: El Mrabet, N., De Feo, L., Duquesne, S. (eds.) AFRICACRYPT 23. LNCS, vol. 14064, pp. 28–52. Springer, Cham (Jul 2023). 10.1007/978-3-031-37679-5_2

[16] Ciampi, M., Liu, X., Tzannetos, I., Zikas, V.: Universal adaptor signatures from blackbox multi-party computation. In: Patra, A. (ed.) CT-RSA 2025. LNCS, vol. 15598, pp. 375–398. Springer, Cham (Apr / May 2025). 10.1007/978-3-031-88661-4_16

[17] Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006), https://eprint.iacr.org/2006/291

[18] Dai, W., Okamoto, T., Yamamoto, G.: Stronger security and generic constructions for adaptor signatures. In: Isobe, T., Sarkar, S. (eds.) INDOCRYPT 2022. LNCS, vol. 13774, pp. 52–77. Springer, Cham (Dec 2022). 10.1007/978-3-031-22912-1_3

[19] D'Alconzo, G., Flamini, A., Meneghetti, A., Signorini, E.: A framework for group action-based multi-signatures and applications to LESS, MEDS, and ALTEQ. In: Jager, T., Pan, J. (eds.) PKC 2025, Part II. LNCS, vol. 15675, pp. 99–133. Springer, Cham (May 2025). 10.1007/978-3-031-91823-0_4

[20] De Feo, L., Galbraith, S.D.: SeaSign: Compact isogeny signatures from class group actions. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 759–789. Springer, Cham (May 2019). 10.1007/978-3-030-17659-4_26

[21] Diffie, W., Hellman, M.E.: New directions in cryptography. IEEE Transactions on Information Theory **22**(6), 644–654 (1976). 10.1109/TIT.1976.1055638

[22] Duman, J., Hartmann, D., Kiltz, E., Kunzweiler, S., Lehmann, J., Riepel, D.: Generic models for group actions. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part I. LNCS, vol. 13940, pp. 406–435. Springer, Cham (May 2023). 10.1007/978-3-031-31368-4_15

[23] Esgin, M.F., Ersoy, O., Erkin, Z.: Post-quantum adaptor signatures and payment channel networks. In: Chen, L., Li, N., Liang, K., Schneider, S.A. (eds.) ESORICS 2020, Part II. LNCS, vol. 12309, pp. 378–397. Springer, Cham (Sep 2020). 10.1007/978-3-030-59013-0_19

[24] Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg (Aug 1987). 10.1007/3-540-47721-7_12

[25] Gerhart, P., Schröder, D., Soni, P., Thyagarajan, S.A.K.: Foundations of adaptor signatures. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part II. LNCS, vol. 14652, pp. 161–189. Springer, Cham (May 2024). 10.1007/978-3-031-58723-8_6

[26] Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions (extended abstract). In: 25th FOCS. pp. 464–479. IEEE Computer Society Press (Oct 1984). 10.1109/SFCS.1984.715949

[27] Goldreich, O., Goldwasser, S., Micali, S.: How to construct random functions. Journal of the ACM **33**(4), 792–807 (Oct 1986). 10.1145/6490.6503

[28] Goldwasser, S., Micali, S., Rivest, R.L.: A digital signature scheme secure against adaptive chosen-message attacks. SIAM Journal on Computing **17**(2), 281–308 (Apr 1988)

[29] Jana, P., Shaw, S., Dutta, R.: Compact adaptor signature from isogenies with enhanced security. In: Kohlweiss, M., Di Pietro, R., Beresford, A.R. (eds.) CANS 2024, Part I. LNCS, vol. 14905, pp. 77–100. Springer, Singapore (Sep 2024). 10.1007/978-981-97-8013-6_4

[30] Joux, A.: MPC in the head for isomorphisms and group actions. Cryptology ePrint Archive, Report 2023/664 (2023), https://eprint.iacr.org/2023/664

[31] Liu, X., Tzannetos, I., Zikas, V.: Adaptor signatures: New security definition and a generic construction for NP relations. In: Chung, K.M., Sasaki, Y. (eds.) ASIACRYPT 2024, Part II. LNCS, vol. 15485, pp. 168–193. Springer, Singapore (Dec 2024). 10.1007/978-981-96-0888-1_6

[32] Orsini, E., Zanotto, R.: Simple two-message OT in the explicit isogeny model. CiC **1**(1), 15 (2024). 10.62056/a39qgy4e-

[33] Renan, F., Kutas, P.: SQIAsignHD: SQIsignHD adaptor signature. Cryptology ePrint Archive, Report 2024/561 (2024), https://eprint.iacr.org/2024/561

[34] Rostovtsev, A., Stolbunov, A.: Public-Key Cryptosystem Based On Isogenies. Cryptology ePrint Archive, Report 2006/145 (2006), https://eprint.iacr.org/2006/145

[35] Sahai, A., Waters, B.: How to use indistinguishability obfuscation: deniable encryption, and more. In: Shmoys, D.B. (ed.) 46th ACM STOC. pp. 475–484. ACM Press (May / Jun 2014). 10.1145/2591796.2591825

[36] Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: Proceedings 35th Annual Symposium on Foundations of Computer Science. pp. 124–134. IEEE Computer Society (1994). 10.1109/SFCS.1994.365700

[37] Tairi, E., Moreno-Sanchez, P., Maffei, M.: Post-quantum adaptor signature for privacy-preserving off-chain payments. In: Borisov, N., Díaz, C. (eds.) FC 2021, Part II. LNCS, vol. 12675, pp. 131–150. Springer, Berlin, Heidelberg (Mar 2021). 10.1007/978-3-662-64331-0_7

[38] Tairi, E., Moreno-Sanchez, P., Schneidewind, C.: LedgerLocks: A security framework for blockchain protocols based on adaptor signatures. In: Meng, W., Jensen, C.D., Cremers, C., Kirda, E. (eds.) ACM CCS 2023. pp. 859–873. ACM Press (Nov 2023). 10.1145/3576915.3623149

[39] Tang, G., Duong, D.H., Joux, A., Plantard, T., Qiao, Y., Susilo, W.: Practical post-quantum signature schemes from isomorphism problems of trilinear forms. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 582–612. Springer, Cham (May / Jun 2022). 10.1007/978-3-031-07082-2_21

[40] Thyagarajan, S.A.K., Malavolta, G.: Lockable signatures for blockchains: Scriptless scripts for all signatures. In: 2021 IEEE Symposium on Security and Privacy. pp. 937–954. IEEE Computer Society Press (May 2021). 10.1109/SP40001.2021.00065