Selectively Linkable Group Signatures — Stronger Security and Preserved Verifiability

Ashley Fraser[1] *, Lydia Garms[23] **, and Anja Lehmann[4]

[1] University of Surrey, a.fraser@surrey.ac.uk
[2] Royal Holloway, University of London
[3] IMDEA Software Institute, lydia.garms@imdea.org
[4] Hasso-Plattner-Institute, University of Potsdam, anja.lehmann@hpi.de

Abstract. Group signatures allow group members to sign on behalf of the group anonymously. They are therefore well suited to storing data in a way that preserves the users' privacy, while guaranteeing its authenticity. Garms and Lehmann (PKC'19) introduced a new type of group signatures that balance privacy with utility by allowing to selectively link subsets of the group signatures via an oblivious entity, the converter. The conversion takes a batch of group signatures and blindly transforms signatures originating from the same user into a consistent representation. Their scheme essentially targets a setting where the entity receiving fully unlinkable signatures and the converted ones is the same: only pseudonyms but not full signatures are converted, and the input to the converter is assumed to be well-formed. Thus, the converted outputs are merely linkable pseudonyms but no longer signatures.
In this work we extend and strengthen such convertibly linkable group signatures. Conversion can now be triggered by malicious entities too, and the converted outputs can be publicly verified. This preserves the authentication of data during the conversion process. We define the security of this scheme and give a provably secure instantiation. Our scheme makes use of controlled-malleable NIZKs, which allow proofs to be mauled in a controlled manner. This allows signatures to be blinded, while still ensuring they can be verified during conversions.

1 Introduction

Group signatures allow members of a group to sign messages anonymously [18, 4, 7, 13, 29, 36, 8]. A valid group signature attests that it was signed by a group member, without revealing the signer's identity, or whether signatures stem from the same user. They are therefore useful when data is collected that should be

* The author was supported by the EPSRC Next Stage Digital Economy Centre in the Decentralised Digital Economy (DECaDE) under grant number EP/T022485/1.
** The author was supported by the EPSRC and the UK government as part of the Centre for Doctoral Training in Cyber Security at Royal Holloway, University of London (EP/K035584/1) and by the InnovateUK funded project AQuaSec, as well as by a research grant from Nomadic Labs and the Tezos Foundation.

authenticated while preserving the privacy of the data sources. However, full anonymity might be undesirable. It may be necessary to know the correlation among *some* data events. For instance, several high value blood pressure measurements might not be critical unless they originate from a single user.

To address the balance between privacy and utility, several linkability mechanisms have been introduced. Standard group signatures [18, 7, 4, 5, 8, 36] have an opening mechanism, that allows a trusted opener to de-anonymise signatures. Less privacy invasive forms exist, where the opener no longer fully de-anonymises users, but merely tests whether two signatures stem from the same one [31, 32, 39, 33, 20]. Another line of work avoids the trusted entity for opening and rather supports *pseudonymous* group signatures where users can choose to sign either with a fresh, and unlinkable pseudonym or re-use an established one, making all signatures under the same pseudonym publicly linkable [9, 14, 6, 11, 10, 24, 23].

Group Signatures with Selective Linkability. Garms and Lehmann [27] argue that none of these schemes provides the flexibility and privacy needed in practice, as they either require the user to decide upon the desired linkability when signatures are computed, or the usage of the linkability gradually erodes the users' privacy.

To overcome these limitations, they proposed a more flexible variant of linkability in the form of the CLS scheme [27]. While all group signatures therein are fully unlinkable per default, i.e. each pseudonym is fresh, certain subsets can be converted into a linked representation. The conversion is performed obliviously by a trusted converter that blindly transforms a batch of pseudonyms, mapping different pseudonyms stemming from the same user into the same one. To avoid the erosion of privacy, due to a user's signatures gradually being linked together as a result of successive convert queries, converted pseudonyms obtained through *different* queries remain unlinkable, i.e., conversions are strictly non-transitive.

Trusted Data Lakes and Data Processors. However, [27] assumes the party receiving/ verifying fully unlinkable signatures (the data lake) and the one obtaining the converted linked ones (the data processor) to be the same entity, or belong to the same trust domain. In their scheme the data lake only inputs pseudonyms to the converter but not the actual signatures, the authenticity of data gets lost in the conversion process. Therefore, a data processor only receives converted pseudonyms from the converter and must trust the data lake that converted data originates from actual user data. It also assumes inputs from the data lake to the converter to be well-formed. The security guarantees only hold when "valid" pseudonyms are converted for which correct group signatures exist. As the converter in [27] receives *blinded* pseudonyms and no signatures, this assumption is impossible to enforce other than by considering honest requests only.

For many applications, this assumption may not be realisable and security from a malicious data lake is vital. For example, in the US the Regional Health Information Organization (a data processor) has the role of integrating the medical records of many hospitals. In practise, this data is often stored by a third party (a data lake) that is not trusted by the hospitals or research organisations that process the data [19]. As discussed above, correlation of data by user can provide additional insights to the data processor in terms of medical analysis.

However, to protect the user's privacy, correlated data should not be revealed to the data lake. Additionally, the data processor will want assurances that the data processed is authentic user data. Also, Google (a data lake) has been collecting anonymised location data during the COVID-19 pandemic to monitor social distancing [1]. In this case the correlation of data by user is valuable for analysis, such as to gain insights into how behaviour changes based on restrictions. However, this comes with a threat to the user's privacy. Ideally, data should be stored anonymously to preserve the users' privacy, but data processors, such as a governmental public health organisation, might be allowed to request to blindly link the data, for example, to provide insight into distances travelled which is only possible if data is correlated by user. Such data processors may not fully trust Google as a data lake. Therefore, the data processor may want assurance that security and privacy holds even if the data lake acts maliciously.

Our Contributions. In this work we strengthen the concept of convertibly linkable group signatures (CLS+) to capture the scenario that the data lake and data processor do not belong to the same trust domain. That is, we guarantee the desired security even when conversion is triggered by malicious data lakes. Further, we leverage the trusted converter to not only blindly transform the pseudonyms but also blindly re-authenticate the associated messages, preserving the authenticity of the data during conversions. We start by lifting the security and privacy definition given in [27] to this stronger setting. Our security model grants the adversary the power to request conversions of arbitrary and blinded inputs. We propose a construction that provably satisfies the desired properties.

Our CLS+ Construction. In the CLS+ model, an issuer joins users to the group. A data lake holds a set of users' (message, pseudonym, signature) tuples. They blind a subset of these to the converter for conversion with respect to the intended data processor's public key. The resulting *converted* (message, pseudonym, signature) tuples are output to the data processor for unblinding. After unblinding the pseudonyms should be consistent, i.e. pseudonyms from the same conversion and from the same user should be the same. As conversions should be non-transitive, pseudonyms should be unlinkable by user across conversion queries.

A user's and issuer's key pair is that of an automorphic signature scheme [25], a structure–preserving signature scheme [2] where the verification keys lie within the message space. When joining the group, the issuer signs the user's verification key, yielding the user's membership credential. During signing, the user encrypts its verification key under an encryption public key held by the converter to form the pseudonym, and proves knowledge of a valid automorphic signature on the message with respect to this verification key, and of a valid credential for this key. In order to blind signatures yet still allow verification during conversions, we use zero-knowledge proofs (NIZKs) that are controlled malleable [16], which can be realised with Groth-Sahai proofs [30]. This allows to encrypt the pseudonym and message under an encryption public key held by the data processor (the blinding public key), and transform the proof accordingly. Because the malleability is controlled, this does not affect the unforgeability of the signatures.

The converter then decrypts the pseudonyms using the converter's encryption secret key. A nested form of ElGamal encryption allows us to remove layers of encryption under different public keys in any order. The resulting pseudonym is re-randomised under the blinding public key and then transformed under a random value r. This value is chosen fresh for every conversion query to ensure non-transitive conversions, but re-used within this query to ensure the resulting pseudonyms are consistent. The converter signs the converted pseudonym and message with the signing key for a standard digital signature scheme to attest that they originate from a valid query containing verifiable signatures. As we assume the converter is at most honest-but-curious, the authentication of converted signatures is carried over from that of the blinded signatures. During unblinding, the data processor decrypts the message and converted pseudonym under the blinding secret key. The final pseudonyms will be consistent, pseudonyms from the same conversion and authored by the same user will be the same. The original output of the converter is included in the signature, along with a proof of correct unblinding. This carries forward the authentication during unblinding. We prove that our construction is secure, assuming the security of the automorphic signatures, standard digital signatures and the controlled malleable NIZKs, as well as the SXDH assumption.

Other Related Work. In [34] group signatures can be converted into standard signatures, but all of a user's signatures are de-anonymised. In [35, 37], the power of the opener is reduced. In [35], they avoid the need for an opener, by allowing users to prove or deny authorship of a signature. The opener can also prove that two signatures originate from the same user without revealing user identities. In [37], another entity is introduced, the admitter. They have the power to specify messages, so that only signatures on those messages can be opened.

2 Syntax and Security Model for $\mathsf{CLS+}$

We define the syntax and security model for $\mathsf{CLS+}$ signatures, an extension of the CLS model [27] that no longer requires the assumption that conversion of signatures is triggered by *honest* verifiers. Whereas CLS only converts pseudonyms, our $\mathsf{CLS+}$ scheme preserves the validity of the associated signatures.

As in CLS, our $\mathsf{CLS+}$ scheme assumes an issuer $\mathcal{I}$, a set of users $\mathcal{U} = \{uid_i\}$, and a converter $\mathcal{C}$. The issuer $\mathcal{I}$ joins new users to the group, who can sign pseudonymously. While signatures are fully unlinkable by default, they can be linked in a controlled manner by the converter $\mathcal{C}$, who blindly converts pseudonym-message-signature tuples into a consistent (now) authenticated form.

In contrast to CLS, we split the verifier role into a data lake $\mathcal{L}$ and data processor $\mathcal{P}$. The data lake (or *any* verifier) can collect and verify the unlinkable signatures w.r.t the group's public key. Additionally, the data lake can request conversions by blinding signatures for a particular data processor $\mathcal{P}$ in a conversion request to the converter. The data processor can ***unblind*** and verify the converted signatures output by the converter. Once unblinded, *any* verifier can check the validity of the converted signature. In this way, we capture the setting where a data processor (in a separate trust domain from the data lake) can verify converted group signatures, whereas, CLS assumed that unlinkable and converted signatures are used by the same entity (or within the same trust domain). Any verifier can take the data lake role, as there are no dedicated keys.

2.1 Syntax of CLS+

We closely follow the notation from the framework for CLS [27], but extend the blinding, conversion and unblinding procedures to transform signatures, as well as pseudonyms. Verification is extended to also handle transformed and linkable signatures. More precisely, a convertibly linkable group signature scheme with preserved verifiability CLS+ consists of the following algorithms:

Setup & Key Generation. Each central entity generates their individual key pair.

CLS+.Setup$(1^\tau) \rightarrow \mathsf{pp}$**:** on input a security parameter, outputs the public parameters pp.

CLS+.IKGen$(\mathsf{pp}) \rightarrow (ipk, isk)$**:** performed by the issuer $\mathcal{I}$, outputs the issuer secret key isk, and the issuer public key ipk.

CLS+.CKGen$(\mathsf{pp}) \rightarrow (cpk, csk)$**:** performed by the converter $\mathcal{C}$, outputs the converter secret key csk, and the converter public key cpk.

CLS+.BKGen$(\mathsf{pp}) \rightarrow (bpk, bsk)$**:** performed by the data processor $\mathcal{P}$, outputs the blinding public key bpk and blinding private key bsk.

We write the group public key gpk to refer to (pp, ipk, cpk) and $\mathcal{BK}$ to denote the public/private key space induced by CLS+.BKGen.

Join, Sign & Verify. A user must join the group via an interactive protocol with the issuer, as is standard in group signatures [5]. Our construction requires that the user already specifies the data processor's key bpk when creating signatures, and thus we reflect this in the syntax. While this limits the flexibility of the data lake (it has to adhere to the choice of the user) it gives the users strong control over the usage of their data, as only they can choose who can unblind the converted (linkable) signatures. Signers still do not need to decide which data should be linked, but only which data processors they trust to process their data.

To handle our setting where converted signatures can be verified too, CLS+.Verify takes as input $type = \{\texttt{tier-}1, \texttt{tier-}2\}$ that indicates the type of signature. We denote standard, fully unlinkable signatures produced by CLS+.Sign as tier-1 signatures, and converted ones (from processing a tier-1 signature with CLS+.Blind − CLS+.Convert − CLS+.Unblind introduced below) as tier-2 signatures.

$\langle$**CLS+.Join**(gpk), **CLS+.Issue**$(isk, gpk)\rangle$**:** an interactive protocol performed by the joining user uid and the issuer $\mathcal{I}$, who perform CLS+.Join and CLS+.Issue respectively. If successful, CLS+.Join outputs $\mathbf{gsk}[uid]$. During the protocol, each algorithm inputs a state and an incoming message, and outputs an updated state, an outgoing message, and a decision cont/ accept/ reject.

CLS+.Sign$(gpk, bpk, \mathbf{gsk}[uid], m) \to (\mu, \sigma)$**:** performed by user uid for a data processor with key bpk, outputs pseudonym μ and signature σ. For ease of expression we treat the pseudonym μ as a dedicated part of the signature.

CLS+.Verify$(type, gpk, bpk, m, \mu, \sigma) \to \{0, 1\}$**:** performed by the data lake (or any verifier), outputs 1 if σ is a valid $\{\texttt{tier-}1, \texttt{tier-}2\}$-signature.

Blind Conversion. As in the CLS model, to allow for blind conversions of signatures, there are the CLS+.Blind, CLS+.Convert and CLS+.Unblind algorithms for the data lake, converter and data processor respectively. The CLS+.Convert algorithm, on input blinded unlinkable pseudonyms, outputs converted pseudonyms that after unblinding are identical when from the same user. Now, all three of these algorithms are extended to handle the signatures as inputs and outputs.

CLS+.Blind$(gpk, bpk, (\mu, \sigma, m)) \to (c\mu, c\sigma, c)$**:** performed by the data lake , outputs a blinded pseudonym, signature and message.

CLS+.Convert$(gpk, bpk, csk, \{(c\mu_i, c\sigma_i, c_i)\}_k) \to \{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_k$**:** performed by the converter, on input k blinded tuples, outputs k converted tuples.

CLS+.Unblind$(bsk, (\overline{c\mu}, \overline{c\sigma}, \overline{c})) \to (\overline{\mu}, \overline{\sigma}, m)$**:** performed by the data processor, outputs an unblinded, converted ($\texttt{tier-}2$) tuple.

2.2 Security Properties of CLS+

We need the CLS+ model to capture (roughly) the same security properties as the CLS model, without the assumption that conversion is triggered by honest parties and such that converted data is verifiable. We describe the properties that CLS+ schemes must satisfy in the table below. The first three constitute the privacy related properties and the final three constitute the unforgeability properties. We no longer include the *join anonymity* requirement from the CLS model, which ensures the corrupted issuer and converter cannot trace signatures to a user's join session. As the converter is corrupted, this requirement only offers weak privacy guarantees, and so we believe this requirement is less important than the positives of a simple modular construction. In all experiments the key generation stage is performed honestly, as standard for group signatures [5].

Our CLS+ schemes still rely on the converter being honest-but-curious. We believe this is an acceptable assumption in practice, as the converter is a central entity that can undergo more scrutiny than the many verifiers and data lakes. In section 6, we discuss how our work could be adapted to achieve security with respect to a fully malicious converter and why we do not take this approach.

Requirement	Corrupted Entities	Overview
Anonymity	Issuer, Data Lake, Data Processor	Group signatures which have not been linked through a conversion request should not leak any information about the signer's identity.
Non-Transitivity	Issuer, Data Lake, Data Processor	While conversion guarantees linkability *within* a batch of converted signatures, the data processor(s) should not be able to link the outputs of *different* convert queries.
Conversion Blindness	Issuer, Converter, Data Lake	The converter should learn no information about the blinded messages and pseudonyms that are input to a conversion.
Traceability	Converter, Data Lake, Data Processor	An adversary should not be able to create more (blinded) `tier`-1 signatures that remain unlinkable in an (honest) conversion than they control corrupt users. We show in Appendix B this implies the same for `tier`-1 signatures that are not blinded.
Conversion Unforgeability	Issuer, Data Lake, Data Processor	All valid `tier`-2 signatures should originate from an honest conversion.
Non-Frameability	Issuer, Converter, Data Lake, Data Processor	An adversary should not be able to output a (blinded) `tier`-1 signature that would be linked to the signature of an honest user. We show in Appendix B this implies the same for `tier`-1 signatures that are not blinded.

Oracles & State. As in the CLS model, our security requirements make use of oracles which keep joint state. We follow the notation of [4, 5] and give the adversary oracle access to honest users, the issuer and the converter (depending on the corruption setting in each game). All oracles have access to the following maintained as global state: a list HUL of *uid*s of honest users, a list CUL of *uid*s of corrupt users (where the issuer is honest), a list SL of requests to the SIGN oracle, and a list UBL containing inputs to the CONVERT oracle and the corresponding unblinded, converted pseudonyms/ messages. We provide an overview of all oracles below.

ADDU (join of honest user & honest issuer) Creates an honest user *uid*, by internally running a join protocol between the honest user and honest issuer. As a result, the oracle stores the secret key $\mathbf{gsk}[uid]$ for later use.

SNDU (Send to User) (join of honest user & corrupt issuer) Creates an honest user *uid*, by running the join protocol on behalf of the honest user with the corrupt issuer. If the join session is successful, the oracle stores the user's secret key $\mathbf{gsk}[uid]$ for later use.

SNDI (Send to Issuer) (join of corrupt user & honest issuer) Creates a corrupt user *uid*, by running the join protocol on behalf of the honest issuer with the corrupt user.

SIGN Outputs signatures on behalf of honest users that have successfully joined (via ADDU or SNDU, depending on whether the issuer is corrupt).

CONVERT The oracle returns a set of converted signatures. In the CLS+ model, the CONVERT oracle is input blinded pseudonym, signature, message tuples instead of `tier`-1 tuples that must be verified and honestly blinded in the oracle. This is because we no longer assume honest inputs from the data verifier. The adversary must input the blinding secret key, which is necessary for our privacy-related security notions, e.g., to ensure that the adversary does not input a re-randomisation of the challenge signature.

Aside from the CONVERT oracle, the oracles are minor adaptations of the oracles in the CLS model. The full description is deferred to Appendix A.1.

Helper Algorithms. In the CLS model, helper algorithms allow for notational simplicity when defining security. Indeed, the algorithms Identify and UnLink respectively test whether a signature originated from a particular user secret key and determine whether signatures would be linked upon conversion. We adapt the helper algorithms for the CLS+ model. These now take as input signatures, because they make use of CLS+.Convert. Identify tests whether a blinded signature belongs to a certain user uid, by creating a second signature for $\mathbf{gsk}[uid]$ and using the converter's secret key to test whether both signatures are linked. This algorithm uses our second helper algorithm UnLink internally, which takes a list of blinded pseudonym-message-signatures pairs and returns 1 only if they are all unlinkable after being converted and unblinded. The tier-2 signatures output as a result of the linking are also now verified. Full details are given in Appendix A.2.

Adapting our unforgeability requirements to the CLS+ setting As well as tier-1 signatures, we now must ensure our unforgeability guarantees for tier-2 signatures, as well as signatures input to and output from conversions. We introduce conversion unforgeability, a new security property that ensures all valid tier-2 signatures stem from an honest conversion. In doing so, our unforgeability guarantees carry over to tier-2 signatures, under the assumption of an honest-but-curious converter. Moreover, in the CLS traceability and non-frameability requirements, the adversary outputs tier-1 signatures, which are then verified and blinded honestly. We update these requirements for the CLS+ setting, so that *blinded* tier-1 signatures are output by the adversary. In the case that conversions are honest, the traceability and non-frameability guarantees carry through to tier-2 signatures. When the converter is corrupted no unforgeability guarantees hold for tier-2 signatures anyway.

We need to ensure that our traceability and non-frameability requirements ensure the CLS definitions, meaning that our unforgeability guarantees hold for tier-1 signatures. In Appendix B, we give reductions that show this is the case. For traceability, we show that if an adversary can output more valid tier-1 signatures that are unlinkable after conversion than they control corrupt users then, by blinding these tier-1 signatures, we can win in our CLS+ traceability game. For non-frameability, we show that if an adversary can output a tier-1 signature that links to that of an honest user in a conversion then, by blinding this signature, we can win in our CLS+ non-frameability game.

To make several of our requirements realisable, we require that the adversary outputs the blinding secret key. We do this for notational simplicity, but alternatively we could add a mechanism for the bsk to be extracted from the bpk. Due to the fact that the adversary outputs blinded tier-1 signatures in our traceability and non-frameability requirements, this is necessary to determine whether signatures are linked. Even if the adversary was to output tier-2 signatures, then the blinding secret key would still be necessary to enforce that the tier-2 signatures originate from an honest conversion.

Correctness and Consistency As in CLS, correctness consists of *correctness of sign*, which ensures that signatures generated honestly will be valid, and *correctness of conversion*, which ensures that honest blinding, conversion and unblinding will result in valid and consistent messages, pseudonyms and signatures.

As in CLS, we require *consistency*, which for CLS+ we relabel *consistency of linking.* This ensures that linking is consistent across multiple convert queries, i.e. if after conversion $c\mu_1$ and $c\mu_2$ are linked, and after conversion $c\mu_2$ and $c\mu_3$ are linked, then $c\mu_1$ and $c\mu_3$ are also linked after conversion. We additionally introduce *consistency of verification.* This is necessary as the verifiability of signatures is now preserved throughout the conversion process. This requirement ensures that a set of valid `tier`-1 signatures will result in valid `tier`-2 signatures for the same set of messages after blinding, conversion and unblinding.

We give the full correctness and consistency definitions in Appendix A.3.

We now provide an overview of all CLS+ security requirements. We present these in full in Appendix A.4.

Anonymity. This requirement ensures that an adversary that has corrupted the issuer, data lake and data processor, while the converter remains honest, cannot link an honest user's signatures or trace them to their join session. The adversary outputs two honest users uid_0^* and uid_1^*, a message m^*, and a blinding public key bpk^* (as this is fixed in signing), and must guess which user authored the resulting `tier`-1 signature.

The adversary has access to the SNDU and SIGN oracles to create honest users and obtain their signatures, as well as the CONVERT oracle. The CLS notion assumed the data lake to be honest, and so the conversion oracle only took unblinded tuples as input, allowing the oracle to check the validity of the input, before blinding and converting them. The adversary was prevented from submitting the challenge signature along with a signature authored by user uid_0^* or uid_1^*, which would allow them to trivially win.

Here we enable the data lake to ask for the conversion of blinded tuples. To prevent trivial wins, we still must be able to detect whether the adversary tries to convert the challenge signature. As signatures will be re-randomisable to satisfy non-transitivity, we opt for an RCCA-style of definition [15]. The CONVERT oracle checks whether any of the blinded signature-tuples would link via Identify to either of the challenge users and match the challenge message m^*. To do so, we require the adversary to input the blinding *secret* key to the oracle. This key is used to check whether the inputs can be traced to a challenge user, but there are no checks that enforce that the inputs are well-formed.

As in CLS, our privacy related requirements do not yield forward anonymity, because the secret keys of honest users cannot be corrupted. It seems difficult to achieve this whilst also ensuring the non-transitivity of conversions [27].

Non-Transitivity. Non-transitivity ensures that the outputs of separate convert queries cannot be linked together across multiple queries, further than what is already possible due to the messages queried. Otherwise, data processor(s) could gradually re-recover the linkability among all signatures. This must hold when the issuer, data lake and processor can be fully corrupt.

As in the CLS model, our non-transitivity definition follows a simulation-based approach, where the adversary must distinguish between the real and ideal world. As the issuer is corrupted, the adversary has access to the SNDU and SIGN oracles for honest users. In the real world, the adversary interacts with the CONVERT oracle, which converts the blinded message-pseudonym-signature tuples that are input. In the ideal world, they interact with the CONVSIM oracle which, for inputs that originate from honest users, uses a simulator SIM that outputs converted pseudonyms/ signatures. SIM only learns which blinded messages belong to the same honest users, without learning the pseudonyms/ signatures input. As in CLS, the CONVSIM oracle generates converted pseudonyms/ signatures for corrupt users normally via CLS+.Convert.

In contrast to the original CLS model, we now allow the data lake to trigger conversions on *blinded* inputs. Similarly to the anonymity game, the adversary must also input the blinding (secret) key with each query to the conversion oracle. Here the key is used to internally unblind the inputs and determine the correlation among the signatures. This is necessary to obtain a security definition that is realisable, as the CONVSIM oracle is still expected to provide consistently transformed outputs *within* a query. Further, the ideal CONVSIM oracle first internally runs the real conversion algorithm and aborts if it fails. This is again necessary to avoid trivial wins where the adversary might input malformed tuples — which the simulator never gets and thus cannot verify.

Conversion Blindness. In the original CLS model, conversion blindness ensures that the converter learns nothing about the pseudonyms and messages it converts. In the CLS+ model, the converter now receives and outputs signatures, which must also be converted obliviously. This must hold when all the entities, except for the data processor, can be fully corrupt. The adversary outputs two pseudonym-message-signature tuples and receives a blinded version of one of them. They must guess which tuple was blinded. We must ensure the adversary's outputs are valid to avoid trivial wins. In the CLS model, no oracles are required because blinding is a public-key operation. However, CLS+.Unblind now outputs a `tier`-2 signature, and so we must ensure that this does not leak anything that might allow for the unblinding of other converted signatures. We therefore give the adversary access to an UNBLIND oracle that blinds, converts and unblinds signatures. We stress that this requirement only provides CPA–level security as in CLS, because the oracle both blinds and unblinds signatures.

Conversion Unforgeability. As converted data is authenticated in the CLS+ model, we introduce the conversion unforgeability requirement. This ensures that all valid `tier`-2 signatures originate from an honest conversion when all entities other than the converter are corrupt. This carries over the traceability and non-frameability guarantees for blinded `tier`-1 signatures, ensuring that both properties hold for `tier`-2 signatures, provided the converter remains honest-but-curious.

We do not differentiate between honest and corrupt users in this requirement, so the adversary only has access to the CONVERT oracle. The adversary must output a valid `tier`-2 signature that does not originate from the CONVERT oracle. In order to check whether the `tier`-2 signature output by the adversary stems from an honest conversion, the adversary must input the blinding secret key to the conversion oracle. The oracle can then unblind all converted outputs and store `tier`-2 pseudonyms/ messages in the UBL list, to compare with the adversary's output.

Traceability. This requirement formalises that an adversary cannot output more unlinkable signatures that the number of corrupted users, when the issuer is honest but the converter, data lake, data processor and some users are corrupt.

As the issuer is honest, the adversary has access to the ADDU and SNDI oracles to create honest and corrupt users respectively, and the SIGN oracle. To lift the CLS traceability notion to the setting where malicious parties can trigger conversions, the adversary outputs a list of *blinded* signatures. As we still assume the converter to be honest-but-curious, the signatures are then *honestly* converted and unblinded by the challenger. For the traceability requirement to be achievable we must ensure that all signatures originate from the same convert query, due to the non-transitivity property. Therefore, it is natural to require the adversary to output a set of blinded `tier`-1 signatures, that are honestly converted. The blinding secret key must then be output by the adversary to determine whether signatures are linked.

In the CLS model, the adversary needed to output more unlinkable `tier`-1 signatures than the number of corrupted users. As the adversary now outputs blinded signatures, we can no longer check if they originate from the signing oracle. Instead, we allow the adversary to output the signatures of honest users. However, for each honest user that could have authored a message, we increase by 1 the number of unlinkable signatures the adversary is required to output. The adversary wins if they output more unlinkable signatures than the number of corrupted users plus the number of signing queries for distinct users.

Our traceability guarantee carries forward to `tier`-2 signatures when the converter is honest. This is because, conversion unforgeability ensures that all valid `tier`-2 signatures originate from a blinded `tier`-1 signature that is input to the honest converter, even when the data processor is corrupted. With a corrupted converter, no guarantees can be made for `tier`-2 signatures anyway.

Non-Frameability. This notion prevents the impersonation of an honest user, whereby an adversary generates signatures that will link to those of this user, when the issuer, converter, data lake and data processor are corrupt.

As the issuer is corrupt, the adversary has access to the SNDU and SIGN oracles to create honest users and obtain their signatures. The adversary outputs a blinded message-pseudonym-signature tuple, along with a blinding public and secret key. As in the CLS model, we use the $\mathsf{Identify}$ algorithm to check whether signatures stem from an honest user. On input a blinded $\texttt{tier}$-1 signature, this creates a second blinded $\texttt{tier}$-1 signature, and converts and unblinds both, checking if they are linked. $\mathsf{Identify}$ now checks the validity of the $\texttt{tier}$-2 signature which is necessary for a framing attack to occur.

When defining the non-frameability of $\texttt{tier}$-2 signatures, firstly consider that the converter is corrupted. As the security guarantees for $\texttt{tier}$-2 signatures depend on the converter being honest-but-curious, in this case the adversary can trivially forge tier-2 signatures and so we can only prevent framing attacks via blinded $\texttt{tier}$-1 signatures. If the converter remains honest, the conversion unforgeability requirement ensures that an adversary can only impersonate an honest user via a blinded signature that is honestly converted and unblinded. Therefore, both cases are captured by the adversary outputting a blinded signature that is converted honestly in the experiment, as in our requirement.

The blinding secret key must be output by the adversary to determine the linkage between signatures. Although in our requirement unblinding is honest, the conversion unforgeability requirement ensures that all valid $\texttt{tier}$-2 signature can be traced to a conversion assuming the converter is honest. Even if the requirement was formulated so that $\texttt{tier}$-2 signatures were output, the blinding secret key would still need to be output by the adversary to check that this signature originated from an honest conversion. Therefore, the adversary must output the blinding secret key for our definition to be realisable.

As in the CLS model, we must prevent trivial wins via the signing oracle. Due to the re-randomisability of $\mathsf{CLS}+$ signatures to allow for non-transitivity, instead of not allowing signatures output by SIGN, we consider whether the attached message was input to the signing oracle. As the adversary outputs a blinded tuple, we must convert and unblind their output to obtain this message.

3 Preliminaries

We now present the building blocks required in this work.

Bilinear Maps. Let $\mathbb{G}_1$, $\mathbb{G}_2$, $\mathbb{G}_T$ be cyclic multiplicative groups with prime order p. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ must satisfy: *bilinearity*, i.e., $e(g_1^x, g_2^y) = e(g_1, g_2)^{xy}$; *non-degeneracy*, i.e., for generators $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$, $e(g_1, g_2)$ generates $\mathbb{G}_T$; and *efficiency*, i.e., there exists efficient algorithms $\mathcal{G}(1^\tau)$ that outputs a bilinear group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$ and to compute $e(a, b)$ for all $a \in \mathbb{G}_1$, $b \in \mathbb{G}_2$. We use type-3 pairings in this work and do not assume $\mathbb{G}_1 = \mathbb{G}_2$ or an efficiently computable isomorphism between groups [26]. Type-3 pairings benefit from the most efficient curves, when balancing the cost of pairings and group operations, the size of the representation of an element of $\mathbb{G}_2$ and the flexibility of parameter choice [26, 17].

Standard and Automorphic Signatures. We use digital signatures which satisfy Existential Unforgeability against Chosen Message Attacks (EUF-CMA), consisting of: $\mathsf{SIG.Setup}$ outputs the parameters $\mathsf{pp}_{\mathsf{sig}}$, $\mathsf{SIG.KeyGen}(\mathsf{pp}_{\mathsf{sig}})$ outputs the signing and verification keys (sk, vk), $\mathsf{SIG.Sign}(sk, m)$ outputs a signature Ω, $\mathsf{SIG.Ver}(\Omega, vk, m)$ checks the signature is valid.

An *automorphic signature* [25] over a bilinear group is an EUF-CMA secure signature whose verification keys are contained in the message space. Moreover, the messages and signatures consist of elements of $\mathbb{G}_1$ and $\mathbb{G}_2$, and the verification predicate is a conjunction of pairing-product equations over the verification key, the message and the signature. They consist of the following algorithms: $\mathsf{ASetup}(1^\tau)$ outputs the parameters $\mathsf{pp_{auto}}$; $\mathsf{AKeyGen}(\mathsf{pp_{auto}})$ outputs the signing and verification keys, (apk, ask); $\mathsf{ASign}(ask, m)$ outputs a signature Ω; and $\mathsf{AVerify}(\Omega, apk, m)$ checks that Ω is a valid signature.

ElGamal Encryption. We use the ElGamal encryption scheme [22], which is chosen-plaintext secure under the DDH assumption. We will use the *homomorphic* property of ElGamal, i.e., if $C_1 \in \mathsf{Enc}(pk, m_1)$, and $C_2 \in \mathsf{Enc}(pk, m_2)$, then $C_1 \odot C_2 \in \mathsf{Enc}(pk, m_1 \cdot m_2)$. ElGamal ciphertexts $c = \mathsf{Enc}(pk, m)$ are *publicly re-randomisable* in the sense that a re-randomised version c' of c looks indistinguishable from a fresh encryption of the plaintext m.

Proof Protocols. When referring to zero-knowledge proofs of knowledge of discrete logarithms, $\mathsf{PK}\{(a, b, c) : y = g^a h^b \wedge \tilde{y} = \tilde{g}^a \tilde{h}^c\}$ denotes a proof of knowledge of integers a, b and c such that $y = g^a h^b$ and $\tilde{y} = \tilde{g}^a \tilde{h}^c$ hold, as in the notation of [12]. SPK denotes a signature proof of knowledge, a non-interactive transformation of a proof PK. We require the proof system to be *sound* and *zero-knowledge.*

Controlled Malleable NIZKs. A controlled malleable proof [16] for a relation $\mathcal{R}$ and transformation class $\mathcal{T}$ consists of three algorithms constituting a regular non-interactive proof. $\mathsf{CRSSetup}(1^\tau)$ generates a common reference string σ_{crs}; $\mathcal{P}(\sigma_{\mathsf{crs}}, x, w)$: takes as input σ_{crs}, a statement x and a witness such that $(x, w) \in \mathcal{R}$, and outputs a proof π; $\mathcal{V}(\sigma_{\mathsf{crs}}, x, \pi)$ outputs 1 if π is valid for statement x. Such a proof is called zero knowledge (NIZK) if there exists a PPT simulator (S_1, S_2) such that an adversary cannot distinguish between proofs formed by the prover and simulator, and a proof of knowledge (NIZKPoK) if there exists a PPT extractor (E_1, E_2) that can produce a valid witness from any valid proof.

The fourth algorithm, specific to malleable proof systems, is: $\mathsf{ZKEval}(\sigma_{\mathsf{crs}}, T, x, \pi)$: which on input σ_{crs}, a transformation $T = (T_{\mathsf{inst}}, T_{\mathsf{wit}})$ (in transformation class $\mathcal{T}$), an instance x, and a proof π, outputs a mauled proof π' for instance $T_{\mathsf{inst}}(x)$.

The controlled-malleable simulation-sound extractability requirement reconciles malleability with simulation-sound extractability [21, 28]. It requires that, for any instance x, if an adversary can produce a valid proof π that $x \in \mathcal{R}$ then an extractor can extract from π either a witness w such that $(x, w) \in \mathcal{R}$ or a previously proved instance x' and transformation $T \in \mathcal{T}$ such that $x = T_{\mathsf{inst}}(x')$. This guarantees that any proof that the adversary produces is either generated from scratch using a valid witness, or formed by applying a transformation from the class $\mathcal{T}$ to an existing proof. We describe the full definition in Appendix C.1.

In [16] strong derivation privacy for such proofs is also defined. This ensures simulated proofs are indistinguishable from those formed via a transformation, as defined formally in Appendix C.2. Putting this together, a cm-NIZK is a proof system that is CM-SSE, strongly derivation private, and zero knowledge.

4 Our CLS+ Construction

Our CLS+ construction uses automorphic signatures, ElGamal encryption, controlled malleable NIZKs, a digital signature scheme, and a signature proof of knowledge as building blocks. Automorphic signatures are structure-preserving signatures, for which the verification key lies within the message space. Controlled-malleable NIZKs allow proofs to be mauled to blind signatures, but because the malleability is controlled the unforgeability properties are still satisfied.

High-Level Idea. We now present a high-level overview of our CLS+ construction, demonstrating how our construction differs from the CLS scheme presented in [27]. The issuer's key pair is that of an automorphic signature [25] as recalled in Section 3. The converter's key pair is an ElGamal encryption key pair and a key pair for a signature scheme. The blinding key pair, held by the data processor, is an ElGamal encryption key pair. Unlike [27], when joining, a user generates a key pair of an automorphic signature as their secret and public key (usk, upk). The issuer signs the user's public key to form a credential, which is possible due to the automorphic property. An automorphic signature is used, instead of a BBS+ signature in [27], to generate credentials, for compatibility with the cm-NIZK proofs, which are necessary to allow signatures to be blinded.

When a user signs a message m, as in [27] they encrypt their public key upk under the converter public key to form a pseudonym, and must prove knowledge of a secret key relating to a valid credential. In the CLS+ scheme to do so, they "normally" sign the message using the automorphic signature. The latter is never revealed, but is only used to generate a cm-NIZK, which proves knowledge of a signature of m under its public key, an issuer's credential on its public key and correctness of the pseudonym. To ensure that conversion queries cannot be used to de-anonymise honest users' signatures, we include upk' as witness such that $e(g_1, upk) = e(upk', g_2)$. The blinding public key must be fixed in signing, as it must be part of the statement proved by the cm-NIZK. This allows for the proof to be transformed in blinding, because cm-NIZKs are defined for relations that are closed under all allowable transformations. However, this allows users to have control when signing over the data processors that can use their data.

Blinding proceeds as in [27] for the pseudonym and message. The pseudonym is re-randomised, and an extra layer of encryption under the blinding public key is added. The message is encrypted under the blinding public key. In the CLS+ model, we need to use the malleability of the cm-NIZK to update the signature. This ensures that the new traceability and non–frameability requirements are satisfied, where the adversary outputs blinded signatures. The mauled proofs still ensure that the user holds a valid secret key corresponding to the public key that is encrypted in the pseudonym to provide non–frameability, and a valid credential on this public key to provide traceability. The strong derivation privacy of the cm-NIZK ensures that the modified conversion blindness requirement holds.

During conversions, each blinded pseudonym, message and now signature is verified. Like [27], the encryption is re-randomised on the message and pseudonym. The converter decrypts the pseudonym using the converter secret key and raises the resulting pseudonym to the power of r which is chosen afresh in every convert query, but used consistently within. Unlike [27], the resulting pseudonym is transformed to the target group to prevent anonymity attacks, and the converter now outputs a standard signature on the converted pseudonym and message.

During unblinding, as in [27], the `tier`-2 pseudonyms and messages are retrieved, by decrypting under the blinding secret key. In the CLS+ scheme, `tier`-2 pseudonyms are of the form $e(g_1, upk)^r$, and so signatures can be linked by author. The final `tier`-2 signature is the blindly signed tuple from the converter and a proof of correct unblinding by the data processor, which can both be publicly verified. This ensures that our new requirement conversion unforgeability is satisfied. To ensure that the converter only blindly signs messages that were authenticated via a group signature, we use that mauled cm-NIZKs can be verified. As the converter is assumed to be honest-but-curious, this transmits the authentication guarantees from the original group signatures to converted ones.

Additional Structural Assumptions of Automorphic Signatures. We make the following assumptions satisfied by our instantiation in section 5. The automorphic signature scheme can be simplified so either messages are elements of $\mathbb{G}_1$ and the verification key is an element of $\mathbb{G}_2$ $(\mathsf{ASetup}_1, \mathsf{AKeyGen}_1, \mathsf{ASign}_1, \mathsf{AVerify}_1)$, or messages are elements of $\mathbb{G}_2$ and the verification key is an element of $\mathbb{G}_1$ $(\mathsf{ASetup}_2, \mathsf{AKeyGen}_2, \mathsf{ASign}_2, \mathsf{AVerify}_2)$. We assume our automorphic signatures are in the type-3 setting, ASetup is input the bilinear group, and the signing key and verification key are of the form $sk \in \mathbb{Z}_p^*$ and $vk = g_j^{sk}$ when $vk \in \mathbb{G}_j$.

4.1 Detailed Description of CLS–CM

Setup & Key Generation. In CLS+.Setup, parameters for all building blocks are generated. The issuing keypair is the keypair of an automorphic signature. The converter's keypair is an ElGamal key pair in $\mathbb{G}_2$ and a keypair for a signature scheme. The blinding keypair is an ElGamal key pair in both $\mathbb{G}_1$ and $\mathbb{G}_2$.

```
CLS+.Setup($1^\tau$)
$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathcal{G}(1^\tau)$
$\mathsf{pp}_{\mathsf{auto}1} \leftarrow \mathsf{ASetup}_1(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{pp}_{\mathsf{auto}2} \leftarrow \mathsf{ASetup}_2(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$
$g \leftarrow\!\$\, \mathbb{G}_1, \hat{g} \leftarrow\!\$\, \mathbb{G}_2, \sigma_{\mathsf{crs}} \leftarrow \mathsf{CRSSetup}(1^\tau), \mathsf{pp}_{\mathsf{sig}} \leftarrow \mathsf{SIG.Setup}(1^\tau)$
return $((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{pp}_{\mathsf{auto}1}, \mathsf{pp}_{\mathsf{auto}2}, g, \hat{g}, \sigma_{\mathsf{crs}}, \mathsf{pp}_{\mathsf{sig}})$
```

```
CLS+.IKGen(pp)
$(ipk, isk) \leftarrow \mathsf{AKeyGen}_2(\mathsf{pp}_{\mathsf{auto}2})$
return $(ipk, isk)$
```

```
CLS+.BKGen(pp)
$bsk_1, bsk_2 \leftarrow\!\$\, \mathbb{Z}_p^*$
$bpk_1 \leftarrow g^{bsk_1}, bpk_2 \leftarrow \hat{g}^{bsk_2}$
return $((bpk_1, bpk_2), (bsk_1, bsk_2))$
```

```
CLS+.CKGen(pp)
$csk_1 \leftarrow\!\$\, \mathbb{Z}_p^*, cpk_1 \leftarrow \hat{g}^{csk_1}$
$(cpk_2, csk_2) \leftarrow \mathsf{SIG.KeyGen}(\mathsf{pp}_{\mathsf{sig}})$
return $((cpk_1, cpk_2), (csk_1, csk_2))$
```

Join. We give the join protocol of our CLS–CM construction in Figure 1. When joining, the users generate a key pair (usk, upk) for an automorphic signature and obtain the issuer's signature on their public key. They also compute $upk' = g_1^{usk}$.

$\mathcal{U}.\mathsf{CLS+.Join}(gpk)$	$\rightleftharpoons$	$\mathcal{I}.\mathsf{CLS+.Issue}(isk, gpk)$
$(upk, usk) \leftarrow \mathsf{AKeyGen}_1(\mathsf{pp}_{\mathsf{auto1}})$		
	$\xrightarrow{upk}$	
		$cred \leftarrow \mathsf{ASign}_2(isk, upk)$
	$\xleftarrow{cred}$	
check that $\mathsf{AVerify}_2(cred, ipk, upk) = 1$		
$upk' \leftarrow g_1^{usk}$		
return $\mathbf{gsk}[uid] \leftarrow (usk, upk, upk', cred)$,		

Fig. 1. Join protocol of our CLS–CM construction.

Sign & Verication of $\mathtt{tier}$*-1 signatures.* When signing, the user's public key is encrypted under the converter public key to form the pseudonym $\mu = (\hat{g}^\alpha, 1 \in \mathbb{G}_2, upk \cdot cpk_1^\alpha)$. The pseudonym can also be seen as an encryption under the blinding key with encryption randomness 0. The ciphertext encrypting m under the blinding key with encryption randomness 0, is of the form $(1 \in \mathbb{G}_1, m)$. This means that blinding encryption can be added by re-randomising both ciphertexts in CLS+.Blind, while maintaining the ability to update the associated proof.

The user then signs m with their user secret key to output Ω. The signature Ω is never output, but instead a cm-NIZK is computed which proves that μ is an encryption of upk, c is an "encryption" of m (with randomness 0), and knowledge of a correct Ω. The latter comprises showing that Ω is a valid signature on m under upk, and it knows a credential $cred$ that is a valid signature on upk under ipk. We also prove knowledge of upk' such that $e(upk', g_2) = e(g_1, upk)$ to prevent attacks on anonymity via conversion queries. This is because usk can no longer be included as a witness as in [27], as it is not a group element. Instead we include upk', which due to the DDH assumption in $\mathbb{G}_2$ and the pairing setting, is hard to derive from upk. This prevents the following attack against anonymity: the adversary uses the upk of an honest user uid to create a new signature with user public key upk^a using a known $a \in \mathbb{Z}_p^*$. They could then test whether another signature belongs to this honest user uid by submitting it alongside the signature they created to the converter. If any of the $\mathtt{tier}$-2 pseudonyms are of the form P, P^a for any P, then they know the signature belongs to uid. The final signature simply consists of the cm-NIZK.

More formally we define the relation R such that $((cpk_1, bpk_1, bpk_2, ipk, \mu, c), (upk', upk, cred, \Omega, g_1^\alpha, g_1^\beta, g_2^\gamma, m)) \in R$ if and only if:

$$e(g_1, \mu_1) = e(g_1^\alpha, \hat{g}), \quad e(g_1, \mu_2) = e(g_1^\beta, \hat{g}), \quad \text{and} \tag{1}$$

$$e(g_1, \mu_3) = e(g_1, upk)e(g_1^\alpha, cpk_1)e(g_1^\beta, bpk_2), \quad \text{and} \tag{2}$$

$$\mathsf{AVerify}_1(\Omega, upk, m) = 1, \quad \mathsf{AVerify}_2(cred, ipk, upk) = 1, \quad \text{and} \tag{3}$$

$$e(c_1, g_2) = e(g, g_2^\gamma), \quad e(c_2, g_2) = e(m, g_2)e(bpk_1, g_2^\gamma), \text{ and} \tag{4}$$

$$e(upk', g_2) = e(g_1, upk). \tag{5}$$

We define the allowable set of transformations for this relation to be:
$\mathcal{T} = \{(r_{\mathsf{enc1}}, r_{\mathsf{enc2}}, r_{\mathsf{enc3}}) : r_{\mathsf{enc1}}, r_{\mathsf{enc2}}, r_{\mathsf{enc3}} \in \mathbb{Z}_p^*\}$, such that for $T = (r_{\mathsf{enc1}}, r_{\mathsf{enc2}}, r_{\mathsf{enc3}})$,

the transformation $T_{\mathsf{inst}}(cpk_1, bpk, ipk, \mu, c) = (cpk_1, bpk, ipk, (\mu_1 \hat{g}^{r_{\mathsf{enc1}}}, \mu_2 \hat{g}^{r_{\mathsf{enc2}}}, \mu_3 \cdot cpk_1^{r_{\mathsf{enc1}}} bpk_2^{r_{\mathsf{enc2}}}), (c_1 g^{r_{\mathsf{enc3}}}, c_2 bpk_1^{r_{\mathsf{enc3}}}))$ and $T_{\mathsf{wit}}(upk', upk, cred, \Omega, g_1^{\alpha}, g_1^{\beta}, g_2^{\gamma}, m) = (upk', upk, cred, \Omega, g_1^{\alpha} g_1^{r_{\mathsf{enc1}}}, g_1^{\beta} g_1^{r_{\mathsf{enc2}}}, g_2^{\gamma} g_2^{r_{\mathsf{enc3}}}, m)$. We show later that this relation and transformation can be instantiated as a $\mathsf{cm\text{-}NIZK}$.

In more detail, $\mathsf{CLS+.Sign}$ and $\mathsf{CLS+.Verify}$ are defined as follows:

```
$\mathsf{CLS+.Sign}(gpk, bpk, \mathbf{gsk}[uid], m)$
parse $\mathbf{gsk}[uid] = (usk, upk, upk', cred), \alpha \leftarrow\$ \mathbb{Z}_p^*, \mu \leftarrow (\hat{g}^{\alpha}, 1, upk \cdot cpk_1^{\alpha})$
$\beta \leftarrow 0, \gamma \leftarrow 0, c \leftarrow (1, m), \Omega \leftarrow \mathsf{ASign}_1(usk, m)$
$\sigma \leftarrow \mathsf{cm\text{-}NIZK}\{(upk', upk, cred, \Omega, g_1^{\alpha}, g_1^{\beta}, g_2^{\gamma}, m) : e(g_1, \mu_1) = e(g_1^{\alpha}, \hat{g}) \wedge e(g_1, \mu_2) = e(g_1^{\beta}, \hat{g}) \wedge$
$e(g_1, \mu_3) = e(g_1, upk) e(g_1^{\alpha}, cpk_1) e(g_1^{\beta}, bpk_2) \wedge \mathsf{AVerify}_1(\Omega, upk, m) = 1 \wedge$
$\mathsf{AVerify}_2(cred, ipk, upk) = 1 \wedge e(c_1, g_2) = e(g, g_2^{\gamma}) \wedge e(c_2, g_2) = e(m, g_2) e(bpk_1, g_2^{\gamma}) \wedge$
$e(upk', g_2) = e(g_1, upk)\}$
**return** $(\mu, \sigma)$

$\mathsf{CLS+.Verify}(\mathtt{tier}\text{-}1, gpk, bpk, m, \mu, \sigma)$
Check $\mu_2 = 1$, Verify $\sigma$ with respect to $(cpk_1, bpk, ipk, \mu, (1, m))$
```

Blind Conversions. During blinding, the pseudonym and message are encrypted under the blinding public key, and the encryption under the converter public key is re-randomised. The $\mathsf{cm\text{-}NIZK}$ is transformed with ZKEval so that it is consistent with the blinded pseudonym, and message, which also re-randomises the proof due to the derivation privacy.

In $\mathsf{CLS+.Convert}$, blinded signatures are now input and verified, leveraging the fact that even fully blinded inputs can be checked for their correctness. The pseudonyms, and blinded messages are re-randomised to ensure non-transitivity. The encryption under the converter public key is then removed from the pseudonyms and they are converted by raising them to the power of r and transforming them into the target group, to ensure non-transitivity. The converted signature is simply a digital signature on the blinded converted pseudonym and message, with respect to the converter's verification key.

In $\mathsf{CLS+.Unblind}$ the converted pseudonym and ciphertext are now decrypted under the blinding secret key. The blinded then converted pseudonym, message and signature are output, along with a proof that the unblinding has been done correctly. During $\mathtt{tier}$-2 verification, the converter's signature on the blinded values and the proof of unblinding are verified.

```
$\underline{\mathsf{CLS+.Blind}(gpk, bpk, (m, \mu, \sigma))}$
$\textbf{if } \mathsf{CLS+.Verify}(\texttt{tier}\text{-}1, gpk, bpk, m, \mu, \sigma) = 0 \quad \textbf{return } \bot$
$\alpha', \beta', \gamma' \leftarrow\$ \mathbb{Z}_p^*, c\mu \leftarrow (\mu_1 \hat{g}^{\alpha'}, \mu_2 \hat{g}^{\beta'}, \mu_3 cpk_1^{\alpha'} bpk_2^{\beta'})$
$c \leftarrow (g^{\gamma'}, m \cdot bpk_1^{\gamma'}), c\sigma \leftarrow \mathsf{ZKEval}(\sigma_{\mathsf{crs}}, (\alpha', \beta', \gamma'), (cpk_1, bpk, ipk, \mu, (1, m)), \sigma)$
$\textbf{return } (c, c\mu, c\sigma)$
$\underline{\mathsf{CLS+.Convert}(gpk, bpk, csk, \{(c\mu_i, c\sigma_i, c_i)\}_k)}$
$r \leftarrow\$ \mathbb{Z}_p^*, \textbf{for } i = 1, \ldots k :$    Verify $c\sigma_i$ with respect to $c\mu_i, c_i, gpk$ and $bpk$
  $\alpha', \beta', \gamma' \leftarrow\$ \mathbb{Z}_p^*, c\mu_i' \leftarrow (c\mu_{i,1} \hat{g}^{\alpha'}, c\mu_{i,2} \hat{g}^{\beta'}, c\mu_{i,3} cpk_1^{\alpha'} bpk_2^{\beta'}), c_i' \leftarrow (c_{i,1} g^{\gamma'}, c_{i,2} bpk_1^{\gamma'})$
  $c\mu_i'' \leftarrow (c\mu_{i,2}', c\mu_{i,3}' \cdot c\mu'^{-csk_1}_{i,1}), c\mu_{i,1}''' \leftarrow e(g_1, c\mu_{i,1}'')^r, c\mu_{i,2}''' \leftarrow e(g_1, c\mu_{i,2}'')^r$
$\sigma_i' \leftarrow \mathsf{SIG.Sign}(csk_2, (c_i', c\mu_i''', bpk))$
choose random permutation $\Pi, \textbf{for } i = 1, \ldots, k : (\overline{c\mu}_i, \overline{c}_i, \overline{c\sigma}_i) \leftarrow (c\mu'''_{\Pi(i)}, c'_{\Pi(i)}, \sigma'_{\Pi(i)})$
$\textbf{return } ((\overline{c\mu}_1, \overline{c}_1, \overline{c\sigma}_1)), ..., (\overline{c\mu}_k, \overline{c}_k, \overline{c\sigma}_k)))$
$\underline{\mathsf{CLS+.Unblind}(bsk, (\overline{c\mu}, \overline{c\sigma}, \overline{c}))}$
$\overline{\mu} \leftarrow \overline{c\mu}_2 \cdot \overline{c\mu}_1^{-bsk_2}, m \leftarrow \overline{c}_2 \overline{c}_1^{-bsk_1}$
$\pi_{\mathsf{ub}} \leftarrow \mathsf{SPK}\{(bsk_1, bsk_2) : \overline{\mu} = \overline{c\mu}_2 \cdot \overline{c\mu}_1^{-bsk_2} \wedge m = \overline{c}_2 \overline{c}_1^{-bsk_1} \wedge bpk_1 = g^{bsk_1} \wedge bpk_2 = \hat{g}^{bsk_2}\}$
$\overline{\sigma} \leftarrow (\overline{c\mu}, \overline{c\sigma}, \overline{c}, \pi_{\mathsf{ub}}) \quad \textbf{return } (\overline{\mu}, m, \overline{\sigma})$
$\underline{\mathsf{CLS+.Verify}(\texttt{tier}\text{-}2, gpk, bpk, \overline{m}, \overline{\mu}, \overline{\sigma})}$
parse $\overline{\sigma} = (\overline{c\mu}, \overline{c\sigma}, \overline{c}, \pi_{\mathsf{ub}})$, Verify $\pi_{\mathsf{ub}}$ holds for $\overline{c\mu}, \overline{\mu}, \overline{c}, \overline{m}, bpk$
$\textbf{if } \mathsf{SIG.Ver}((\overline{c\mu}, \overline{c}, bpk), cpk_2, \overline{c\sigma}) = 1 \quad \textbf{return } 1 \quad \textbf{else return } \bot$
```

4.2 Security of CLS–CM.

In Appendix D, we show that our scheme satisfies all security properties defined in Section 2. More precisely, we show that the following theorem holds.

Theorem 1. *The CLS–CM construction presented in Sec. 4.1 is a secure CLS+ as defined in Sec. 2 if: the automorphic signatures schemes are EUF-CMA secure and satisfy the additional structural assumptions given in Section 4; the cm-NIZK is zero knowledge, strongly derivation private and controlled-malleable simulation-sound extractable (cm-SSE); the SPK is a sound zero-knowledge proof; the DDH assumption holds in $\mathbb{G}_1$ and $\mathbb{G}_2$; and the SIG is EUF-CMA secure.*

5 Concrete Instantiation of CLS–CM construction

We show the building blocks of our CLS–CM construction can be instantiated.

Automorphic Signatures and Standard Signatures. An instantiation of automorphic signatures that is EUF-CMA secure based on the Asymmetric Double Hidden SDH (ADHSDH) assumption is given in [25]. It is easy to see that this scheme satisfies the additional structural assumptions needed for our construction. For the digital signature scheme, we will make use of Schnorr signatures [38].

Controlled Malleable NIZKs. We demonstrate that $\mathsf{cm\text{-}NIZK}$s for the relation $\mathcal{R}$, and set of allowable transformations $\mathcal{T}$ defined above can be instantiated. It is shown in Theorem 4.5 in [16] that $\mathsf{cm\text{-}NIZKS}$ for $(\mathcal{R}, \mathcal{T})$ can be instantiated if $(\mathcal{R}, \mathcal{T})$ are CM-friendly, which we show in Appendix E. This instantiation makes use of Groth Sahai proofs [30] to build malleable NIWIPOKs and structure preserving signatures (SPS) based on the DLIN assumption. However for our instantiation, we make use of Groth Sahai proofs [30] in the type-3 setting based on the SXDH assumption, that the DDH assumption holds in both groups $\mathbb{G}_1$ and $\mathbb{G}_2$. We make use of a different Structure Preserving Signature scheme [3] in the type 3 setting, with better efficiency and based on the SXDH assumption.

Instantiating the Proof of Unblinding. For transforming interactive into non-interactive zero-knowledge proofs we rely on the Fiat-Shamir heuristic that ensures security in the random oracle model.

Efficiency. We compare the computational cost of our construction to that of [27] in Table 2. We denote k exponentiations in group $\mathbb{G}_i$ by $k\mathsf{exp}_{\mathbb{G}_i}$, k pairing operations by $k\mathsf{pair}$, and k exponentiations in $\mathbb{Z}^*_{n^2}$ by $k\mathsf{exp}_{\mathbb{Z}^*_{n^2}}$. In Table 3, we compare the combined sizes of pseudonyms and signatures in terms of the amount of group elements to [27]. We denote the length required to represent k elements in $\mathbb{G}_i$ as $k\mathbb{G}_i$, k elements in $\mathbb{Z}_p$ as $k\mathbb{Z}_p$ and k elements in $\mathbb{Z}^*_{n^2}$ as $k\mathbb{Z}^*_{n^2}$. Our construction is significantly less efficient than that of [27], particularly in terms of the signing, verification and size of $\texttt{tier}$-1 signatures. However, we demonstrate that stronger security can be achieved and the assumption of trusted data lakes can be avoided.

Algorithm	Computational cost [27]	Computational cost (this work)
Sign	$16\mathsf{exp}_{\mathbb{G}_1} + 15\mathsf{exp}_{\mathbb{Z}_{n^2}}$	$668\mathsf{exp}_{\mathbb{G}_1} + 703\mathsf{exp}_{\mathbb{G}_2}$
$\mathsf{Verify}(\texttt{tier}\text{-}1)$	$12\mathsf{exp}_{\mathbb{G}_1} + 11\mathsf{exp}_{\mathbb{Z}_{n^2}} + 2\mathsf{pair}$	$1548\mathsf{pair}$
$\mathsf{Verify}(\texttt{tier}\text{-}2)$	-	$6\mathsf{exp}_{\mathbb{G}_1} + 2\mathsf{exp}_{\mathbb{G}_2} + 2\mathsf{exp}_{\mathbb{G}_T}$
Blind	$6\mathsf{exp}_{\mathbb{G}_1}$	$2\mathsf{exp}_{\mathbb{G}_1} + 4\mathsf{exp}_{\mathbb{G}_2}$
$\mathsf{Unblind}$	$2\mathsf{exp}_{\mathbb{G}_1}$	$3\mathsf{exp}_{\mathbb{G}_1} + 1\mathsf{exp}_{\mathbb{G}_2} + 2\mathsf{exp}_{\mathbb{G}_T}$
$\mathsf{Convert}$ (k pseudonyms input)	$7k\mathsf{exp}_{\mathbb{G}_1}$	$k(3\mathsf{exp}_{\mathbb{G}_1} + 7\mathsf{exp}_{\mathbb{G}_2} + 2\mathsf{pair})$

Fig. 2. Computational costs.

	$\texttt{tier}$-1 (μ, σ)	Blinded $(c\mu, c\sigma)$	Converted $(\overline{c\mu}, \overline{c\sigma})$	$\texttt{tier}$-2 $(\overline{\mu}, \overline{\sigma})$
Size [27]	$5\mathbb{G}_1$ $7\mathbb{Z}_p$ $6\mathbb{Z}^*_{n^2}$	$3\mathbb{G}_1$	$2\mathbb{G}_1$	$1\mathbb{G}_1$
Size (this work)	$446\mathbb{G}_1$ $541\mathbb{G}_2$	$446\mathbb{G}_1$ $541\mathbb{G}_2$	$2\mathbb{G}_T$ $2\mathbb{Z}_p$	$3\mathbb{G}_T$ $2\mathbb{G}_1$ $5\mathbb{Z}_p$

Fig. 3. Sizes of pseudonyms and signatures.

6 Conclusion and Future Work

We have extended the work of [27], allowing for authentication to be preserved during convert queries, and removing the assumption that inputs to the converter are well formed. We have extended the CLS model to formalise this strengthened security and given a provably secure construction in this model, making use of automorphic signatures, ElGamal encryption, controlled malleable NIZKs, a digital signature scheme, and a signature proof of knowledge A simple extension of this work (and [27]) would be to provide anonymity for `tier`-1 signatures with respect to the converter. An additional anonymity requirement could be included, where the converter and issuer are corrupted but the data processor is honest. Our construction could very simply be adapted by adding the layer of encryption under the blinding public key during `tier`-1 signature generation, instead of during blinding. We have not made this extension in this work to avoid adding complexity to an already complex security model.

Fully Malicious Converters. Another direction would be to investigate how to achieve security against fully malicious converters. The unforgeability guarantees for `tier`-2 signatures would no longer carry over from the blinded verifiable signatures input to the converter. This would require the converter to prove that their outputs were computed honestly from valid blinded inputs, and also that they have converted each input in a convert query consistently, i.e. they have not treated one convert query as several separate convert queries. The converter could re-randomise the blinded signatures input and output these along with the converted pseudonyms. They would also need to include a proof that the converted pseudonyms were computed correctly from the re-randomised blinded pseudonyms. However, to ensure that each signature was converted consistently, the size of the `tier`-2 signatures would increase with the number of inputs to the convert query, which would lead to a significant efficiency loss. We stress that this is not specific to our particular construction, but necessary to ensure all inputs were converted consistently.

References

1. Helping public health officials combat covid-19. `https://blog.google/technology/health/covid-19-community-mobility-reports/`
2. Abe, M., Fuchsbauer, G., Groth, J., Haralambiev, K., Ohkubo, M.: Structure–preserving signatures and commitments to group elements. In: Crypto (2010)
3. Abe, M., Hofheinz, D., Nishimaki, R., Ohkubo, M., Pan, J.: Compact structure-preserving signatures with almost tight security. In: Crypto (2017)
4. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In: Eurocrypt (2003)
5. Bellare, M., Shi, H., Zhang, C.: Foundations of group signatures: The case of dynamic groups. In: CT-RSA (2005)
6. Bernhard, D., Fuchsbauer, G., Ghadafi, E., Smart, N.P., Warinschi, B.: Anonymous attestation with user-controlled linkability. International Journal of Information Security 12(3) (2013)
7. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Crypto (2004)
8. Bootle, J., Cerulli, A., Chaidos, P., Ghadafi, E., Groth, J.: Foundations of fully dynamic group signatures. In: ACNS (2016)

9. Brickell, E., Camenisch, J., Chen, L.: Direct anonymous attestation. In: ACM CCS (2004)
10. Camenisch, J., Drijvers, M., Lehmann, A.: Anonymous attestation using the strong diffie hellman assumption revisited. In: TRUST (2016)
11. Camenisch, J., Drijvers, M., Lehmann, A.: Universally composable direct anonymous attestation. In: PKC (2016)
12. Camenisch, J., Kiayias, A., Yung, M.: On the portability of generalized schnorr proofs. In: Eurocrypt (2009)
13. Camenisch, J., Stadler, M.: Efficient group signature schemes for large groups. In: Crypto (1997)
14. Canard, S., Schoenmakers, B., Stam, M., Traoré, J.: List signature schemes. Discrete Applied Mathematics 154(2) (2006)
15. Canetti, R., Krawczyk, H., Nielsen, J.: Relaxing chosen-ciphertext security. In: Crypto (2003)
16. Chase, M., Kohlweiss, M., Lysyanskaya, A., Meiklejohn, S.: Malleable proof systems and applications. In: Eurocrypt (2012)
17. Chatterjee, S., Menezes, A.: On cryptographic protocols employing asymmetric pairings— the role of ψ revisited. Discrete Applied Mathematics 159(13) (2011)
18. Chaum, D., van Heyst, E.: Group signatures. In: Eurocrypt (1991)
19. Chow, S., Lee, J., Subramanian, L.: Two-party computation model for privacy-preserving queries over distributed databases. In: NDSS (2009)
20. Chow, S., Susilo, W., Yuen, T.: Escrowed linkability of ring signatures and its applications. In: VIETCRYPT (2006)
21. De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Crypto (2001)
22. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. IEEE transactions on information theory 31(4) (1985)
23. Emura, K., Hayashi, T.: Road-to-vehicle communications with time-dependent anonymity: A lightweight construction and its experimental results. IEEE Transactions on Vehicular Technology 67(2) (2017)
24. Franklin, M., Zhang, H.: Unique group signatures. In: ESORICS (2012)
25. Fuchsbauer, G.: Automorphic signatures in bilinear groups and an application to round-optimal blind signatures. In: Cryptology ePrint Archive 2009/320 (2009)
26. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156(16) (2008)
27. Garms, L., Lehmann, A.: Group signatures with selective linkability. In: PKC (2019)
28. Groth, J.: Simulation-sound nizk proofs for a practical language and constant size group signatures. In: Asiacrypt (2006)
29. Groth, J.: Fully anonymous group signatures without random oracles. In: Asiacrypt (2007)
30. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Eurocrypt (2008)
31. Hwang, J.Y., Lee, S., Chung, B.H., Cho, H.S., Nyang, D.: Short group signatures with controllable linkability. In: Workshop on Lightweight Security & Privacy: (LightSec) (2011)
32. Hwang, J.Y., Lee, S., Chung, B.H., Cho, H.S., Nyang, D.: Group signatures with controllable linkability for dynamic membership. Information Sciences 222 (2013)
33. Kiayias, A., Tsiounis, Y., Yung, M.: Traceable signatures. In: Eurocrypt (2004)
34. Kim, S., Park, S., Won, D.: Convertible group signatures. In: Asiacrypt (1996)

35. Krenn, S., Samelin, K., Striecks, C.: Practical group-signatures with privacy-friendly openings. In: ARES (2019)
36. Libert, B., Peters, T., Yung, M.: Scalable group signatures with revocation. In: Eurocrypt (2012)
37. Sakai, Y., Emura, K., Hanaoka, G., Kawai, Y., Matsuda, T., Omote, K.: Group signatures with message-dependent opening. In: PAIRING (2012)
38. Schnorr, C.P.: Efficient identification and signatures for smart cards. In: Crypto (1989)
39. Slamanig, D., Spreitzer, R., Unterluggauer, T.: Adding controllable linkability to pairing-based group signatures for free. In: International Conference on Information Security (2014)
40. Young, A., Yung, M.: Semantically secure anonymity: Foundations of re- encryption. In: SCN (2018)

```
$\mathsf{ADDU}(uid)$
if $uid \in \mathsf{HUL} \cup \mathsf{CUL}$   return $\perp$
$\mathsf{HUL} \leftarrow \mathsf{HUL} \cup \{uid\}, \mathbf{gsk}[uid] \leftarrow \perp$
$\mathbf{dec}^{uid} \leftarrow \mathsf{cont}, \mathsf{st}^{uid}_{\mathsf{Join}} \leftarrow gpk$
$\mathsf{st}^{uid}_{\mathsf{Issue}} \leftarrow (isk, gpk)$
$(\mathsf{st}^{uid}_{\mathsf{Join}}, M_{\mathsf{Issue}}, \mathbf{dec}^{uid}) \leftarrow \mathsf{CLS+.Join}(\mathsf{st}^{uid}_{\mathsf{Join}}, \perp)$
while $\mathbf{dec}^{uid} = \mathsf{cont}$
  $(\mathsf{st}^{uid}_{\mathsf{Issue}}, M_{\mathsf{Join}}, \mathbf{dec}^{uid}) \leftarrow \mathsf{CLS+.Issue}(\mathsf{st}^{uid}_{\mathsf{Issue}}, M_{\mathsf{Issue}})$
  $(\mathsf{st}^{uid}_{\mathsf{Join}}, M_{\mathsf{Issue}}, \mathbf{dec}^{uid}) \leftarrow \mathsf{CLS+.Join}(\mathsf{st}^{uid}_{\mathsf{Join}}, M_{\mathsf{Join}})$
if $\mathbf{dec}^{uid} = \mathsf{accept}$   $\mathbf{gsk}[uid] \leftarrow \mathsf{st}^{uid}_{\mathsf{Join}}$
return accept
```

```
$\mathsf{SIGN}(uid, m, bpk)$
if $uid \notin \mathsf{HUL}$ or $\mathbf{gsk}[uid] = \perp$   return $\perp$
$(\mu, \sigma) \leftarrow \mathsf{CLS+.Sign}(gpk, bpk, \mathbf{gsk}[uid], m)$
$\mathsf{SL} \leftarrow \mathsf{SL} \cup \{(uid, m, bpk)\}$
return $(\sigma, \mu)$
```

```
$\mathsf{SNDI}(uid, M_{\mathsf{in}})$
if $uid \in \mathsf{HUL}$   return $\perp$
if $uid \notin \mathsf{CUL}$   $\mathsf{CUL} \leftarrow \mathsf{CUL} \cup \{uid\}, \mathbf{dec}^{uid} \leftarrow \mathsf{cont}$
if $\mathbf{dec}^{uid} \neq \mathsf{cont}$   return $\perp$
if  undefined $\mathsf{st}^{uid}_{\mathsf{Issue}} \leftarrow (isk, gpk)$
$(\mathsf{st}^{uid}_{\mathsf{Issue}}, M_{\mathsf{out}}, \mathbf{dec}^{uid}) \leftarrow \mathsf{CLS+.Issue}(\mathsf{st}^{uid}_{\mathsf{Issue}}, M_{\mathsf{in}})$
return $(M_{\mathsf{out}}, \mathbf{dec}^{uid})$
```

```
$\mathsf{SNDU}(uid, M_{\mathsf{in}})$
if $uid \in \mathsf{CUL}$   return $\perp$
if $uid \notin \mathsf{HUL}$   $\mathsf{HUL} \leftarrow \mathsf{HUL} \cup \{uid\}$
  $\mathbf{gsk}[uid] \leftarrow \perp, M_{\mathsf{in}} \leftarrow \perp, \mathbf{dec}^{uid} \leftarrow \mathsf{cont}$
if $\mathbf{dec}^{uid} \neq \mathsf{cont}$   return $\perp$
if $\mathsf{st}^{uid}_{\mathsf{Join}}$ undefined    $\mathsf{st}^{uid}_{\mathsf{Join}} \leftarrow gpk$
$(\mathsf{st}^{uid}_{\mathsf{Join}}, M_{\mathsf{Out}}, \mathbf{dec}^{uid}) \leftarrow \mathsf{CLS+.Join}(\mathsf{st}^{uid}_{\mathsf{Join}}, M_{\mathsf{in}})$
if $\mathbf{dec}^{uid} = \mathsf{accept}$   $\mathbf{gsk}[uid] \leftarrow \mathsf{st}^{uid}_{\mathsf{Join}}$
return $(M_{\mathsf{Out}}, \mathbf{dec}^{uid})$
```

```
$\mathsf{CONVERT}((c\mu_1, c\sigma_1, c_1), \ldots, (c\mu_k, c\sigma_k, c_k), bpk, bsk)$
if $(bpk, bsk) \notin \mathcal{BK}$   return $\perp$
Compute $\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_k \leftarrow \mathsf{CLS+.Convert}(gpk, bpk, csk, \{(c\mu_i, c\sigma_i, c_i)\}_k)$
Parse permutation shuffling signatures in this run of CLS+.Convert as $\Pi$
$\forall i \in [1, k]$   $(\overline{\mu}_i, \overline{\sigma}_i, m_i) \leftarrow \mathsf{CLS+.Unblind}(bsk, (\overline{c\mu_{\Pi(i)}}, \overline{c\sigma_{\Pi(i)}}, \overline{c_{\Pi(i)}}))$,
  $\mathsf{UBL} \leftarrow \mathsf{UBL} \cup \{(\overline{\mu}_i, m_i, c_i, c\mu_i, c\sigma_i, bpk)\}$
if $\exists i \in [k]$ s.t $\mathsf{Identify}(uid^*_d, c_i, c\mu_i, c\sigma_i) = 1$ for some $d \in \{0, 1\}$ and $m_i = m^*$
  // Note $m^*$ is the challenged message in the anonymity requirement
  if $\exists j \in [k] \backslash \{i\}$ s.t. $\mathsf{Identify}(uid^*_d, c_j, c\mu_j, c\sigma_j) = 1$ for some $d \in \{0, 1\}$   return $\perp$
else return $(\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_k)$
```

Fig. 4. Oracles used in our CLS+ security games

A Omitted Definitions for CLS+

A.1 Oracles in CLS+

In Figure 4, we present in full the omitted oracles in our CLS+ model.

A.2 Helper Algorithms

We now present the algorithms Identify and UnLink in full.

```
$\mathsf{Identify}(gpk, bpk, csk, bsk, uid, c, c\mu, c\sigma)$
$(\mu', \sigma') \leftarrow \mathsf{CLS+.Sign}(gpk, bpk, \mathbf{gsk}[uid], 0), (c\mu', c\sigma', c') \leftarrow \mathsf{CLS+.Blind}(gpk, bpk, (\mu', \sigma', 0))$
if $\mathsf{UnLink}(gpk, csk, bpk, bsk, ((c\mu, c\sigma, c), (c\mu', c\sigma', c'))) = 0$   return $1$   else return $0$
```

```
$\mathsf{UnLink}(gpk, csk, bpk, bsk, ((c\mu_1, c\sigma_1, c_1), \cdots, (c\mu_k, c\sigma_k, c_k)))$
$\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_k \leftarrow \mathsf{CLS+.Convert}(gpk, bpk, csk, \{(c\mu_i, c\sigma_i, c_i)\}_k)$
$\forall i \in [1, k]$   $(\overline{\mu}_i, \overline{\sigma}_i, m_i) \leftarrow \mathsf{CLS+.Unblind}(bsk, (\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i))$
if $\exists i \in [1, k]$ s.t $\mathsf{CLS+.Verify}(\mathtt{tier\text{-}2}, gpk, bpk, \overline{m}_i, \overline{\mu}_i, \overline{\sigma_i}) = 0$   return $\perp$
if $\exists (i, j)$ with $i \neq j$ s.t. $\overline{\mu}_i = \overline{\mu}_j$   return $0$   else return $1$
```

When the keys input are clear from context, we often write:
$\mathsf{Identify}(uid, c, c\mu, c\sigma)$ and $\mathsf{UnLink}((c_1, c\mu, c\sigma_1), \cdots, (c_k, c\mu_k, c\sigma_k))$.

A.3 Correctness and Consistency of CLS+

The detailed definitions for correctness and consistency are given in Figure 5.

Definition 1 (Correctness). *A CLS+ scheme satisfies correctness if for all adversaries* $\mathcal{A}$, $\Pr[\mathbf{Exp}^{corr-sig}_{\mathcal{A},CLS+}(\tau) = 1] = 0$, *and* $\Pr[\mathbf{Exp}^{corr-conv}_{\mathcal{A},CLS+}(\tau) = 1] \leqslant \mathsf{negl}(\tau)$.

Definition 2 (Consistency). *A CLS+ scheme satisfies consistency if for all adversaries* $\mathcal{A}$, $\Pr[\mathbf{Exp}^{consist-link}_{\mathcal{A},CLS+}(\tau) = 1] \leqslant \mathsf{negl}(\tau)$, *and* $\Pr[\mathbf{Exp}^{consist-verif}_{\mathcal{A},CLS+}(\tau) = 1] \leqslant \mathsf{negl}(\tau)$.

A.4 Security requirements for CLS+

We provide the CLS+ security requirements in full.

Definition 3 (CLS+ Anonymity). *A CLS+ scheme satisfies anonymity if for all ppt adversaries* $\mathcal{A}$ *the following advantage is negligible in* τ :
$\left|\Pr[\mathbf{Exp}^{anon-0}_{\mathcal{A},CLS+}(\tau) = 1] - \Pr[\mathbf{Exp}^{anon-1}_{\mathcal{A},CLS+}(\tau) = 1]\right|$.

Experiment: $\mathbf{Exp}^{anon-b}_{\mathcal{A},\mathsf{CLS+}}(\tau)$

```
$\mathsf{pp} \leftarrow \mathsf{CLS+.Setup}(1^\tau), (ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp}), (cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp})$
$gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$(uid_0^*, uid_1^*, m^*, bpk^*, st) \leftarrow \mathcal{A}^{\mathsf{SNDU,SIGN,CONVERT}}(\mathsf{choose}, gpk, isk)$
$\textbf{if } uid_0^* \notin \mathsf{HUL} \text{ or } \mathbf{gsk}[uid_0^*] = \perp \text{ or } uid_1^* \notin \mathsf{HUL} \text{ or } \mathbf{gsk}[uid_1^*] = \perp \quad \textbf{return } \perp$
$(\mu^*, \sigma^*) \leftarrow \mathsf{CLS+.Sign}(gpk, bpk^*, \mathbf{gsk}[uid_b^*], m^*)$
$b^* \leftarrow \mathcal{A}^{\mathsf{SNDU,SIGN,CONVERT}}(\mathsf{guess}, st, \mu^*, \sigma^*) \quad \textbf{return } b^*$
```

Definition 4 (CLS+ Non-Transitivity). *A CLS+ scheme satisfies non-transitivity if for all ppt adversaries* $\mathcal{A}$ *there exists an efficient simulator SIM such that the following advantage is negligible in* τ*:*

$$\left|\Pr[\mathbf{Exp}^{nontrans-0}_{\mathcal{A},CLS+}(\tau) = 1] - \Pr[\mathbf{Exp}^{nontrans-1}_{\mathcal{A},CLS+}(\tau) = 1]\right.$$

Experiment: $\mathsf{Exp}^{corr-sig}_{\mathcal{A},\mathsf{CLS}}(\tau)$

```
$\mathsf{pp} \leftarrow \mathsf{CLS+.Setup}(1^\tau), (ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp}), (cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp})$
$(bpk, bsk) \leftarrow \mathsf{CLS+.BKGen}(\mathsf{pp}), gpk \leftarrow (\mathsf{pp}, ipk, cpk), \mathsf{HUL},\ \mathsf{CUL} \leftarrow \emptyset$
$(uid, m) \leftarrow \mathcal{A}^{\mathsf{ADDU}}(gpk),$ if $\mathbf{gsk}[uid] = \perp$   return $0$
$(\mu, \sigma) \leftarrow \mathsf{CLS+.Sign}(gpk, bpk, \mathbf{gsk}[uid], m)$
if $\mathsf{CLS+.Verify}(\mathtt{tier}\text{-}1, gpk, bpk, m, \mu, \sigma) = 0$   return $1$   else return $0$
```

Experiment: $\mathsf{Exp}^{corr-conv}_{\mathcal{A},\mathsf{CLS}}(\tau)$

```
$\mathsf{pp} \leftarrow \mathsf{CLS+.Setup}(1^\tau), (ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp}), (cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp}),$
$(bpk, bsk) \leftarrow \mathsf{CLS+.BKGen}(\mathsf{pp}), gpk \leftarrow (\mathsf{pp}, ipk, cpk), \mathsf{HUL},\ \mathsf{CUL} \leftarrow \emptyset$
$((uid_1, m_0), ..., (uid_k, m_k)) \leftarrow \mathcal{A}^{\mathsf{ADDU}}(gpk)$
if $\exists i \in [1,k]$ st $\mathbf{gsk}[uid_i] = \perp$   return $0$
$\forall i \in [1,k]$   $(\mu_i, \sigma_i) \leftarrow \mathsf{CLS+.Sign}(gpk, bpk, \mathbf{gsk}[uid_i], m_i)$
$\forall j \in [1,k]$   $(c\mu_j, c\sigma_j, c_j) \leftarrow \mathsf{CLS+.Blind}(gpk, bpk, (\mu_j, \sigma_j, m_j))$
$\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_k \leftarrow \mathsf{CLS+.Convert}(gpk, bpk, csk, \{(c\mu_i, c\sigma_i, c_i)\}_k)$
$\forall j \in [1,k]$   $(\overline{\mu}_j, \overline{\sigma_j}, \overline{m}_j) \leftarrow \mathsf{CLS+.Unblind}((\overline{c\mu}_j, c\sigma_j, \overline{c}_j), bsk)$
if $\exists j \in [1,k]$ s.t $\mathsf{CLS+.Verify}(\mathtt{tier}\text{-}2, gpk, bpk, \overline{m}_j, \overline{\mu}_j, \overline{\sigma}_j) = 0$   return $1$
if $\exists$ permutation $\Pi : [1,k] \to [1,k]$ s.t.   return $0$
  1.$\forall i \in [1,k]$   $m_{\Pi(i)} = m_i$
  2.$\forall (i,j) \in [1,k]$ with $uid_i = uid_j$   $\overline{\mu}_{\Pi(i)} = \overline{\mu}_{\Pi(j)}$
  3.$\forall (i,j) \in [1,k]$ with $uid_i \neq uid_j$   $\overline{\mu}_{\Pi(i)} \neq \overline{\mu}_{\Pi(j)}$
else return $1$
```

Experiment: $\mathsf{Exp}^{consist-link}_{\mathcal{A},\mathsf{CLS}}(\tau)$

```
$\mathsf{pp} \leftarrow \mathsf{CLS+.Setup}(1^\tau), (ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp}), (cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp}),$
$(bpk, bsk) \leftarrow \mathsf{CLS+.BKGen}(\mathsf{pp}), gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$((c_0, c\mu_0, c\sigma_0), (c_1, c\mu_1, c\sigma_1), (c_2, c\mu_2, c\sigma_2)) \leftarrow \mathcal{A}(gpk, isk, csk, bsk)$
if $\mathsf{UnLink}((c_0, c\mu_0, c\sigma_0), (c_1, c\mu_1, c\sigma_1), 0) \neq 0$ or   $\mathsf{UnLink}((c_1, c\mu_1, c\sigma_1), (c_2, c\mu_2, c\sigma_2), 0) \neq 0$
  return $0$
if $\mathsf{UnLink}((c_0, c\mu_0, c\sigma_0), (c_2, c\mu_2, c\sigma_2), 0) = 1$   return $1$   else return $0$
```

Experiment: $\mathsf{Exp}^{consist-verif}_{\mathcal{A},\mathsf{CLS}}(\tau)$

```
$\mathsf{pp} \leftarrow \mathsf{CLS+.Setup}(1^\tau), (ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp}), (cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp})$
$gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$((m_0, \mu_0, \sigma_0), \cdots, (m_k, \mu_k, \sigma_k), bpk, bsk) \leftarrow \mathcal{A}(gpk, isk, csk)$
if $\exists j \in [k]$ such that $\mathsf{CLS+.Verify}(\mathtt{tier}\text{-}1, gpk, bpk, m_j, \mu_j, \sigma_j) = 0$ or $(bpk, bsk) \notin \mathcal{BK}$   return $0$
$\forall i \in [1,k]$   $(c\mu_i, c\sigma_i, c_i) \leftarrow \mathsf{CLS+.Blind}(gpk, bpk, (\mu_i, \sigma_i, m_i))$
$\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_k \leftarrow \mathsf{CLS+.Convert}(gpk, bpk, csk, \{(c\mu_i, c\sigma_i, c_i)\}_k)$
$\forall i \in [1,k]$   $(\overline{\mu}_i, \overline{\sigma_i}, \overline{m}_i) \leftarrow \mathsf{CLS+.Unblind}((\overline{c\mu}_i, c\sigma_i, \overline{c}_i), bsk)$
if $\{\overline{m}_i\}_k \neq \{m_i\}_k$ or $\exists j \in [k]$ such that $\mathsf{CLS+.Verify}(\mathtt{tier}\text{-}2, gpk, bpk, \overline{m}_j, \overline{\mu}_j, \overline{\sigma_j}) = 0$   return $1$
else return $0$
```

Fig. 5. Security games for correctness of CLS

```
Experiment: $\mathsf{Exp}^{nontrans-b}_{\mathcal{A},\mathsf{CLS+}}(\tau)$
$\mathsf{pp} \leftarrow \mathsf{CLS+.Setup}(1^\tau), (ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp})$
$(cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp}), gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$b^* \leftarrow \mathcal{A}^{\mathsf{SNDU,SIGN,CONVX}}(\mathsf{guess}, gpk, isk)$   return $b^*$
  where the oracle CONVX works as follows:
  if $b = 0$ (real world)  then  CONVX is the standard CONVERT oracle
  if $b = 1$ (ideal world)  then  CONVX is the simulated CONVSIM oracle
```

```
$\mathsf{CONVSIM}((c\mu_1, c\sigma_1, c_1), \ldots, (c\mu_k, c\sigma_k, c_k), bpk, bsk)$
if $(bpk, bsk) \notin \mathcal{BK}$ or $\mathsf{CLS+.Convert}(gpk, bpk, csk, (c\mu_1, c\sigma_1, c_1), \ldots, (c\mu_k, c\sigma_k, c_k)) = \perp$
  return $\perp$
$\mathsf{CL} \leftarrow \emptyset, \forall i \in [1, k]$
  if $\exists uid \in \mathsf{HUL}$ s.t $\mathsf{Identify}(bpk, bsk, uid, c_i, c\mu_i, c\sigma_i) = 1$
    if $\mathsf{L}_{uid}$ does not exist, create $\mathsf{L}_{uid} \leftarrow \{c_i\}$ else  set $\mathsf{L}_{uid} \leftarrow \mathsf{L}_{uid} \cup \{c_i\}$
  else $\mathsf{CL} \leftarrow \mathsf{CL} \cup \{(c_i, c\sigma_i, c\mu_i)\}$
$(\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_{i=1,\ldots k'}) \leftarrow \mathsf{CLS+.Convert}(gpk, bpk, csk, \mathsf{CL})$ for $k' \leftarrow |\mathsf{CL}|$
Let $\mathsf{L}_{uid_1}, \ldots \mathsf{L}_{uid_{k''}}$ be the non-empty message clusters created above
$\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_{i=k'+1,\ldots k} \leftarrow \mathsf{SIM}(gpk, bpk, csk, \mathsf{L}_{uid_1}, \ldots \mathsf{L}_{uid_{k''}})$
Let $\{(\overline{c\mu}'_i, \overline{c\sigma}'_i, \overline{c}'_i)\}_{i=1,\ldots k}$ be a random permutation of $\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_{i=1,\ldots k}$
return $(\{(\overline{c\mu}'_i, \overline{c\sigma}'_i, \overline{c}'_i)\}_{i=1,\ldots k})$
```

Definition 5 (CLS+ Conversion Blindness). *A CLS+ scheme satisfies conversion blindness if: for all polynomial time adversaries $\mathcal{A}$ the following advantage is negligible in τ:*

$$\left|\Pr[\mathsf{Exp}^{blind-conv-0}_{\mathcal{A},CLS+}(\tau) = 1] - \Pr[\mathsf{Exp}^{blind-conv-1}_{\mathcal{A},CLS+}(\tau) = 1]\right|.$$

```
$\mathsf{UNBLIND}((\mu_1, \sigma_1, m_1), \ldots, (\mu_k, \sigma_k, m_k))$
if $\exists i \in [k]$ s.t $\mathsf{CLS+.Verify}(\texttt{tier-}1, gpk, bpk, m_i, \mu_i, \sigma_i) = 0$   return $\perp$
$\forall i \in [k]$   $(c\mu_i, c\sigma_i, c_i) \leftarrow \mathsf{CLS+.Blind}(gpk, bpk, (\mu_i, \sigma_i, m_i; r_i))$
$\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_k \leftarrow \mathsf{CLS+.Convert}(gpk, bpk, csk, \{(c\mu_i, c\sigma_i, c_i)\}_k; r')$
$\forall i \in [1, k]$   $(\overline{\mu}_i, \overline{\sigma_i}, m_i) \leftarrow \mathsf{CLS+.Unblind}(bsk, (\overline{c\mu}_i, \overline{c\sigma_i}, \overline{c_i}))$
return $(\{(\overline{\mu}_i, \overline{\sigma_i}, m_i)\}_k, \{(r_i)\}_k, r')$
```

```
Experiment: $\mathsf{Exp}^{blind-conv-b}_{\mathcal{A},\mathsf{CLS+}}(\tau)$
$\mathsf{pp} \leftarrow \mathsf{CLS+.Setup}(1^\tau), (ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp})$
$(cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp}), (bpk, bsk) \leftarrow \mathsf{CLS+.BKGen}(\mathsf{pp}), gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$(st, (\mu_0, \sigma_0, m_0), (\mu_1, \sigma_1, m_1)) \leftarrow \mathcal{A}^{\mathsf{UNBLIND}}(\mathsf{choose}, gpk, bpk, isk, csk)$
if $\exists d \in \{0, 1\}$ s.t $\mathsf{CLS+.Verify}(\texttt{tier-}1, gpk, bpk, m_d, \mu_d, \sigma_d) = 0$   return $0$
$(c\mu^*, c\sigma^*, c^*) \leftarrow \mathsf{CLS+.Blind}(gpk, bpk, (\mu_b, \sigma_b, m_b))$
$b^* \leftarrow \mathcal{A}^{\mathsf{UNBLIND}}(\mathsf{guess}, st, c\mu^*, c\sigma^*, c^*)$   return $b^*$
```

Definition 6 (Conversion Unforgeability). *A CLS+ scheme satisfies conversion unforgeability if for all ppt adversaries $\mathcal{A}$, the advantage $\Pr[\mathsf{Exp}^{uf-conv}_{\mathcal{A},CLS+}(\tau) = 1]$ is negligible in τ.*

```
Experiment: $\mathsf{Exp}^{uf-conv}_{\mathcal{A},\mathsf{CLS+}}(\tau)$
$\mathsf{pp} \leftarrow \mathsf{CLS+.Setup}(1^\tau), (ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp})$
$(cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp}), gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$(\overline{\mu}, \overline{\sigma}, m, bpk) \leftarrow \mathcal{A}^{\mathsf{CONVERT}}(gpk, isk)$
return 1 if all of the following conditions are satisfied:
  $\mathsf{CLS+.Verify}(\mathtt{tier}\text{-}2, gpk, bpk, m, \overline{\mu}, \overline{\sigma}) = 1$ and $(\overline{\mu}, m, \cdot, \cdot, \cdot, bpk) \notin \mathsf{UBL}$
```

Definition 7 (CLS+ Traceability). *A CLS+ scheme satisfies traceability if for all ppt adversaries $\mathcal{A}$, the advantage* $\Pr[\mathsf{Exp}^{trace}_{\mathcal{A},\mathsf{CLS+}}(\tau) = 1]$ *is negligible in τ.*

```
Experiment: $\mathsf{Exp}^{trace}_{\mathcal{A},\mathsf{CLS+}}(\tau)$
$\mathsf{pp} \leftarrow \mathsf{CLS+.Setup}(1^\tau), (ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp})$
$(cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp}), gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$(\{(c_i, c\mu_i, c\sigma_i)\}_{i=1,\cdots,k}, bpk, bsk) \leftarrow \mathcal{A}^{\mathsf{ADDU,SNDI,SIGN}}(gpk, csk)$
if $(bpk, bsk) \notin \mathcal{BK}$   return 0
$\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_k \leftarrow \mathsf{CLS+.Convert}(gpk, bpk, csk, \{(c\mu_i, c\sigma_i, c_i)\}_k)$
$\forall i \in [1, k]$   $(\overline{\mu}_i, \overline{\sigma}_i, m_i) \leftarrow \mathsf{CLS+.Unblind}(bsk, (\overline{c\mu_i}, \overline{c\sigma_i}, \overline{c}_i))$
$L \leftarrow 0, \forall uid \in \mathsf{HUL}$   if $\exists i \in [k]$ s.t $(uid, m_i, bpk) \in \mathsf{SL}$   $L \leftarrow L + 1$
return 1 if all of the following conditions are satisfied:
  $k > |\mathsf{CUL}| + L$ and   $\forall (i, j) \in [k]$ s.t $i \neq j$   $\overline{\mu}_i \neq \overline{\mu}_j$ and
  $\forall i \in [k]$   $\mathsf{CLS+.Verify}(\mathtt{tier}\text{-}2, gpk, bpk, m_i, \overline{\mu}_i, \overline{\sigma}_i) = 1$
```

Definition 8 (CLS+ Non-Frameability). *A CLS+ scheme satisfies non-frameability if for all polynomial time adversaries $\mathcal{A}$, the advantage* $\Pr[\mathsf{Exp}^{nonframe}_{\mathcal{A},\mathsf{CLS+}}(\tau) = 1]$ *is negligible in τ.*

```
Experiment: $\mathsf{Exp}^{nonframe}_{\mathcal{A},\mathsf{CLS+}}(\tau)$
$\mathsf{pp} \leftarrow \mathsf{CLS+.Setup}(1^\lambda), (ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp}), (cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp})$
$gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$(uid, (c, c\mu, c\sigma), bpk, bsk) \leftarrow \mathcal{A}^{\mathsf{SNDU,SIGN}}(gpk, isk, csk)$
if $(bpk, bsk) \notin \mathcal{BK}$   return 0
$(\overline{\mu}, \overline{\sigma}, m) \leftarrow \mathsf{CLS+.Unblind}(bsk, \mathsf{CLS+.Convert}(gpk, bpk, csk, (c, c\mu, c\sigma)))$
return 1 if all of the following conditions are satisfied:
  $\mathsf{Identify}(uid, c, c\mu, c\sigma) = 1$ where $uid \in \mathsf{HUL}$ and $((uid, m, bpk) \notin \mathsf{SL}$
```

B Reduction of CLS unforgeability definitions to CLS+ definitions

We provide reductions that show our CLS+ unforgeability requirements imply the CLS unforgeability requirements in our adjusted setting. We how that our CLS+ model ensures the traceability and non-frameability requirements from the CLS model, adjusted to our updated setting. This ensures that our unforgeability guarantees also hold for tier-1 signatures before blinding. We now give the adjusted CLS requirements.

Traceability (of `tier`-1 signatures) We now update the traceability requirement from the CLS model to our new setting. This ensures that the unforgeability guarantees hold for `tier`-1 signatures before blinding, when the converter, data lake and data processor are corrupted and the issuer is honest. More precisely, we need that an adversary should not be able to output more valid `tier`-1 signatures that will be unlinkable upon an honest conversion than the number of corrupted users. Unlike for CLS, as the blinding public key must be fixed in signing, this must be output by the adversary. The corresponding blinding secret key must also be output to allow for the signatures to be unblinded after conversion. Due to the re-randomisability of signatures to ensure non-transitivity, we cannot simply restrict signatures obtained from the signing oracle from being output. Instead, we allow the adversary to output the signatures of honest users. However, similarly to in the CLS+ requirement, for each honest user that could have output a signature, we increase the number of unlinkable signatures required by 1. To match this notation, we also explicitly blind, convert and unblind signatures instead of using the UnLink algorithm. This avoids checking the validity of `tier`-2 signatures which should not be necessary for the adversary to win. This ensures an attack is captured where the adversary does not corrupt any users but still outputs a valid `tier`-1 signature without using the signing oracle.

Definition 9 (CLS+ `tier`-1 Traceability). *A CLS+ scheme satisfies `tier`-1 traceability if for all polytime adversaries $\mathcal{A}$, the advantage* $\Pr[\mathbf{Exp}^{trace-tier1}_{\mathcal{A},CLS+}(\tau) = 1]$ *is negligible in* τ.

```
Experiment: $\mathbf{Exp}^{trace-tier1}_{\mathcal{A},\mathsf{CLS+}}(\tau)$
$\mathsf{pp} \leftarrow \mathsf{CLS+.Setup}(1^\tau), (ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp}), (cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp})$
$gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$((m_1, \mu_1, \sigma_1), ..., (m_k, \mu_k, \sigma_k), bpk, bsk) \leftarrow \mathcal{A}^{\mathsf{ADDU,SNDI,SIGN}}(gpk, csk)$
$\textbf{if } (bpk, bsk) \notin \mathcal{BK} \quad \textbf{return } 0$
$\forall i \in [k] \quad (c\mu_i, c\sigma_i, c_i) \leftarrow \mathsf{CLS+.Blind}(gpk, bpk, (\mu_i, \sigma_i, m_i))$
$\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_k \leftarrow \mathsf{CLS+.Convert}(gpk, bpk, csk, \{(c\mu_i, c\sigma_i, c_i)\}_k)$
$\forall i \in [k] \quad (\overline{\mu}_i, \overline{\sigma}_i, \overline{m}_i) \leftarrow \mathsf{CLS+.Unblind}(bsk, (\overline{c\mu_i}, \overline{c\sigma_i}, \overline{c}_i))$
$L \leftarrow 0, \forall uid \in \mathsf{HUL} \quad \textbf{if } \exists i \in [k] \text{ s.t } (uid, m_i, bpk) \in \mathsf{SL} \quad L \leftarrow L + 1$
$\textbf{return } 1 \text{ if all of the following conditions are satisfied:}$
    $k > |\mathsf{CUL}| + L \text{ and } \forall (i, j) \in [k] \text{ s.t } i \neq j \quad \overline{\mu}_i \neq \overline{\mu}_j \text{ and}$
    $\forall i \in [k] \quad \mathsf{CLS+.Verify}(\mathtt{tier}\text{-}1, gpk, bpk, m_i, \mu_i, \sigma_i) = 1$
```

Non-Frameability (of `tier`-1 signatures). We now update the non-frameability requirement from the CLS model to our new setting. This ensures that the unforgeability guarantees hold for `tier`-1 signatures before blinding, when the converter, issuer, data lake and data processor are corrupted. More precisely, we need that an adversary should not be able to output a valid `tier`-1 signatures that will identify to an honest user. Again, unlike in the CLS model, the blinding public and secret key must be output because these are set during signing and $\mathsf{Identify}$ is now input blinded signatures. Also, we now require that the message should not have been input to the signing oracle, due to the re-randomisability of signatures. The $\mathsf{Identify}$ algorithm also now checks that the resulting $\texttt{tier}$-2 signatures are valid.

Definition 10 ($\mathsf{CLS+}$ $\texttt{tier}$-1 Non-Frameability). *A CLS+ scheme satisfies $\texttt{tier}$-1 non-frameability if for all polytime adversaries $\mathcal{A}$, the advantage* $\Pr[\mathbf{Exp}^{nonframe-tier1}_{\mathcal{A},CLS+}(\tau) = 1]$ *is negligible in τ.*

```
Experiment: $\mathbf{Exp}^{nonframe-tier1}_{\mathcal{A},\mathsf{CLS+}}(\tau)$
$\mathsf{pp} \leftarrow \mathsf{CLS+.Setup}(1^\tau), (ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp}), (cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp})$
$gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$(uid, m^*, \mu^*, \sigma^*, bpk, bsk) \leftarrow \mathcal{A}^{\mathsf{SNDU},\mathsf{SIGN}}(gpk, isk, csk)$
**if** $(bpk, bsk) \notin \mathcal{BK}$   **return** 0
**return** 1 if all of the following conditions are satisfied:
  $\mathsf{CLS+.Verify}(\texttt{tier-1}, gpk, bpk, m^*, \mu^*, \sigma^*) = 1$ and
  $\mathsf{Identify}(uid, \mathsf{CLS+.Blind}(gpk, bpk, \mu^*, \sigma^*, m^*)) = 1$ where $uid \in \mathsf{HUL}$ and
  $(uid, m^*, bpk) \notin \mathsf{SL}$
```

Reduction from $\texttt{tier}$-1 Traceability to $\mathsf{CLS+}$ Traceability We build an adversary $\mathcal{A}'$ that wins in the $\mathsf{CLS+}$ traceability game, given $\mathcal{A}$ that wins in the $\texttt{tier}$-1 traceability game. We give $\mathcal{A}'$ in Figure 6, and below explain why the simulation input to $\mathcal{A}$ given in Figure 6 is identically distributed to the $\texttt{tier}$-1 traceability experiment, and that $\mathcal{A}'$ successfully breaks $\mathsf{CLS+}$ traceability.

The inputs to the adversary in both the $\mathsf{CLS+}$ and $\texttt{tier}$-1 traceability experiments are the same. Therefore, the inputs to $\mathcal{A}$ are identical to in the $\texttt{tier}$-1 traceability game.

Reduction to CLS+ Traceability We assume $\mathcal{A}$ is successful. Therefore, $(bpk, bsk) \in \mathcal{BK}$ and $\forall(i,j) \in [k]$ s.t $i \neq j \quad \overline{\mu}_i \neq \overline{\mu}_j$.

We need to ensure that $k > |\mathsf{CUL}| + L$. The lists of corrupted users, honest users and outputs from the signing oracle are the same in both games. Let $\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_k \leftarrow \mathsf{CLS+.Convert}(gpk, bpk, csk, \{(c\mu_i, c\sigma_i, c_i)\}_k)$ and for all $i \in [1,k]$ let $(\overline{\mu}_i, \overline{\sigma}_i, \overline{m}_i) = \mathsf{CLS+.Unblind}(bsk, (\overline{c\mu_i}, \overline{c\sigma}_i, \overline{c}_i))$. Due to the consistency of conversions, $\{\overline{m}_i\}_k = \{m_i\}_k$. Therefore, the value of L will be the same in both games, and so $k > |\mathsf{CUL}| + L$.

Due to the consistency of conversions and that for all $i \in [k]$
$\mathsf{CLS+.Verify}(\texttt{tier-1}, gpk, bpk, m_i, \mu_i, \sigma_i) = 1$, for all $i \in [k]$
$\mathsf{CLS+.Verify}(\texttt{tier-2}, gpk, bpk, \overline{m}_i, \overline{\mu}_i, \overline{\sigma}_i) = 1$.

Therefore, $\mathcal{A}'$ successfully breaks $\mathsf{CLS+}$ traceability.

Reduction from $\texttt{tier}$-1 Non-Frameability to $\mathsf{CLS+}$ Non-Frameability We build an adversary $\mathcal{A}'$, that successfully wins in the $\mathsf{CLS+}$ non-frameability

```
$\mathsf{ADDU}(uid)$
$\textbf{return } \mathsf{ADDU}'(uid)$

$\mathsf{SNDI}(uid, M_{\mathsf{in}})$
$\textbf{return } \mathsf{SNDI}'(uid, M_{\mathsf{in}})$

$\mathsf{SIGN}(uid, m, bpk)$
$\textbf{return } \mathsf{SIGN}'(uid, m, bpk)$

$\mathcal{A}'^{\mathsf{ADDU}',\mathsf{SNDI}',\mathsf{SIGN}'}(gpk, csk)$
$((m_1, \mu_1, \sigma_1), ..., (m_k, \mu_k, \sigma_k), bpk, bsk) \leftarrow \mathcal{A}^{\mathsf{ADDU},\mathsf{SNDI},\mathsf{SIGN}}(gpk, csk)$
$\forall i \in [k] \quad (c\mu_i, c\sigma_i, c_i) \leftarrow \mathsf{CLS+.Blind}(gpk, bpk, \mu_i, \sigma_i, m_i)$
$\textbf{return } (\{(c_i, c\mu_i, c\sigma_i)\}_{i=1,\cdots,k}, bpk, bsk)$
```

Fig. 6. $\mathcal{A}'$ which wins in the CLS+ traceability game

game, given $\mathcal{A}$ that wins in the `tier`-1 non-frameability game. We give $\mathcal{A}'$ in Figure 7, and below explain why the simulation input to $\mathcal{A}$ given in Figure 7 is identically distributed to the `tier`-1 non-frameability experiment, and that $\mathcal{A}'$ successfully breaks CLS+ non-frameability.

```
$\mathsf{SNDU}(uid, M_{\mathsf{in}})$
$\textbf{return } \mathsf{SNDU}'(uid, M_{\mathsf{in}})$

$\mathsf{SIGN}(uid, m, bpk)$
$\textbf{return } \mathsf{SIGN}'(uid, m, bpk)$

$\mathcal{A}'^{\mathsf{SNDU}',\mathsf{SIGN}'}(gpk, isk, csk)$
$(uid, m^*, \mu^*, \sigma^*, bpk, bsk) \leftarrow \mathcal{A}^{\mathsf{SNDU},\mathsf{SIGN}}(gpk, isk, csk)$
$(c\mu, c\sigma, c) \leftarrow \mathsf{CLS+.Blind}(gpk, bpk, \mu^*, \sigma^*, m^*)$
$\textbf{return } (uid, (c, c\mu, c\sigma), bpk, bsk)$
```

Fig. 7. $\mathcal{A}'$ which wins in the CLS+ non-frameability game

The inputs to the adversary in both the CLS+ and `tier`-1 non-frameability experiments are the same. Therefore, the inputs to $\mathcal{A}$ are identical to in the `tier`-1 non-frameability game.

Reduction to $\mathsf{CLS+}$ *Non-Frameability* We assume $\mathcal{A}$ is successful. Therefore, $(bpk, bsk) \in \mathcal{BK}$ and $uid \in \mathsf{HUL}$. Also, $\mathsf{Identify}(uid, \mathsf{CLS+.Blind}(gpk, bpk, \mu^*, \sigma^*, m^*)) = 1$ and $(uid, m^*, bpk) \notin \mathsf{SL}$. We need to ensure that for $(\overline{\mu}, \overline{\sigma}, \overline{m}) =$ $\mathsf{CLS+.Unblind}(bsk, \mathsf{CLS+.Convert}(gpk, bpk, csk, (c, c\mu, c\sigma)))$ that $(uid, \overline{m}, bpk) \notin$ SL. Due to the consistency of conversions and the fact that $\mathsf{CLS+.Verify}(\mathtt{tier}\text{-}1,$ $gpk, bpk, m^*, \mu^*, \sigma^*) = 1$, $\overline{m} = m^*$. Therefore, $\mathcal{A}'$ successfully breaks $\mathsf{CLS+}$ non-frameability.

C Controlled Malleable NIZKs

C.1 Controlled Malleable Simulation Soundness Extractability

Definition 11. *Let* $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V}, \mathsf{ZKEval})$ *be a NIZKPoK system for an efficient relation* $\mathcal{R}$*, with a simulator* (S_1, S_2) *and an extractor* (E_1, E_2)*. Let* $\mathcal{T}$ *be an allowable set of unary transformations for the relation* $\mathcal{R}$ *such that membership in* $\mathcal{T}$ *is efficiently testable. Let* SE_1 *be an algorithm that, on input* (1^τ)*, outputs* $(\sigma_{\mathsf{crs}}, \tau_s, \tau_e)$ *such that* $(\sigma_{\mathsf{crs}}, \tau_s)$ *is distributed identically to the output of* S_1*. Let* $\mathcal{A}$ *be given, and consider the following game:*

- *Step 1.* $(\sigma_{\mathsf{crs}}, \tau_s, \tau_e) \leftarrow SE_1(1^\tau)$
- *Step 2.* $(x, \pi) \leftarrow \mathcal{A}^{S_2(\sigma_{\mathsf{crs}}, \tau_s, \cdot)}(\sigma_{\mathsf{crs}}, \tau_e)$
- *Step 3.* $(w, x', T) \leftarrow E_2(\sigma_{\mathsf{crs}}, \tau_e, x, \pi)$.

The proof system satisfies controlled-malleable simulation-sound extractability (CM-SSE) with respect to $\mathcal{T}$ *if for all PPT algorithms* $\mathcal{A}$ *there exists a negligible function* negl *such that the probability (over the choices of* SE_1*,* $\mathcal{A}$*, and* S_2*) that* $\mathcal{V}(\sigma_{\mathsf{crs}}, x, \pi) = 1$ *and* $(x, \pi) \notin Q$ *(where* Q *is the set of queried statements and their responses) but either (1)* $w \neq \perp$ *and* $(x, w) \notin \mathcal{R}$*; (2)* $(x', T) \neq (\perp, \perp)$ *and either* $x' \notin Q_{\mathsf{inst}}$ *(the set of queried instances),* $x \neq T_{\mathsf{inst}}(x')$*, or* $T \notin \mathcal{T}$ *; or (3)* $(w, x', T) = (\perp, \perp, \perp)$ *is at most* $\mathsf{negl}(\tau)$*.*

C.2 Strong Derivation Privacy

Definition 12. *For a malleable NIZK* $(\mathsf{CRSSetup}, \mathcal{P}, \mathcal{V}, \mathsf{ZKEval})$ *with an associated simulator* (S_1, S_2)*, a given adversary* $\mathcal{A}$*, and bit* b*, let* $p_b^{\mathcal{A}}(\tau)$ *be the probability of the event that* $b' = 0$ *in the following game:*

- *Step 1.* $(\sigma_{\mathsf{crs}}, \tau_s) \leftarrow S_1(1^\tau)$.
- *Step 2.* $(\mathsf{state}, x_1, \pi_1, ..., x_q, \pi_q, T) \leftarrow \mathcal{A}(\sigma_{\mathsf{crs}}, \tau_s)$.
- *Step 3. If* $\mathcal{V}(\sigma_{\mathsf{crs}}, x_i, \pi_i) = 0$ *for some* i*,* $(x_1, \cdots, x_q)$ *is not in the domain of* T_{inst}*, or* $T \notin \mathcal{T}$*, abort and output* $\perp$*. Otherwise, form*

$$\pi \leftarrow \begin{cases} S_2(\sigma_{\mathsf{crs}}, \tau_s, T_{\mathsf{inst}}(x_1, \cdots, x_q)), & \text{if } b = 0. \\ \mathsf{ZKEval}(\sigma_{\mathsf{crs}}, T, \{x_i, \pi_i\}_{i \in [q]}), & \text{if } b = 1. \end{cases}$$

- *Step 4.* $b' \leftarrow \mathcal{A}(\mathsf{state}, \pi)$.

The proof system is strongly derivation private if for all PPT algorithms $\mathcal{A}$ *there exists a negligible function* negl *such that* $|p_0^{\mathcal{A}}(\tau) - p_1^{\mathcal{A}}(k)| < \mathsf{negl}(\tau)$*.*

D Proofs of Security

We now prove that our CLS–CM construction satisfies the Correctness, Consistency, Anonymity, Non-Transitivity, Conversion Blindness, Conversion Unforgeability, Non-Frameability and Traceability requirements given in Section 2, i.e Lemmas 1, 2, 3, 4, 5, 6 are satisfied.

D.1 Correctness

Correctness of sign is clearly satisfied, due to the correctness of the cm-NIZKs used.

Let upk_i be the user public key for the user with identifier uid_i. $\forall i \in k$, $\mu_i, c\mu_i, c_i$ are honestly generated and blinded and so for a_i, a'_i, a''_i chosen randomly, $\mu_i = (\hat{g}^{a_i}, 1, cpk_1^{a_i} upk_i)$, $c\mu_i = (\hat{g}^{a_i+a'_i}, \hat{g}^{a''_i}, cpk_1^{a_i+a'_i} bpk_2^{a''_i} upk_i)$. In CLS+.Convert the encryption on the pseudonyms is re-randomised. For $r'_i, r''_i \leftarrow\$ \{0,1\}^*$, $c\mu'_i = (\hat{g}^{a_i+a'_i+r'_i}, \hat{g}^{a''_i+r''_i}, cpk_1^{a_i+a'_i+r'_i} bpk_2^{a''_i+r''_i} upk_i)$. The decryption under the converter public key is then removed:
$c\mu''_i = (\hat{g}^{a''_i+r''_i}, cpk_1^{a_i+a'_i+r'_i} bpk_2^{a''_i+r''_i} upk_i \cdot \hat{g}^{-csk_1(a_i+a'_i+r'_i)}) = (\hat{g}^{a''_i+r''_i}, bpk_2^{a''_i+r''_i} upk_i)$.

$(e(g_1, \hat{g}^{a''_{\Pi(i)}+r''_{\Pi(i)}})^r, e(g_1, bpk_2^{a''_{\Pi(i)}+r''_{\Pi(i)}} upk_{\Pi(i)})^r)$, will be the pseudonym output by convert, and $e(g_1, bpk_2^{a''_{\Pi(i)}+r''_{\Pi(i)}} upk_{\Pi(i)})^r \cdot e(g_1, \hat{g}^{a''_{\Pi(i)}+r''_{\Pi(i)}})^{-rbsk_2} = e(g_1, upk_{\Pi(i)})^r$ will be the pseudonym output by CLS+.Unblind. Therefore $\overline{\mu}_{\Pi(i)} = \overline{\mu}_{\Pi(j)}$ if and only if $upk_i = upk_j$ and therefore $uid_i = uid_j$.

$\forall i \in [1, k]$, for α chosen randomly, $c_i = (g^{\alpha}, m_i bpk_1^{\alpha})$, for r'''_i chosen randomly, $\overline{c}_{\Pi(i)} = (g^{\alpha+r'''_i}, m_i bpk_1^{\alpha+r'''_i})$, therefore $\overline{m}_{\Pi(i)} = m_i$. CLS+.Convert will not fail, due to the correctness of the cm-NIZKs used. Therefore, due to the correctness of the SPK and the digital signature used, the signature $\overline{\sigma}$ output will be valid. Therefore, the construction satisfies **correctness of conversion**.

D.2 Consistency

Assume $\mathsf{UnLink}(gpk, csk, ((c\mu_0, c_0, c\sigma_0), (c\mu_1, c_1, c\sigma_1))) = 0$ and $\mathsf{UnLink}(gpk, csk, ((c\mu_1, c_1, c\sigma_1), (c\mu_2, c_2, c\sigma_2))) = 0$. This ensures that $c\sigma_0, c\sigma_1, c\sigma_2$ are all valid cm-NIZKs as otherwise UnLink would output $\perp$. Therefore $c\mu_i = (\hat{g}^{\alpha_i}, \hat{g}^{\beta_i}, cpk_1^{\alpha_i} bpk_2^{\beta_i} upk_i)$ for some $\alpha_i, \beta_i \in \mathbb{Z}_p^*, upk_i \in \mathbb{G}_2$.

Due to the above argument for correctness of conversions, letting r_1, r_2 be the randomness chosen in convert, $e(g_1, upk_0)^{r_1} = e(g_1, upk_1)^{r_1}$ and $e(g_1, upk_1)^{r_2} = e(g_1, upk_2)^{r_2}$. Therefore, $e(g_1, upk_0) = e(g_1, upk_1) = e(g_1, upk_2)$. However, if $\mathsf{UnLink}(gpk, csk, ((c\mu_0, c_0, c\sigma_0), (c\mu_2, c_2, c\sigma_2))) = 1$, then $e(g_1, upk_0)^{r_3} \neq e(g_1, upk_2)^{r_3}$, where r_3 was chosen during Convert. This is a contradiction. Therefore, the construction satisfies **consistency of linking**.

In the **consistency of verification** game, all signatures returned are valid, and so they contain valid cm-NIZKs. Due to the strong derivation privacy of the cm-NIZK, valid cm-NIZKs will be returned after blinding. Therefore, CLS+.Convert will not fail and $\overline{m}_i$ will be a shuffle of the original messages, due to the above argument in correctness of conversion. As conversion does not fail, the `tier-2` signature will consist of a digital signature on the pseudonym and ciphertext output by CLS+.Convert and a proof SPK that these were correctly unblinded. Due to the correctness of the digital signature scheme and the SPK, the `tier-2` signatures will be valid.

D.3 Anonymity

Lemma 1. *The CLS–CM construction satisfies* ***anonymity*** *if the DDH assumption holds in $\mathbb{G}_2$, and the cm-NIZK is zero knowledge, and cm-SSE.*

We assume that an adversary $\mathcal{A}$ exists, that makes q queries to the SNDU oracle for distinct user identifiers, and q_{conv} queries to the CLS+.Convert oracle, that guesses b correctly in the anonymity game and wins with probability $\epsilon + 1/2$.

We define Game (0,0) to be the anonymity experiment, with b chosen randomly at the beginning, using the CLS–CM construction. Let $S_{0,0}$ be the event that an adversary $\mathcal{A}$ correctly guesses b after Game (0,0).

Game $(0, j)$ is identical to Game (0,0) except during the first j queries to the CONVERT oracle, when $c, c\mu, c\sigma$ is queried to the CONVERT, such that c unblinds to $\overline{m}$, $\mathsf{Identify}(uid^*_d, c, c\mu, c\sigma) = 1$ and $\overline{m} = m^*$. We give the new convert oracle used for the first j queries of Game $(0, j)$ in Figure 8. Let $S_{0,j}$ be the event that the adversary $\mathcal{A}$ correctly guesses b after Game $(0, j)$.

```
$\mathsf{CONVERT}((c\mu_1, c\sigma_1, c_1), \ldots, (c\mu_k, c\sigma_k, c_k), bpk, bsk)$
if $(bpk, bsk) \notin \mathcal{BK}$   return $\perp$
if $\exists i^* \in [k], d \in \{0,1\}$ s.t $\mathsf{Identify}(uid^*_d, c_{i*}, c\mu_{i*}, c\sigma_{i*}) = 1$
 and $c_{i*,2} c_{i*,1}^{-bsk_1} = m^*$ and $bpk = bpk^*$
  if $\exists i \in [k] \backslash \{i^*\}$ s.t. $\mathsf{Identify}(uid^*_d, c_i, c\mu_i, c\sigma_i) = 1$ for $d \in \{0,1\}$   return $\perp$
  $(\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_{k-1}) \leftarrow \mathsf{CLS+.Convert}(gpk, bpk, csk, \{(c\mu_i, c\sigma_i, c_i) : i \in [k]\backslash\{i^*\}\})$
  if $(\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_{k-1}) = \perp$   return $\perp$
  $a_1 \leftarrow\$ \mathbb{Z}_p^*, \overline{c}_k \leftarrow (c_{i*,1} g^{a_1}, c_{i*,2} bpk_1^{a_1})$
  $upk^* \leftarrow\$ \mathbb{G}_2, a_2 \leftarrow\$ \mathbb{Z}_p^*, \overline{c\mu}_k \leftarrow (e(g_1, \hat{g}^{a_2}), e(g_1, upk^* bpk_2^{a_2}))$
  $\overline{c\sigma}_k \leftarrow \mathsf{SIG.Sign}(csk_2, (\overline{c}_k, \overline{c\mu}_k, bpk))$
  choose random permutation $\Pi$, for $i = 1, \ldots, k$ :
  $(\overline{c\mu}_i, \overline{c}_i, \overline{c\sigma}_i) \leftarrow (\overline{c\mu}_{\Pi(i)}, \overline{c}_{\Pi(i)}, \overline{c\sigma}_{\Pi(i)})$
else compute $\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_k \leftarrow \mathsf{CLS+.Convert}(gpk, bpk, csk, \{(c\mu_i, c\sigma_i, c_i)\}_k)$
return $(\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_k)$
```

Fig. 8. Convert oracle used during first j queries of Game $(0, j)$

We show that Game (0,j) and Game (0, j+1) are indistinguishable assuming the DDH assumption. We give a distinguishing algorithm $\mathcal{D}_j$ in Figures 9 and 10, and below explain why $\mathcal{D}_j$ simulates inputs to $\mathcal{A}$ that are distributed identically to in Game $(0, j)$ if a DDH tuple is input, and $\mathcal{D}_j$ simulates inputs to $\mathcal{A}$ that are distributed identically to in Game $(0, j+1)$ if a DDH tuple is not input.

```
$\mathsf{CONVERT}((c\mu_1, c\sigma_1, c_1), \ldots, (c\mu_k, c\sigma_k, c_k), bpk, bsk)$
if $(bpk, bsk) \notin \mathcal{BK}$   return $\perp$
$s \leftarrow s+1$   if $s \leqslant j$ perform $\mathsf{CONVERT}$ given in Figure 12
if $s > j+1$ perform $\mathsf{CONVERT}$ given in anonymity experiment
If $\exists i \in [k]$ s.t $c\sigma_i$ is not valid with respect to $c\mu_i, c_i, gpk$ and $bpk$   return $\perp$
if $\exists i^* \in [k], d \in \{0,1\}$ s.t $c\mu_{i^*,3} c\mu_{i^*,1}^{-csk_1} c\mu_{i^*,1}^{-bsk_2} = upk_{uid_d^*}$ and $c_{i^*,2} c_{i^*,1}^{-bsk_1} = m^*$
and $bpk = bpk^*$
  if $uid_d^* \neq uid'$ abort $\mathcal{D}_j$
  if $\exists i \in [k] \backslash \{i^*\}, d \in \{0,1\}$ s.t. $c\mu_{i,3} c\mu_{i,1}^{-csk_1} c\mu_{i,1}^{-bsk_2} = upk_{uid_d^*}$   return $\perp$
  for $i = [k] \backslash \{i^*\}$ :
    if $\exists uid \in \mathsf{HUL} \backslash \{uid'\}$ s.t $\mathsf{Identify}(uid, c_i, c\mu_i, c\sigma_i) = 1, upk'_i \leftarrow g_1^{usk_{uid}}$
    else $(upk'_i, \cdots) \leftarrow E_{2,\mathsf{cm\text{-}NIZK}}(\sigma_{\mathsf{crs}}, \tau_e, (cpk_1, bpk, ipk, c\mu_i, c_i), c\sigma_i)$
    if $upk'_i = \perp$ return $\perp$
    $a_1, a_2 \leftarrow\$ \mathbb{Z}_p^*, \bar{c}_i \leftarrow (c_{i,1} g^{a_2}, c_{i,2} bpk_1^{a_2}), \overline{c\mu_i} \leftarrow (e(upk'_i, \hat{g}^{a_1}), e(upk'_i, D_3 bpk_2^{a_1}))$
    $\overline{c\sigma}_i \leftarrow \mathsf{SIG.Sign}(csk_2, (\overline{c_i}, \overline{c\mu_i}, bpk))$
  $b_1, b_2 \leftarrow\$ \mathbb{Z}_p^*, \mu_{i^*} \leftarrow D_4, \overline{c\mu}_{i^*} \leftarrow (e(g_1, \hat{g}^{b_1}), e(g_1, bpk_2^{b_1} \mu_{i^*}))$
  $\bar{c}_{i^*} \leftarrow (c_{i^*,1} g^{b_2}, c_{i^*,2} bpk_1^{b_2}), \overline{c\sigma}_{i^*} \leftarrow \mathsf{SIG.Sign}(csk_2, (\bar{c}_{i^*}, \overline{c\mu}_{i^*}, bpk))$
  choose random permutation $\Pi$, for $i = 1, \ldots, k$ :
  $(\overline{c\mu}_i, \bar{c}_i, \overline{c\sigma}_i) \leftarrow (\overline{c\mu}_{\Pi(i)}, \bar{c}_{\Pi(i)}, \overline{c\sigma}_{\Pi(i)})$
else compute $\{(\overline{c\mu}_i, \overline{c\sigma}_i, \bar{c}_i)\}_k \leftarrow \mathsf{CLS+.Convert}(gpk, bpk, csk, \{(c\mu_i, c\sigma_i, c_i)\}_k)$
return $(\{(\overline{c\mu}_i, \overline{c\sigma}_i, \bar{c}_i)\}_k)$
```

Fig. 9. $\mathcal{D}_j$ a distinguishing algorithm for the DDH problem

The values gpk, csk, isk are distributed identically to in the anonymity game, as everything but $g_2, \sigma_{\mathsf{crs}}$ are chosen in the same way. $SE_{1,\mathsf{cm\text{-}NIZK}}$ outputs a σ_{crs} that is identically distributed to in CRSSetup.

Simulating the SNDU Oracle The SNDU oracle only differs from the oracle in the anonymity experiment when uid' is input. In this case upk is distributed identically, and the CLS+.Join protocol can be simulated without knowledge of usk. Therefore, outputs are distributed identically. usk, upk' are set to $\perp$, but we will show these are not used later.

Simulating the SIGN Oracle The SIGN oracle is identical to the oracle in the anonymity experiment, when $uid \neq uid'$ is queried. When uid' is queried, then

```
$\mathsf{SNDU}(uid, M_{\mathsf{in}})$
$\textbf{if } uid \in \mathsf{CUL} \quad \textbf{return } \perp$
$\textbf{if } uid \notin \mathsf{HUL}$
    $\mathsf{HUL} \leftarrow \mathsf{HUL} \cup \{uid\}, l \leftarrow l+1, \mathbf{gsk}[uid] \leftarrow \perp, \mathbf{dec}^{uid} \leftarrow \mathsf{cont}$
    $\textbf{if } l = q^* \quad uid' \leftarrow uid, upk_{uid} \leftarrow D_2 \quad \textbf{return } (upk_{uid}, \mathsf{cont})$
$\textbf{if } uid = uid'$
    continue from line 4 of oracle in anonymity experiment but set $upk' = \perp$
$\textbf{else}$ Continue from line 4 of oracle in anonymity experiment
```

```
$\mathsf{SIGN}(uid, m, bpk)$
$\textbf{if } uid \neq uid'$ perform $\mathsf{SIGN}$ oracle from anonymity experiment
$\textbf{if } uid = uid'$
    $\textbf{if } \mathbf{dec}^{uid} \neq \mathsf{accept} \quad \textbf{return } \perp$
    $\alpha \leftarrow\$ \mathbb{Z}_p^*, \mu \leftarrow (\hat{g}^{\alpha}, 1, upk_{uid} cpk_1^{\alpha}), \beta \leftarrow 0, \gamma \leftarrow 0, c \leftarrow (1, m)$
    $\sigma \leftarrow S_{2,\mathsf{cm\text{-}NIZK}}(\sigma_{\mathsf{crs}}, \tau_s, (cpk_1, bpk, ipk, \mu, c)) \quad \textbf{return } (\mu, \sigma)$
    $\mathsf{SL} \leftarrow \mathsf{SL} \cup \{(uid, m, bpk)\}$
    $\textbf{return } (\sigma, \mu)$
```

```
$\mathcal{D}_j(D_1, D_2, D_3, D_4)$
$s, l \leftarrow 0, q^* \leftarrow\$ [1, q], b \leftarrow\$ \{0,1\}, g_1 \leftarrow\$ \mathbb{G}_1, g_2 \leftarrow D_1$
$\mathsf{pp}_{\mathsf{auto}1} \leftarrow \mathsf{ASetup}_1(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2))$
$\mathsf{pp}_{\mathsf{auto}2} \leftarrow \mathsf{ASetup}_2(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2))$
$g \leftarrow\$ \mathbb{G}_1, \hat{g} \leftarrow\$ \mathbb{G}_2, (\sigma_{\mathsf{crs}}, \tau_s, \tau_e) \leftarrow SE_{1,\mathsf{cm\text{-}NIZK}}(1^{\tau})$
$\mathsf{pp}_{\mathsf{sig}} \leftarrow \mathsf{SIG.Setup}(1^{\tau})$
$\mathsf{pp} \leftarrow ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{pp}_{\mathsf{auto}1}, \mathsf{pp}_{\mathsf{auto}2}, g, \hat{g}, \sigma_{\mathsf{crs}}, \mathsf{pp}_{\mathsf{sig}})$
$(isk, ipk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp}), (csk, cpk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp})$
$gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$(uid_0^*, uid_1^*, m^*, bpk^*, st) \leftarrow \mathcal{A}^{\mathsf{SNDU,SIGN,CONVERT}}(\mathsf{choose}, gpk, isk)$
$\textbf{if } uid_0^* \notin \mathsf{HUL} \text{ or } \mathbf{dec}^{uid_0^*} \neq \mathsf{accept} \text{ or } uid_1^* \notin \mathsf{HUL} \text{ or } \mathbf{dec}^{uid_1^*} \neq \mathsf{accept} \quad \textbf{return } \perp$
$(\mu^*, \sigma^*) \leftarrow \mathsf{SIGN}(uid_b^*, m^*, bpk^*)$
$b^* \leftarrow \mathcal{A}^{\mathsf{SNDU,SIGN,CONVERT}}(\mathsf{guess}, st, \mu^*, \sigma^*)$
$\textbf{if } b^* = b \quad \textbf{return } 1$
```

Fig. 10. $\mathcal{D}_j$ a distinguishing algorithm for the DDH problem

μ, c can be computed as normal and $S_{2,\mathsf{cm\text{-}NIZK}}$ can be used to simulate the $\mathsf{cm\text{-}NIZK}$ σ. This will be identically distributed, due to the zero knowledge of $\mathsf{cm\text{-}NIZK}$. Therefore, the outputs of SIGN are distributed identically to in the anonymity experiment.

Simulating the $\mathsf{CONVERT}$ Oracle Other than the $(j+1)$th query, the $\mathsf{CONVERT}$ oracle is identical to in both Games $(0, j)$ and $(0, j+1)$.

For the $(j+1)$th query, if the input to $\mathcal{D}_j$ is a DDH tuple, then outputs from the $\mathsf{CONVERT}$ oracle are identically distributed to in the anonymity game. This is because, if $c, c\mu, c\sigma$ is not queried to the $\mathsf{CONVERT}$, such that c will decrypt to $\overline{m}$, $\mathsf{Identify}(uid^*_d, c, c\mu, c\sigma) = 1$ for $d \in \{0,1\}$ and $\overline{m} = m^*$ and $bpk = bpk^*$, the oracle behaves identically to in the anonymity game, as in both games. If an invalid blinded signature is input, the oracle will abort, as in the original anonymity game.

If this is queried, we show the simulation is correctly distributed. Firstly $\overline{c}_i$ and $\overline{c\sigma}_i$ are generated identically to in $\mathsf{CLS{+}.Convert}$.

As we do not have access to $\mathbf{gsk}[uid']$, we cannot perform $\mathsf{Identify}$ on input uid', therefore instead we decrypt the pseudonym and check whether it matches $upk_{uid'}$. Only one signature identifies to uid^*_d otherwise the oracle will abort as in the anonymity game. We assume $uid^*_d = uid'$ which happens with probability $1/q$, therefore extraction of the upk'_i is always successful, because $c\mu_i$ does not decrypt to the user public key of uid'. This is because $E_{2,\mathsf{cm\text{-}NIZK}}$ will either successfully extract upk'_i, or instead will extract a transformation T and statement $(cpk_1, bpk, ipk, c\mu'_i, c'i)$ such that $(cpk_1, bpk, ipk, c\mu_i, c_i) = T_{\mathsf{inst}}(cpk_1, bpk, ipk, c\mu'_i, c'_i)$. If it outputs the latter $(c\mu'_i, c'_i)$ was input to $S_{2,\mathsf{cm\text{-}NIZK}}$, and $(c\mu'_i, c'_i)$ is a re-randomisation of $(c\mu_i, c_i)$. This is not possible as $S_{2,\mathsf{cm\text{-}NIZK}}$ is only used in the signing oracle for uid'.

Letting $r' = \log_{D_1}(D_3)$, and $upk'_i = g_1^{usk_i}$, $upk_i = g_2^{usk_i}$, then $\overline{c\mu}_i = (e(upk'_i, \hat{g}^{a_1}), e(upk'_i, D_3 bpk_2^{a_1})) = (e(g_1, \hat{g}^{a_1})^{usk_i}, e(g_1, D_3 bpk_2^{a_1})^{usk_i})$ $= (e(g_1, \hat{g}^{a_1 usk_i}), e(g_1, D_3^{usk_i} bpk_2^{a_1 usk_i})) = (e(g_1, \hat{g}^{a_1 usk_i}), e(g_1, upk_i^{r'} bpk_2^{a_1 usk_i}))$ $= (e(g_1, \hat{g}^{a_1 usk_i r'^{-1}})^{r'}, e(g_1, upk_i bpk_2^{a_1 usk_i r'^{-1}})^{r'})$, which is correctly distributed as a_1 is chosen randomly and independently. Also $\overline{c\mu}_{i*} =$ $(e(g_1, \hat{g}^{b_1}), e(g_1, bpk_2^{b_1} D_4)) = (e(g_1, \hat{g}^{b_1 r'^{-1}})^{r'}, e(g_1, bpk_2^{b_1 r'^{-1}} upk)^{r'})$ where $upk = upk_{uid'}$. These are distributed identically, due to the fact that b_1 is chosen randomly and independently. The $\overline{c\mu}_i$ are then shuffled with a random permutation. Therefore, the outputs of $\mathsf{CONVERT}$ are distributed identically to the $\mathsf{CONVERT}$ oracle in the anonymity game, and so Game $(0, j)$.

Simulating (μ^*, σ^*). (μ^*, σ^*) input to $\mathcal{A}$ in the guessing stage is distributed identically to in the anonymity game, due to outputs of the SIGN oracle being distributed identically to the anonymity game.

Reduction to the DDH problem If the input to $\mathcal{D}_j$ is not a DDH tuple, then outputs of the $\mathsf{CONVERT}$ oracle are identically distributed to in Figure 8. This is because if $c, c\mu, c\sigma$ is not queried to the $\mathsf{CONVERT}$ oracle, such that c decrypts to $\overline{m}$, $\mathsf{Identify}(uid_d^*, c, c\mu, c\sigma) = 1$ and $\overline{m} = m^*$ for $d \in \{0,1\}$ and $bpk = bpk^*$, then the oracle behaves identically to both games. If this is the case, as D_4 is now chosen randomly and independently, $\overline{c}_i^*, \overline{c\mu}_i^*, \overline{c\sigma}_i^*$ are now chosen identically to in Figure 8. Therefore, if the input to $\mathcal{D}_j$ is not a DDH tuple, then the outputs to $\mathcal{A}$ are identically distributed to Game $(0, j+1)$

$\mathcal{D}_j$ only aborts early if if $c, c\mu, c\sigma$ is queried to the $\mathsf{CONVERT}$ oracle, such that c decrypts to $\overline{m}$, $\mathsf{Identify}(uid_d^*, c, c\mu, c\sigma) = 1$ and $\overline{m} = m^*$, for $d \in \{0,1\}$ and $uid' \neq uid_d^*$. This occurs with probability at most $q - 1/q$. Therefore, the probability that $\mathcal{D}_j$ outputs 1 given a DDH tuple was input is $Pr[S_{0,j}]/q$. The probability that $\mathcal{D}_j$ outputs 1 given a DDH tuple was not input is $Pr[S_{0,j+1}]/q$. The advantage of $\mathcal{D}_j$ is then $|Pr[S_{0,j}] - Pr[S_{0,j+1}]|/q$, therefore $|Pr[S_{0,j}] - Pr[S_{0,j+1}]| = q\epsilon_{\mathsf{DDH}}$.

We define Game 1 to be Game $(0, q_{\mathsf{conv}})$, where q_{conv} is the number of queries to the $\mathsf{CONVERT}$ oracle. Let S_1 be the event that an adversary $\mathcal{A}$ correctly guesses b after Game 1. As $|Pr[S_{0,j}] - Pr[S_{0,j+1}]| = q\epsilon_{\mathsf{DDH}}$, then $|Pr[S_{0,0}] - Pr[S_1]| \leqslant q_{\mathsf{conv}} q\epsilon_{\mathsf{DDH}}$.

Next, we show that $|Pr[S_1] - 1/2| \leqslant \epsilon_{\mathsf{DDH}}$. We build an adversary $\mathcal{A}'$ that distinguishes DDH tuples, given $\mathcal{A}$ that guesses successfully in Game 1 with $Pr[S_1]$. We give $\mathcal{A}'$ in Figure 11, and below explain why the simulation input to $\mathcal{A}$ given in Figure 11 is identically distributed to Game 1 given a DDH tuple is input, and that $\mathcal{A}'$ successfully distinguishes DDH tuples.

The values gpk, isk are distributed identically to in the anonymity game, as everything but $\hat{g}, cpk_1, \sigma_{\mathsf{crs}}$ are chosen in the same way. $SE_{1,\mathsf{cm\text{-}NIZK}}$ outputs a σ_{crs} that is identically distributed to in $\mathsf{CRSSetup}$. $\hat{g}, cpk_1$ are distributed identically. The SNDU, and SIGN oracles are identical to the anonymity experiment. If a DDH tuple is input, μ^* is distributed identically to Game 1, because letting $\alpha = \log_{D_1}(D_3)$, then $\mu^* = (\hat{g}^\alpha, 1, cpk_1^\alpha upk_{uid_b^*})$ which is identically distributed.

Simulating the $\mathsf{CONVERT}$ oracle The $\mathsf{CONVERT}$ oracle can no longer use $\mathsf{Identify}$, without csk_1. However, if $\exists i^* \in [k]$ s.t $\mathsf{Identify}(uid_d^*, c_{i^*}, c\mu_{i^*}, c\sigma_{i^*}) = 1$ for $d \in \{0,1\}$ and $c_{i^*,2}c_{i^*,1}^{-bsk_1} = m^*$ and $bpk = bsk^*$, then clearly either $upk_{i^*} = \perp$ or $upk_{i^*} = upk_{uid_d}$ s.t $d \in \{0,1\}$ and $m_{i^*} = m^*$. If there does not exist $i^* \in [k]$ s.t $\mathsf{Identify}(uid_d^*, c_{i^*}, c\mu_{i^*}, c\sigma_{i^*}) = 1$ for $d \in \{0,1\}$ and $c_{i^*,2}c_{i^*,1}^{-bsk_1} = m^*$, there clearly there does not exist $i^* \in [k]$ s.t $upk_{i^*} = upk_{uid_d}$ s.t $d \in \{0,1\}$ and $m_{i^*} = m^*$. If $upk_{i^*} = \perp$, then the extraction algorithm outputs a statement x', and a transformation T, where $x' = (cpk_1, bpk^*, ipk, \mu^*, c^*)$ as this is the only statement for which a simulation is generated, and $x = T_x(x')$ and T is a valid transform. This means that $c\mu_{i^*}$, and c_{i^*}, are re-randomisations of μ^*, c^*, and so c_{i^*} is an encryption of m^*, $c\mu_{i^*}$ is an encryption of $upk_{uid_b^*}$, and $bpk = bpk^*$. This is a contradiction and so the statements are equivalent.

The condition $\exists i \in [k] \backslash \{i^*\}$ s.t $\mathsf{Identify}(uid_d^*, c_i, c\mu_i, c\sigma_i) = 1$ for $d \in \{0,1\}$, is equivalent to $\exists i \in [k] \backslash \{i^*\}$ s.t. $upk_i = \perp$ or $upk_{uid_d^*}$ for $d \in \{0,1\}$ by the same reasoning.

Finally the only difference, is that $c\mu_i'' \leftarrow (\hat{g}^{a_1}, upk_i bpk_2^{a_1})$ is used in $\mathsf{CLS{+}.Convert}$. Clearly due to the above conditions, upk_i has been extracted

```
$\mathsf{SNDU}(uid, M_{\mathsf{in}})$
Identical to the anonymity experiment.
```

```
$\mathsf{SIGN}(uid, m, bpk)$
Identical to the anonymity experiment.
```

```
$\mathsf{CONVERT}((c\mu_1, c\sigma_1, c_1), \ldots, (c\mu_k, c\sigma_k, c_k), bpk, bsk)$
**if** $\exists i \in [1,k]$ s.t $c\sigma_i$ is not valid with respect to $c\mu_i, c_i, gpk$ and $bpk$ or $(bpk, bsk) \notin \mathcal{BK}$
  **return** $\perp$
$\forall i \in [k] \quad ((\cdot, upk_i, \cdot, \cdot, \cdot, \cdot, \cdot), \cdot, \cdot) \leftarrow E_{2,\mathsf{cm\text{-}NIZK}}(\sigma_{\mathsf{crs}}, \tau_e, (cpk_1, bpk, ipk, c\mu_i, c_i), c\sigma_i)$
  $a_1 \leftarrow\$ \mathbb{Z}_p^*, c\mu_i'' \leftarrow (\hat{g}^{a_1}, upk_i bpk_2^{a_1}), m_i \leftarrow c_{i,2} c_{i,1}^{-bsk_1}$
**if** $\exists i^* \in [k]$ s.t $upk_{i*} = \perp$
or $upk_{i*} = upk_{uid_d}$ s.t $d \in \{0,1\}$ and $m_{i*} = m^*$ and $bpk = bpk^*$
  **if** $\exists i \in [k] \backslash \{i^*\}$ s.t. $upk_i = \perp$ or $upk_{uid_d^*}$ for $d \in \{0,1\}$
    **return** $\perp$
  Using the $c\mu_i''$ computed above instead of using $csk_1$ compute
  $(\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_{k-1}) \leftarrow \mathsf{CLS+.Convert}(gpk, bpk, csk, \{(c\mu_i, c\sigma_i, c_i) : i \in [k] \backslash \{i^*\}\})$
  **if** $(\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_{k-1}) = \perp$ **return** $\perp$
  $b_1, b_2 \leftarrow\$ \mathbb{Z}_p^*, \overline{c}_k \leftarrow (c_{i*,1} g^{b_2}, c_{i*,2} bpk_1^{b_2}), upk^* \leftarrow\$ \mathbb{G}_2, \overline{c\mu}_k \leftarrow (e(g_1, \hat{g}^{b_1}), e(g_1, upk^* bpk_2^{b_1}))$
  $\overline{c\sigma}_k \leftarrow \mathsf{SIG.Sign}(csk_2, (\overline{c}_k, \overline{c\mu}_k, bpk))$
  choose random permutation $\Pi$, **for** $i = 1, \ldots, k$ :
  $(\overline{c\mu}_i, \overline{c}_i, \overline{c\sigma}_i) \leftarrow (\overline{c\mu}_{\Pi(i)}, \overline{c}_{\Pi(i)}, \overline{c\sigma}_{\Pi(i)})$
**else** Using the $c\mu_i''$ computed above instead of using $csk_1$ compute
$\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_k \leftarrow \mathsf{CLS+.Convert}(gpk, bpk, csk, \{(c\mu_i, c\sigma_i, c_i)\}_k)$
**return** $(\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_k)$
```

```
$\mathcal{A}'(D_1, D_2, D_3, D_4)$
$b \leftarrow\$ \{0,1\}, (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathcal{G}(1^\tau)$
$\mathsf{pp}_{\mathsf{auto}1} \leftarrow \mathsf{ASetup}_1(1^\tau), \mathsf{pp}_{\mathsf{auto}2} \leftarrow \mathsf{ASetup}_2(1^\tau)$
$g \leftarrow\$ \mathbb{G}_1, \hat{g} \leftarrow D_1, (\sigma_{\mathsf{crs}}, \tau_s, \tau_e) \leftarrow \mathsf{SE}_{1,\mathsf{cm\text{-}NIZK}}(1^\tau), \mathsf{pp}_{\mathsf{sig}} \leftarrow \mathsf{SIG.Setup}(1^\tau)$
$\mathsf{pp} \leftarrow ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{pp}_{\mathsf{auto}1}, \mathsf{pp}_{\mathsf{auto}2}, g, \hat{g}, \sigma_{\mathsf{crs}}, \mathsf{pp}_{\mathsf{sig}})$
$(isk, ipk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp}), (cpk_2, csk_2) \leftarrow \mathsf{SIG.KeyGen}(1^\tau), cpk \leftarrow (D_2, cpk_2)$
$gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$(uid_0^*, uid_1^*, m^*, bpk^*, st) \leftarrow \mathcal{A}^{\mathsf{SNDU},\mathsf{SIGN},\mathsf{CONVERT}}(\mathsf{choose}, gpk, isk)$
**if** $uid_0^*, uid_1^* \notin \mathsf{HUL}$ or $\mathbf{gsk}[uid_0^*], \mathbf{gsk}[uid_1^*] = \perp$  **return** $\perp$
$\mu^* \leftarrow (D_3, 1, D_4 upk_{uid_b^*}), c^* \leftarrow (1, m^*)$
$\sigma^* \leftarrow S_{2,\mathsf{cm\text{-}NIZK}}(\sigma_{\mathsf{crs}}, \tau_s, (cpk_1, bpk^*, ipk, \mu^*, c^*))$
$b^* \leftarrow \mathcal{A}^{\mathsf{SNDU},\mathsf{SIGN},\mathsf{CONVERT}}(\mathsf{guess}, st, \mu^*, \sigma^*)$
**if** $b^* = b$  **return** 1  **else return** 0
```

Fig. 11. $\mathcal{A}'$ which distinguishes DDH tuples successfully. Then, $c\mu_i''$ is distributed identically to the value computed in CLS+.Convert because this is a re-randomised encryption of upk_i. Therefore, this oracle is distributed identically to Figure 8.

Reduction to the DDH assumption If a DDH tuple is not input, D_4 was chosen randomly, and so the input to $\mathcal{A}$ is independent of b. Therefore $\mathcal{A}$ guesses correctly with probability 1/2.

If a DDH tuple is input, the inputs to $\mathcal{A}$ are distributed identically to in Game 1, and they will guess correctly with probability S_1. Therefore $|Pr[S_1] - 1/2| \leqslant \epsilon_{\mathsf{DDH}}$.

Therefore $|Pr[S_{0,0}] - 1/2| \leqslant \epsilon_{\mathsf{DDH}} + q_{\mathsf{conv}} q \epsilon_{\mathsf{DDH}}$. As this is negligible, our construction satisfies anonymity.

D.4 Non-Transitivity

Lemma 2. *The CLS–CM construction satisfies* ***non-transitivity*** *if the DDH assumption holds in $\mathbb{G}_2$, and the cm-NIZK is zero knowledge, and cm-SSE.*

For proving non-transitivity, we have to show that there exists an efficient simulator SIM that makes the real and simulated game indistinguishable. We start by describing the simulator and then explain why the real and simulated conversion oracles CONVERT and CONVSIM are indistinguishable.

```
$\mathsf{SIM}(gpk, bpk, csk, \mathsf{L}_{uid_1}, \ldots \mathsf{L}_{uid_{k'}})$
$l \leftarrow 0, \forall j \in [1, k']$
  $\mu' \leftarrow\$ \mathbb{G}_2; \forall c \in L_{uid_j}$
    $l \leftarrow l+1, b_1, b_2 \leftarrow\$ \mathbb{Z}_p^*, \overline{c\mu}_l \leftarrow (e(g_1, \hat{g}^{b_1}), e(g_1, \mu' bpk_2^{b_1})), \bar{c}_l \leftarrow (g^{b_2} c_1, bpk_1^{b_2} c_2)$
    $\overline{c\sigma}_l \leftarrow \mathsf{SIG.Sign}(csk_2, (\overline{c\mu}_l, \bar{c}_l, bpk))$
return $((\overline{c\mu}_1, \bar{c}_1, \overline{c\sigma}_1), \ldots (\overline{c\mu}_l, \bar{c}_l, \overline{c\sigma}_l))$
```

We assume that an adversary $\mathcal{A}$ exists, that makes q queries to the SNDU oracle for distinct user identifiers, and q_{conv} queries to the CONVX oracle, that guesses b correctly in the non-transitivity game with SIM given above and wins with probability $\epsilon + 1/2$.

We will stepwise make the real-world (b=0) and the simulated world (b=1) equivalent, using a sequence of Games $\mathcal{H}_j$ for $j = 0, \ldots, q$. The idea is that in Game $\mathcal{H}_j$ we will not use simulated conversions for all users $uid_1, \ldots, uid_j$ in order of when they were queried to SNDU. More precisely, we define Game $\mathcal{H}_j$ to be as given in Figure 12 with all other oracles identical to in the non-transitivity experiment. Let S_j be the event that $\mathcal{A}$ guesses b correctly in Game $\mathcal{H}_j$, with the simulator given above. Game $\mathcal{H}_j$ keeps track of the queries to SNDU, adding the first j queries of uid to a set UL. Then during queries to CONVSIM, if a signature of a user in UL is queried, these are treated in the same way as pseudonyms for corrupted users, i.e., they are normally converted using the CLS+.Convert algorithm. If a signature of an honest user that is not in UL is queried, we add this user to NUL. These conversions are simulated as usual.

Game $\mathcal{H}_0$ is identical to the non-transitivity game, because UL is empty. Therefore, $\Pr[S_0] = \epsilon + 1/2$. In Game $\mathcal{H}_q$, UL contains all honest users, and so the CONVSIM oracle is now identical to the CONVERT oracle, and inputs to the adversary are now independent of b , therefore $\Pr[S_q] = 1/2$.

```
Game $\mathcal{H}_j$
$t \leftarrow 0, b \leftarrow\$ \{0,1\}, \mathsf{pp} \leftarrow \mathsf{CLS+.Setup}(1^\tau), (ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp})$
$(cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp}), gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
return $\mathcal{A}^{\mathsf{SNDU},\mathsf{SIGN},\mathsf{CONVX}}(\mathsf{guess}, gpk, isk)$

$\mathsf{SNDU}(uid, M_{\mathsf{in}})$
if $uid \notin \mathsf{HUL}, t \leftarrow t+1,$ if $t \leqslant j$   $\mathsf{UL} \leftarrow \mathsf{UL} \cup \{uid\}$
Continue from line 5 of standard SNDU oracle

$\mathsf{CONVSIM}((c\mu_1, c\sigma_1, c_1), \ldots, (c\mu_k, c\sigma_k, c_k), bpk, bsk)$
if $(bpk, bsk) \notin \mathcal{BK}$   return $\perp$
if $\mathsf{CLS+.Convert}(gpk, bpk, csk, (c\mu_1, c\sigma_1, c_1), \ldots, (c\mu_k, c\sigma_k, c_k)) = \perp$   return $\perp$
Set $\mathsf{CL} \leftarrow \varnothing$
$\forall i \in [1, k]$
  if $\exists uid \in \mathsf{HUL}$ s.t $\mathsf{Identify}(uid, c_i, c\mu_i, c\sigma_i) = 1$
    if $\mathsf{L}_{uid}$ does not exist, create $\mathsf{L}_{uid} \leftarrow \{c_i\}, \mathsf{CL}_{uid} \leftarrow \{(c_i, c\sigma_i, c\mu_i)\}$
    else  set $\mathsf{L}_{uid} \leftarrow \mathsf{L}_{uid} \cup \{c_i\}, \mathsf{CL}_{uid} \leftarrow \mathsf{CL}_{uid} \cup \{(c_i, c\sigma_i, c\mu_i)\}$
  else $\mathsf{CL} \leftarrow \mathsf{CL} \cup \{(c_i, c\sigma_i, c\mu_i)\}$
$(\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_{i=1,\ldots k'}) \leftarrow \mathsf{CLS+.Convert}(gpk, bpk, csk, \mathsf{CL} \cup \bigcup_{uid \in \mathsf{UL}} \mathsf{CL}_{uid}))$
 for $k' \leftarrow |\mathsf{CL} \cup \bigcup_{uid \in \mathsf{UL}} \mathsf{CL}_{uid}|$
Let $\mathsf{L}_{uid_1}, \ldots \mathsf{L}_{uid_{k''}}$ be the non-empty message clusters created above
Let $\mathsf{NUL} \leftarrow \{uid_1, \ldots uid_{k''}\} \backslash \mathsf{UL}$
$\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_{i=k'+1,\ldots k} \leftarrow \mathsf{SIM}(gpk, bpk, csk, \bigcup_{uid \in \mathsf{NUL}} \mathsf{L}_{uid})$
Let $\{(\overline{c\mu}'_i, \overline{c\sigma}'_i, \overline{c}'_i)\}_{i=1,\ldots k}$ be a random permutation of $\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_{i=1,\ldots k}$
return $(\{(\overline{c\mu}'_i, \overline{c\sigma}'_i, \overline{c}'_i)\}_{i=1,\ldots k})$
```

Fig. 12. Description of Game $\mathcal{H}_j$ and the changes to the SNDU and CONVSIM oracles.

We now show that if an adversary can distinguish Games $\mathcal{H}_j$ and $\mathcal{H}_{j+1}$, we can turn this into a distinguisher $\mathcal{D}_j$ that can break the DDH assumption in $\mathbb{G}_2$. We describe the reduction and the additional simulation that is needed therein in Figures 13 and 14.

```
$\mathcal{D}_j(D'_1, D'_2, D'_3, D'_4)$
Expand $D'_1, D'_2, D'_3, D'_4$ into $D_1, D_2, \{D_{3,i}, D_{4,i} : i \in [q_{\mathsf{conv}}]\}$ using DDH random self-reduction [40]
$s, t \leftarrow 0, g_1 \leftarrow\$ \mathbb{G}_1, g_2 \leftarrow D_1$
$\mathsf{pp}_{\mathsf{auto}1} \leftarrow \mathsf{ASetup}_1(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2))$
$\mathsf{pp}_{\mathsf{auto}2} \leftarrow \mathsf{ASetup}_2(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2))$
$g \leftarrow\$ \mathbb{G}_1, \hat{g} \leftarrow\$ \mathbb{G}_2, (\sigma_{\mathsf{crs}}, \tau_s, \tau_e) \leftarrow SE_{1,\mathsf{cm\text{-}NIZK}}(1^\tau), \mathsf{pp}_{\mathsf{sig}} \leftarrow \mathsf{SIG.Setup}(1^\tau)$
$\mathsf{pp} \leftarrow ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{pp}_{\mathsf{auto}1}, \mathsf{pp}_{\mathsf{auto}2}, g, \hat{g}, \sigma_{\mathsf{crs}}, \mathsf{pp}_{\mathsf{sig}})$
$(isk, ipk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp})$
$(csk, cpk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp}), gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
return $\mathcal{A}^{\mathsf{SNDU},\mathsf{SIGN},\mathsf{CONVX}}(gpk, csk, isk)$

$\mathsf{SNDU}(uid, M_{\mathsf{in}})$
if $uid \in \mathsf{CUL}$   return $\perp$
if $uid \notin \mathsf{HUL}, t \leftarrow t+1,$ if $t \leqslant j$   $\mathsf{UL} \leftarrow \mathsf{UL} \cup \{uid\}$
  $\mathsf{HUL} \leftarrow \mathsf{HUL} \cup \{uid\}, \mathbf{gsk}[uid] \leftarrow \perp, \mathbf{dec}^{uid} \leftarrow \mathsf{cont}$
  if $t = j+1$   $uid' \leftarrow uid, upk_{uid} \leftarrow D_2,$   return $(upk_{uid}, \mathsf{cont})$
if $uid = uid'$
  Continue from line 4 of standard SNDU oracle but set $upk' = \perp$
else Continue from line 4 of standard SNDU oracle

$\mathsf{SIGN}(uid, m, bpk)$
if $uid \neq uid'$ perform SIGN oracle from non-transitivity experiment
if $uid = uid'$
  if $\mathbf{dec}^{uid} \neq \mathsf{accept}$   return $\perp$
  $\alpha \leftarrow\$ \mathbb{Z}_p^*, \mu \leftarrow (\hat{g}^\alpha, 1, upk_{uid} cpk_1^\alpha), \beta \leftarrow 0, \gamma \leftarrow 0, c \leftarrow (1, m)$
  $\sigma \leftarrow S_{2,\mathsf{cm\text{-}NIZK}}(\sigma_{\mathsf{crs}}, \tau_s, (cpk_1, bpk, ipk, \mu, c))$   return $(\mu, \sigma)$
  $\mathsf{SL} \leftarrow \mathsf{SL} \cup \{(uid, m, bpk)\}$
  return $(\sigma, \mu)$
```

Fig. 13. $\mathcal{D}_j$ our distinguishing algorithm for the DDH problem. The CONVERT oracle remains unchanged, and the CONVSIM oracle using the DDH challenge is given in Figure 13.

We now argue that when a DDH tuple (D'_1, D'_2, D'_3, D'_4) is input to $\mathcal{D}_j$, the inputs to $\mathcal{A}$ are distributed identically to in Game $\mathcal{H}_{j+1}$; when a DDH tuple is not input, the inputs to $\mathcal{A}$ are distributed identically to in Game $\mathcal{H}_j$. That is for $D'_1 = h, D'_2 = h^a, D'_3 = h^b, D'_4 = h^c$, the oracles provided by $\mathcal{D}_j$ will be exactly as in $\mathcal{H}_{j+1}$ when $c = ab$, and as in $\mathcal{H}_j$ otherwise.

We first note that due to the DDH random self-reduction given in [40], if a DDH tuple is input to $\mathcal{D}_j$, then for all $i \in [q_{\mathsf{conv}}]$, $D_1, D_2, D_{3,i}, D_{4,i}$ is a DDH tuple. If a DDH tuple is not input to $\mathcal{D}_j$, then $D_1, D_2, D_{3,1}, D_{4,1}, \cdots, D_{3,q_{\mathsf{conv}}}, D_{4,q_{\mathsf{conv}}}$ are randomly and independently distributed.

The values gpk, csk, isk are distributed identically to in the non-transitivity game, as everything but $g_2, \sigma_{\mathsf{crs}}$ are chosen in the same way. $SE_{1,\mathsf{cm\text{-}NIZK}}$ and $\mathsf{CRSSetup}$ outputs identically distributed σ_{crs}.

Simulating the SNDU Oracle The SNDU oracle only differs from the oracle in the non-transitivity experiment during the $(j+1)$-th query by embedding D_2 of the DDH challenger into the user's "public key" upk. Clearly, upk is distributed identically as when computed normally. Note that usk, upk' are not defined for this user, but this is not output to $\mathcal{A}$, or used in the next stage of the protocol.

Simulating the SIGN *Oracle* The SIGN oracle is identical to the oracle in the non-transitivity experiment, when $uid \neq uid'$ is queried. When uid' is queried, then μ, c can be computed as normal and $S_{2,\mathsf{cm\text{-}NIZK}}$ can be used to simulate the $\mathsf{cm\text{-}NIZK}$ σ. This will be identically distributed, due to the zero knowledge of $\mathsf{cm\text{-}NIZK}$. Therefore the outputs of SIGN are distributed identically to in the non-transitivity experiment.

Simulating the CONVSIM *Oracle* What remains to be shown is that the $\mathsf{CONVSIM}$ oracle created by $\mathcal{D}_j$ either behaves identically to the $\mathsf{CONVSIM}$ oracle in Game $\mathcal{H}_j$ or as in $\mathcal{H}_{j+1}$, depending on whether its input was a DDH tuple or not. upk' is only extracted if $c_i, c\mu_i, c\sigma_i$ does not identify to any honest user, and therefore this will be successful. This is because otherwise $E_{2,\mathsf{cm\text{-}NIZK}}$ will extract a transformation T and statement $(cpk_1, bpk, ipk, c\mu', c')$ such that $(cpk_1, bpk, ipk, c\mu_i, c_i) = T_{\mathsf{inst}}(cpk_1, bpk, ipk, c\mu', c')$. In this case $(c\mu', c')$ was input to $S_{2,\mathsf{cm\text{-}NIZK}}$, and so $c\mu'$ is an encryption of $upk_{uid'}$. Therefore, $c\mu_i$ is also an encryption of $upk_{uid'}$ and so would have identified to an honest user. Although in the case of uid', $\mathbf{usk}[uid']$ is not defined, as $uid' \notin \mathsf{UL}$ then $\mathsf{CL}_{uid'}$ will never be used. $\overline{c}, \overline{c\sigma}$ are computed in the same way as in $\mathsf{CLS{+}.Convert}$. We know that $D_{3,s} = g_2^{\tilde{r}}$ for some $\tilde{r}$, and $upk' = g_1^{usk}$ for some usk, and thus it must hold that $\overline{c\mu} = (e(upk', \hat{g}^{a_1}), e(upk', D_{3,s}bpk_2^{a_1})) = (e(g_1, \hat{g}^{a_1})^{usk}, e(g_1, g_2^{\tilde{r}}bpk_2^{a_1})^{usk}) =$
$(e(g_1, \hat{g}^{a_1 usk\tilde{r}^{-1}})^{\tilde{r}}, e(g_1, upkbpk_2^{a_1 usk\tilde{r}^{-1}})^{\tilde{r}})$. This is distributed identically, as a_1 is chosen randomly and independently.

If (D'_1, D'_2, D'_3, D'_4) is a DDH tuple, then $D_{4,s} = D_2^{\tilde{r}}$. Therefore as $upk_{uid'} = D_2$, and as $(e(g_1, \hat{g}^{a_1}), e(g_1, D_{4,s}bpk_2^{a_1})) = (e(g_1, \hat{g}^{a_1}), e(g_1, upk_{uid'}^{\tilde{r}}bpk_2^{a_1})) =$
$(e(g_1, \hat{g}^{a_1\tilde{r}^{-1}})^{\tilde{r}}, e(g_1, upk_{uid'}bpk_2^{a_1\tilde{r}^{-1}})^{\tilde{r}})$ the inputs to $\mathcal{A}$ are also distributed identically to in Game $\mathcal{H}_{j+1}$. Whereas if (D'_1, D'_2, D'_3, D'_4) is *not* a DDH tuple, then $D_{4,s}$, is distributed identically to μ', which was chosen randomly and independently. Therefore the inputs to $\mathcal{A}$ are distributed identically to in Game $\mathcal{H}_j$.

Reduction to the DDH problem Therefore the probability that $\mathcal{D}_j$ outputs 1 if it was given a valid DDH tuple as input is $\Pr[S_{j+1}]$, and $\Pr[S_j]$ is the probability that $\mathcal{D}_j$ outputs 1 when the input was not a DDH tuple. The advantage of $\mathcal{D}_j$ is then $|\Pr[S_j] - \Pr[S_{j+1}]|$, therefore $|\Pr[S_j] - \Pr[S_{j+1}]| = \epsilon_{\mathsf{DDH}}$.

Overall, for our sequence of games $\mathcal{H}_0$ to $\mathcal{H}_q$ it holds that $|\Pr[S_0] - \Pr[S_q]| \leqslant q\epsilon_{\mathsf{DDH}}$ and thus $\epsilon \leqslant q\epsilon_{\mathsf{DDH}}$ is negligible. This concludes our proof that the $\mathsf{CLS\text{–}CM}$ construction satisfies non-transitivity.

D.5 Conversion Blindness

Lemma 3. *The CLS–CM construction satisfies* ***conversion blindness*** *if the DDH assumption holds in $\mathbb{G}_1$ and $\mathbb{G}_2$, the cm-NIZK is zero knowledge, and strongly derivation private and the SPK is Zero Knowledge.*

```
$\mathsf{CONVSIM}((c\mu_1, c\sigma_1, c_1), \ldots, (c\mu_k, c\sigma_k, c_k), bpk, bsk)$
$s \leftarrow s+1,$ if $(bpk, bsk) \notin \mathcal{BK}$  return $\perp$
if $\mathsf{CLS{+}.Convert}(gpk, bpk, csk, (c\mu_1, c\sigma_1, c_1), \ldots, (c\mu_k, c\sigma_k, c_k)) = \perp$  return $\perp$
Set $\mathsf{CL} \leftarrow \emptyset$
$\forall i \in [1,k]$
  if $\exists uid \in \mathsf{HUL}$ s.t $c\mu_{i,3} c\mu_{i,2}^{-bsk_2} c\mu_{i,1}^{-csk_1} = upk_{uid}$
    if $\mathsf{L}_{uid}$ does not exist $\mathsf{L}_{uid} \leftarrow \{c_i\}, \mathsf{CL}_{uid} \leftarrow \{(c_i, g_1^{\mathbf{usk}[uid]})\}$
    else $L_{uid} \leftarrow \mathsf{L}_{uid} \cup \{c_i\}, \mathsf{CL}_{uid} \leftarrow \mathsf{CL}_{uid} \cup \{(c_i, g_1^{\mathbf{usk}[uid]})\}$
  else $(upk_i', \cdot, \cdot, \cdot, \cdot \ldots) \leftarrow E_{2,\mathsf{cm\text{-}NIZK}}(\sigma_{\mathsf{crs}}, \tau_e, (cpk_1, bpk, ipk, c\mu_i, c_i), c\sigma_i)$
    $\mathsf{CL} \leftarrow \mathsf{CL} \cup \{(c_i, upk_i')\}$
$n \leftarrow 0$;  $\forall (c, upk') \in \mathsf{CL} \cup \bigcup_{uid \in \mathsf{UL}} \mathsf{CL}_{uid}$
  $n \leftarrow n+1, a_1, a_2 \leftarrow\$ \mathbb{Z}_p^*, \overline{c\mu}_n \leftarrow (e(upk', \hat{g}^{a_1}), e(upk', D_{3,s} bpk_2^{a_1}))$
  $\overline{c}_n \leftarrow (c_1 g^{a_2}, c_2 bpk_1^{a_2}), \overline{c\sigma}_n \leftarrow \mathsf{Sig}(csk_2, (\overline{c\mu}_n, \overline{c}_n, bpk))$
if $\mathsf{L}_{uid'}$ exists $\forall c \in L_{uid'}$
  $n \leftarrow n+1, a_1, a_2 \leftarrow\$ \mathbb{Z}_p^*, \overline{c\mu}_n \leftarrow (e(g_1, \hat{g}^{a_1}), e(g_1, D_{4,s} bpk_2^{a_1}))$
  $\overline{c}_n \leftarrow (c_1 g^{a_2}, c_2 bpk_1^{a_2}), \overline{c\sigma}_n \leftarrow \mathsf{Sig}(csk, (\overline{c\mu}_n, \overline{c}_n, bpk))$
let $\mathsf{L}_{uid_1}, \ldots \mathsf{L}_{uid_{k''}}$ be the non-empty message clusters created above
Let $\mathsf{NUL} \leftarrow \{uid_1, \ldots uid_{k''}\} \backslash \mathsf{UL}$
$\{(\overline{c\mu}_i, \overline{c}_i, \overline{c\sigma}_i)\}_{i=n+1,\ldots k} \leftarrow \mathsf{SIM}(gpk, bpk, csk \bigcup_{uid \in \mathsf{NUL}, uid \neq uid'} \mathsf{L}_{uid})$
Let $\{(\overline{c\mu}_i', \overline{c\sigma}_i', \overline{c}_i')\}_{i=1,\ldots k}$ be a random permutation of $\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_{i=1,\ldots k}$
return $(\{(\overline{c\mu}_i', \overline{c\sigma}_i', \overline{c}_i')\}_{i=1,\ldots k})$
```

Fig. 14. The CONVSIM oracle used by distinguisher $\mathcal{D}_j$ given in Figure 14. To avoid confusion, we write uid' to refer to the $j+1$-th user that has joined the group (and for which $\mathcal{D}_j$ embedded the DDH challenge).

We define Game 0 be the conversion blindness experiment for the CLS–CM construction. Let P_0 be the event that an adversary $\mathcal{A}$ correctly guesses b after Game 0.

We define Game 1 to be identical to Game 0, except that σ_{crs} is generated by $S_{1,\mathsf{cm\text{-}NIZK}}$ instead of CRSSetup, and a simulation trapdoor is also generated. As outputs of $S_{1,\mathsf{cm\text{-}NIZK}}$ and CRSSetup are identically distributed, letting P_1 be the event that the adversary $\mathcal{A}$ correctly guesses b after Game 1, then $Pr[P_0] = Pr[P_1]$.

We define Game 2 to be identical to Game 1, except during blinding instead of transforming the proof with ZKEval, instead the proof is simulated. We give Game 2 in full below. Let P_2 be the event that the adversary $\mathcal{A}$ correctly guesses b after Game 2.

$\underline{\mathsf{UNBLIND}((\mu_1, \sigma_1, m_1), \ldots, (\mu_k, \sigma_k, m_k))}$

As in Blindness experiment

Game 2

```
$b \leftarrow\!\$\, \{0,1\}, (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathcal{G}(1^\tau), g \leftarrow\!\$\, \mathbb{G}_1, \hat{g} \leftarrow\!\$\, \mathbb{G}_2$
$\mathsf{pp}_{\mathsf{auto}\,1} \leftarrow \mathsf{ASetup}_1(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2))$
$\mathsf{pp}_{\mathsf{auto}\,2} \leftarrow \mathsf{ASetup}_2(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)), (\sigma_{\mathsf{crs}}, \tau_s) \leftarrow S_{1,\mathsf{cm\text{-}NIZK}}(1^\tau)$
$\mathsf{pp}_{\mathsf{sig}} \leftarrow \mathsf{SIG.Setup}(1^\tau)$
$\mathsf{pp} \leftarrow ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{pp}_{\mathsf{auto}\,1}, \mathsf{pp}_{\mathsf{auto}\,2}, g, \hat{g}, \sigma_{\mathsf{crs}}, \mathsf{pp}_{\mathsf{sig}})$
$(ipk, isk) \leftarrow \mathsf{CLS{+}.IKGen}(\mathsf{pp}), (cpk, csk) \leftarrow \mathsf{CLS{+}.CKGen}(\mathsf{pp})$
$(bpk, bsk) \leftarrow \mathsf{CLS{+}.BKGen}(\mathsf{pp}), gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$(st, (\mu_0, \sigma_0, m_0), (\mu_1, \sigma_1, m_1)) \leftarrow \mathcal{A}^{\mathsf{UNBLIND}}(\mathsf{choose}, gpk, bpk, isk, csk)$
$\textbf{if } \exists d \in \{0,1\} \text{ s.t } \mathsf{CLS{+}.Verify}(\mathtt{tier}\text{-}1, gpk, bpk, m_d, \mu_d, \sigma_d) = 0 \quad \textbf{return } 0$
$\alpha, \beta, \gamma \leftarrow\!\$\, \{0,1\}^*, c\mu^* \leftarrow (\mu_{b,1}\hat{g}^\alpha, \mu_{b,2}\hat{g}^\beta, \mu_{b,3} cpk_1^\alpha bpk_2^\beta),$
$c^* \leftarrow (g^\gamma, m_b bpk_1^\gamma), c\sigma^* \leftarrow S_{2,\mathsf{cm\text{-}NIZK}}(\sigma_{\mathsf{crs}}, \tau_s, (cpk_1, bpk, ipk, c\mu^*, c^*))$
$b^* \leftarrow \mathcal{A}^{\mathsf{UNBLIND}}(\mathsf{guess}, st, c\mu^*, c\sigma^*, c^*) \quad \textbf{if } b^* = b \quad \textbf{return } 1 \quad \textbf{else return } 0$
```

We show that Game 1 and Game 2 are indistinguishable assuming the $\mathsf{cm\text{-}NIZK}$ proof is strongly derivation private. We give a distinguishing algorithm $\mathcal{D}_1$ in Figure 15, that aims to guess b' in the strongly derivation private security game.

We now show that when $b' = 1$ inputs to $\mathcal{A}$ are identical to Game 1, and when $b' = 0$ inputs to $\mathcal{A}$ are identical to Game 2.

The gpk, bpk, isk, csk input to $\mathcal{A}$ are identical to the input in both Game 1 and Game 2. The $\mathsf{UNBLIND}$ oracle is also identical. $c\mu^*$ and c^* are re-randomisations of μ_b and $(1, m_b)$ which is identical to in the $\mathsf{CLS{+}.Blind}$ algorithm, and therefore identical to in both Game 1 and Game 2.

If $b' = 0$, $\mathcal{D}_1$ is returned with the simulation of a proof for the statement $T(cpk_1, bpk, ipk, \mu_b, (1, m_b)) = (cpk_1, bpk, ipk, c\mu^*, c^*)$, which is identical to in Game 2.

If $b' = 1$, $\mathcal{D}_1$ is returned with the transformation of the proof under the transformation defined by (α, β, γ), which are used to re-randomise μ_b and $(1, m_b)$. This is identical to in Game 1.

Therefore $|Pr[P_1] - Pr[P_2]| \leqslant \epsilon_{sdp}$, where ϵ_{sdp} is the advantage in breaking the strong derivation privacy of the $\mathsf{cm\text{-}NIZK}$.

We define Game 3 to be identical to Game 2, except $c\mu^*$ is chosen randomly. We give Game 3 in full below. Let P_3 be the event that the adversary $\mathcal{A}$ correctly guesses b after Game 3.

```
$\underline{\mathsf{UNBLIND}((\mu_1, \sigma_1, m_1), \ldots, (\mu_k, \sigma_k, m_k))}$
As in Blindness experiment

$\underline{\mathcal{D}_1(\sigma_{\mathsf{crs}})}$
$b \leftarrow\$ \{0,1\}, (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathcal{G}(1^\tau), g \leftarrow\$ \mathbb{G}_1, \hat{g} \leftarrow\$ \mathbb{G}_2$
$\mathsf{pp}_{\mathsf{auto}\,1} \leftarrow \mathsf{ASetup}_1(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2))$
$\mathsf{pp}_{\mathsf{auto}\,2} \leftarrow \mathsf{ASetup}_2(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)), (\sigma_{\mathsf{crs}}, \tau_s) \leftarrow S_{1,\mathsf{cm\text{-}NIZK}}(1^\tau)$
$\mathsf{pp}_{\mathsf{sig}} \leftarrow \mathsf{SIG.Setup}(1^\tau)$
$\mathsf{pp} \leftarrow ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{pp}_{\mathsf{auto}\,1}, \mathsf{pp}_{\mathsf{auto}\,2}, g, \hat{g}, \sigma_{\mathsf{crs}}, \mathsf{pp}_{\mathsf{sig}})$
$(ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp}), (cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp})$
$(bpk, bsk) \leftarrow \mathsf{CLS+.BKGen}(\mathsf{pp}), gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$(st, (\mu_0, \sigma_0, m_0), (\mu_1, \sigma_1, m_1)) \leftarrow \mathcal{A}^{\mathsf{UNBLIND}}(\mathsf{choose}, gpk, bpk, isk, csk)$
**if** $\exists d \in \{0,1\}$ s.t $\mathsf{CLS+.Verify}(\texttt{tier}\text{-}1, gpk, bpk, m_d, \mu_d, \sigma_d) = 0$   **return** $0$
$\alpha, \beta, \gamma \leftarrow\$ \{0,1\}^*$
**return** $(st, (cpk_1, bpk, ipk, \mu_b, (1, m_b)), \sigma_b, (\alpha, \beta, \gamma))$

$\underline{\mathcal{D}_1(st, \pi^*)}$
$c\mu_1^* \leftarrow \mu_{b,1}\hat{g}^\alpha, c\mu_2^* \leftarrow \mu_{b,2}\hat{g}^\beta, c\mu_3^* \leftarrow \mu_{b,3} cpk_1^\alpha bpk_2^\beta$
$c_1^* \leftarrow g^\gamma, c_2^* \leftarrow m_b bpk_1^\gamma$
$b^* \leftarrow \mathcal{A}(\mathsf{guess}, st, cupk, c^*, c\mu^*, \pi^*)$,   **if** $b^* = b$   **return** $1$
```

Fig. 15. $\mathcal{D}_1$ that distinguishes between Game 1 and Game 2

```
$\underline{\mathsf{UNBLIND}((\mu_1, \sigma_1, m_1), \ldots, (\mu_k, \sigma_k, m_k))}$
As in Blindness experiment
```

```
Game 3
$b \leftarrow\$ \{0,1\}, (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathcal{G}(1^\tau), g \leftarrow\$ \mathbb{G}_1, \hat{g} \leftarrow\$ \mathbb{G}_2$
$\mathsf{pp}_{\mathsf{auto}\,1} \leftarrow \mathsf{ASetup}_1(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2))$
$\mathsf{pp}_{\mathsf{auto}\,2} \leftarrow \mathsf{ASetup}_2(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)), (\sigma_{\mathsf{crs}}, \tau_s) \leftarrow S_{1,\mathsf{cm\text{-}NIZK}}(1^\tau)$
$\mathsf{pp}_{\mathsf{sig}} \leftarrow \mathsf{SIG.Setup}(1^\tau)$
$\mathsf{pp} \leftarrow ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{pp}_{\mathsf{auto}\,1}, \mathsf{pp}_{\mathsf{auto}\,2}, g, \hat{g}, \sigma_{\mathsf{crs}}, \mathsf{pp}_{\mathsf{sig}})$
$(ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp}), (cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp})$
$(bpk, bsk) \leftarrow \mathsf{CLS+.BKGen}(\mathsf{pp}), gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$(st, (\mu_0, \sigma_0, m_0), (\mu_1, \sigma_1, m_1)) \leftarrow \mathcal{A}^{\mathsf{UNBLIND}}(\mathsf{choose}, gpk, bpk, isk, csk)$
$\mathbf{if}\ \exists d \in \{0,1\}$ s.t $\mathsf{CLS+.Verify}(\mathtt{tier}\text{-}1, gpk, bpk, m_d, \mu_d, \sigma_d) = 0 \quad \mathbf{return}\ 0$
$\gamma \leftarrow\$ \{0,1\}^*, c\mu^* \leftarrow\$ \mathbb{G}_2^3, c_1^* \leftarrow g^\gamma, c_2^* \leftarrow m_b bpk_1^\gamma$
$c\sigma^* \leftarrow S_{2,\mathsf{cm\text{-}NIZK}}(\sigma_{\mathsf{crs}}, \tau_s, (cpk_1, bpk, ipk, c\mu^*, c^*))$
$b^* \leftarrow \mathcal{A}^{\mathsf{UNBLIND}}(\mathsf{guess}, st, c\mu^*, c\sigma^*, c^*) \quad \mathbf{if}\ b^* = b \quad \mathbf{return}\ 1 \quad \mathbf{return}\ 0$
```

We define Game 4 to be identical to Game 3, except c^* is chosen randomly. We give Game 4 in full below. Let P_4 be the event that the adversary $\mathcal{A}$ correctly guesses b after Game 4.

```
$\underline{\mathsf{UNBLIND}((\mu_1, \sigma_1, m_1), \ldots, (\mu_k, \sigma_k, m_k))}$
As in Blindness experiment
```

```
Game 4
$b \leftarrow\$ \{0,1\}, (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathcal{G}(1^\tau), g \leftarrow\$ \mathbb{G}_1, \hat{g} \leftarrow\$ \mathbb{G}_2$
$\mathsf{pp}_{\mathsf{auto}\,1} \leftarrow \mathsf{ASetup}_1(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2))$
$\mathsf{pp}_{\mathsf{auto}\,2} \leftarrow \mathsf{ASetup}_2(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)), (\sigma_{\mathsf{crs}}, \tau_s) \leftarrow S_{1,\mathsf{cm\text{-}NIZK}}(1^\tau)$
$\mathsf{pp}_{\mathsf{sig}} \leftarrow \mathsf{SIG.Setup}(1^\tau)$
$\mathsf{pp} \leftarrow ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{pp}_{\mathsf{auto}\,1}, \mathsf{pp}_{\mathsf{auto}\,2}, g, \hat{g}, \sigma_{\mathsf{crs}}, \mathsf{pp}_{\mathsf{sig}})$
$(ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp}), (cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp})$
$(bpk, bsk) \leftarrow \mathsf{CLS+.BKGen}(\mathsf{pp}), gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$(st, (\mu_0, \sigma_0, m_0), (\mu_1, \sigma_1, m_1)) \leftarrow \mathcal{A}^{\mathsf{UNBLIND}}(\mathsf{choose}, gpk, bpk, isk, csk)$
$\mathbf{if}\ \exists d \in \{0,1\}$ s.t $\mathsf{CLS+.Verify}(\mathtt{tier}\text{-}1, gpk, bpk, m_d, \mu_d, \sigma_d) = 0 \quad \mathbf{return}\ 0$
$\gamma \leftarrow\$ \{0,1\}^*, c\mu^* \leftarrow\$ \mathbb{G}_2^3, c^* \leftarrow\$ \mathbb{G}_1^2$
$c\sigma^* \leftarrow S_{2,\mathsf{cm\text{-}NIZK}}(\sigma_{\mathsf{crs}}, \tau_s, (cpk_1, bpk, ipk, c\mu^*, c^*))$
$b^* \leftarrow \mathcal{A}^{\mathsf{UNBLIND}}(\mathsf{guess}, st, c\mu^*, c\sigma^*, c^*) \quad \mathbf{if}\ b^* = b \quad \mathbf{return}\ 1 \quad \mathbf{return}\ 0$
```

We show that Game 2 and Game 3 are indistinguishable assuming the DDH assumption in $\mathbb{G}_2$. We give a distinguishing algorithm $\mathcal{D}_2$ in Figure 16.

```
$\mathsf{UNBLIND}((\mu_1,\sigma_1,m_1),\ldots,(\mu_k,\sigma_k,m_k))$
if $\exists i \in [k]$ s.t $\mathsf{CLS+.Verify}(\mathtt{tier}\text{-}1, gpk, bpk, m_i, \mu_i, \sigma_i) = 0$ return $\perp$
$r \leftarrow\$ \mathbb{Z}_p^*, \forall i \in [1,k]$
    $r'_{i,1} r'_{i,2}, r'_{i,3} \leftarrow\$ \mathbb{Z}_p^*, c\mu_i \leftarrow (\mu_{i,1}\hat{g}^{r'_{i,1}}, \mu_{i,2}\hat{g}^{r'_{i,2}}, \mu_{i,3} cpk_1^{r'_{i,1}} bpk_2^{r'_{i,2}})$
    $c_i \leftarrow (g^{r'_{i,3}}, m_i \cdot bpk_1^{r'_{i,3}})$
    $r''_{i,1}, r''_{i,2}, r''_{i,3} \leftarrow\$ \mathbb{Z}_p^*, \overline{c_i} \leftarrow (c_{i,1} g^{r''_{i,3}}, c_{i,2} bpk_1^{r''_{i,3}})$
    $\overline{\mu}_i \leftarrow e(g_1, \mu_{i,3}\mu_{i,1}^{-csk_1})^r$
    $\overline{c\mu}_i \leftarrow (e(g_1, \hat{g}^{r'_{i,2}+r''_{i,2}})^r, e(g_1, bpk_2^{r'_{i,2}+r''_{i,2}} \mu_{i,3}\mu_{i,1}^{-csk_1})^r)$
    $\overline{c\sigma} \leftarrow \mathsf{SIG.Sign}(csk_2, (\overline{c_i}, \overline{c\mu}_i, bpk))$
    Simulate $\pi_i$ with $\overline{\mu}_i, \overline{c\mu}_i, m_i, \overline{c_i}$
    $\overline{\sigma_i} \leftarrow (\overline{c\mu_i}, \overline{c\sigma}_i, \overline{c_i}, \pi_i)$
choose random permutation $\Pi$, for $i = 1, \ldots, k : (\overline{\mu}_i, \overline{\sigma_i}, m_i) \leftarrow (\overline{\mu}_{\Pi(i)}, \overline{\sigma_{\Pi(i)}}, m_{\Pi(i)})$
return $(\{(\overline{\mu}_i, \overline{\sigma_i}, m_i)\}_k, \{r'_{i,1}, r'_{i,2}, r'_{i,3}\}_k, \{r''_{i,1}, r''_{i,2}, r''_{i,3}\}_k, r, \Pi)$
```

```
$\mathcal{D}_2(D_1, D_2, D_3, D_4)$
$b \leftarrow\$ \{0,1\}, (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathcal{G}(1^\tau), g \leftarrow\$ \mathbb{G}_1, \hat{g} \leftarrow D_1$
$\mathsf{pp}_{\mathsf{auto}1} \leftarrow \mathsf{ASetup}_1(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2))$
$\mathsf{pp}_{\mathsf{auto}2} \leftarrow \mathsf{ASetup}_2(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)), (\sigma_{\mathsf{crs}}, \tau_s) \leftarrow S_{1,\mathsf{cm\text{-}NIZK}}(1^\tau)$
$\mathsf{pp}_{\mathsf{sig}} \leftarrow \mathsf{SIG.Setup}(1^\tau)$
$\mathsf{pp} \leftarrow ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{pp}_{\mathsf{auto}1}, \mathsf{pp}_{\mathsf{auto}2}, g, \hat{g}, \sigma_{\mathsf{crs}}, \mathsf{pp}_{\mathsf{sig}})$
$(ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp}), (cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp})$
$bsk_1 \leftarrow\$ \mathbb{Z}_p^*, bpk_1 \leftarrow g^{bsk_1}, bpk_2 \leftarrow D_2, gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$(st, (\mu_0, \sigma_0, m_0), (\mu_1, \sigma_1, m_1)) \leftarrow \mathcal{A}^{\mathsf{UNBLIND}}(\mathsf{choose}, gpk, bpk, isk, csk)$
if $\exists d \in \{0,1\}$ s.t $\mathsf{CLS+.Verify}(\mathtt{tier}\text{-}1, gpk, bpk, m_d, \mu_d, \sigma_d) = 0$ return $0$
$\gamma, \beta \leftarrow\$ \mathbb{Z}_p^*, c\mu^* \leftarrow (\mu_{b,1}\hat{g}^\beta, D_3, \mu_{b,3} D_4 cpk_1^\beta), c_1^* \leftarrow g^\gamma, c_2^* \leftarrow m_b bpk_1^\gamma$
$c\sigma^* \leftarrow S_{2,\mathsf{cm\text{-}NIZK}}(\sigma_{\mathsf{crs}}, \tau_s, (cpk_1, bpk, ipk, c\mu^*, c^*))$
$b^* \leftarrow \mathcal{A}^{\mathsf{UNBLIND}}(\mathsf{guess}, st, c\mu^*, c\sigma^*, c^*)$ if $b^* = b$ return $1$ return $0$
```

Fig. 16. $\mathcal{D}_2$ that distinguishes between Game 2 and Game 3

If $\mathcal{D}_2$ is input a DDH tuple, all inputs to $\mathcal{A}$ are distributed identically to in Game 2. This is because, letting $\alpha = \log_{\hat{g}} D_3$, then $D_4 = bpk_2^{\alpha}$, and therefore $c\mu^*$ is distributed identically to in Game 2.

If $\mathcal{D}_2$ is not input a DDH tuple, all inputs to $\mathcal{A}$ are distributed identically to in Game 3. This is because, β, D_3, D_4 are now chosen independently and randomly.

Simulating the $\mathsf{UNBLIND}$ *oracle.* Other than this all inputs are generated identically, except for the $\mathsf{UNBLIND}$ oracle, which must be simulated as bsk_2 is not known. This is distributed correctly because: $c\mu_i, c_i, \overline{c\mu_i}, \overline{c}_i$, are distributed identically to in $\mathsf{CLS{+}.Blind}$, $\mathsf{CLS{+}.Convert}$ with randomness $r'_{i,1}, r'_{i,2}, r'_{i,3}$ and $(r''_{i,1}, r''_{i,2}, r''_{i,3}), r$ respectively. $\overline{c\sigma}_i$ is computed identically to in $\mathsf{CLS{+}.Convert}$. The unblinded pseudonym $\overline{\mu}_i = e(g_1, \mu_{i,3}\mu_{i,1}^{-csk_1})^r = \overline{c\mu_{i,2}} \cdot \overline{c\mu_{i,1}}^{-bsk_2}$, and so is distributed correctly. $\overline{m}_i = m_i$ due to the correctness of the ElGamal encryption scheme. Finally, π_i can be simulated due to the zero knowledge property of the SPK. Therefore all outputs of $\mathsf{UNBLIND}$ are distributed identically to in the experiment.

Therefore $|Pr[P_2] - Pr[P_3] \leqslant \epsilon_{DDH}$, where ϵ_{DDH} is the DDH advantage, and therefore negligible.

We show that Game 3 and Game 4 are indistinguishable assuming the DDH assumption in $\mathbb{G}_1$. We give a distinguishing algorithm $\mathcal{D}_3$ in Figure 17.

If $\mathcal{D}_3$ is input a DDH tuple, all inputs to $\mathcal{A}$ are distributed identically to in Game 3. This is because, letting $\gamma = \log_g D_3$, then $D_4 = bpk_1^{\gamma}$, and so c^* is distributed identically to Game 3.

If $\mathcal{D}_3$ is not input a DDH tuple, all inputs to $\mathcal{A}$ are distributed identically to in Game 4. This is because c_1^*, c_2^* are now chosen independently and randomly.

Simulating the $\mathsf{UNBLIND}$ *oracle.* Other than this all inputs are generated identically, except for the $\mathsf{UNBLIND}$ oracle, which must be simulated as bsk_1 is not known. This is done identically to in the previous distinguisher $\mathcal{D}_2$.

Therefore $|Pr[P_3] - Pr[P_4] \leqslant \epsilon_{DDH}$ and therefore $|Pr[P_0] - Pr[P_4]| \leqslant 2\epsilon_{DDH} + \epsilon_{\mathsf{sdp}}$, and so $|Pr[P_0] - 1/2| \leqslant 2\epsilon_{DDH} + \epsilon_{\mathsf{sdp}}$. Therefore assuming the DDH assumption and the strong derivation privacy of the $\mathsf{cm\text{-}NIZK}$, the advantage of any polynomial time adversary in the conversion blindness game with the $\mathsf{CLS\text{–}CM}$ construction is negligible

D.6 Conversion Unforgeability

Lemma 4. *The $\mathsf{CLS\text{–}CM}$ construction satisfies **conversion unforgeability** if the signature scheme is EUF-CMA secure, and the SPK is sound.*

We build an adversary $\mathcal{A}'$, that successfully wins the EUF-CMA game for a digital signature scheme, given $\mathcal{A}$ that wins the conversion unforgeability game for the $\mathsf{CLS\text{–}CM}$ construction with non-negligible probability ϵ. We give $\mathcal{A}'$ in Figure 18, and below explain why the simulation input to $\mathcal{A}$ given in Figure 18 is identically distributed to the conversion unforgeability experiment for the

```
$\mathsf{UNBLIND}(\mu_1, \sigma_1, m_1), \ldots, (\mu_k, \sigma_k, m_k))$
$\textbf{if } \exists i \in [k] \text{ s.t } \mathsf{CLS+.Verify}(\texttt{tier-}1, gpk, bpk, m_i, \mu_i, \sigma_i) = 0 \quad \textbf{return } \perp$
$r \leftarrow\$ \mathbb{Z}_p^*, \forall i \in [1, k]$
    $r'_{i,1}r'_{i,2}, r'_{i,3} \leftarrow\$ \mathbb{Z}_p^*, c\mu_i \leftarrow (\mu_{i,1}\hat{g}^{r'_{i,1}}, \mu_{i,2}\hat{g}^{r'_{i,2}}, \mu_{i,3}cpk_1^{r'_{i,1}} bpk_2^{r'_{i,2}})$
    $c_i \leftarrow (g^{r'_{i,3}}, m_i \cdot bpk_1^{r'_{i,3}})$
    $r''_{i,1}, r''_{i,2}, r''_{i,3} \leftarrow\$ \mathbb{Z}_p^*, \overline{c_i} \leftarrow (c_{i,1}g^{r''_{i,3}}, c_{i,2}bpk_1^{r''_{i,3}})$
    $\overline{\mu}_i \leftarrow e(g_1, \mu_{i,3}\mu_{i,1}^{-csk_1})^r$
    $\overline{c\mu}_i \leftarrow (e(g_1, \hat{g}^{r'_{i,2}+r''_{i,2}})^r, e(g_1, bpk_2^{r'_{i,2}+r''_{i,2}}\mu_{i,3}\mu_{i,1}^{-csk_1})^r)$
    $\overline{c\sigma} \leftarrow \mathsf{SIG.Sign}(csk_2, (\overline{c_i}, \overline{c\mu}_i, bpk))$
    Simulate $\pi_i$ with $\overline{\mu}_i, \overline{c\mu}_i, m_i, \overline{c_i}$
    $\overline{\sigma_i} \leftarrow (\overline{c\mu_i}, \overline{c\sigma}_i, \overline{c_i}, \pi_i)$
choose random permutation $\Pi$, $\textbf{for } i = 1, \ldots, k : (\overline{\mu}_i, \overline{\sigma_i}, m_i) \leftarrow (\overline{\mu}_{\Pi(i)}, \overline{\sigma_{\Pi(i)}}, m_{\Pi(i)})$
$\textbf{return } (\{(\overline{\mu}_i, \overline{\sigma_i}, m_i)\}_k, \{r'_{i,1}, r'_{i,2}, r'_{i,3}\}_k, \{r''_{i,1}, r''_{i,2}, r''_{i,3}\}_k, r, \Pi)$
```

```
$\mathcal{D}_2(D_1, D_2, D_3, D_4)$
$b \leftarrow\$ \{0, 1\}, (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathcal{G}(1^\tau), g \leftarrow\$ D_1, \hat{g} \leftarrow \mathbb{G}_1$
$\mathsf{pp}_{\mathsf{auto}1} \leftarrow \mathsf{ASetup}_1(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2))$
$\mathsf{pp}_{\mathsf{auto}2} \leftarrow \mathsf{ASetup}_2(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)), (\sigma_{\mathsf{crs}}, \tau_s) \leftarrow S_{1,\mathsf{cm\text{-}NIZK}}(1^\tau)$
$\mathsf{pp}_{\mathsf{sig}} \leftarrow \mathsf{SIG.Setup}(1^\tau)$
$\mathsf{pp} \leftarrow ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{pp}_{\mathsf{auto}1}, \mathsf{pp}_{\mathsf{auto}2}, g, \hat{g}, \sigma_{\mathsf{crs}}, \mathsf{pp}_{\mathsf{sig}})$
$(ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp}), (cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp})$
$bsk_2 \leftarrow\$ \mathbb{Z}_p^*, bpk_2 \leftarrow \hat{g}^{bsk_2}, bpk_1 \leftarrow D_2, gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$(st, (\mu_0, \sigma_0, m_0), (\mu_1, \sigma_1, m_1)) \leftarrow \mathcal{A}^{\mathsf{UNBLIND}}(\mathsf{choose}, gpk, bpk, isk, csk)$
$\textbf{if } \exists d \in \{0, 1\} \text{ s.t } \mathsf{CLS+.Verify}(\texttt{tier-}1, gpk, bpk, m_d, \mu_d, \sigma_d) = 0 \quad \textbf{return } 0$
$c\mu^* \leftarrow\$ \mathbb{G}_2^3, c_1^* \leftarrow D_3, c_2^* \leftarrow m_b D_4$
$c\sigma^* \leftarrow S_{2,\mathsf{cm\text{-}NIZK}}(\sigma_{\mathsf{crs}}, \tau_s, (cpk_1, bpk, ipk, c\mu^*, c^*))$
$b^* \leftarrow \mathcal{A}^{\mathsf{UNBLIND}}(\mathsf{guess}, st, c\mu^*, c\sigma^*, c^*) \quad \textbf{if } b^* = b \quad \textbf{return } 1 \quad \textbf{return } 0$
```

Fig. 17. $\mathcal{D}_3$ that distinguishes between Game 3 and Game 4

CLS–CM construction, and that $\mathcal{A}'$ successfully breaks the EUF-CMA security of the digital signature scheme.

```
$\mathsf{CONVERT}(c\mu_1, c\sigma_1, c_1), \ldots, (c\mu_k, c\sigma_k, c_k), bpk, bsk)$
As normal but instead of $\sigma'_i \leftarrow \mathsf{SIG.Sign}(csk_2, (c'_i, c\mu'''_i, bpk))$ use $\mathsf{SIG}$ oracle

$\mathcal{A}'^{\mathsf{SIG}}(\mathsf{pp}_{\mathsf{sig}}, pk_{\mathsf{sig}})$
$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2) \leftarrow \mathcal{G}(1^\tau), \mathsf{pp}_{\mathsf{auto}1} \leftarrow \mathsf{ASetup}_1(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2),$
$\mathsf{pp}_{\mathsf{auto}2} \leftarrow \mathsf{ASetup}_2(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$
$g \leftarrow\$ \mathbb{G}_1, \hat{g} \leftarrow\$ \mathbb{G}_2, \sigma_{\mathsf{crs}} \leftarrow \mathsf{CRSSetup}(1^\tau)$
$\mathsf{pp} \leftarrow ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{pp}_{\mathsf{auto}1}, \mathsf{pp}_{\mathsf{auto}2}, g, \hat{g}, \sigma_{\mathsf{crs}}, \mathsf{pp}_{\mathsf{sig}})$
$(ipk, isk) \leftarrow \mathsf{CLS+.IKGen}(\mathsf{pp})$
$csk_1 \leftarrow\$ \mathbb{Z}_p^*, cpk_1 \leftarrow \hat{g}^{csk_1}, cpk_2 \leftarrow pk_{\mathsf{sig}}, cpk \leftarrow (cpk_1, cpk_2)$
$gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$(\overline{\mu}, \overline{\sigma}, m, bpk) \leftarrow \mathcal{A}^{\mathsf{CONVERT}}(gpk, isk)$
Parse $\overline{\sigma} = (\overline{c\mu}, \overline{c\sigma}, \overline{c}, \pi_{\mathsf{ub}})$
return $(\overline{c\mu}, \overline{c}, bpk), \overline{c\sigma}$
```

Fig. 18. $\mathcal{A}'$ which breaks the EUF-CMA security of a digital signature scheme using $\mathcal{A}$

(gpk, isk) are computed identically to in the conversion unforgeability game, except cpk_2 and $\mathsf{pp}_{\mathsf{sig}}$ are the public key and parameters input to $\mathcal{A}'$ and so distributed identically. The CONVERT oracle is identical to in the conversion unforgeability game, except as csk_2 is not known, the signing oracle is used instead to generate signatures.

Reduction to the EUF-CMA security of the digital signature scheme We assume $\mathcal{A}$ is successful, then $\mathsf{CLS+.Verify}(\mathtt{tier}\text{-}2, gpk, bpk, m, \overline{\mu}, \overline{\sigma}) = 1$. Therefore, $\overline{c\sigma}$ is a valid signature on $(\overline{c\mu}, \overline{c}, bpk)$. We now need to show that $(\overline{c\mu}, \overline{c}, bpk)$ was not input to SIG. We have that $(\overline{\mu}, m, \cdot, \cdot, \cdot, bpk) \notin \mathsf{UBL}$. If $(\overline{c\mu}, \overline{c}, bpk)$ was input to SIG, then due to the soundness of the SPK, the same $\overline{\mu}, m$ would have been generated in the CLS+.Convert oracle and so saved in UBL. This is a contradiction.

Therefore $\mathcal{A}'$ successfully breaks the EUF-CMA of the digital signature scheme with probability ϵ. Therefore, assuming the EUF-CMA security of the digital signature scheme, the CLS–CM construction satisfies conversion unforgeability.

D.7 Non-Frameability

Lemma 5. *The CLS–CM construction satisfies* ***non-frameability*** *if the automorphic signatures are EUF-CMA secure, and the cm-NIZK is zero knowledge, and cm-SSE.*

We build an adversary $\mathcal{A}'$, that breaks the EUF-CMA security of the automorphic signature scheme with non-negligible probability, given $\mathcal{A}$ that wins the non-frameability game for the $\mathsf{CLS\text{–}CM}$ construction with non-negligible probability ϵ and makes q queries to the SNDU oracle for distinct users. We give $\mathcal{A}'$ in Figure 19, and below explain why the simulation input to $\mathcal{A}$ given in Figure 19 is identically distributed to the non-frameability experiment for the $\mathsf{CLS\text{–}CM}$ construction, and that $\mathcal{A}'$ successfully breaks the EUF-CMA security of the automorphic signature scheme.

(gpk, isk) are computed identically to in the non–frameability game, except that $\mathsf{pp}_{\mathsf{auto1}}$ are the automorphic signature scheme parameters input to $\mathcal{A}'$ and so distributed identically.

Simulating the SNDU oracles: When $uid \neq uid^*$, the SNDU oracle is identical to in the non-frameability experiment. When $uid = uid^*$, upk_{uid^*} is set to apk, which is distributed identically to in $\mathsf{CLS{+}.Join}$. $\mathbf{usk}[uid], upk'$ are set to $\perp$ but these are not used later.

Simulating the SIGN oracle: In the case of $uid \neq uid^*$, this is identical to in the non-frameability experiment. When $uid = uid^*$, μ, c are generated identically to in $\mathsf{CLS{+}.Sign}$. σ can be simulated due to the zero knowledge property of the $\mathsf{cm\text{-}NIZK}$ proofs used.

Reduction to EUF-CMA security of automorphic signatures. We assume $\mathcal{A}$ is successful. We now argue that $\mathcal{A}'$ successfully breaks the EUF-CMA security of the automorphic signature scheme.

We assume with probability $1/q$ that $\mathcal{A}'$ guesses correctly and $uid = uid^*$. First, we need to show that $\mathcal{A}'$ will successfully extract apk, Ω, m. $c\sigma$ is a valid $\mathsf{cm\text{-}NIZK}$ as otherwise $\mathsf{Identify}$ would not be successful. As $\mathsf{Identify}(uid^*, c, c\mu, c\sigma) = 1$, then $c\mu$ is an encryption of $upk_{uid^*} = apk$. $E_{2,\mathsf{cm\text{-}NIZK}}$ will either extract a valid witness, which would ensure that $\mathsf{AVerify}_1(\Omega, apk, m) = 1$; or $E_{2,\mathsf{cm\text{-}NIZK}}$ will extract a statement x' and transformation T such that $x = T_{\mathsf{inst}}(x')$, where x is the statement output, $T \in \mathcal{T}$ and $x' \in Q$. Therefore a proof must have been simulated during signing for a statement $x' = (cpk_1, bpk^*, ipk, c\mu', c')$ such that $c\mu$ is a re-randomisation of $c\mu'$ and c is a re-randomisation of c'. Therefore $c\mu'$ must be an encryption of upk_{uid^*} and c' must be an encryption of m, and so (uid^*, m, bpk) was queried to the SIGN oracle. This is a contradiction given that $\mathcal{A}$ is successful.

In order for $\mathcal{A}'$ to win, (m, Ω) must be a valid under apk, which is true as the extraction was successful. The signing oracle $\mathsf{SIGN}_{\mathsf{auto}}$ is not used, and therefore $\mathcal{A}'$ wins with probability ϵ/q.

Therefore $\mathcal{A}'$ succesfully breaks the EUF-CMA security of the automorphic signature scheme with probability ϵ/q. Therefore, assuming the EUF-CMA security of the automorphic signature scheme, the $\mathsf{CLS\text{–}CM}$ construction satisfies non-frameability.

```
$\mathsf{SNDU}(uid, M_{\mathsf{in}})$
if $uid \in \mathsf{CUL}$   return $\perp$
if $uid \notin \mathsf{HUL}$
    $\mathsf{HUL} \leftarrow \mathsf{HUL} \cup \{uid\}, Q \leftarrow Q + 1, \mathbf{gsk}[uid] \leftarrow \perp, \mathbf{dec}^{uid} \leftarrow \mathsf{cont}$
    if $Q = k$   $uid^* \leftarrow uid, upk_{uid} \leftarrow apk$   return $(upk_{uid}, \mathsf{cont})$
if $uid = uid^*$
    Continue from line 4 of oracle in non-frameability experiment but set $upk' = \perp$
Continue from line 4 of oracle in non-frameability experiment
```

```
$\mathsf{SIGN}(uid, m, bpk)$
if $uid = uid^*$
    if $\mathbf{dec}^{uid} \neq \mathsf{accept}$   return $\perp$
    $\alpha \leftarrow\!\$\, \mathbb{Z}_p^*, \mu \leftarrow (\hat{g}^{\alpha}, 1, upk_{uid}cpk_1^{\alpha}), c \leftarrow (1, m)$
    $\sigma \leftarrow S_{2,\mathsf{cm\text{-}NIZK}}(\sigma_{\mathsf{crs}}, \tau_s, (cpk_1, bpk, ipk, \mu, c))$
    $\mathsf{SL} \leftarrow \mathsf{SL} \cup \{(uid, m, bpk)\}$
    return $(\mu, \sigma)$
else  Identical to non-frameability experiment
```

```
$\mathcal{A}'^{\mathsf{SIGN}_{\mathsf{auto}}}((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{pp}_{\mathsf{auto}}, apk)$
$Q \leftarrow 0, k \leftarrow\!\$\, [q], \mathsf{pp}_{\mathsf{auto}1} \leftarrow \mathsf{pp}_{\mathsf{auto}}, \mathsf{pp}_{\mathsf{auto}2} \leftarrow \mathsf{ASetup}_2(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)$
$g \leftarrow\!\$\, \mathbb{G}_1, \hat{g} \leftarrow\!\$\, \mathbb{G}_2, (\sigma_{\mathsf{crs}}, \tau_s, \tau_e) \leftarrow \mathsf{SE}_{1,\mathsf{cm\text{-}NIZK}}(1^{\tau}), \mathsf{pp}_{\mathsf{sig}} \leftarrow \mathsf{SIG.Setup}(1^{\tau})$
$\mathsf{pp} \leftarrow ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{pp}_{\mathsf{auto}1}, \mathsf{pp}_{\mathsf{auto}2}, g, \hat{g}, \sigma_{\mathsf{crs}}, \mathsf{pp}_{\mathsf{sig}})$
$(ipk, isk) \leftarrow \mathsf{CLS{+}.IKGen}(\mathsf{pp}), (cpk, csk) \leftarrow \mathsf{CLS{+}.CKGen}(\mathsf{pp})$
$gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$(uid, (c, c\mu, c\sigma), bpk, bsk) \leftarrow \mathcal{A}^{\mathsf{SNDU},\mathsf{SIGN}}(gpk, isk, csk)$
if $(bpk, bsk) \notin \mathcal{BK}$   return 0
$((\cdot, upk, \cdot, \Omega, \cdot, \cdot, \cdot, m), \cdot, \cdot) \leftarrow E_{2,\mathsf{cm\text{-}NIZK}}(\sigma_{\mathsf{crs}}, \tau_e, (cpk_1, bpk, ipk, c\mu, c), c\sigma)$
if $upk = apk$   return $(m, \Omega)$   else return $\perp$
```

Fig. 19. $\mathcal{A}'$ which breaks the EUF-CMA security of the automorphic signature scheme using $\mathcal{A}$ which breaks the non-frameability of the CLS–CM construction.

D.8 Traceability

Lemma 6. *The CLS–CM construction satisfies* ***traceability*** *if the automorphic signatures are EUF-CMA secure, and the cm-NIZK is zero knowledge, and cm-SSE.*

We build an adversary $\mathcal{A}'$, that breaks the EUF-CMA security of the automorphic signature scheme with non-negligible probability, given $\mathcal{A}$ that wins the traceability game for the CLS–CM construction with non-negligible probability ϵ. We give $\mathcal{A}'$ in Figure 20, and below explain why the simulation input to $\mathcal{A}$ given in Figure 20 is identically distributed to the traceability experiment for the CLS–CM construction, and that $\mathcal{A}'$ successfully breaks the EUF-CMA security of the automorphic signature scheme.

```
$\mathsf{ADDU}(uid)$
Identical to traceability experiment except $cred \leftarrow \mathsf{SIGN}_{\mathsf{auto}}(upk)$

$\mathsf{SNDI}(uid, M_{\mathsf{in}})$
Identical to traceability experiment except $cred \leftarrow \mathsf{SIGN}_{\mathsf{auto}}(upk)$

$\mathsf{SIGN}(uid, m, bpk)$
Identical to in traceability experiment

${\mathcal{A}'}^{\mathsf{SIGN}_{\mathsf{auto}}}((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{pp}_{\mathsf{auto}}, apk)$
$\mathsf{pp}_{\mathsf{auto}1} \leftarrow \mathsf{ASetup}_1(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2)), \mathsf{pp}_{\mathsf{auto}2} \leftarrow \mathsf{pp}_{\mathsf{auto}}$
$g \leftarrow\$ \mathbb{G}_1, \hat{g} \leftarrow\$ \mathbb{G}_2, (\sigma_{\mathsf{crs}}, \tau_s, \tau_e) \leftarrow \mathsf{SE}_{1,\mathsf{cm\text{-}NIZK}}(1^\tau), \mathsf{pp}_{\mathsf{sig}} \leftarrow \mathsf{SIG.Setup}(1^\tau)$
$\mathsf{pp} \leftarrow ((p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, g_2), \mathsf{pp}_{\mathsf{auto}1}, \mathsf{pp}_{\mathsf{auto}2}, g, \hat{g}, \sigma_{\mathsf{crs}}, \mathsf{pp}_{\mathsf{sig}})$
$ipk \leftarrow apk, (cpk, csk) \leftarrow \mathsf{CLS+.CKGen}(\mathsf{pp})$
$gpk \leftarrow (\mathsf{pp}, ipk, cpk)$
$((c_1, c\mu_1, c\sigma_1), ..., (c_k, c\mu_k, c\sigma_k), bpk, bsk) \leftarrow \mathcal{A}^{\mathsf{ADDU,SNDI,SIGN}}(gpk, csk)$
if $(bpk, bsk) \notin \mathcal{BK}$   return $\perp$
if $\exists i \in [1, k]$ s.t $\forall uid \in \mathsf{CUL} \cup \mathsf{HUL}$   $c\mu_{i,3} c\mu_{i,1}^{-csk_1} c\mu_{i,2}^{-bsk_2} \neq upk_{uid}$
  $((\cdot, upk, cred, \cdot, \cdot, \cdot, \cdot, \cdot), \cdot, \cdot) \leftarrow E_{2,\mathsf{cm\text{-}NIZK}}(\sigma_{\mathsf{crs}}, \tau_e, (cpk_1, bpk, ipk, c\mu_i, c_i), c\sigma_i)$
  return $(upk, cred)$
else return $\perp$
```

Fig. 20. $\mathcal{A}'$ which breaks the EUF-CMA security of the automorphic signature scheme using $\mathcal{A}$ which breaks the traceability of the CLS–CM construction.

First note that all inputs that $\mathcal{A}'$ provides to $\mathcal{A}$ are distributed identically to in the traceability experiment. This is because $\mathsf{SE}_{1,\mathsf{cm\text{-}NIZK}}$ outputs a σ_{crs} that is identical to in CRSSetup, and ipk and $\mathsf{pp}_{\mathsf{auto}2}$ are distributed identically.

Simulating the oracles: The ADDU and SNDI oracles are only different to in the traceability experiment, when $cred$ is generated. As the signing oracle in the EUF-CMA experiment outputs a valid signature under isk, this is distributed identically.

The SIGN oracle is identical to in the traceability experiment.

Reduction to EUF-CMA security of automorphic signatures. Assume $\mathcal{A}$ is successful. We first show that $\mathcal{A}'$ can find $i \in [1, k]$ s.t $\forall uid \in \mathsf{CUL} \cup \mathsf{HUL}$ $c\mu_{i,3} c\mu_{i,1}^{-csk_1} c\mu_{i,2}^{-bsk_2} \neq upk_{uid}$.

Let $L' = |\{i \in [k] : c\mu_{i,3} c\mu_{i,1}^{-csk_1} c\mu_{i,2}^{-bsk_2} = upk_{uid} \text{ and } uid \in \mathsf{HUL}\}|$

Let $C' = |\{i \in [k] : c\mu_{i,3} c\mu_{i,1}^{-csk_1} c\mu_{i,2}^{-bsk_2} = upk_{uid} \text{ and } uid \in \mathsf{CUL}\}|$.

Let $\{(\overline{c\mu}_i, \overline{c\sigma}_i, \overline{c}_i)\}_k \leftarrow \mathsf{CLS+.Convert}(gpk, bpk, csk, \{(c\mu_i, c\sigma_i, c_i)\}_k)$ and for all $i \in [1, k]$ let $(\overline{\mu}_i, \overline{\sigma}_i, m_i) = \mathsf{CLS+.Unblind}(bsk, (\overline{c\mu_i}, \overline{c\sigma_i}, \overline{c}_i))$.

If $\mathcal{A}'$ aborts then $k \leqslant C' + L'$. However if $\mathcal{A}$ is successful, then $k > \mathsf{CUL} + L$, where $L = |\{uid \in \mathsf{HUL} : \exists i \text{ s.t } (uid, m_i, bpk) \in \mathsf{SL}\}|$.

Therefore, either $C' > \mathsf{CUL}$ or $L' > L$.

If $C' > \mathsf{CUL}$, two signatures both decrypt to the same corrupted user. Therefore for some $(i, j) \in [k]^2$, $c\mu_{i,3} c\mu_{i,1}^{-csk_1} c\mu_{i,2}^{-bsk_2} = c\mu_{j,3} c\mu_{j,1}^{-csk_1} c\mu_{j,2}^{-bsk_2}$.

However as the two signatures are unlinked we have for $\beta'_1, \beta'_2 \leftarrow\$ \mathbb{Z}_p^*$,

$$e(g_1, c\mu_{i,3} bpk_2^{\beta'_1} c\mu_{i,1}^{-csk_1})^r e(g_1, c\mu_{i,2} \hat{g}^{\beta'_1})^{-rbsk_2}$$

$$\neq e(g_1, c\mu_{j,3} bpk_2^{\beta'_2} c\mu_{j,1}^{-csk_1})^r e(g_1, c\mu_{j,2} \hat{g}^{\beta'_2})^{-rbsk_2},$$

Therefore, $e(g_1, c\mu_{i,3} c\mu_{i,2}^{-bsk_2} c\mu_{i,1}^{-csk_1})^r \neq e(g_1, c\mu_{j,3} c\mu_{j,2}^{-bsk_2} c\mu_{j,1}^{-csk_1})^r$, which is a contradiction.

If $L' > L$, let $L'' = |\{uid \in \mathsf{HUL} : \exists i \in [k] \text{ s.t } c\mu_{i,3} c\mu_{i,1}^{-csk_1} c\mu_{i,2}^{-bsk_2} = upk_{uid}\}|$, as no two signatures will decrypt to the same honest user, due to the above argument, $L' = L''$, and so $L'' > L$.

Therefore there exists an honest user $uid*$ and $i \in [k]$ such that $c\mu_{i,3} c\mu_{i,1}^{-csk_1} c\mu_{i,2}^{-bsk_2} = upk_{uid*}$, but $(uid, m_i, bpk) \notin \mathsf{SL}$. Clearly then $(c\mu_i, c\sigma_i, c_i)$ would identify to $uid*$, because $c\sigma_i$ is a valid $\mathsf{cm\text{-}NIZK}$ (otherwise $\mathsf{CLS+.Convert}$ would fail) and decrypts to upk_{uid*} under csk_1 and bsk_2. This is not possible as we have already proven our $\mathsf{CLS\text{–}CM}$ construction satisfies non-frameability. Therefore, $\mathcal{A}'$ aborts with negligible probability.

All signatures output by $\mathcal{A}$ are valid $\mathsf{cm\text{-}NIZK}$s because they do not cause $\mathsf{CLS+.Convert}$ to fail. We have not used $S_{1,\mathsf{cm\text{-}NIZK}}$ to obtain simulations of the $\mathsf{cm\text{-}NIZK}$ proofs used, therefore $E_{2,\mathsf{cm\text{-}NIZK}}$ will extract a valid witness, with $1 - \mathsf{negl}(n)$ probability. This ensures that $\mathsf{AVerify}_2(cred, ipk, upk) = 1$. As upk is not the public key of any corrupted or honest users, upk was not queried to $\mathsf{SIGN}_{\mathsf{auto}}$ by ADDU or SNDI. Therefore $cred$ is a valid forgery under apk, and $\mathcal{A}'$ is successful.

E Instantiation of the $\mathsf{cm\text{-}NIZK}$

It is shown in Theorem 4.5 in [16] that $\mathsf{cm\text{-}NIZKS}$ for $(\mathcal{R}, \mathcal{T})$ can be instantiated assuming DLIN holds if $(\mathcal{R}, \mathcal{T})$ are CM-friendly which they define fully in Section C.1.

More formally we define the relation R such that $((cpk_1, bpk_1, bpk_2, ipk, c\mu, c),$ $(upk', upk, cred, \Omega, g_1^{\alpha}, g_1^{\beta}, g_2^{\gamma}, m)) \in R$ if and only if $e(g_1, \mu_1) = e(g_1^{\alpha}, \hat{g})$, $e(g_1, \mu_2) = e(g_1^{\beta}, \hat{g})$ and $e(g_1, \mu_3) = e(g_1, upk)e(g_1^{\alpha}, cpk_1)e(g_1^{\beta}, bpk_2)$; , $\mathsf{AVerify}_1(\Omega, upk, m) = 1$, $\mathsf{AVerify}_2(cred, ipk, upk) = 1$ and
$e(c_1, g_2) = e(g, g_2^{\gamma})$, $e(c_2, g_2) = e(m, g_2)e(bpk_1, g_2^{\gamma})$, and $e(upk', g_2) = e(g_1, upk)$.

We define the allowable set of transformation $\mathcal{T} = \{(r_{\mathsf{enc1}}, r_{\mathsf{enc2}}, r_{\mathsf{enc3}}) : r_{\mathsf{enc1}}, r_{\mathsf{enc2}}, r_{\mathsf{enc3}} \in \mathbb{Z}_p^*\}$, such that for $T = (r_{\mathsf{enc1}}, r_{\mathsf{enc2}}, r_{\mathsf{enc3}})$, the transformation $T_{\mathsf{inst}}(cpk_1, bpk, ipk, \mu, c) = (cpk_1, bpk, ipk, (\mu_1\hat{g}^{r_{\mathsf{enc1}}}, \mu_2\hat{g}^{r_{\mathsf{enc2}}}, \mu_3 cpk_1^{r_{\mathsf{enc1}}} bpk_2^{r_{\mathsf{enc2}}}),$ $(c_1 g^{r_{\mathsf{enc3}}}, c_2 bpk_1^{r_{\mathsf{enc3}}}))$, $T_{\mathsf{wit}}(upk', upk, cred, \Omega, g_1^{\alpha}, g_1^{\beta}, g_2^{\gamma}, m) = (upk', upk, cred, \Omega,$ $g_1^{\alpha} g_1^{r_{\mathsf{enc1}}}, g_1^{\beta} g_1^{r_{\mathsf{enc2}}}, g_2^{\gamma} g_2^{r_{\mathsf{enc3}}}, m)$.

We now show that $(\mathcal{R}, \mathcal{T})$ is CM-Friendly which means 6 conditions are satisfied.

1. *Representable statements: any instance and witness of $\mathcal{R}$ can be represented as a set of group elements*
   Verification keys, messages and signatures of autormorphic signatures are all group elements and so $ipk, upk, cred, \Omega, m$ are all group elements. $upk' = g_1^{usk}$ and so is a group element.
   $cpk_1, bpk_1, bpk_2, c\mu = (c\mu_1, c\mu_2, c\mu_3), c = (c_1, c_2)$ can all clearly be represented by group elements due to the elgamal encryption used. Represent encryption randomness α, β, γ with $(g_1^{\alpha}, g_1^{\beta}, g_2^{\gamma})$.
2. *Representable transformations: any transformation in T can be represented as a set of group elements*
   Represent randomness $r_{\mathsf{enc1}}, r_{\mathsf{enc2}}, r_{\mathsf{enc3}}$ with $(g_1^{r_{\mathsf{enc1}}}, \hat{g}^{r_{\mathsf{enc1}}}, cpk_1^{r_{\mathsf{enc1}}}, g_1^{r_{\mathsf{enc2}}}, \hat{g}^{r_{\mathsf{enc2}}},$ $bpk_2^{r_{\mathsf{enc2}}}, g_2^{r_{\mathsf{enc3}}},$
   $g^{r_{\mathsf{enc3}}}, bpk_1^{r_{\mathsf{enc3}}})$.
3. *Provable statements: we can prove the statement $(x, w) \in \mathcal{R}$ (using the above representation for x and w) using pairing product equations.*
   $\mathsf{AVerify}_1(\Omega, upk, m) = 1$, and $\mathsf{AVerify}_2(cred, ipk, upk) = 1$ can be written as a conjuction of pairing product equations over $upk, m, \Omega, cred, ipk$ due to the properties of automorphic signatures.
   All other equations are already in the form of pairing product equations, using the representation given above.
4. *Provable transformations: we can prove the statement "$T_{\mathsf{inst}}(x) = x'$ for $T \in \mathcal{T}$" (using the above representations for x and T) using a pairing product equation.*
   Given $x = (cpk_1, bpk_1, bpk_2, ipk, (c\mu_1, c\mu_2, c\mu_3), (c_1, c_2))$, $x' = (cpk_1', bpk_1',$ $bpk_2', ipk',$ $(c\mu_1', c\mu_2', c\mu_3'), (c_1', c_2'))$, $T = (R_1, R_1', R_1'', R_2, R_2', R_2'', R_3, R_3', R_3'')$, then $T_{\mathsf{inst}}(x) = x'$ and $T \in \mathcal{T}$ iff:
   $e(g_1, cpk_1) = e(g_1, cpk_1')$,$e(bpk_1, g_2) = e(bpk_1', g_2)$, $e(g_1, bpk_2) = e(g_1, bpk_2')$,
   $e(ipk, g_2) = e(ipk', g_2)$ and

$e(c_2', g_2) = e(bpk_1, R_3)e(c_2, g_2)$, $e(c_1', g_2) = e(g, R_3)e(c_1, g_2)$, and
$e(g_1, c\mu_3') = e(R_1, cpk_1)e(R_2, bpk)e(g_1, c\mu_3)$, $e(g_1, c\mu_1') = e(R_1, \hat{g})e(g_1, c\mu_1)$,
$e(g_1, c\mu_2') = e(R_2, \hat{g})e(g_1, c\mu_2)$.

5. *Transformable statements: for any $T \in \mathcal{T}$, there is a valid transformation $s(T)$ that takes the statement "$(x, w) \in \mathcal{R}$" (phrased using pairing products as above) and produces the statement "$(T_{\mathsf{inst}}(x), T_{\mathsf{wit}}(w)) \in \mathcal{R}$.*
We transform the pairing product equations for an instance $(cpk_1, bpk_1, bpk_2, ipk, (c\mu_1, c\mu_2, c\mu_3), (c_1, c_2))$ into mauled equations for an instance $(cpk_1, bpk_1, bpk_2, ipk, (c\mu_1\hat{g}^{\alpha'}, c\mu_2\hat{g}^{\beta'}, c\mu_3 cpk_1^{\alpha'} bpk_2^{\beta'}), (c_1 g^{\gamma'}, c_2 bpk_1^{\gamma'}))$ as follows:
First of all we re-randomise $c\mu$.
   - $\mathsf{Add}(eq_1 := e(g_1, \hat{g}^{\alpha'})^{-1}e(g_1^{\alpha'}, \hat{g}) = 1)$
     $\mathsf{Add}(eq_2 := e(g_1, \hat{g}^{\beta'})^{-1}e(g_1^{\beta'}, \hat{g}) = 1)$
     $\mathsf{Add}(eq_3 : e(g_1, cpk_1^{\alpha'} bpk_2^{\beta'})^{-1}e(g_1^{\alpha'}, cpk_1)e(g_1^{\beta'}, bpk_2) = 1)$
   - $\mathsf{MergeEq}(eq_1,\ e(g_1, c\mu_1)^{-1}e(g_1^{\alpha}, \hat{g}) = 1\ )$ to create equation eq_4.
     $\mathsf{MergeEq}(eq_2,\ e(g_1, c\mu_2)^{-1}e(g_1^{\beta}, \hat{g}) = 1)$ to create equation eq_5.
     $\mathsf{MergeEq}(eq_3,\ e(g_1, c\mu_3)^{-1}e(g_1^{\alpha}, cpk_1)e(g_1^{\beta}, bpk_2)e(g_1, upk) = 1\ )$ to create equation eq_6.
   - $\mathsf{MergeVar}(c\mu_1, \hat{g}^{\alpha'}, \hat{c\mu_1}, \{g_1\})$, will create equation
     $e(g_1, c\mu_1\hat{g}^{\alpha'})e(g_1, \hat{c\mu_1})^{-1} = 1$.
     $\mathsf{MergeEq}(e(g_1, c\mu_1\hat{g}^{\alpha'})e(g_1, \hat{c\mu_1})^{-1} = 1, \mathsf{eq}_4)$ to create equation eq_7.
     $\mathsf{MergeVar}(c\mu_2, \hat{g}^{\beta'}, \hat{c\mu_2}, \{g_1\})$, will create equation
     $e(g_1, c\mu_2\hat{g}^{\beta'},)e(g_1, \hat{c\mu_2})^{-1} = 1$.
     $\mathsf{MergeEq}(e(g_1, c\mu_2\hat{g}^{\beta'},)e(g_1, \hat{c\mu_2})^{-1} = 1, \mathsf{eq}_5\ )$ to create equation eq_8.
     $\mathsf{MergeVar}(c\mu_3, cpk_1^{\alpha'} bpk_2^{\beta'}, \hat{c\mu_3}, \{g_1\})$, will create equation
     $e(g_1, c\mu_3 cpk_1^{\alpha'} bpk_2^{\beta'})e(g_1, \hat{c\mu_3})^{-1} = 1$.
     $\mathsf{MergeEq}(e(g_1, c\mu_3 cpk_1^{\alpha'} bpk_2^{\beta'})e(g_1, \hat{c\mu_3})^{-1} = 1, \mathsf{eq}_6)$ to create equation eq_9.
   - $\mathsf{MergeVar}(g_1^{\alpha'}, g_1^{\alpha}, g_1^{\tilde{\alpha}}, \{\hat{g}, cpk_1\})$, will create equations
     $e(g_1^{\alpha'} g_1^{\alpha}, \hat{g})^{-1}e(g_1^{\tilde{\alpha}}, \hat{g}) = 1$ and
     $e(g_1^{\alpha'} g_1^{\alpha}, cpk_1)^{-1}e(g_1^{\tilde{\alpha}}, cpk_1) = 1$.
     $\mathsf{MergeEq}(e(g_1^{\alpha'} g_1^{\alpha}, \hat{g})^{-1}e(g_1^{\tilde{\alpha}}, \hat{g}) = 1, \mathsf{eq}_7\ )$.
     $\mathsf{MergeVar}(g_1^{\beta'}, g_1^{\beta}, g_1^{\tilde{\beta}}, \{\hat{g}, bpk_2\})$, will create equations
     $e(g_1^{\beta'} g_1^{\beta}, \hat{g})^{-1}e(g_1^{\tilde{\beta}}, \hat{g}) = 1$ and
     $e(g_1^{\beta'} g_1^{\beta}, bpk_2)^{-1}e(g_1^{\tilde{\beta}}, bpk_2) = 1$.
     $\mathsf{MergeEq}(e(g_1^{\beta'} g_1^{\beta}, \hat{g})^{-1}e(g_1^{\tilde{\beta}}, \hat{g}) = 1, \mathsf{eq}_8\ )$.
     $\mathsf{MergeEq}(e(g_1^{\alpha'} g_1^{\alpha}, cpk_1)^{-1}e(g_1^{\tilde{\alpha}}, cpk_1) = 1, e(g_1^{\beta'} g_1^{\beta}, bpk_2)^{-1}e(g_1^{\tilde{\beta}}, bpk_2) = 1)$ to create equation eq_{10} .
     $\mathsf{MergeEq}(\mathsf{eq}_9,\ \mathsf{eq}_{10}\ )$.
     Remove obsolete equations and variables with $\mathsf{RemoveEq}$ and $\mathsf{RemoveVar}$.

We now re-randomise c.
   - $\mathsf{Add}(eq_{11} := e(g^{\gamma'}, g_2)^{-1}e(g, g_2^{\gamma'}) = 1)$
     $\mathsf{Add}(eq_{12} := e(bpk_1^{\gamma'}, g_2)^{-1}e(bpk_1, g_2^{\gamma'}) = 1)$

– $\mathsf{MergeEq}(eq_{11}, e(c_1, g_2)^{-1} e(g, g_2^{\gamma}) = 1\ )$ to create equation eq_{13}.
  $\mathsf{MergeEq}(eq_{12},\ e(c_2, g_2)^{-1} e(bpk_1, g_2^{\gamma}) e(m, g_2) = 1\ )$ to create equation eq_{14}.
– $\mathsf{MergeVar}(c_1, g^{\gamma'}, \hat{c_1}, \{g_2\})$, will create equation
  $e(c_1 g^{\gamma'}, g_2) e(\hat{c_1}, g_2)^{-1} = 1$.
  $\mathsf{MergeEq}(e(c_1 g^{\gamma'}, g_2) e(\hat{c_1}, g_2)^{-1} = 1, \mathsf{eq}_{13}\ )$ to create eq_{15}.
  $\mathsf{MergeVar}(c_2, bpk_1^{\gamma'}, \hat{c_2}, \{g_2\})$, will create equation
  $e(c_2 bpk_1^{\gamma'}, g_2) e(\hat{c_2}, g_2)^{-1} = 1$.
  $\mathsf{MergeEq}(e(c_2 bpk_1^{\gamma'}, g_2) e(\hat{c_2}, g_2)^{-1} = 1, \mathsf{eq}_{14})$ to create eq_{16}.
– $\mathsf{MergeVar}(g_2^{\gamma}, g_2^{\gamma'}, g_2^{\tilde{\gamma}}, \{bpk_1, g\})$, will create equations
  $e(g, g_2^{\gamma} g_2^{\gamma'})^{-1} e(g, g_2^{\tilde{\gamma}}) = 1$ and
  $e(bpk_1, g_2^{\gamma} g_2^{\gamma'})^{-1} e(bpk_1, g_2^{\tilde{\gamma}}) = 1$.
  $\mathsf{MergeEq}(e(g, g_2^{\gamma} g_2^{\gamma'})^{-1} e(g, g_2^{\tilde{\gamma}}) = 1, \mathsf{eq}_{15}\ )$.
  $\mathsf{MergeEq}(e(bpk_1, g_2^{\gamma} g_2^{\gamma'})^{-1} e(bpk_1, g_2^{\tilde{\gamma}}) = 1, \mathsf{eq}_{16}\ )$.

Finally remove obsolete equations and variables with RemoveEq and RemoveVar.
(6) *Transformable transformations: for any $T, T' \in \mathcal{T}$ there is a valid transformation $t(T)$ that takes the statement "$T_{inst}(x') = x$ for $T \in \mathcal{T}$" (phrased using pairing products as above) and produces the statement "$T'_{inst} \circ T_{inst}(x') = T'_{inst}(x)$ for $T' \circ T \in \mathcal{T}$" and that preserves the variables in x' (does not perform RemoveVar on variables in x').*
We transform the pairing product equations defined in (4) for proving knowledge of a transformation from instance $(cpk_1, bpk, ipk, c\mu', c')$ to
$(cpk_1, bpk, ipk, c\mu, c)$, into mauled equations for proving knowledge of a transformation from instance $(cpk_1, bpk, ipk, c\mu', c')$to
$(cpk_1, bpk, ipk, (c\mu_1 \hat{g}^{\alpha}, c\mu_2 \hat{g}^{\beta}, c\mu_3 bpk_2^{\beta} cpk_1^{\alpha}), (c_1 g^{\gamma}, c_2 bpk_1^{\gamma}))$ we can use the same strategy (and the same constant equations) as for transforming statements.

– $\mathsf{MergeVar}(c_1, g^{\gamma}, \hat{c_1}, \{g_2\})$, will create equation $e(c_1 g^{\gamma}, g_2) e(\hat{c_1}, g_2)^{-1} = 1$.
  $\mathsf{MergeEq}(e(c_1, g_2)^{-1} e(g, R_3) e(c_1', g_2),\ e(c_1 g^{\gamma}, g_2) e(\hat{c_1}, g_2)^{-1} = 1)$.
  $\mathsf{MergeVar}(R_3, g_2^{\gamma}, \hat{R_3}, \{g, bpk_1\})$, will create equations
  $e(g, R_3 g_2^{\gamma})^{-1} e(g, \hat{R_3}) = 1$ and
  $e(bpk_1, R_3 g_2^{\gamma})^{-1} e(bpk_1, \hat{R_3}) = 1$.
  $\mathsf{MergeEq}(e(\hat{c_1}, g_2)^{-1} e(g, R_3) e(g, g_2^{\gamma}) e(c_1', g_2),\ e(g, R_3 g_2^{\gamma})^{-1} e(g, \hat{R_3}) = 1)$.
  $\mathsf{MergeVar}(c_2, bpk_1^{\gamma}, \hat{c_2}, \{g_2\})$, will create equation
  $e(c_2 bpk_1^{\gamma}, g_2) e(\hat{c_2}, g_2)^{-1} = 1$.
  $\mathsf{MergeEq}(e(c_2, g_2)^{-1} e(bpk_1, R_3) e(c_2', g_2) = 1,\ e(c_2 bpk_1^{\gamma}, g_2) e(\hat{c_2}, g_2)^{-1} = 1)$.
  $\mathsf{MergeEq}(e(\hat{c_2}, g_2)^{-1} e(bpk_1, R_3 g_2^{\gamma}) e(c_2', g_2) = 1,\ e(bpk_1, R_3 g_2^{\gamma})^{-1}$
  $e(bpk_1, \hat{R_3}) = 1\ )$.
– $\mathsf{MergeVar}(c\mu_1, \hat{g}^{\alpha}, \hat{c\mu_1}, \{g_1\})$, will create equation
  $e(g_1, c\mu_1 \hat{g}^{\alpha}) e(g_1, \hat{c\mu_1})^{-1} = 1$.
  $\mathsf{MergeEq}(e(g_1, c\mu_1)^{-1} e(R_1, \hat{g}) e(g_1, c\mu_1') = 1,\ e(g_1, c\mu_1 \hat{g}^{\alpha}) e(g_1, \hat{c\mu_1})^{-1} = 1)$.

MergeVar$(R_1, g_1^\alpha, \hat{R}_1, \{\hat{g}, cpk_1\})$, will create equations
$e(R_1 g_1^\alpha, \hat{g})^{-1} e(\hat{R}_1, \hat{g}) = 1$ and
$e(R_1 g_1^\alpha, cpk_1)^{-1} e(\hat{R}_1, cpk_1) = 1$.
MergeEq$(e(g_1, c\hat{\mu}_1)^{-1} e(R_1, \hat{g}) e(g_1^\alpha, \hat{g}) e(g_1, c\mu_1') = 1,\ e(R_1 g_1^\alpha, \hat{g})^{-1}$
$e(\hat{R}_1, \hat{g}) = 1)$.
MergeVar$(c\mu_2, \hat{g}^\beta, c\hat{\mu}_2, \{g_1\})$, will create equation
$e(g_1, c\mu_2 \hat{g}^\beta) e(g_1, c\hat{\mu}_2)^{-1} = 1$.
MergeEq$(e(g_1, c\mu_2)^{-1} e(R_2, \hat{g}) e(g_1, c\mu_2') = 1,\ e(g_1, c\mu_2 \hat{g}^\beta) e(g_1, c\hat{\mu}_2)^{-1} = 1)$.
MergeVar$(R_2, g_1^\beta, \hat{R}_2, \{\hat{g}, bpk_2\})$, will create equations
$e(R_2 g_1^\beta, \hat{g})^{-1} e(\hat{R}_2, \hat{g}) = 1$ and
$e(R_2 g_1^\beta, bpk_2)^{-1} e(\hat{R}_2, bpk_2) = 1$.
MergeEq$(e(g_1, c\hat{\mu}_2)^{-1} e(R_2, \hat{g}) e(g_1^\beta, \hat{g}) e(g_1, c\mu_2') = 1,$
$e(R_2 g_1^\beta, \hat{g})^{-1} e(\hat{R}_2, \hat{g}) = 1)$.
MergeVar$(c\mu_3, cpk_1^\alpha bpk_2^\beta, c\hat{\mu}_3, \{g_1\})$, will create equation
$e(g_1, c\mu_3 cpk_1^\alpha bpk_2^\beta) e(g_1, c\hat{\mu}_3)^{-1} = 1$.
MergeEq$(e(g_1, c\mu_3)^{-1} e(R_1, cpk_1) e(R_2, bpk_2) e(g_1, c\mu_3') = 1,$
$e(g_1, c\mu_3 cpk_1^\alpha bpk_2^\beta) e(g_1, c\hat{\mu}_3)^{-1} = 1)$.
MergeEq$(e(g_1, c\hat{\mu}_3)^{-1} e(R_1, cpk_1) e(g_1^\alpha, cpk_1) e(R_2, bpk_2) e(g_1^\beta, bpk_2)$
$e(g_1, c\mu_3') = 1,\ e(R_1 g_1^\alpha, cpk_1)^{-1} e(\hat{R}_1, cpk_1) = 1\ )$.
MergeEq$(e(g_1, c\hat{\mu}_3)^{-1} e(\hat{R}_1, cpk_1) e(R_2, bpk_2) e(g_1^\beta, bpk_2) e(g_1, c\mu_3') = 1,$
$e(R_2 g_1^\beta, bpk_2)^{-1} e(\hat{R}_2, bpk_2) = 1\ )$.

Finally remove obsolete equations and variables with RemoveEq and RemoveVar, $(c\mu', c')$ will be unaffected by RemoveVar.