Extending Updatable Encryption: Public Key, Tighter Security and Signed Ciphertexts

Chen Qian [1], Yao Jiang Galteland [2], and Gareth T. Davies [3]

[1] Shandong University
chen.qian@sdu.edu.cn
[2] Qredo
yao.jiang@qredo.com
[3] Bergische Universität Wuppertal
davies@uni-wuppertal.de

Abstract. Updatable encryption is a useful primitive that enables key rotation for storing data on an untrusted storage provider without the leaking anything about the plaintext or the key. In this work, we make two contributions. Firstly, we extend updatable encryption to the public-key setting, providing its security model and three different efficient constructions. Using a public-key updatable encryption scheme, a user can receive messages directly in the cloud from multiple senders without revealing their secret key. Secondly, we add signatures on ciphertexts to guarantee plaintext integrity and authenticity. We call our new primitive *Public-Key Signable Updatable Encryption* (PSigUE). Our approach ensures that only legitimate ciphertexts are accepted by the server, and the adversary cannot compromise the message integrity in the database. We bypass the conflict between public integrity verification and the malleability that comes from the update functionality.
We provide three pairing-based constructions of public-key signable updatable encryption. The first scheme, PSigUE_1, is built using a dual-mode zero-knowledge proof of knowledge system under an assumption closely related to the k-linear assumption. The second scheme, PSigUE_2, provides unlinkability in addition to public authenticity. In the third scheme, $\mathsf{PSigUE}_\mathsf{T}$, we achieve the tight security with respect of number of epochs. The construction of $\mathsf{PSigUE}_\mathsf{T}$ is inspired by tag-based tightly-secure PKE schemes.

Keywords. Updatable Encryption; Outsourced Storage; Pairing-based Cryptography.

1 Introduction

Recent years have seen two consistent trends for individuals and businesses: the storage of more and more data with third-party providers and growing awareness of the importance of mitigations in the event of key exposures. Forward security and protection from corruption of past key material are the default expectation for key exchange protocols and the protocols underpinning instant messaging services.

Reflecting these concerns, academic work has recently given considerable focus to the topic of symmetric-key *updatable encryption* (SKUE): a primitive that allows a data owner to encrypt their plaintext material, send it to an untrusted storage provider (e.g. a cloud-based storage business), and perform key rotation by sending one or more tokens to the storage provider. The update operation performed by the storage provider rotates the encryption key(s). It thus advances to the next 'epoch' in a way that leaks no information about the plaintext other than what can be trivially observed by looking at the ciphertext. This primitive is particularly attractive for two reasons. First up is the obvious bandwidth saving compared to downloading, decrypting, re-encrypting, and re-uploading. Secondly, the computational burden is passed on from the user—who may be operating from a relatively limited device such as a smartphone—to the server-grade storage provider.

UE is by now a mature primitive, and since the early treatments [6,22] there have been advances in security properties and efficiency [20,7,18,8,5,28,25,24,14] in both the ciphertext-independent setting (where the data owner encrypts all ciphertexts with the same key and sends a single token to update) and the ciphertext-dependent setting (individual keys per ciphertext). While schemes in the ciphertext-dependent setting are generally more efficient, the necessary key management often limits their practicality to specific use cases. In this work, we focus on the simpler, ciphertext-independent version.

In this work, we extend UE in two directions. Firstly, we treat *public-key updatable encryption* (PKUE), where multiple parties may encrypt plaintexts to a single user. This natural extension enables us to capture many application scenarios that were not possible before. With this framework in place, we investigate the benefits of adding signatures to UE ciphertexts, focusing on the PKUE setting. Our new primitive is called Public-Key Signable Updatable Encryption (PSigUE).

We note that PKUE is fundamentally different from SKUE. In PKUE, any user possessing Alice's public key can create ciphertexts that Alice could decrypt. However, PKUE is inherently vulnerable to a denial-of-service attack where encrypting parties fill up Alice's storage area, so we will always need some way only to allow certain encryptors to insert ciphertexts into the database. Hence, we cannot directly follow the definition and security notions of SKUE to define PKUE. Another issue is whether or not the database is public because this affects what it actually means for an adversary to make corruption queries to the system. This work will discuss the challenges in providing formal PKUE models. In Section 1.4, we provide scenarios where our primitive may be helpful.

Security properties for UE schemes focus on confidentiality, integrity and unlinkability. Confidentiality makes sure that freshly encrypted messages are indistinguishable, so the indistinguishability focuses on evaluating the encryption algorithm. Integrity guarantees no (new) ciphertexts can be forged by any adversary, which prevents modification. Unlinkability captures the fact that it may be desirable for an adversary that is given a ciphertext in the current epoch, to know which ciphertext it was updated from in the previous epoch. However,

unlinkability may not be necessary in some use cases. In PSigUE, we focus on protecting the private content and are happy to reveal the public content because this gives more usability to the data owner and does not require unlinkability is achieved for the verifier. Moreover, if PSigUE is re-randomizable via updating to the current epoch with a trivial token, then the statistical unlinkability is unachievable without additional mechanisms. More precisely, the adversary can firstly re-randomize the challenge ciphertext, then update it to another epoch e', and corrupt the secret key $\mathsf{sk}_{\mathsf{e}'}$. The challenger cannot refuse such corruption queries due to statistical unlinkability.

Closely Related Literature. In Section 1.3 we wil discuss related primitives in more detail, but for now we mention two ideas that are closely related to ours yet represent subtly different approaches.

Updatable signatures on messages have been investigated in the work of Cini et al. [9], but their focus was on updating the keypair for signing/verification for a fixed message, while our aim is to provide a *signature* that remains valid *on a UE ciphertext* as its encryption key is rotated.

In very recent work, Knapp and Quaglia [21] presented a short paper investigating the public-key updatable encryption setting. Their extended abstract in the proceedings version contains very few details regarding their security model and constructions, except that they use a UE flavor of replayable CCA security where the adversary is not allowed to query its update oracle(s) on ciphertexts that decrypt to the challenge messages (this restriction is necessary for schemes where perfect re-randomization of ciphertexts is possible). As discussed earlier, statistical unlinkability is incompatible with UE in the public key setting. Therefore, we choose to forego unlinkability in some of our schemes to provide what we believe to be more realistic adversarial powers.

1.1 Technical overview

There are two main challenges in our construction. First, we notice that achieving security in the public-key setting is non-trivial. Moreover, there is a contradiction between the signature on ciphertexts and updatability of PSigUE.

Security Challenges and Two Solutions. We follow a strong adaptive corruption model for security where the adversary can update arbitrary ciphertexts in the database (the game must rule out any trivial wins). To prove the confidentiality of an updatable encryption scheme, it is necessary to simulate the update queries for both challenge and non-challenge ciphertexts in the security game to any adversary. However, such a simulation will be challenging without knowledge of the update token and message-related information. The typical proof strategies for symmetric setting in the work of [22,20] use fresh encryptions of the underlying message to simulate updated ciphertexts, or as in [7] where some secret information about the message and randomness is used to simulate updated non-challenge ciphertexts. Therefore, it seems complicated to update any non-challenge ciphertexts; the

challenger should be able to get some secret information about the non-challenge ciphertext (message or randomness) in the public-key setting.

In [21], they bypassed this problem by not allowing the adversary to encrypt any challenge message in the security model. With the help of this additional restriction, they can use the decryption oracle of RCCA security to get the plaintext of any non-challenge ciphertext. However, we argue that this restriction is artificial and does not adequately reflect the PKUE problem setting: the storage provider will almost certainly have to retain some metadata for each ciphertext to enable ciphertext retrieval, meaning that unlinkability is only achievable from an entity that does not have persistent view access to the database. This may be desirable in specific scenarios, but it is generally difficult to imagine why this would be necessary.

Instead of limiting the adversary, we solve this problem by modifying the mechanism of the updatable encryption scheme, that is, adding a registration phase for each ciphertext. The registration phase is used to verify the validity of a ciphertext before inserting it into the database. This method is important for both practice and our security proofs, as it prevents malicious data injection by attackers. To insert a ciphertext into the database, we introduce two new approaches to verify the ciphertext. Looking forward, the signer will check these proofs, and if they pass, the signer will add a signature and insert the ciphertext on behalf of the encrypting party.

The first verification method is using a zero-knowledge proof of knowledge (ZKPoK) of some secret information about the ciphertext (message or randomness), only ciphertexts with valid ZKPoK can be added into the database. Moreover, to simplify the protocol every ZKPoK is never stored and updated in the database (this invoking an assumption of deletion of the proof and associated values by the signer). The additional ZKPoK helps us to prove the security in the following two ways:

- The challenger can use the knowledge extractor to get secret information about the non-challenge ciphertexts. Moreover, since no proof is ever stored in the database, the adversary cannot use database corruption to access these proofs and learn any additional information.
- When we reduce the security to the underlying PKE scheme, the returned challenge ciphertext is without any ZKPoK. However, since every ZKPoK in our model is discarded right after the registration phase, the challenger does not need to simulate the ZKPoK of the challenge ciphertext.

Our second solution to this problem leads to our tightly-secure construction (Section 5). Compared to our first solution, it is more elegant but less generic. We observe that the problem of exponential security losses in proofs already exists in multi-user public key encryption schemes. Therefore, we borrowed some ideas from tightly-secure public-key encryption. In our construction in Section 5, every ciphertext is associated with a tag, and this tag remains the same when we update the ciphertext. Using this tag-based technique, we can construct our scheme in a way that only the ciphertexts with a special tag $\mathsf{t}^\star$ information theoretically hide all information about the message. Moreover, this tag $\mathsf{t}^\star$ is hidden for the

adversary even with secret key corruption queries. The proof strategy forces only the challenge ciphertext to have this special tag $\mathsf{t}^\star$.

Signatures on Updatable Ciphertexts. Public verifiability of PSigUE is inherently a difficult task. Since the signature are mostly used to guarantee the message integrity, it contradicts the updatability of PSigUE. Therefore, we propose two different approaches. Our first solution is used for the schemes that we call PSigUE_1 and $\mathsf{PSigUE}_\mathsf{T}$. The intuition is that the ciphertext is of the form $\mathbf{c} = (\mathbf{c}_{\mathsf{Fix}}, \mathbf{c}_{\mathsf{Aux}})$. All information of the message m is included in $\mathbf{c}_{\mathsf{Fix}}$. Moreover, $\mathbf{c}_{\mathsf{Fix}}$ is not changing while updating, and $\mathbf{c}_{\mathsf{Aux}}$ uniquely defined by $\mathbf{c}_{\mathsf{Fix}}$ and pk_e. The advantage of this approach is that we can simply sign $\mathbf{c}_{\mathsf{Fix}}$ to guarantee the integrity using any signature scheme. However, since $\mathbf{c}_{\mathsf{Fix}}$ is not changing while updating. We cannot achieve any meaningful unlinkability for this approach.

Our second solution for integrity is used in PSigUE_2. We follow a similar strategy as in [3], first used in blind signature schemes. This approach can achieve unlinkability since all ciphertexts and signatures are re-randomizable. However, we cannot combine this blind signature technique with our tightly secure one. We leave the construction of efficient, unlinkable, and tightly secure PSigUE as one of the major open problems for future work.

1.2 Contributions

We identify two main contributions in this work. The first contribution is the public-key security model with signability of updatable ciphertexts. We believe that public verifiability can protect databases' integrity while allowing key rotation.

Our second contribution is three different constructions that achieve different properties, and the results are summarized in Table 1. PSigUE_1 is our most efficient construction, only 4 group elements is sufficient to encrypt 1 group message. However, it does not have unlinkability, we can only prove that it is $\mathsf{wIND\text{-}ENC}$ secure (a weaker version of confidentiality where the adversary is limited in where it can make corruption queries), and we need to add an additional ZKPoK while trusting the database to not store this proof. Our second construction PSigUE_2 improves PSigUE_1 by adding unlinkability. However, the main drawback is the efficiency, it can only encrypt short messages, and the ciphertext is very large compared to PSigUE_1. Our final construction $\mathsf{PSigUE}_\mathsf{T}$ improves PSigUE_1 and PSigUE_2 in multiple dimensions. $\mathsf{PSigUE}_\mathsf{T}$ is tightly secure in the number of epochs and does not need to add ZKPoK. We also believe that our construction $\mathsf{PSigUE}_\mathsf{T}$ is optimal because it is impossible to have perfect unlinkability, rerandomizability, and tight security while not needing additional ZKPoK.

1.3 Related Primitives

We now describe a number of primitives which are similar to the problem that we tackle in one or more aspects.

Schemes	IND-ENC	Integrity	Unlinkability	without ZKPoK	Tight	Efficiency
PSigUE_1	$\checkmark^{\mathsf{w}}$	✓	✗	✗	✗	$2\mathbb{G}_1 + 1\mathbb{G}_2 + 1\mathbb{G}_\mathsf{T}$
PSigUE_2	$\checkmark^{\mathsf{w}}$	✓	✓	✗	✗	$(6\ell+6)\mathbb{G}_1 + (6\ell+2)\mathbb{G}_2$
$\mathsf{PSigUE}_\mathsf{T}$	✓	✓	✗	✓	✓	$9\mathbb{G}_1 + 2\mathbb{G}_2$

Table 1. Contributions of our work. $\checkmark^{\mathsf{w}}$ indicates the schemes achieves the weaker security model wIND-ENC.

Proxy re-encryption. Proxy re-encryption (PRE) allows the holder of pk_i to derive a re-encryption key $rk_{i,j}$ that allows an untrusted proxy to convert a ciphertext encrypted under pk_i to one that is decryptable under some pk_j. PRE schemes have no ordering of encryption keys, so public-key updatable encryption can be seen as a special case of (mutli-hop) PRE, where the restriction that the owner of a public key can only generate a re-encryption key between the current epoch and the next: this is the what we call the token.

Comparison of our work with existing security models for PRE is challenging because the vast majority of the PRE literature considers selective security, where the adversary indicates which parties it wishes to corrupt at the beginning of the security experiment. Most papers in the (symmetric-key) UE literature focus on adaptive security to provide fine-grained results regarding forward security and post-compromise security, and we follow this approach in our work. Fuchsbauer et al. [13] gave the first treatment of adaptive security in this context and produced a surprising result: it is possible to avoid an exponential security loss due to guessing corrupted parties by using pebbling techniques, however this leads to security losses that depend on the adversary's queries and the graphs that their re-encryption queries invoke. We will discuss tightness of our reductions in more detail in Section 5. Davidson et al. [11] provided the first analysis of post-compromise security for PRE, which is analogous to the indistinguishability of updates that is normally expected for symmetric-key UE.

Updatable Signatures (and MACs) Cini et al. [9] gave constructions of signatures and MACs where the signing/MAC key owner wishes to periodically rotate their secret key values to provide forward security. The focus was on plaintext data that remained static, whereas we wish to sign a ciphertext that is having its key rotated in regular updatable encryption. The signing keys in our approach do not rotate, however it may be interesting future work to combine the two approaches to get updatable signatures on updatable ciphertexts.

Updatable Public Key Encryption. In Updatable PKE [19,1,12], the regular syntax of epoch-based (*i.e.* forward-secure) public key encryption is extended to allow any sender to produce a so-called *update ciphertext* which is associated with the public key of the next epoch, allowing the receiver to compute the corresponding secret key. This approach enables forward security even before a receiver has chosen to update their key pair, and is particularly suited to group messaging. In

UPKE there is no mechanism for old ciphertexts to be 'updated' to encryptions under the new key, so forward security is not obtained for the messages sent under the 'previous' key of a party.

Signcryption. Signcryption is a combination of public-key encryption and digital signatures in a single primitive, at a cost lower than performing the two operations separately. A single epoch version of PSigUE reduces to signcryption if the encryptor and signer are the same entity, however in the applications that we target in this paper we do not expect that this is the case. Furthermore, adapting signcryption schemes to the updatable setting appears to be a very challenging problem: among other issues, many schemes involve a hash of message and key material, which destroys the structure that is essentially required to perform UE operations.

1.4 Applications of Our Schemes

File Transfer from Whistleblowers. Consider a scenario where a whistleblower works at (or has privileged access to) an organisation or company and wants to inform a journalist about some grave misdeed. The whistleblower is in possession of an image, video or document that contains inciminating evidence and wishes to expose this file to the press, but is concerned for their own safety. The current solution to this problem is to use a tool such as SecureDrop[4] which is used[5] by The Guardian, The New York Times and other major news outlets: after navigating a Tor circuit, the file is encrypted using PKE. Further details are not so important, but the decryption key of this PKE scheme is stored in an air-gapped machine which makes key rotation (and authentication of public keys related to this decryption key) a challenging task. Given the sensitivity of the documents that are transmitted using such a service, key rotation appears to be of paramount importance so an public-key updatable encryption is already an attractive proposition. But the newspaper doesn't want just anyone to be able to insert files, so their journalists hold a signing key that verifies the usefulness of the whistleblower's file. In this application, the signer (journalist) is only[6] verifying that what is contained in the whistleblower's file is useful to the newspaper and if the whistleblower is in fact associated with the organization that the files refer to (source authentication). Our solution enables key rotation to protect past data and in addition the newspaper (database owner) can be sure of the legitimacy of data before inserting it into the database, without knowing the underlying secret message beforehand.

[4] `https://securedrop.org/overview/encrypted-and-air-gapped/`

[5] `https://www.theguardian.com/securedrop` and `https://www.nytimes.com/tips#securedrop`

[6] Depending on the application and requirements for the signature, the signer may be responsible for the content, in that case, the signer should check the underlying message before signing ciphertexts.

Updatability from anyone. We modify the classic token generation setting by taking the old public key as input and outputting a new public key and an update token. This design approach enables automated updates that do not require interaction with the secret key holder. The epoch advancements could be triggered by time periods (e.g. end of the calendar month), or activated in the event of a security incident elsewhere in the data owner's ICT ecosystem.

Storage only for authenticated data. A cloud server can be sure if a file was uploaded by some party that has access to the storage area, and not a network adversary. The aforementioned 'attack' that our work prevents is arguably unlikely: an adversary fills up a party's cloud storage area with garbage ciphertexts to make the client pay more money or lose access to their legitimate files. The major benefit however is that our approach prevents the cloud from storing garbage data. This application makes sure all data on the cloud are authenticated, it is beneficial to both the cloud and the client.

Public Verifiability for Proof-of-Retrievability. A data storage center must prove to a verifier that it is actually storing all of a client's data. With signatures on ciphertexts, anyone can verify if the cloud server stored the client's data. This enables a service that any third party can verify the proof-of-retrievability without any storage or computational effort by the clients.

Public Verifiability for Counterfeit Media. A seller (for example, a media-services provider) is an intermediate party who can buy products from producers and sell to customers. Updatable encryption allows the seller to alter the data access key to a fresh key in a situation where a customer's product viewing period expires. Counterfeiting is a serious issue and sellers wish to protect their intellectual property. The common way to prove if a product is not a counterfeit is that the customer uses the access key to get the data and then checks the data with its credential. However, such verification is done after the customer bought the product and got the access key, that is, the verification is on plaintext. Our solution enables verification on ciphertext, therefore, any potential customer can check the legitimacy of product before paying for the product.

2 Preliminary

2.1 Notation

Let pairing groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T})$ are cyclic groups of order p, g_1 and g_2 are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_\mathsf{T}$ is a bilinear map, for simplicity we also denote the pairing operator by $\bullet$ and $\mathsf{g}_\mathsf{T} = \mathsf{g}_1 \bullet \mathsf{g}_2$ is a generator in group $\mathbb{G}_\mathsf{T}$.

For $\mathsf{s} \in \{1, 2, \mathsf{T}\}$ and $a \in \mathbb{Z}_\mathsf{p}$, define $\left[a\right]_\mathsf{s} = a \cdot \mathsf{g}_\mathsf{s} \in \mathbb{G}_\mathsf{s}$. For a matrix $\mathbf{A} = (a_{i,j}) \in \mathbb{Z}_\mathsf{p}^{n \times m}$, we define $\left[\mathbf{A}\right]_\mathsf{s} = \begin{pmatrix} a_{1,1} \cdot \mathsf{g}_\mathsf{s} & \dots & a_{1,m} \cdot \mathsf{g}_\mathsf{s} \\ \vdots & & \vdots \\ a_{n,1} \cdot \mathsf{g}_\mathsf{s} & \dots & a_{n,m} \cdot \mathsf{g}_\mathsf{s} \end{pmatrix}$.

Scalar exponentiation. Given $[a]_{\mathsf{s}}$ and a scalar $x \in \mathbb{Z}_{\mathsf{p}}$, we can compute $[ax]_{\mathsf{s}}$ by computing $[a]_{\mathsf{s}} \cdot x$. In particular, for matrix of $\mathbb{Z}_{\mathsf{p}}$ elements $\mathbf{A}, \mathbf{B}$, we can compute $[\mathbf{AB}]_{\mathsf{s}}$ by the knowledge of $[\mathbf{A}]_s, \mathbf{B}$, we denote $[\mathbf{AB}]_{\mathsf{s}} = [\mathbf{A}]_{\mathsf{s}} \cdot \mathbf{B}$.

Pairing computation. Given $[a]_1, [b]_2$, we can compute $[ab]_{\mathsf{T}}$ by computing $[a]_1 \bullet [b]_2$). In particular, for matrix $\mathbf{A}, \mathbf{B}$, we can compute $[\mathbf{AB}]_{\mathsf{T}}$ by $[\mathbf{A}]_1 \bullet [\mathbf{B}]_2$, we denote $[\mathbf{A}]_1 \bullet [\mathbf{B}]_2 = [\mathbf{AB}]_{\mathsf{T}}$.

Algorithms. We use code-based games [2] to present our definitions and proofs. We implicitly assume all Boolean flags to be initialized to 0 (**false**), numerical variables to 0, sets to $\emptyset$ and strings to $\perp$. We make the

convention that a procedure terminates once it has returned an output. Additionally, we introduce the operator **Check** $<$ *condition* $>$ which means: if $<$ *condition* $>$ is true then go to the next step, else return $\perp$.

Symbols. We use . to present "such that" in the first order logic. For example, for all x such that $x \in \mathcal{X}$ is denoted as $\forall x.x \in \mathcal{X}$.

2.2 Public-Key Signable Updatable Encryption

We define a new type of updatable encryption in the public-key setting and with signed ciphertexts. We denote this new primitive as PSigUE, a public-key encryption scheme with updated functionality to provide forward and post-compromise security. Additionally, PSigUE supports signature on ciphertexts to guarantee the integrity of the encryption.

Recall that we have two different approaches as described in the technical overview (Section 1.1) to solve the exponential security loss in the public-key setting. Therefore, we define the PSigUE in two different settings (with and without proof of knowledge (ZKPoK)) as follows.

Definition 1 (Public-Key Signable Updatable Encryption). *A public-key signable updatable encryption scheme consists of four PPT algorithms* $\mathsf{PSigUE} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *together with update functionalities* $(\mathsf{TokGen}, \mathsf{Usk}, \mathsf{Upd})$ *with the following syntax:*

- $\mathsf{Setup}(\mathsf{parG})$ *takes group parameters* parG *as input, and outputs public parameters* pp.
- $\mathsf{KGen}(\mathsf{pp})$ *takes public parameters* pp *as input, and outputs a public encryption key pair* (pk_0, sk_0) *of the epoch* 0.
- $\mathsf{Enc}(\mathsf{pk}_\mathsf{e}, \mathsf{m})$ *takes a public key* pk_e *of the epoch* e *and a message* m *as input, and outputs a ciphertext* $\mathbf{c}_\mathsf{e}$ *of the epoch* e.
- $\mathsf{Dec}(\mathsf{sk}_\mathsf{e}, \mathbf{c}_\mathsf{e})$ *takes a secret key* sk_e *and a ciphertext* $\mathbf{c}_\mathsf{e}$ *as input, and outputs a message* m.
- $\mathsf{TokGen}(\mathsf{pk}_\mathsf{e})$ *takes a public key* pk_e *of the epoch* e *as input, and outputs an update token* $\Delta_{\mathsf{e}+1}$ *and an updated public key* $\mathsf{pk}_{\mathsf{e}+1}$ *of the epoch* $\mathsf{e}+1$.
- $\mathsf{Usk}(\Delta_{\mathsf{e}+1}, \mathsf{sk}_\mathsf{e})$ *takes an update token* $\Delta_{\mathsf{e}+1}$ *and a secret key* sk_e *as input, and outputs an updated secret key* $\mathsf{sk}_{\mathsf{e}+1}$ *of the epoch* $\mathsf{e}+1$.

– $\mathsf{Upd}(\Delta_{\mathsf{e}+1}, \mathbf{c}_{\mathsf{e}}, \sigma_{\mathsf{e}})$ *takes an update token* $\Delta_{\mathsf{e}+1}$*, a ciphertext* $\mathbf{c}_{\mathsf{e}}$ *and a signature* σ_{e} *as input, and outputs an updated ciphertext* $\mathbf{c}_{\mathsf{e}+1}$ *and an updated signature* $\sigma_{\mathsf{e}+1}$*.*

SIGNED CIPHERTEXTS:[7] *Additionally, we provide public authentication with the following three PPT algorithms* $(\mathsf{SKGen}, \mathsf{Sig}, \mathsf{Ver})$*:*

– $\mathsf{SKGen}(\mathsf{pp})$ *takes public parameters* pp *as input, and outputs a singing key* ssk *and a verification key* svk*.*
– $\mathsf{Sig}(\mathsf{ssk}, \mathsf{pk}_{\mathsf{e}}, \mathbf{c}_{\mathsf{e}})$ *takes as input a signing key* ssk*, a public key* pk_{e} *and a ciphertext* $\mathbf{c}_{\mathsf{e}}$ *as input, and outputs a signature* σ_{e}*.*
– $\mathsf{Ver}(\mathsf{svk}, \mathsf{pk}_{\mathsf{e}}, \mathbf{c}_{\mathsf{e}}, \sigma_{\mathsf{e}})$ *takes a verification key, a public key* pk_{e}*, a ciphertext* $\mathbf{c}_{\mathsf{e}}$ *and a signature* σ_{e} *as input, and outputs* 0 *or* 1*.*

PSigUE WITHOUT PROOF OF KNOWLEDGE (ZKPoK): *In the no* ZKPoK *setting, we require an additional public verification algorithm for the ciphertexts. Informally, we need to check that every ciphertext is well-formed before adding it to the database.*

– $\mathsf{PVer}(\mathsf{pk}_{\mathsf{e}}, \mathbf{c})$ *takes a public key* pk_{e} *of epoch* e*, a ciphertext* $\mathbf{c}$ *as input, and outputs* 0 *or* 1*.*

PSigUE WITH PROOF OF KNOWLEDGE (ZKPoK): *For* PSigUE *with* ZKPoK*, we require a zero-knowledge proof of knowledge component. Depending on the property of the* PSigUE*, this zero-knowledge proof could be a proof of knowledge of the secret information* $\mathsf{x} \in \{\mathsf{m}, \mathsf{r}\}$ *(message or randomness). We recall that the zero-knowledge proof is used only in the registration phase and never stored in the database.*

– $\mathsf{Prove}_{\mathsf{x}}(\mathsf{pk}_{\mathsf{e}}, \mathbf{c}, \mathsf{x})$ *takes a public key* pk_{e} *of epoch* e*, a ciphertext* $\mathbf{c}$*, and a secret information* x *as input, and outputs a zero-knowledge proof of knowledge* π*.*
– $\mathsf{Ver}_{\mathsf{x}}(\mathsf{pk}_{\mathsf{e}}, \mathbf{c}, \pi)$ *takes the public key* pk_{e} *of the epoch* e*, a ciphertext* $\mathbf{c}$*, and a proof* π*, and outputs* 0 *or* 1*. In particular,* $\mathsf{x} \in \{\mathsf{m}, \mathsf{r}\}$*.*

In the remaining definition section, we denote the additional part for PSigUE without ZKPoK in dashed boxes, and the additional part for PSigUE with ZKPoK in gray boxes.

Firstly, we define the correctness for PSigUE in the expected way.

[7] Note that our syntax does not require that the signatures change as the ciphertexts are updated when the Upd operation is performed (i.e. we allow $\sigma_{\mathsf{e}+1} = \sigma_{\mathsf{e}}$), and we will use such static signatures in PSigUE_1 and $\mathsf{PSigUE}_{\mathsf{T}}$. Note also that the signer will be signing ciphertexts rather than plaintexts. This means that a signature is valid on an encryption of some plaintext for multiple epochs, and consequently a signature is valid on a set of ciphertexts (in a similar manner to linearly homomorphic signatures).

Definition 2 (Correctness). PSigUE *is correct if for any security parameter* λ*, any message* m*, and any polynomially large integers* $\mathsf{e}_1 \leq \mathsf{e}_2$*, we have:*

$$\Pr\left[\begin{array}{l|l} \mathsf{Dec}(\mathsf{sk}_{\mathsf{e}_2}, \mathbf{c}_{\mathsf{e}_2}) = \mathsf{m} & \mathsf{parG} \xleftarrow{\$} \mathsf{GGen}(1^\lambda);\ \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(\mathsf{parG}) \\ \wedge \mathsf{Ver}(\mathsf{svk}, \mathbf{c}_{\mathsf{e}_2}, \sigma_{\mathsf{e}_2}) = 1 & (\mathsf{pk}_0, \mathsf{sk}_0) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp}) \\ \boxed{\wedge \mathsf{PVer}(\mathsf{pk}_{\mathsf{e}_1}, \mathbf{c}) = 1} & (\mathsf{svk}, \mathsf{ssk}) \xleftarrow{\$} \mathsf{SKGen}(\mathsf{pp}) \\ \wedge \mathsf{Ver}_{\mathsf{x}}(\mathsf{pk}_{\mathsf{e}_1}, \mathbf{c}, \pi) = 1 & \textbf{for } j = 0 \textbf{ to } \mathsf{e}_1 - 1 \\ & \quad (\mathsf{pk}_{j+1}, \Delta_{j+1}) \xleftarrow{\$} \mathsf{TokGen}(\mathsf{pk}_j) \\ & \quad \mathsf{sk}_{j+1} \xleftarrow{\$} \mathsf{Usk}(\Delta_{j+1}, \mathsf{sk}_j) \\ & \mathbf{c}_{\mathsf{e}_1} \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}_{\mathsf{e}_1}, \mathsf{m});\ \sigma_{\mathsf{e}_1} \xleftarrow{\$} \mathsf{Sig}(\mathsf{ssk}, \mathbf{c}_{\mathsf{e}_1}) \\ & \pi \xleftarrow{\$} \mathsf{Prove}_{\mathsf{x}}(\mathsf{pk}_{\mathsf{e}_1}, \mathbf{c}, \mathsf{x}) \\ & \textbf{for } j = \mathsf{e}_1 \textbf{ to } \mathsf{e}_2 - 1 \\ & \quad (\mathsf{pk}_{j+1}, \Delta_{j+1}) \xleftarrow{\$} \mathsf{TokGen}(\mathsf{pk}_j) \\ & \quad \mathsf{sk}_{j+1} \xleftarrow{\$} \mathsf{Usk}(\Delta_{j+1}, \mathsf{sk}_j) \\ & \quad (\mathbf{c}_{j+1}, \sigma_{j+1}) \xleftarrow{\$} \mathsf{Upd}(\Delta_{j+1}, \mathbf{c}_j, \sigma_j) \end{array}\right] = 1.$$

2.3 Integrity for PSigUE Scheme

In this section, we define the integrity game for PSigUE schemes. This notion captures an adversary that attempts to provide a ciphertext-signature pair that correspond to a plaintext for which it has not asked to its signing oracle and for which the ciphertext decrypts correctly (even when given the secret decryption key in all epochs). Intuitively this means that a scheme will not be deemed to provide integrity protection if signatures are re-randomizable, nor if it is easy to find collisions in ciphertexts in the underlying encryption scheme. As we have mentioned already, signatures are static and are not rotated with the ciphertexts.

Definition 3 (Integrity). *For any PPT adversary* $\mathcal{A}$ *and any security parameter* λ*, we define the integrity experiment* $\mathbf{Exp}^{\mathsf{Int}}_{\mathsf{PSigUE},\mathcal{A}}(1^\lambda)$ *and oracles* $\mathcal{O}_{\mathsf{Int}} = (\mathcal{O}_{\mathsf{Sig}}, \mathcal{O}_{\mathsf{TokGen}})$ *as in Figure 1.*

We say that PSigUE *satisfies the integrity if there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that,*

$$\Pr\left[\mathbf{Exp}^{\mathsf{Int}}_{\mathsf{PSigUE},\mathcal{A}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\cdot).$$

2.4 IND-ENC and wIND-ENC Security for PSigUE Scheme

In this section, we define the new confidentiality games for the PSigUE scheme. The adversary can ask for moving to the new epoch via $\mathcal{O}_{\mathsf{Next}}$, signing ciphertext via $\mathcal{O}_{\mathsf{Sig}}$, inserting data via $\mathcal{O}_{\mathsf{Insert}}$, corrupting secret key and update token via $\mathcal{O}_{\mathsf{CorrK}}$ and $\mathcal{O}_{\mathsf{CorrT}}$, and updating data via $\mathcal{O}_{\mathsf{Upd}}$. At some moment, the adversary requests for a challenge query $\mathcal{O}_{\mathsf{Chall}_\mathsf{b}}$ with a challenge message, the challenger responses either an encryption of the challenge message or an encryption of a random value. The adversary can continue asking for the above oracles again. In the end, it outputs a guess bit. The adversary wins the game if it guesses

$\mathbf{Exp}^{\mathsf{Int}}_{\mathsf{PSigUE},\mathcal{A}}(1^\lambda)$:

01 $\mathsf{parG} \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$
02 $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(\mathsf{parG})$
03 $(\mathsf{pk}_0, \mathsf{sk}_0) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp})$
04 $(\mathsf{svk}, \mathsf{ssk}) \xleftarrow{\$} \mathsf{SKGen}(\mathsf{pp})$
05 $(\mathbf{c}^*, \sigma^*) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{Int}}}(\mathsf{pp}, \mathsf{pk}_0, \mathsf{sk}_0, \mathsf{svk})$
06 $\mathsf{m}' \leftarrow \mathsf{Dec}(\mathsf{sk}_{\mathsf{e}^*}, \mathbf{c}^*)$
07 **if** $\mathsf{m}' \neq \perp$ **and** $\mathsf{Ver}(\mathsf{svk}, \mathbf{c}^*, \sigma^*) = 1$ **and** $(\mathsf{m}', \cdot, \cdot) \notin \mathcal{L}_\sigma$
08 **return** 1
09 **else**
10 **return** 0

Oracle $\mathcal{O}_{\mathsf{Sig}}(\mathbf{c}_\mathsf{e})$:

11 $\sigma_\mathsf{e} \xleftarrow{\$} \mathsf{Sig}(\mathsf{ssk}, \mathbf{c}_\mathsf{e})$
12 $\mathsf{m} := \mathsf{Dec}(\mathsf{sk}_e, \mathbf{c}_\mathsf{e})$
13 $\mathcal{L}_\sigma := \mathcal{L}_\sigma \cup \{(\mathsf{m}, \mathbf{c}_\mathsf{e}, \sigma)\}$
14 **return** σ_e

Oracle $\mathcal{O}_{\mathsf{TokGen}()}$:

15 $\mathsf{e}^\star := \mathsf{e}$
16 $(\mathsf{pk}_{\mathsf{e}^\star+1}, \Delta_{\mathsf{e}^\star+1}) \xleftarrow{\$} \mathsf{TokGen}(\mathsf{pk}_{\mathsf{e}^\star})$
17 $\mathsf{sk}_{\mathsf{e}^\star+1} := \mathsf{Usk}(\mathsf{sk}_{\mathsf{e}^\star}, \Delta_{\mathsf{e}^\star+1})$
18 $\mathsf{e} := \mathsf{e} + 1$
19 **return** $(\mathsf{pk}_{\mathsf{e}^\star+1}, \Delta_{\mathsf{e}^\star+1}, \mathsf{sk}_{\mathsf{e}^\star+1})$

Fig. 1. We give the security experiment $\mathbf{Exp}^{\mathsf{Int}}_{\mathsf{PSigUE},\mathcal{A}}$, with the definition of the oracles $\mathcal{O}_{\mathsf{Int}} = (\mathcal{O}_{\mathsf{Sig}}, \mathcal{O}_{\mathsf{TokGen}})$. The list $\mathcal{L}_\sigma$ keeps track on all signatures generated by the signing oracle.

correctly. In particular, the oracle $\mathcal{O}_{\mathsf{Insert}}$ is introduced in this work, it is used to check ciphertexts before storage. We emphasize that for both PSigUE with and without ZKPoK the adversary can only insert ciphertexts with valid proofs into the database. Moreover, for PSigUE with ZKPoK once a ciphertext is inserted into the database, the corresponding proof will be completely deleted from the database. Therefore, we do not need to update this proof, and the adversary can not get access to this proof via corruption queries.

Definition 4 (IND-ENC). *Let* PSigUE *be a public-key signable updatable encryption. For all PPT adversary, we define the security experiment* $\mathbf{Exp}^{\mathsf{IND\text{-}ENC}_\mathsf{b}}_{\mathsf{PSigUE},\mathcal{A}}$ *as in Figure 2.*

We say that PSigUE *satisfies the* IND-ENC *security if, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that,*

$$\left| \Pr\left[\mathbf{Exp}^{\mathsf{IND\text{-}ENC}}_{\mathsf{PSigUE},\mathcal{A}}(1^\lambda) = 1 \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

It is non-trivial to construct a public-key updatable encryption with IND-ENC. One common approach to prove confidentiality for updatable encryption is reducing the UE security to the security of the underlying encryption scheme. However, such approach would lead to an exponential security loss in constructing updatable encryption in the public-key setting. This is because adversary can obtain all the public keys before any corruption or challenge queries. As a challenger in the security proof, we need to already decide which public key should include a trapdoor at this point. If we did not guess the challenge epochs correctly, the adversary can simply require the key corruption oracles to distinguish the simulated security game from the real one. We emphasize that this problem is specifically only for public key setting. Since in the symmetric setting, no public keys are required.

We propose two different solutions for the above problem. The first one is to construct our PSigUE in a weaker security model (Section 3, Section 4). The second solution borrows idea from tightly-secure public-key constructions (Section 5).

Our first solution to this problem is by weakening the security model. We propose a new security model wIND-ENC which is a weaker version of IND-ENC. More precisely, in the wIND-ENC security model, the adversary is only allowed to learn challenge ciphertexts in one consecutive sequence of epochs. In this model, as a challenger we only need to guess the starting and the ending epochs of such epoch periods. Therefore, the security loss is only n^2 for n epochs.

We argue that the weaker security notion wIND-ENC is already meaningful in the real life applications. For a publicly accessible database, the adversary could copy the entire database and have access to all ciphertexts at any moment. In this setting, if the database stops maintaining part of the database, for example, data deletion. Then the database will not be able to provide future version of such part of database. Therefore, there is only one consecutive sequence of epochs that the adversary knows all (updated) version of ciphertexts, including challenge ciphertexts.

We give the formal definition of wIND-ENC as follows.

Definition 5 (wIND-ENC)**.** *Let* PSigUE *be a public-key signable updatable encryption. For all PPT adversary, we define the security experiment* $\mathbf{Exp}^{\mathsf{wIND\text{-}ENC}_\mathsf{b}}_{\mathsf{PSigUE},\mathcal{A}}$ *as in Figure 2.*

We say that PSigUE *satisfies the* wIND-ENC *security if, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that,*

$$\left|\Pr\left[\mathbf{Exp}^{\mathsf{wIND\text{-}ENC}}_{\mathsf{PSigUE},\mathcal{A}}(1^\lambda)=1\right]-\frac{1}{2}\right|\leq \mathsf{negl}(\lambda).$$

2.5 Unlinkability

We also define unlinkability as follows. The unlinkability measures if the adversary can distinguish an updated real ciphertext from an updated random ciphertext.

Definition 6 (Unlink)**.** *Let* PSigUE *be a public-key signable updatable encryption. For all PPT adversaries* $\mathcal{A}$*, we define the security experiment* $\mathbf{Exp}^{\mathsf{Unlink}_\mathsf{b}}_{\mathsf{PSigUE},\mathcal{A}}$ *as in Figure 4.*

We say that PSigUE *satisfies the* Unlink *security if, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that,*

$$\left|\Pr\left[\mathbf{Exp}^{\mathsf{Unlink}}_{\mathsf{PSigUE},\mathcal{A}}(1^\lambda)=1\right]-\frac{1}{2}\right|\leq \mathsf{negl}(\lambda).$$

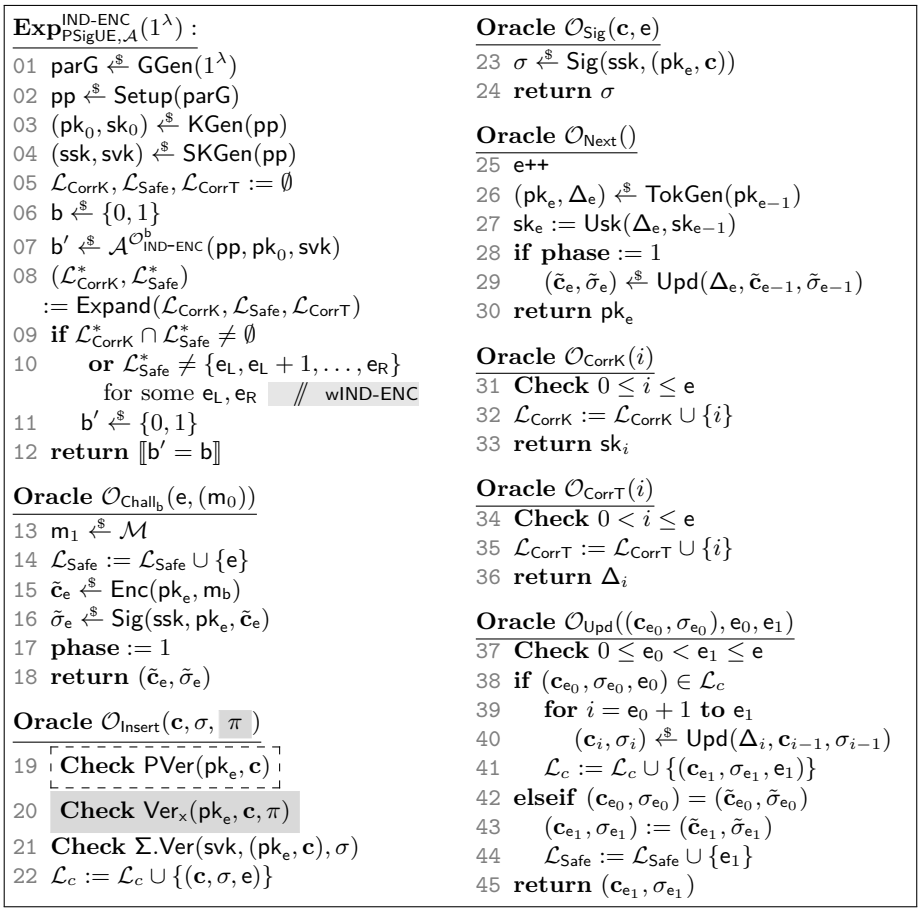

Fig. 2. Security experiment of IND-ENC and wIND-ENC security game of the PSigUE scheme. The codes ending with // wIND-ENC are only for wIND-ENC security game. Expand algorithm (Line 08) will be used to check trivial win condition and it is described in Fig. 3. For both IND-ENC and wIND-ENC security, the adversary $\mathcal{A}$ has access to the oracles $\mathcal{O}^{\mathsf{b}}_{\mathsf{IND\text{-}ENC}} = (\mathcal{O}_{\mathsf{Chall_b}}, \mathcal{O}_{\mathsf{Sig}}, \mathcal{O}_{\mathsf{Insert}}, \mathcal{O}_{\mathsf{Next}}, \mathcal{O}_{\mathsf{CorrK}}, \mathcal{O}_{\mathsf{CorrT}}, \mathcal{O}_{\mathsf{Upd}})$. The challenge ciphertext and its updated ciphertexts are denoted as $(\tilde{\mathbf{c}}_i, \tilde{\sigma}_i)$.

2.6 Dual-Mode Non-Interactive Zero-Knowledge Proof of Knowledge

We recall the Dual-Mode Non-Interactive Zero-Knowledge Proof of Knowledge (DM-ZKPoK) [15] in this section, this will be a essential tool in our construction. The DM-ZKPoK is a special type of zero-knowledge proof system in which the common reference string (crs) generation is dual-mode. More precisely, the dual-mode property means that the setup algorithm Setup can generate crs in soundness or zero-knowledge modes. In the soundness mode, we have perfect soundness, and we have perfect zero-knowledge in the zero-knowledge mode.

Alg $\mathsf{Expand}(\mathcal{L}_{\mathsf{CorrK}}, \mathcal{L}_{\mathsf{Safe}}, \mathcal{L}_{\mathsf{CorrT}})$:

$\mathcal{L}^*_{\mathsf{CorrK}} := \{\mathsf{e} \in \{0, \dots l\} \mid \mathsf{CorrK}(\mathsf{e}) = 1\}$
  $\mathsf{CorrK}(\mathsf{e}) = 1 \iff (\mathsf{e} \in \mathcal{L}_{\mathsf{CorrK}})$ **or** $(\mathsf{CorrK}(\mathsf{e}-1)$ **and** $\mathsf{e} \in \mathcal{L}_{\mathsf{CorrT}})$ **or**
    or $(\mathsf{CorrK}(\mathsf{e}+1)$ **and** $\mathsf{e}+1 \in \mathcal{L}_{\mathsf{CorrT}})$
$\mathcal{L}^*_{\mathsf{CorrT}} := \{\mathsf{e} \in \{0, \dots l\} \mid (\mathsf{e} \in \mathcal{L}_{\mathsf{CorrT}})$ **or** $(\mathsf{e}-1 \in \mathcal{L}^*_{\mathsf{CorrK}}$ **and** $\mathsf{e} \in \mathcal{L}^*_{\mathsf{CorrK}})\}$
$\mathcal{L}^*_{\mathsf{Safe}} := \{\mathsf{e} \in \{0, \dots l\} \mid \mathsf{Safe}(\mathsf{e}) = 1\}$
  $\mathsf{Safe}(\mathsf{e}) = 1 \iff (\mathsf{e} \in \mathcal{L}_{\mathsf{Safe}})$ **or** $(\mathsf{Safe}(\mathsf{e}-1)$ **and** $\mathsf{e} \in \mathcal{L}^*_{\mathsf{CorrT}})$ **or**
    or $(\mathsf{Safe}(\mathsf{e}+1)$ **and** $\mathsf{e}+1 \in \mathcal{L}^*_{\mathsf{CorrT}})$
return $(\mathcal{L}^*_{\mathsf{CorrK}}, \mathcal{L}^*_{\mathsf{Safe}})$

Fig. 3. Expand algorithm. We follow the formulas in [22,7,18] to compute the leakage sets. This algorithm computes which keys and challenge ciphertexts are known to the adversary.

$\mathbf{Exp}^{\mathsf{Unlink}}_{\mathsf{PSigUE},\mathcal{A}}(1^\lambda)$:

$\mathsf{parG} \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$
$\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(\mathsf{parG})$
$(\mathsf{pk}_0, \mathsf{sk}_0) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp})$
$(\mathsf{ssk}, \mathsf{svk}) \xleftarrow{\$} \mathsf{SKGen}(\mathsf{pp})$
$\mathcal{L}_{\mathsf{CorrK}}, \mathcal{L}_{\mathsf{Safe}}, \mathcal{L}_{\mathsf{CorrT}} := \emptyset$
$\mathsf{b} \xleftarrow{\$} \{0,1\}$
$\mathsf{b}' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}^{\mathsf{b}}_{\mathsf{Unlink}}}(\mathsf{pp}, \mathsf{pk}_0, \mathsf{svk})$
return $[\![\mathsf{b}' = \mathsf{b}]\!]$

Oracle $\mathcal{O}^{\mathsf{Unlink}}_{\mathsf{Chall}_\mathsf{b}}(\mathsf{e}, (\mathbf{c}_0))$

$\mathbf{c}_1 \xleftarrow{\$} \mathcal{C}$
$\mathcal{L}_{\mathsf{Safe}} := \mathcal{L}_{\mathsf{Safe}} \cup \{\mathsf{e}\}$
$\tilde{\sigma}_{\mathsf{e}-1} \xleftarrow{\$} \mathsf{Sig}(\mathsf{ssk}, \mathsf{pk}_{\mathsf{e}-1}, \mathbf{c}_\mathsf{b})$
$(\tilde{\mathbf{c}}_\mathsf{e}, \tilde{\sigma}_\mathsf{e}) \xleftarrow{\$} \mathsf{Upd}(\Delta_\mathsf{e}, \mathbf{c}_\mathsf{b}, \tilde{\sigma}_{\mathsf{e}-1})$
phase $:= 1$
return $(\tilde{\mathbf{c}}_\mathsf{e}, \tilde{\sigma}_\mathsf{e})$

Fig. 4. Security experiment of Unlink security game of the PSigUE scheme. The adversary $\mathcal{A}$ has access to the oracles $\mathcal{O}^{\mathsf{b}}_{\mathsf{Unlink}} = (\mathcal{O}^{\mathsf{Unlink}}_{\mathsf{Chall}_\mathsf{b}}, \mathcal{O}_{\mathsf{Sig}}, \mathcal{O}_{\mathsf{Insert}}, \mathcal{O}_{\mathsf{Next}}, \mathcal{O}_{\mathsf{CorrK}}, \mathcal{O}_{\mathsf{CorrT}}, \mathcal{O}_{\mathsf{Upd}})$.

Definition 7. *A binary relation* R *is polynomially bounded if it is decidable in polynomial time and there exists a polynomial* p *such that for all* $(\mathsf{x}, \mathsf{w}) \in \mathsf{R}$*, we have* $|\mathsf{w}| \leq p(|\mathsf{x}|)$*. We also define the language related to the relation* $\mathcal{L}_\mathsf{R} := \{\mathsf{x} | \exists \mathsf{w}.(\mathsf{x}, \mathsf{w}) \in \mathsf{R}\}$*.*

Definition 8 (DM-ZKPoK)**.** *A dual-mode non-interactive zero-knowledge proof of knowledge scheme (*DM-ZKPoK*) for the relation* R *consists of six PPT algorithms* $\mathsf{DM\text{-}ZKPoK} = (\mathsf{Setup}_{\mathsf{Snd}}, \mathsf{Setup}_{\mathsf{ZK}}, \mathsf{Prove}, \mathsf{Ver}, \mathsf{Sim}, \mathsf{Ext})$ *with the following syntax:*

- $\mathsf{Setup}_{\mathsf{Snd}}(1^\lambda)$ *takes the security parameter* 1^λ *as input, and output a common reference string in the soundness mode* $\mathsf{crs}_{\mathsf{Snd}}$ *and an extraction key* ExtK*.*
- $\mathsf{Setup}_{\mathsf{ZK}}(1^\lambda)$ *takes the security parameter* 1^λ *as input, and outputs a common reference string in the zero-knowledge mode* $\mathsf{crs}_{\mathsf{ZK}}$ *and a simulation key* SimK*.*
- $\mathsf{Prove}(\mathsf{crs}, \mathsf{x}, \mathsf{w})$ *takes a common reference string* crs*, a statement* x*, and a witness* w *as input, and outputs a proof* π*.*
- $\mathsf{Ver}(\mathsf{crs}, \mathsf{x}, \pi)$ *takes a common reference string* crs*, a statement* x*, and a proof* π *as input, and outputs a boolean* 0 *(false) or* 1 *(true).*

- $\mathsf{Ext}(\mathsf{crs}_{\mathsf{Snd}}, \mathsf{ExtK}, \mathsf{x}, \pi)$ *takes a common reference string in the soundness mode, an extraction key* ExtK*, a statament* x*, and a proof* π *as input, and outputs a witness* w*.*
- $\mathsf{Sim}(\mathsf{crs}_{\mathsf{ZK}}, \mathsf{SimK}, \mathsf{x})$ *takes a common reference string in the zero-knowledge mode, a simulation key* SimK*, and a statament* x *as input, and outputs a proof* π*.*

We require the following properties for DM-ZKPoK *schemes:*

Correctness: *For all* $(\mathsf{x}, \mathsf{w}) \in \mathsf{R}$*, we have*

$$\Pr\left[\mathsf{Ver}(\mathsf{crs}_{\mathsf{Snd}}, \mathsf{x}, \pi) = 1 \;\middle|\; \begin{array}{l}(\mathsf{crs}_{\mathsf{Snd}}, \mathsf{ExtK}) \xleftarrow{\$} \mathsf{Setup}_{\mathsf{Snd}}(1^\lambda) \\ \pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}_{\mathsf{Snd}}, \mathsf{x}, \mathsf{w})\end{array}\right] = 1.$$

Perfect Soundness: *For all* $\mathsf{x} \in \mathsf{X}$ *and for all proof* π*, we have*

$$\Pr\left[\mathsf{Ver}(\mathsf{crs}_{\mathsf{Snd}}, \mathsf{x}, \pi) = 1 \implies (\mathsf{x}, \mathsf{w}) \in \mathsf{R} \;\middle|\; \begin{array}{l}(\mathsf{crs}_{\mathsf{Snd}}, \mathsf{ExtK}) \xleftarrow{\$} \mathsf{Setup}_{\mathsf{Snd}}(1^\lambda) \\ \mathsf{w} \xleftarrow{\$} \mathsf{Ext}(\mathsf{crs}_{\mathsf{Snd}}, \mathsf{ExtK}, \mathsf{x}, \pi)\end{array}\right] = 1.$$

Perfect Zero-Knowledge: *For all* $\mathsf{x} \in \mathsf{X}$*, we have*

$$\Pr\left[\mathsf{Ver}(\mathsf{crs}_{\mathsf{ZK}}, \mathsf{x}, \pi) = 1 \;\middle|\; \begin{array}{l}(\mathsf{crs}_{\mathsf{ZK}}, \mathsf{SimK}) \xleftarrow{\$} \mathsf{Setup}_{\mathsf{ZK}}(1^\lambda) \\ \pi \xleftarrow{\$} \mathsf{Sim}(\mathsf{crs}_{\mathsf{ZK}}, \mathsf{SimK}, \mathsf{x})\end{array}\right] = 1.$$

Mode-indistinguishability: *The common reference string* crs *generated by* $\mathsf{Setup}_{\mathsf{Snd}}$ *and* $\mathsf{Setup}_{\mathsf{ZK}}$ *are computationally indistinguishable. Formally, for all PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\begin{aligned}|\Pr\big[\mathcal{A}(\mathsf{crs}_{\mathsf{Snd}}) = 1 \;\big|\; (\mathsf{crs}_{\mathsf{Snd}}, \mathsf{ExtK}) \xleftarrow{\$} \mathsf{Setup}_{\mathsf{Snd}}(1^\lambda)\big] \\ - \Pr\big[\mathcal{A}(\mathsf{crs}_{\mathsf{ZK}}) = 1 \;\big|\; (\mathsf{crs}_{\mathsf{ZK}}, \mathsf{SimK}) \xleftarrow{\$} \mathsf{Setup}_{\mathsf{ZK}}(1^\lambda)\big]| \leq \mathsf{negl}(\lambda).\end{aligned}$$

3 Our efficient Construction of PSigUE without Unlinkability

Our first scheme PSigUE_1 is a simple and efficient construction consisting of only $2\mathbb{G}_1 + 1\mathbb{G}_2 + 1\mathbb{G}_\mathsf{T}$ element. It is based on a generalization of k-linear assumption in the asymmetric pairing setting. We call this new assumption the Bilateral k-linear assumption. Note that this construction is unlinkable and provides wIND-ENC security.

3.1 Bilateral k-Linear assumption

To build an efficient PSigUE scheme, we consider the following bilateral k-Lin assumption. We firstly define a uniform diagonal matrix distribution $\mathcal{L}_k = \{\begin{pmatrix} a_1 & & \\ & \ddots & \\ & & a_k \end{pmatrix} \mid a_1, \ldots, a_k \xleftarrow{\$} \mathbb{Z}_\mathsf{p}\}$.

Definition 9 (k-BLin). *Let* $\mathsf{parG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}, e, \mathsf{p})$ *be an asymmetric pairing groups with order* p. *We say that the* k-BLin *assumption holds on pairing groups* parG *if, the following two distributions are computationally indistinguishable:*

$$([\mathbf{A}]_1, [\mathbf{Ar}]_1, [\mathbf{A}]_2, [\mathbf{Ar}]_2) \approx_c ([\mathbf{A}]_1, [\mathbf{u}]_1, [\mathbf{A}]_2, [\mathbf{u}]_2),$$

where $\bar{\mathbf{A}} \xleftarrow{\$} \mathcal{L}_k$, $\mathbf{A} := \begin{pmatrix} \bar{\mathbf{A}} \\ \mathbf{1}^\intercal \end{pmatrix}$, $\mathbf{r} \xleftarrow{\$} \mathbb{Z}_\mathsf{p}^k$, *and* $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_\mathsf{p}^{k+1}$.

We can notice that by multiplying a uniformly random matrix $\bar{\mathbf{C}} \xleftarrow{\$} \mathcal{L}_k$ on the left side of $[\bar{\mathbf{A}}]_2$ and $[\bar{\mathbf{A}}\mathbf{r}]_2$, we can directly have the following lemma:

Lemma 1. *Let* $\mathsf{parG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}, e, \mathsf{p})$ *be an asymmetric pairing groups with order* p. *The following two distributions are computationally indistinguishable under the* k-BLin *assumption:*

$$([\mathbf{A}]_1, [\mathbf{Ar}]_1, [\bar{\mathbf{B}}]_2, [\bar{\mathbf{B}}\mathbf{r}]_2) \approx_c ([\mathbf{A}]_1, [\mathbf{u}]_1, [\bar{\mathbf{B}}]_2, [\bar{\mathbf{v}}]_2),$$

where $\bar{\mathbf{A}}, \bar{\mathbf{B}} \xleftarrow{\$} \mathcal{L}_k$, $\mathbf{A} := \begin{pmatrix} \bar{\mathbf{A}} \\ \mathbf{1}^\intercal \end{pmatrix}$, $\mathbf{r}, \bar{\mathbf{v}} \xleftarrow{\$} \mathbb{Z}_\mathsf{p}^k$, *and* $\mathbf{u} \xleftarrow{\$} \mathbb{Z}_\mathsf{p}^{k+1}$.

We note that the k-BLin assumption in the symmetric setting is the k-linear assumption [4,16,27].

By pairing the element $[\mathbf{1}^\intercal \cdot \mathbf{r}]_1$ in the Lemma 1 with $[1]_2$, we can get $[\mathbf{1}^\intercal \cdot \mathbf{r}]_\mathsf{T}$ and the following lemma which we will use in our construction.

Lemma 2. *Let* $\mathsf{parG} = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}, e, \mathsf{p})$ *be an asymmetric pairing groups with order* p. *The following two distributions are computationally indistinguishable under the* k-BLin *assumption:*

$$([\bar{\mathbf{A}}]_1, [\bar{\mathbf{A}}\mathbf{r}]_1, [\bar{\mathbf{B}}]_2, [\bar{\mathbf{B}}\mathbf{r}]_2, [\mathbf{1}^\intercal \cdot \mathbf{r}]_\mathsf{T}) \approx_c ([\bar{\mathbf{A}}]_1, [\bar{\mathbf{u}}]_1, [\bar{\mathbf{B}}]_2, [\bar{\mathbf{v}}]_2, [\mathbf{w}]_\mathsf{T}),$$

where $\bar{\mathbf{A}}, \bar{\mathbf{B}} \xleftarrow{\$} \mathcal{L}_k$, $\mathbf{r}, \bar{\mathbf{u}}, \bar{\mathbf{v}} \xleftarrow{\$} \mathbb{Z}_\mathsf{p}^k$, *and* $\mathbf{w} \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$.

3.2 Our Instantiation of PSigUE from k-BLin assumption

In this section, we give our first efficient construction of our public-key signable updatable encryption scheme PSigUE_1. Our first observation is that the signature on ciphertext and the updatability are incompatible. However, what we really need to authenticate is the underlying message. Our first intuition is that, to support signature scheme, we could separate the message and the public key in the ciphertext. The advantage of this approach is that since we only update the key part in the ciphertext and the signature is on the message part, we don't need to update the signature. Therefore, our starting point is an encryption scheme from k linear assumption. More precisely, an encryption of the message m consists of 2 group elements $\mathbf{c} = ([\mathbf{Ar}], [\mathbf{1}^\intercal \cdot \mathbf{r} + \mathsf{m}])$ in $\mathbb{G}$, where $\mathbf{A} \xleftarrow{\$} \mathcal{L}_k$. We can notice that in this ciphertext the first element only contains information about the public key and the second element only contains the message information. Thus,

we can provide a signature only on $[\mathbf{1}^\intercal \cdot \mathbf{r} + \mathsf{m}]$ to authenticate the underlying message.

However, this first attempt fails quickly by considering the following trivial attack on the integrity property. The adversary can simply modify the first element in the ciphertext $[\mathbf{A} \cdot \mathbf{r}]$ to $[2 \cdot \mathbf{A} \cdot \mathbf{r}]$. Since the first element is not signed, this move is legitimate. But when we decrypt the ciphertext we will get $[\mathbf{1}^\intercal \cdot \mathbf{r} + \mathsf{m} - 2 \cdot \mathbf{1}^\intercal \mathbf{r}] \neq \mathsf{m}$. This attack works because the randomness $\mathbf{r}$ is not fixed inside of the ciphertext. Our solution is to add another component $[\mathbf{B} \cdot \mathbf{r}]$ in the ciphertext with $[\mathbf{B}]$ in the public key together with a proof π of the fact that $[\mathbf{A} \cdot \mathbf{r}], [\mathbf{B} \cdot \mathbf{r}]$ share the same discret log $[\mathbf{r}]$. Then the signature is on $[\mathbf{B} \cdot \mathbf{r}]$ and $[\mathbf{1}^\intercal \cdot \mathbf{r} + \mathsf{m}]$. This new element $[\mathbf{B} \cdot \mathbf{r}]$ help us to fix the underlying randomness $\mathbf{r}$. The only problem for our second attempt is that the proof π may not be updatable when we modify the public key $[\mathbf{A}]$.

Fortunately, by using the bilateral k-linear assumption, we can get this proof π for free. More precisely, by putting $[\mathbf{A}], [\mathbf{A} \cdot \mathbf{r}]$ into $\mathbb{G}_1$, and $[\mathbf{B}], [\mathbf{B} \cdot \mathbf{r}]$ into $\mathbb{G}_2$, we can verify that they share the same discret log publicly without any proof by the following computation:

$$([\mathbf{Ar}]_1^\intercal \bullet [\mathbf{B}]_2)^\intercal = [\mathbf{A}]_1 \bullet [\mathbf{Br}]_2$$

We note that the above equality is only true because $\mathbf{A}$ and $\mathbf{B}$ are generated from the distribution $\mathcal{L}_k$. Consequently, we have $\mathbf{B}^\intercal \cdot \mathbf{A} = \mathbf{A} \cdot \mathbf{B}$. Since the zero-knowledge proof is no longer needed in this approach, we do not need to consider the updatability of the proof system anymore.

Summarying the above intuitions, we give our first construction of PSigUE_1 in Figure 5 by using a signature scheme $\Sigma = (\mathsf{SKGen}, \mathsf{Sig}, \mathsf{Ver})$ and a DM-ZKPoK proof system $\Pi_{\mathsf{DM}} = (\mathsf{Setup}_{\mathsf{Snd}}, \mathsf{Setup}_{\mathsf{ZK}}, \mathsf{Prove}, \mathsf{Ver}, \mathsf{Sim}, \mathsf{Ext})$.

3.3 Security of the construction PSigUE_1

In this section, we will formally analyse the security of PSigUE_1 as constructed in Figure 5.

Theorem 1 (wIND-ENC security). *Let* PSigUE_1 *be the public-key signable updatable encryption described in Figure 5. Suppose* n *is the total number of epochs. For any* wIND-ENC *adversary* $\mathcal{A}$ *against* PSigUE_1*, there exists two PPT adversaries* $\mathcal{B}_1, \mathcal{B}_2$ *such that*

$$\mathsf{Adv}^{\mathsf{wIND\text{-}ENC}}_{\mathsf{PSigUE}_1,\mathcal{A}}(1^\lambda) \leq \mathsf{n}^2 \cdot (\mathsf{Adv}^{\mathsf{Ind\text{-}Mode}}_{\Pi_{\mathsf{DM}},\mathcal{B}_1}(1^\lambda) + \mathsf{Adv}^{k\text{-}\mathsf{BLin}}_{\mathsf{parG},\mathcal{B}_2}(1^\lambda)).$$

Proof. We recall that, for the wIND-ENC security, the adversary cannot corrupt any secret key between two challenge epochs. Therefore, we firstly guess the starting and the ending challenge epoch. The success probability is $1/\mathsf{n}^2$ for n epochs in total. Assuming the first challenge epoch is e_L and the last e_R.

We will introduce a simple hybrid game that change the crs generation.

Game $\mathbf{G}_0$: This is the initial wIND-ENC security game with crs generation in

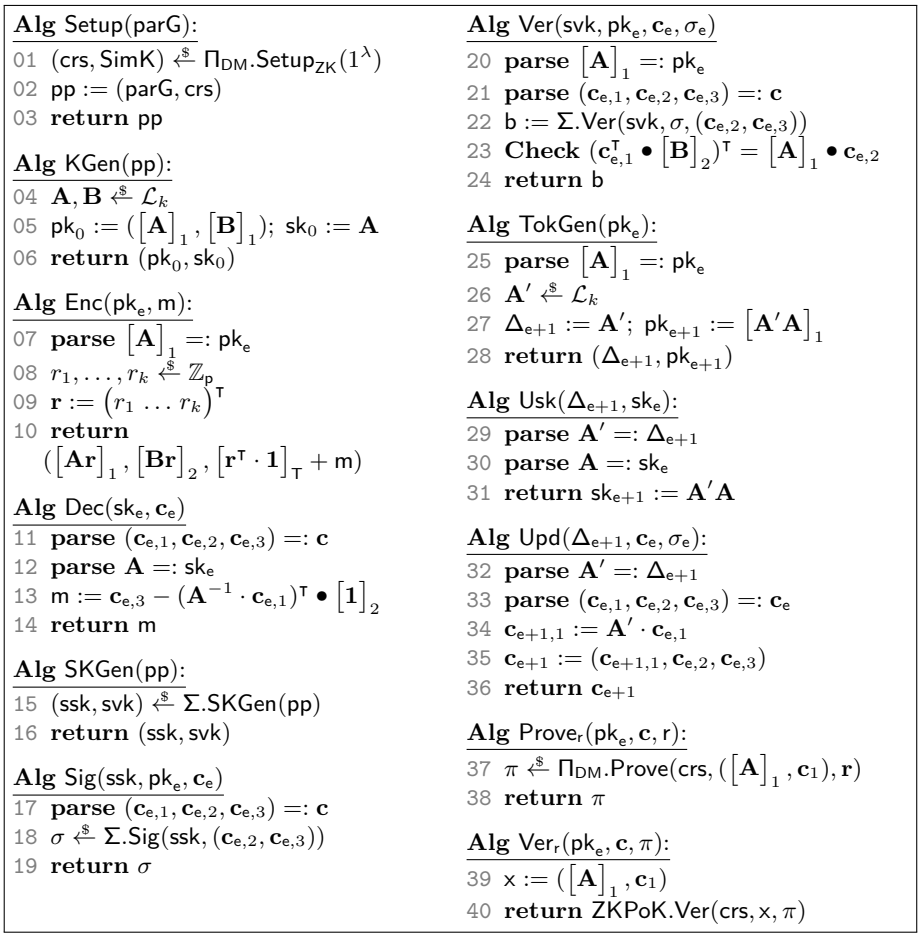

Fig. 5. Our construction of efficient PSigUE_1 without unlinkability. In the Prove_r algorithm, we use the Π_{DM} to prove the knowledge of r such that $\mathbf{c}_1 = [\mathbf{A}]_1 \cdot \mathbf{r}$.

the perfect zero-knowledge mode while the guess of challenge epochs are correct. Therefore, we have

$$\mathsf{pr}_0 = \frac{1}{\mathsf{n}^2} \cdot \mathsf{Adv}^{\mathsf{wIND\text{-}ENC}}_{\mathsf{PSigUE}_1,\mathcal{A}}(1^\lambda).$$

Game $\mathbf{G}_1$: In $\mathbf{G}_1$, we change the crs generation in the perfect sound setting. Together with crs, we also get an extraction key ExtK. The only difference between $\mathbf{G}_0$ and $\mathbf{G}_1$ is the mode switching. Therefore we have

$$|\mathsf{pr}_1 - \mathsf{pr}_0| \leq \mathsf{Adv}^{\mathsf{Ind-Mode}}_{\Pi_{\mathsf{DM}},\mathcal{B}_1}(1^\lambda).$$

Let $\mathcal{A}$ be an adversary against wIND-ENC security of PSigUE_1 with challenge epoch from e_0 to e_1, we will construct an adversary $\mathcal{B}_2$ against the k-BLin assumption. The explicit construction of $\mathcal{B}_2$ is given in Figure 6. We denote the challenge distribution $\mathsf{CH} = (\left[\mathbf{A}^\star\right]_1, \left[\mathbf{u}^\star\right]_1, \left[\mathbf{B}^\star\right]_2, \left[\mathbf{v}^\star\right]_2, \left[\mathbf{w}^\star\right]_\mathsf{T})$, where $\mathbf{A}^\star, \mathbf{B}^\star \xleftarrow{\$} \mathcal{L}_k$.

$\underline{\mathcal{B}_2(\mathsf{CH}, \mathsf{e}_\mathsf{L}, \mathsf{e}_\mathsf{R}, \mathcal{A}) :}$
01 $(\mathsf{crs}, \mathsf{ExtK}) \xleftarrow{\$} \mathsf{Setup}_{\mathsf{Snd}}(1^\lambda)$
02 $\mathsf{pp} := (\mathsf{parG}, \mathsf{crs})$
03 $\mathbf{A}_0, \mathbf{A}_{\mathsf{e}_\mathsf{R}+1} \xleftarrow{\$} \mathcal{L}_k$
04 $\mathsf{pk}_0 := (\left[\mathbf{A}_0\right]_1, \left[\mathbf{B}^\star\right]_2)$; $\mathsf{sk}_0 := \mathbf{A}_0$
05 $\mathsf{pk}_{\mathsf{e}_\mathsf{R}+1} \xleftarrow{\$} (\left[\mathbf{A}_{\mathsf{e}_\mathsf{R}+1}\right]_1, \left[\mathbf{B}^\star\right]_2)$
06 $\mathsf{sk}_{\mathsf{e}_\mathsf{R}+1} := \mathbf{A}_{\mathsf{e}_\mathsf{R}+1}$
07 **for** $i \in \{1,\ldots,\mathsf{e}_\mathsf{L}-1\} \cup \{\mathsf{e}_\mathsf{R}+2,\ldots,\mathsf{n}\}$
08 $\quad \mathbf{A}_i \xleftarrow{\$} \mathcal{L}_k$
09 $\quad \mathsf{pk}_i := (\left[\mathbf{A}_i\right]_1, \left[\mathbf{B}^\star\right]_1)$
10 $\quad \mathsf{sk}_i := \mathbf{A}_i$; $\Delta_i := \mathbf{A}_i \cdot \mathbf{A}_{i-1}^{-1}$
11 $\mathsf{pk}_{\mathsf{e}_\mathsf{L}} := (\left[\mathbf{A}^\star\right]_1)$
12 **for** $i = \mathsf{e}_\mathsf{L} + 1$ **to** e_R
13 $\quad \Delta_i \xleftarrow{\$} \mathcal{L}_k$; $\mathsf{pk}_i = \Delta_i \cdot \mathsf{pk}_{i-1}$
14 $(\mathsf{ssk}, \mathsf{svk}) \xleftarrow{\$} \Sigma.\mathsf{SKGen}(\mathsf{pp})$
15 $\mathsf{b}' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{wIND-ENC}}}(\mathsf{pp}, \mathsf{pk}_0, \mathsf{svk})$
16 **return** b'

$\underline{\mathcal{O}^{\mathsf{wIND-ENC}}_{\mathsf{Chall}_\mathsf{b}}(\mathsf{e}, (\mathsf{m}_0))}$
17 $\tilde{\mathbf{c}}_\mathsf{e} := (\left[\mathbf{u}^\star\right]_1, \left[\mathbf{v}^\star\right]_2, \left[\mathbf{w}^\star\right]_\mathsf{T} + \mathsf{m}_0)$
18 $\tilde{\sigma}_\mathsf{e} \xleftarrow{\$} \mathsf{Sig}(\mathsf{ssk}, \mathbf{c}_\mathsf{e})$
19 $\mathcal{L}_{\mathsf{Chall}} := \{(\tilde{\mathbf{c}}_\mathsf{e}, \tilde{\sigma}_\mathsf{e})\}$
20 **phase** $:= 1$
21 **return** $(\tilde{\mathbf{c}}_\mathsf{e}, \tilde{\sigma}_\mathsf{e})$

Oracle $\underline{\mathcal{O}_{\mathsf{Sig}}(\mathbf{c}, \mathsf{e})}$
01 $\sigma \xleftarrow{\$} \mathsf{Sig}(\mathsf{ssk}, (\mathsf{pk}_\mathsf{e}, \mathbf{c}))$
02 **return** σ

Oracle $\underline{\mathcal{O}_{\mathsf{Insert}}(\mathbf{c}, \pi)}$
18 **if** $\mathsf{Ver}_\mathsf{r}(\mathsf{pk}_\mathsf{e}, \mathbf{c}, \pi) = 1$
19 $\quad \sigma \xleftarrow{\$} \mathsf{Sig}(\mathsf{ssk}, \mathbf{c})$
20 $\quad \mathcal{L}_c := \mathcal{L}_c \cup \{(\mathbf{c}, \sigma, \mathsf{e}, \pi)\}$

Oracle $\underline{\mathcal{O}_{\mathsf{Next}}()}$
21 e**++**
22 **return** pk_e

Oracle $\underline{\mathcal{O}_{\mathsf{CorrK}}(i)}$
23 **Check** $0 \le i \le \mathsf{e}$
24 **return** sk_i

Oracle $\underline{\mathcal{O}_{\mathsf{CorrT}}(i)}$
25 **Check** $0 < i \le \mathsf{e}$
26 **return** Δ_i

Oracle $\underline{\mathcal{O}_{\mathsf{Upd}}((\mathbf{c}_{\mathsf{e}_0}, \sigma_{\mathsf{e}_0}), \mathsf{e}_0, \mathsf{e}_1)}$
27 **Check** $0 \le \mathsf{e}_0 < \mathsf{e}_1 \le \mathsf{e}$
28 **parse** $(\mathbf{c}_{\mathsf{e}_0,1}, \mathbf{c}_{\mathsf{e}_0,2}, \mathbf{c}_{\mathsf{e}_0,3}) =: \mathbf{c}_{\mathsf{e}_0}$
29 **if** $\exists \pi.(\mathbf{c}_{\mathsf{e}_0}, \sigma_{\mathsf{e}_0}, \mathsf{e}_0, \pi) \in \mathcal{L}_c$
30 $\quad \mathbf{r} := \Pi_{\mathsf{DM}}.\mathsf{Ext}(\mathsf{crs}, \mathsf{ExtK}, \pi)$
31 $\quad \mathbf{c}_{\mathsf{e}_1} := (\left[\mathbf{A}_{\mathsf{e}_1}\mathbf{r}\right]_1, \mathbf{c}_{\mathsf{e}_0,2}, \mathbf{c}_{\mathsf{e}_0,3})$
32 $\quad \sigma_{\mathsf{e}_1} := \sigma_{\mathsf{e}_0}$
33 $\quad \mathcal{L}_c := \mathcal{L}_c \cup \{(\mathbf{c}_{\mathsf{e}_1}, \sigma_{\mathsf{e}_0}, \mathsf{e}_1, \pi)\}$
34 **elseif** $(\mathbf{c}_{\mathsf{e}_0}, \sigma_{\mathsf{e}_0}) \in \mathcal{L}_{\mathsf{Chall}}$
35 $\quad$ **Check** $\mathsf{e}_\mathsf{L} \le \mathsf{e}_0 < \mathsf{e}_1 \le \mathsf{e}_\mathsf{R}$
36 $\quad$ **for** $i = \mathsf{e}_0 + 1$ **to** e_1
37 $\quad\quad \mathbf{c}_{i,1} := \Delta_i \cdot \mathbf{c}_{i-1,1}$
38 $\quad \mathbf{c}_{\mathsf{e}_1} := (\mathbf{c}_{\mathsf{e}_1,1}, \mathbf{c}_{\mathsf{e}_0,2}, \mathbf{c}_{\mathsf{e}_0,3})$
39 $\quad \sigma_{\mathsf{e}_1} := \sigma_{\mathsf{e}_0}$
40 $\quad \mathcal{L}_{\mathsf{Chall}} := \mathcal{L}_{\mathsf{Chall}} \cup \{(\mathbf{c}_{\mathsf{e}_1}, \sigma_{\mathsf{e}_1})\}$
41 **else abort**
42 **return** $(\mathbf{c}_{\mathsf{e}_1}, \sigma_{\mathsf{e}_1})$

Fig. 6. The construction of $\mathcal{B}_2$ that use wIND-ENC adversary $\mathcal{A}$ to break the k-BLin assumption. We recall that e_L and e_R are the starting and ending epoch for challenge ciphertexts.

We can notice that if the challenge distribution CH is k-BLin tuple, then the view of the the adversary $\mathcal{A}$ is identical to the security game $\mathbf{Exp}^{\mathsf{wIND-ENC}_0}_{\mathsf{PSigUE}_1,\mathcal{A}}(1^\lambda)$ in $\mathbf{G}_1$. Moreover, if $(\left[\mathbf{u}^\star\right]_1, \left[\mathbf{v}^\star\right]_2, \left[\mathbf{w}^\star\right]_\mathsf{T})$ are chosen uniformly random, then the view of the adversary $\mathcal{A}$ is identical to $\mathbf{Exp}^{\mathsf{wIND-ENC}_1}_{\mathsf{PSigUE}_1,\mathcal{A}}(1^\lambda)$ in $\mathbf{G}_1$. Therefore,

the success probability of the adversary $\mathcal{B}_2$ in breaking the k-BLin assumption is at least pr_1. More precisely, we have $\mathsf{pr}_1 \leq \mathsf{Adv}^{k\text{-}\mathsf{BLin}}_{\mathsf{parG},\mathcal{B}_2}(1^\lambda)$. Applying an union bound over the hybrids, we have done with the proof. □

Theorem 2 (Integrity). *Let* PSigUE_1 *be the public-key signable updatable encryption described in Figure 5. For any* Int *adversary* $\mathcal{A}$ *against* PSigUE_1*, there exists an adversary* $\mathcal{B}$ *such that*

$$\mathsf{Adv}^{\mathsf{Int}}_{\mathsf{PSigUE}_1,\mathcal{A}}(1^\lambda) \leq \mathsf{Adv}^{\mathsf{Unforg}}_{\Sigma,\mathcal{B}}(1^\lambda).$$

Proof. We can notice that the PSigUE_1 ciphertext given in Figure 5 is of the form $\mathbf{c}_\mathsf{e} = (\mathbf{c}_{\mathsf{e},1}, \mathbf{c}_{\mathsf{e},2}, \mathbf{c}_{\mathsf{e},3})$, and the signature is given for the message $(\mathbf{c}_{\mathsf{e},2}, \mathbf{c}_{\mathsf{e},3})$. Thus, the only part of the ciphertext can be modified by the adversary is $\mathbf{c}_{\mathsf{e},1}$. However, a valid ciphertext of pk_e should verify that $(\mathbf{c}^\intercal_{\mathsf{e},1} \bullet [\mathbf{B}]_2)^\intercal = [\mathbf{A}]_1 \bullet \mathbf{c}_{\mathsf{e},2}$. Note that $\mathbf{c}_{\mathsf{e},1}$ is fully determined by pk_e and $(\mathbf{c}_{\mathsf{e},2}, \mathbf{c}_{\mathsf{e},3})$. Therefore, we have the desired result. □

4 Unlinkable PSigUE scheme with ZKPoK

In this section, we give our second construction of public-key signable updatable encryption scheme PSigUE_2. It is a PSigUE with ZKPoK, this construction has the advantage that it has unlinkability.

4.1 $\mathcal{T}$-Updatable Non-Interactive Zero-Knowledge proof system

We first introduce the Updatable Non-Interactive Zero-Knowledge proof system ($\mathcal{T}$-UNIZK), which is a special family of Non-Interactive Zero-Knowledge proof system (NIZK). We will use $\mathcal{T}$-UNIZK to build our PSigUE_2 in Section 4.3.

Definition 10 ($\mathcal{T}$-UNIZK). *An Updatable Non-Interactive Zero-Knowledge proof system for the transformation* $\mathcal{T}$ ($\mathcal{T}$-UNIZK) *consists of five PPT algorithms* $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Ver}, \mathsf{Upd}, \mathsf{Rand})$ *with the following syntax:*

- $\mathsf{Setup}(1^\lambda)$ *takes security parameters* 1^λ *as input, and outputs a common reference string* crs.
- $\mathsf{Prove}(\mathsf{crs}, \mathcal{L}, \mathsf{x}, \mathsf{w})$ *takes the common reference string* crs*, an* **NP** *language* $\mathcal{L}$*, a statement* x *and a witness* w *as input, and outputs a proof* π.
- $\mathsf{Ver}(\mathsf{crs}, \mathcal{L}, \mathsf{x}, \pi)$ *takes the common reference string* crs*, an* **NP** *language* $\mathcal{L}$*, a statement* x *and a proof* π*, and outputs* 0 *or* 1.
- $\mathsf{Upd}(\mathcal{L}, \mathsf{x}, \pi, \mathcal{T})$ *takes a proof* π *for the language* $\mathcal{L}$ *and a transformation* $\mathcal{T} = (\mathcal{T}_\mathsf{x}, \mathcal{T}_\mathcal{L})$*, and outputs a new proof* π' *for* $\mathcal{T}_\mathsf{x}(\mathsf{x}) \in \mathcal{T}_\mathcal{L}(\mathcal{L})$.
- $\mathsf{Rand}(\pi)$ *takes a proof* π *and outputs a re-randomized proof* π'.

We require the correctness, updatability, rerandomizability, soundness and zero-knowledge properties for our $\mathcal{T}$-UNIZK.

CORRECTNESS: *For an* **NP** *language* $\mathcal{L}$*, and for all* $\mathsf{x} \in_{\mathsf{w}} \mathcal{L}$*, we have*

$$\Pr\left[\mathsf{Ver}(\mathsf{crs}_{\mathsf{Snd}}, \mathcal{L}, \mathsf{x}, \pi) = 1 \;\middle|\; \begin{array}{l}(\mathsf{crs}_{\mathsf{Snd}}, \mathsf{ExtK}) \xleftarrow{\$} \mathsf{Setup}_{\mathsf{Snd}}(1^\lambda) \\ \pi \xleftarrow{\$} \mathsf{Prove}(\mathsf{crs}_{\mathsf{Snd}}, \mathcal{L}, \mathsf{x}, \mathsf{w})\end{array}\right] = 1.$$

UPDATABILITY: *For an* **NP** *language* $\mathcal{L}$*, and for all* x, π*, we have that*

$$\Pr\left[\mathsf{Ver}(\mathsf{crs}, \mathcal{L}', \mathsf{x}', \pi') = 1 \;\middle|\; \begin{array}{l}\mathsf{Ver}(\mathsf{crs}, \mathcal{L}, \mathsf{x}, \pi) = 1 \\ \pi' \xleftarrow{\$} \mathsf{Upd}(\mathcal{L}, \mathsf{x}, \pi, \mathcal{T}) \\ \mathsf{x}' := \mathcal{T}_{\mathsf{x}}(\mathsf{x});\ \mathcal{L}' := \mathcal{T}_{\mathsf{x}}(\mathcal{L})\end{array}\right] = 1.$$

RERANDOMIZABILITY: *For an* **NP** *language* $\mathcal{L}$*, and for all* $\mathsf{x} \in_{\mathsf{w}} \mathcal{L}$*. The following two distributions are identical*

$$\pi \equiv \mathsf{Rand}(\mathsf{Prove}(\mathsf{crs}, \mathcal{L}, \mathsf{x}, \mathsf{w})).$$

SOUNDNESS: *There exists two PPT algorithms* $(\mathsf{Setup}_{\mathsf{Snd}}, \mathsf{Ext})$*, such that for all PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\begin{aligned}|\Pr\big[\mathcal{A}(\mathsf{crs}) = 1 \;\big|\; \mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)\big] - \\ \Pr\big[\mathcal{A}(\mathsf{crs}_{\mathsf{Snd}}) = 1 \;\big|\; (\mathsf{crs}_{\mathsf{Snd}}, \mathsf{ExtK}) \xleftarrow{\$} \mathsf{Setup}_{\mathsf{Snd}}(1^\lambda)\big]| \leq \mathsf{negl}(1^\lambda).\end{aligned}$$

Moreover, for every statement $\mathsf{x} \in \mathsf{X}$ *and for all proof* π*, we have*

$$\Pr\left[\mathsf{Ver}(\mathsf{crs}_{\mathsf{Snd}}, \mathcal{L}, \mathsf{x}, \pi) = 1 \implies (\mathsf{x}, \mathsf{w}) \in \mathsf{R} \;\middle|\; \begin{array}{l}(\mathsf{crs}_{\mathsf{Snd}}, \mathsf{ExtK}) \xleftarrow{\$} \mathsf{Setup}_{\mathsf{Snd}}(1^\lambda) \\ \mathsf{w} \xleftarrow{\$} \mathsf{Ext}(\mathsf{crs}_{\mathsf{Snd}}, \mathcal{L}, \mathsf{ExtK}, \mathsf{x}, \pi)\end{array}\right] = 1.$$

ZERO-KNOWLEDGE: *There exists two PPT algorithm* $(\mathsf{Setup}_{\mathsf{ZK}}, \mathsf{Sim})$*, such that for all PPT adversary* $\mathcal{A}$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\begin{aligned}|\Pr\big[\mathcal{A}(\mathsf{crs}) = 1 \;\big|\; \mathsf{crs} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)\big] - \\ \Pr\big[\mathcal{A}(\mathsf{crs}_{\mathsf{ZK}}) = 1 \;\big|\; (\mathsf{crs}_{\mathsf{ZK}}, \mathsf{SimK}) \xleftarrow{\$} \mathsf{Setup}_{\mathsf{ZK}}(1^\lambda)\big]| \leq \mathsf{negl}(1^\lambda).\end{aligned}$$

Moreover, for every statement $x \in \mathsf{X}$*, we have*

$$\Pr\left[\mathsf{Ver}(\mathsf{crs}_{\mathsf{ZK}}, \mathsf{x}, \pi) = 1 \;\middle|\; \begin{array}{l}(\mathsf{crs}_{\mathsf{ZK}}, \mathsf{SimK}) \xleftarrow{\$} \mathsf{Setup}_{\mathsf{ZK}}(1^\lambda) \\ \pi \xleftarrow{\$} \mathsf{Sim}(\mathsf{crs}_{\mathsf{ZK}}, \mathsf{SimK}, \mathsf{x})\end{array}\right] = 1.$$

4.2 Instantiation of $\mathcal{T}$-UNIZK from SXDH assumption

We notice that Groth-Sahai proof system already has some malleability properties. For specific language and transformation family, we show in Figure 10 that, Groth-Sahai proof system is a $\mathcal{T}$-UNIZK.

Let $\mathsf{R}_{\mathbf{A},\mathbf{h}}$ be a binary relation such that $(\mathbf{C},(r,\mathsf{m})) \in \mathsf{R}_{\mathbf{A},\mathbf{h}}$ is defined as

$$\mathbf{C} = \mathbf{A} \cdot r + \begin{bmatrix} 0 \\ \mathsf{WH}(\mathsf{m}) \end{bmatrix}_1,$$

where $\mathbf{C}, \mathbf{A} \in \mathbb{G}_1^2$, $r \in \mathbb{Z}_\mathsf{p}$, $\mathsf{m} \in \{0,1\}^\ell$, and $\mathsf{WH}(\mathsf{m}) = \begin{bmatrix} 1 \ \mathsf{m}_1 \cdots \mathsf{m}_\ell \end{bmatrix} \cdot \mathbf{h}$ with $\mathbf{h} \in \mathbb{G}_1^{\ell+1}$.

We also define the transform $\mathcal{T} = (\mathcal{T}_\mathsf{x}, \mathcal{T}_\mathcal{L})$ as follows.

$$\mathcal{T}_\mathsf{x}(\mathbf{C}) = \begin{pmatrix} \Delta_\mathsf{e} \cdot (\mathbf{C}_0 + \mathbf{A}_0 \cdot r') \\ \mathbf{C}_1 + \begin{bmatrix} r' \end{bmatrix}_1 \end{pmatrix} \qquad \mathcal{T}_\mathcal{L}(\mathsf{R}_{\mathbf{A},\mathbf{h}}) = \mathsf{R}_{\mathbf{A}',\mathbf{h}},$$

where $\mathbf{A}' = \begin{pmatrix} \Delta_\mathsf{e} \cdot \mathbf{A}_0 \\ \mathbf{A}_1 \end{pmatrix}$, and $\Delta_\mathsf{e}, r' \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$.

We give the explicit construction of $\mathcal{T}$-UNIZK for the relation $RelGS$, including the update and re-randomization algorithms in Figure 10.

4.3 Instantiation of PSigUE from SXDH assumption

In this section, we give the instantiation of our PSigUE_2 scheme by using the $\mathcal{T}$-UNIZK system Π constructed in Figure 10 as a building block.

The correctness and the unlinkability of PSigUE_2 in Figure 7 is straightforward. We will provide the proof of the wIND-ENC security and integrity.

Theorem 3 (wIND-ENC). *Let* PSigUE_2 *be the public-key signable updatable encryption described in Figure 7. For any* wIND-ENC *adversary* $\mathcal{A}$ *against* PSigUE_1*, there exists two PPT adversaries* $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3$ *such that,*

$$\mathsf{Adv}^{\mathsf{wIND\text{-}ENC}}_{\mathsf{PSigUE}_2,\mathcal{A}}(1^\lambda) \leq \mathsf{n}^2 \cdot (\mathsf{Adv}^{\mathsf{ZK}}_{\Pi,\mathcal{B}_1}(1^\lambda) + \mathsf{Adv}^{\mathsf{Dual\text{-}Mode}}_{\Pi_{\mathsf{DM}},\mathcal{B}_2}(1^\lambda) + \mathsf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathcal{B}_3}(1^\lambda)).$$

We give the detailed proof of Theorem 3 in Supplementary Material Section B.1.

4.4 Integrity

To argue the integrity of of PSigUE_2, we need to firstly show an extended version of Waters signature. Then, as in [3], we prove that the extended version of Waters signature is extended unforgeable under chosen-message attack (Ext-UF-CMA). Then we will reduce the integrity property of PSigUE_2 to Ext-UF-CMA of the underlying Waters signature.

Theorem 4 (Integrity). *Let* PSigUE_2 *be the public-key signable updatable encryption described in Figure 7. For any* Int *adversary* $\mathcal{A}$ *(that makes max* $\mathsf{Q}_{\mathsf{Sig}}$ *signing queries), there exists an adversary* $\mathcal{B}$ *with* $\mathsf{T}(\mathcal{A}) \approx \mathsf{T}(\mathcal{B})$*, such that*

$$\mathsf{Adv}^{\mathit{Int}}_{\mathsf{PSigUE}_2,\mathcal{A}}(1^\lambda) \leq \mathsf{Adv}^{\mathsf{SXDH}_1}_{\mathcal{B},\mathbb{G}_1}(1^\lambda) \cdot O(\frac{1}{\sqrt{\ell \cdot \mathsf{Q}_{\mathsf{Sig}}}}).$$

We give the detailed proof in Supplementary Material Section B.2

Alg $\mathsf{Setup}(1^\lambda)$:
01 $\mathsf{parG} \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$
02 $\mathsf{crs} \xleftarrow{\$} \Pi.\mathsf{Setup}(1^\lambda)$
03 $\mathsf{crs}_{\mathsf{DM}} \xleftarrow{\$} \Pi_{\mathsf{DM}}.\mathsf{Setup}_{\mathsf{ZK}}(1^\lambda)$
04 $\mathbf{h} \xleftarrow{\$} \mathbb{G}_1^{\ell+1}$
05 **return** $\mathsf{pp} := (\mathsf{parG}, \mathsf{crs}, \mathsf{crs}_{\mathsf{DM}}, \mathbf{h})$

Alg $\mathsf{KGen}(\mathsf{pp})$:
06 $x \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$
07 **return** $(\mathsf{pk}, \mathsf{sk}) = (\left[x\right]_1, x)$

Alg $\mathsf{Enc}(\mathsf{pk}_\mathsf{e}, \mathsf{m} \in \{0,1\}^\ell)$:
08 $r \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$
09 $\mathbf{A} := \left[x\ 1\right]_1^\intercal$
10 $\mathbf{C} := \begin{pmatrix} \mathbf{C}_0 \\ \mathbf{C}_1 \end{pmatrix} := \mathbf{A} \cdot r + \begin{bmatrix} 0 \\ \mathsf{WH}(\mathsf{m}) \end{bmatrix}_1$
11 $\pi \xleftarrow{\$} \Pi.\mathsf{Prove}(\mathsf{crs}, \mathbf{C}, (r, \mathsf{m}))$
12 $\mathbf{c}_\mathsf{e} := (\mathbf{C}, \pi)$
13 **return** $\mathbf{c}_\mathsf{e}$

Alg $\mathsf{Dec}(\mathsf{sk}_\mathsf{e}, \mathbf{c}_\mathsf{e})$:
14 **parse** $\mathbf{c}_\mathsf{e} =: (\mathbf{C}, \pi)$
15 **Find** $\mathsf{m} : \mathsf{WH}(\mathsf{m}) = \left(-\frac{1}{x}\ 1\right) \cdot \mathbf{C}$
16 **return** m

Alg $\mathsf{SKGen}(1^\lambda)$:
17 $\mathsf{u} \xleftarrow{\$} \mathbb{G}_1$
18 $y \xleftarrow{\$} \mathbb{Z}_\mathsf{p};\ \hat{\mathsf{v}} := \left[y\right]_2 \in \mathbb{G}_2$
19 $\mathsf{w} := y \cdot \mathsf{u}$
20 $\mathsf{svk} := (\mathsf{u}, \hat{\mathsf{v}}) \in \mathbb{G}_1 \times \mathbb{G}_2$
21 $\mathsf{ssk} := \mathsf{w} \in \mathbb{G}_1$
22 **return** $(\mathsf{svk}, \mathsf{ssk})$

Alg $\mathsf{Sig}(\mathsf{ssk}, \mathbf{c})$:
23 $s \xleftarrow{\$} \mathbb{Z}_\mathsf{p};\ \mathbf{Z} := s \cdot \mathbf{A};\ \hat{\mathsf{z}} := \left[s\right]_2$
24 $\mathbf{D} := \begin{pmatrix} \mathbf{D}_0 \\ \mathbf{D}_1 \end{pmatrix} := \mathbf{C} \cdot s + \begin{pmatrix} 0 \\ \mathsf{w} \end{pmatrix}$
25 **return** $\sigma_\mathsf{e} = (\mathbf{D}, \mathbf{Z}, \hat{\mathsf{z}})$

Alg $\mathsf{Ver}(\mathsf{svk}, \mathbf{c}, \sigma)$:
26 **parse** $\mathbf{c}_\mathsf{e} =: (\mathbf{C}, \pi)$
27 **Check** $\mathbf{D}_0 \bullet \left[1\right]_2 = \mathbf{C}_0 \bullet \hat{\mathsf{z}}$
28 **Check** $\mathbf{D}_1 \bullet \left[1\right]_2 = \mathbf{C}_1 \bullet \hat{\mathsf{z}} + \mathsf{u} \bullet \hat{\mathsf{v}}$
29 **Check** $\Pi.\mathsf{Ver}(\mathsf{crs}, \mathbf{C}, \pi)$

Alg $\mathsf{TokGen}(\mathsf{pk}_\mathsf{e})$:
30 $\Delta_\mathsf{e} \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$
31 $\mathsf{pk}_{\mathsf{e}+1} := \Delta_\mathsf{e} \cdot \mathsf{pk}_\mathsf{e} \in \mathbb{G}_1$
32 **return** $(\mathsf{pk}_{\mathsf{e}+1}, \Delta_\mathsf{e})$

Alg $\mathsf{Usk}(\Delta_\mathsf{e}, \mathsf{sk}_\mathsf{e})$:
33 $\mathsf{sk}_{\mathsf{e}+1} := \Delta_\mathsf{e} \cdot \mathsf{sk}_\mathsf{e} \in \mathbb{Z}_\mathsf{p}$
34 **return** $\mathsf{sk}_{\mathsf{e}+1}$

Alg $\mathsf{Upd}(\Delta_\mathsf{e}, \mathbf{c}_\mathsf{e}, \sigma_\mathsf{e})$:
35 **parse** $(\mathbf{C}, \pi) =: \mathbf{c}_\mathsf{e}$
36 **parse** $(\mathbf{D}, \mathbf{Z}, \hat{\mathsf{z}}) =: \sigma_\mathsf{e}$
37 $r' \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$
38 $\mathbf{C}' := \begin{pmatrix} \Delta_\mathsf{e} \cdot \mathbf{C}_0 + r' \Delta_\mathsf{e} \cdot \mathsf{pk}_\mathsf{e} \\ \mathbf{C}_1 + \left[r'\right]_1 \end{pmatrix}$
39 $\pi' \xleftarrow{\$} \Pi.\mathsf{Upd}(\pi, \Delta_\mathsf{e}, r')$
40 $\mathbf{D}' := \begin{pmatrix} \Delta_\mathsf{e} \cdot \mathbf{D}_0 + r' \Delta_\mathsf{e} \cdot \mathbf{Z}_0 \\ \mathbf{D}_1 + r' \cdot \mathbf{Z}_1 \end{pmatrix}$
41 $\mathbf{Z}' := \begin{pmatrix} \Delta_\mathsf{e} \cdot \mathbf{Z}_0 \\ \mathbf{Z}_1 \end{pmatrix};\ \hat{\mathsf{z}}' := \hat{\mathsf{z}}$
42 $\mathbf{c}_{\mathsf{e}+1} := (\mathbf{C}', \pi')$
43 $\sigma_{\mathsf{e}+1} := (\mathbf{D}', \mathbf{Z}', \hat{\mathsf{z}}')$
44 **return** $\mathbf{c}_{\mathsf{e}+1}, \sigma_{\mathsf{e}+1}$

Alg $\mathsf{Prove}_\mathsf{m}(\mathsf{pk}_\mathsf{e}, \mathbf{c}, \mathsf{m})$:
45 $\pi_{\mathsf{DM}} \xleftarrow{\$} \Pi_{\mathsf{DM}}.\mathsf{Prove}(\mathsf{crs}_{\mathsf{DM}}, (\mathsf{pk}_\mathsf{e}, \mathbf{c}), \mathsf{m})$
46 **return** π_{DM}

Alg $\mathsf{Ver}_\mathsf{m}(\mathsf{pk}_\mathsf{e}, \mathbf{c}, \pi)$:
47 **return** $\Pi_{\mathsf{DM}}.\mathsf{Ver}(\mathsf{crs}_{\mathsf{DM}}, (\mathsf{pk}_\mathsf{e}, \mathbf{c}))$

Alg $\mathsf{WH}(\mathsf{m} \in \{0,1\}^\ell)$:
48 **return** $\left[1\ \mathsf{m}_1 \cdots \mathsf{m}_\ell\right] \cdot \mathbf{h}$

Fig. 7. Instantiation of PSigUE_2, where $\mathsf{parG} := (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_\mathsf{T}, \mathsf{p}, e, \mathsf{g}_1, \mathsf{g}_2, \mathsf{g}_\mathsf{T})$ generated by using the group generation algorithm $\mathsf{GGen}(1^\lambda)$. Π_{DM} is the dual-mode ZKPoK as defined in Definition 8.

5 Construction of tightly-secure PSigUE without ZKPoK

In this section we give our tightly-secure PSigUE without ZKPoK. In addition, this PSigUE scheme achieves IND-ENC security, which is stronger than the weaker security model wIND-ENC.

Our starting point are the tag-based tightly-secure encryption schemes. We want to take advantage of lossy trapdoor functions [26], and construct our scheme in a such way that only the challenge ciphertext and its updated version will have a lossy tag, and all other ciphertexts provided by the adversary will have an injective tag. However, there are several difficulties.

- If the tag of a ciphertext is changing while updating, then we need to make sure that the tag of the challenge ciphertext is still a lossy tag after the update. Moreover, for security we need to prove that the lossy tag and injective tag during the challenge epoch is indistinguishable for the adversary with token and secret key corruption. We encounter the same type of problem while trying to tightly prove the security of PSigUE.
- Another choice is that the tag remains the same while updating the ciphertext. Moreover, to our knowledge in all all-but-one lossy trapdoor functions [26], the lossy tag is included in the invertion key. In this case, it leads to a straight-forward attack. The adversary can simply corrupt any epoch other than the challenge one to obtain the lossy tag, then it can see that the challenge ciphertext is generated with a lossy tag instead of an injective one.

To solve the above issues, we borrow some ideas from the Cramer-Shoup like CCA encryption schemes [10,23]. Therefore, for our second attempt the core part of the ciphertext consists of 3 group elements:

$$\mathbf{C}_1 = \mathbf{A}_1 \cdot r \qquad \mathbf{C}_2 = \mathbf{A}_2 \cdot r \qquad \mathbf{C}_3 = \mathbf{X} \cdot r + \mathsf{m},$$

with $\mathsf{pk} = (\mathbf{A}_0, \mathbf{A}_1, \mathbf{X}) \in \mathbb{G}_1^3$, $\mathsf{sk} = (x_1, x_2) \in \mathbb{Z}_\mathsf{p}$, and $\mathbf{X} = \mathbf{A}_1 \cdot x_1 + \mathbf{A}_2 \cdot x_2 \in \mathbb{G}_1$. In addition to the core part, there is a tagged Groth-Sahai proof of $\log_{\mathbf{A}_1}(\mathbf{C}_1) = \log_{\mathbf{A}_2}(\mathbf{C}_2)$. The idea behind [23] is that every ciphertext is associated with a tag t, the common reference string crs of the Groth-Sahai proof is parameterized with t in a such way that only $\mathsf{crs}_{\mathsf{t}^\star}$ of the challenge ciphertext is in the perfect zero-knowledge setting, other crs_t are all in the perfect sound setting. Therefore, the challenger can use the Groth-Sahai simulation trapdoor to fake the proof in the challenge ciphertext while keeping other proofs perfectly sound. We can notice that the exact value of x_1, x_2 leaks only if we decrypt a ciphertext with $\log_{\mathbf{A}_1}(\mathbf{C}_1) \neq \log_{\mathbf{A}_2}(\mathbf{C}_2)$. Therefore all information of which x_2 is used is completely hidden for the adversary without using any assumption. We will exploit this to hide the underlying message.

However, it is difficult to make the above scheme updatable. To guarantee the non-malleability of the above scheme, $\mathbf{C}_3$ needs to be signed. This means $\mathbf{X}$ cannot be updated. On the other hand, to fully update $x_1, x_2, \mathbf{A}_1, \mathbf{A_2}$ while keeping $\mathbf{X} = \mathbf{A}_1 \cdot x_1 + \mathbf{A}_2 \cdot x_2$, we need know the $\log_{\mathbf{A}_1}(\mathbf{A}_2)$ which breaks the security. Our solution is to add a third element $\mathbf{A}_0$ into the public key such that $\mathbf{X} = \mathbf{A}_0 + \mathbf{A}_1 \cdot x_1 + \mathbf{A}_2 \cdot x_2$. Now, we can successfully update the core part of the ciphertext, by updating $\mathbf{A}_0$ while keeping $\mathbf{A}_1, \mathbf{A}_2, \mathbf{X}$ unchanged. Fortunately, we can also update the rest parts of the ciphertext as well.

We give our detailed construction in Figure 8.

Alg $\mathsf{Setup}(\mathsf{parG})$:
$\mathsf{pp} := \mathsf{parG}$
return pp

Alg $\mathsf{KGen}(\mathsf{pp})$:
$\mathbf{u} = \begin{bmatrix} u_1 & u_2 \end{bmatrix}_2^\intercal$
$\mathsf{td} \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$
$\mathbf{v} := \mathbf{u} \cdot \mathsf{td} = \begin{bmatrix} u_1 \cdot \mathsf{td} & u_2 \cdot \mathsf{td} \end{bmatrix}_2^\intercal$
$\mathsf{crs} := (\mathbf{u}, \mathbf{v})$
$a_0, a_1, a_2, b, x_1, x_2 \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$
$\mathbf{A} := \begin{pmatrix} \mathbf{A}_0 \\ \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix} = \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix}_1$
$\mathbf{B} := \begin{bmatrix} b \end{bmatrix}_1$
$\mathbf{X} := \begin{pmatrix} 1 & x_1 & x_2 \end{pmatrix} \cdot \mathbf{A}$
$\mathsf{pk} := (\mathbf{A}, \mathbf{B}, \mathbf{X})$; $\mathsf{sk} := (x_1, x_2)$
return $(\mathsf{pk}, \mathsf{sk})$

Alg $\mathsf{Enc}(\mathsf{pk}_\mathsf{e}, \mathsf{m})$:
$(\mathsf{osk}, \mathsf{ovk}) \xleftarrow{\$} \mathsf{OT.KGen}(1^\lambda)$
$\mathsf{t} \xleftarrow{\$} \mathsf{H}(\mathsf{ovk}) \in \mathbb{Z}_\mathsf{p}$
$r \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$
$\mathbf{C} := \begin{pmatrix} \mathbf{C}_0 & \mathbf{C}_1 & \mathbf{C}_2 \end{pmatrix}^\intercal = \mathbf{A} \cdot r$
$\mathbf{D} := \mathbf{B} \cdot r$
$\mathbf{Y} := \mathbf{X} \cdot r + \mathsf{m}$
$\mathbf{v}_\mathsf{t} := \mathbf{v} + \begin{bmatrix} 0 & \mathsf{t} \end{bmatrix}_2$
$\mathsf{crs}_\mathsf{t} := (\mathbf{u}, \mathbf{v}_\mathsf{t})$
$s \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$
$\mathsf{Com}_r := \mathbf{u} \cdot s + \mathbf{v}_\mathsf{t} \cdot r$
$\pi_\mathbf{A} := \begin{pmatrix} \pi_{a_0} & \pi_{a_1} & \pi_{a_2} \end{pmatrix}^\intercal := \mathbf{A} \cdot s$
$\pi_\mathbf{B} := \mathbf{B} \cdot s$
$\pi := (\mathsf{Com}_r, \pi_\mathbf{A}, \pi_\mathbf{B})$
$\sigma_\mathsf{OT} \xleftarrow{\$} \mathsf{OT.Sig}(\mathsf{osk}, (\mathbf{D}, \mathbf{Y}, \mathsf{Com}_r))$
$\mathbf{c} := (\mathsf{ovk}, \mathbf{C}, \mathbf{D}, \mathbf{Y}, \pi, \sigma_\mathsf{OT})$
return $\mathbf{c}$

Alg $\mathsf{Dec}(\mathsf{sk}_\mathsf{e}, \mathbf{c}_\mathsf{e})$
parse $(\mathsf{ovk}, \mathbf{C}, \mathbf{D}, \mathbf{Y}, \sigma_\mathsf{OT}, \pi) =: \mathbf{c}_\mathsf{e}$
parse $(x_1, x_2) =: \mathsf{sk}_\mathsf{e}$
Check $\mathsf{OT.Ver}(\mathsf{ovk}, \sigma_\mathsf{OT}, (\mathbf{D}, \mathbf{Y}, \mathsf{Com}_r))$
Check $\mathbf{A}_0 \bullet \mathsf{Com}_r = \mathbf{C}_0 \bullet \mathbf{v}_\mathsf{t} + \pi_{a_0} \bullet \mathbf{u}$
Check $\mathbf{A}_1 \bullet \mathsf{Com}_r = \mathbf{C}_1 \bullet \mathbf{v}_\mathsf{t} + \pi_{a_1} \bullet \mathbf{u}$
Check $\mathbf{A}_2 \bullet \mathsf{Com}_r = \mathbf{C}_2 \bullet \mathbf{v}_\mathsf{t} + \pi_{a_2} \bullet \mathbf{u}$
Check $\mathbf{B} \bullet \mathsf{Com}_r = \mathbf{D} \bullet \mathbf{v}_\mathsf{t} + \pi_\mathbf{B} \bullet \mathbf{u}$
$\mathsf{m} := \mathbf{Y} - \mathbf{C}_0 - \mathbf{C}_1 \cdot x_1 - \mathbf{C}_2 \cdot x_2$
return m

Alg $\mathsf{SKGen}(\mathsf{pp})$:
$(\mathsf{svk}, \mathsf{ssk}) \xleftarrow{\$} \Sigma.\mathsf{KGen}(1^\lambda)$
return $(\mathsf{svk}, \mathsf{ssk})$

Alg $\mathsf{Sig}(\mathsf{ssk}, \mathsf{pk}_\mathsf{e}, \mathbf{c}_\mathsf{e})$
$\sigma \xleftarrow{\$} \Sigma.\mathsf{Sig}(\mathsf{ssk}, (\mathbf{D}, \mathbf{Y}))$
return σ

Alg $\mathsf{Ver}(\mathsf{svk}, \mathsf{pk}_\mathsf{e}, \mathbf{c}_\mathsf{e}, \sigma_\mathsf{e})$
return $\Sigma.\mathsf{Ver}(\mathsf{svk}, (\mathbf{D}, \mathbf{Y}), \sigma_\mathsf{e})$

Alg $\mathsf{TokGen}(\mathsf{pk}_\mathsf{e})$:
parse $(\mathbf{A}, \mathbf{B}, \mathbf{X}) =: \mathsf{pk}_\mathsf{e}$
$\delta_1, \delta_2 \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$
$\mathbf{A}' := \begin{pmatrix} \mathbf{A}_0 - \mathbf{A}_1 \cdot \delta_1 - \mathbf{A}_2 \cdot \delta_2 \\ \mathbf{A}_1 \\ \mathbf{A}_2 \end{pmatrix}$
$\mathsf{pk}_{\mathsf{e}+1} := (\mathbf{A}', \mathbf{B}, \mathbf{X})$
$\Delta_{\mathsf{e}+1} := (\delta_1, \delta_2)$
return $(\mathsf{pk}_{\mathsf{e}+1}, \Delta_{\mathsf{e}+1})$

Alg $\mathsf{Usk}(\Delta_{\mathsf{e}+1}, \mathsf{sk}_\mathsf{e})$:
$\mathsf{sk}_{\mathsf{e}+1} := (x_1 + \delta_1, x_2 + \delta_2)$
return $\mathsf{sk}_{\mathsf{e}+1}$

Alg $\mathsf{Upd}(\Delta_{\mathsf{e}+1}, \mathbf{c}_\mathsf{e}, \sigma_\mathsf{e})$:
parse $(\mathsf{ovk}, \mathbf{C}, \mathbf{D}, \mathbf{Y}, \sigma_\mathsf{OT}, \pi) =: \mathbf{c}_\mathsf{e}$
$\mathbf{C}' := \begin{pmatrix} \mathbf{C}_0 - \mathbf{C}_1 \cdot \delta_1 - \mathbf{C}_2 \cdot \delta_2 \\ \mathbf{C}_1 \\ \mathbf{C}_2 \end{pmatrix}$
$\pi'_\mathbf{A} := \begin{pmatrix} \pi_{a_0} - \pi_{a_1} \cdot \delta_1 - \pi_{a_2} \cdot \delta_2 \\ \pi_{a_1} \\ \pi_{a_2} \end{pmatrix}$
$\pi' := (\mathsf{Com}_r, \pi'_\mathbf{A}, \pi_\mathbf{B})$
$\mathbf{c}_{\mathsf{e}+1} := (\mathsf{ovk}, \mathbf{C}', \mathbf{D}, \mathbf{Y}, \sigma_\mathsf{OT}, \pi')$
$\sigma_{\mathsf{e}+1} := \sigma_\mathsf{e}$
return $(\mathbf{c}_{\mathsf{e}+1}, \sigma_{\mathsf{e}+1})$

Fig. 8. Our construction of tightly-secure $\mathsf{PSigUE}_\mathsf{T}$ without ZKPoK.

Theorem 5. *Let* $\mathsf{PSigUE}_{\mathsf{T}}$ *be the public-key signable updatable encryption described in Figure 8. For any* IND-ENC *adversary* $\mathcal{A}$ *against* $\mathsf{PSigUE}_{\mathsf{T}}$*, there exists two PPT adversaries* $\mathcal{B}_1, \mathcal{B}_2$ *such that,*

$$Adv^{\mathsf{IND\text{-}ENC}}_{\mathsf{PSigUE}_{\mathsf{T}},\mathcal{A}}(1^\lambda) \leq Adv^{\mathsf{CR}}_{\mathsf{H},\mathcal{B}_1}(1^\lambda) + Adv^{\mathsf{Unforg}}_{\mathsf{OT},\mathcal{B}_2}(1^\lambda) + Adv^{\mathsf{DDH}}_{\mathbb{G}_1}(\lambda) + Adv^{\mathsf{DDH}}_{\mathbb{G}_2}(\lambda).$$

Proof. We will prove the above theorem by hybrid games. The advantage of the adversary $\mathcal{A}$ against the IND-ENC security game is denoted by pr_i. For every i, we define the security game $\mathbf{G}_i$ as follows.

The game changes are described in the Figure 9.

Game $\mathbf{G}_0$: This is the original IND-ENC security game with challenge bit $\mathsf{b} = 0$.

Game $\mathbf{G}_1$: In $\mathbf{G}_1$, we change the challenge ciphertext. $\mathbf{Y}$ is now computed with the secret key x_1, x_2:

$$\mathbf{Y} = \mathbf{C}_0 + \mathbf{C}_1 \cdot x_1 + \mathbf{C}_2 \cdot x_2 + \mathsf{m}_0.$$

We can notice that since every crs_t is in the perfect sound setting, $\mathbf{Y}$ has exactly the same value as in $\mathbf{G}_0$. Therefore we have $|\mathsf{pr}_1 - \mathsf{pr}_0| = 0$.

Game $\mathbf{G}_2$: In this game, insteed of generating t and $(\mathsf{osk}, \mathsf{ovk})$ when the challenge oracle is happening. We generate t and $(\mathsf{osk}, \mathsf{ovk})$ at the beginning of the security game. Since this change is oblivious to the adversary we have $|\mathsf{pr}_2 - \mathsf{pr}_1| = 0$.

Game $\mathbf{G}_3$: In $\mathbf{G}_3$, we change $\mathbf{v}$ in generating the crs. In $\mathbf{G}_2$, $\mathbf{v}$ is linear dependent on $\mathbf{u}$ with coefficient td. However, when we prove the statement, we use $\mathbf{u}$ and $\mathbf{v}_\mathsf{t} = \mathbf{v} + \left[0\ \mathsf{t}\right]_2$ as the crs. This means, $\mathbf{u}$ and $\mathbf{v}_\mathsf{t}$ are linearly independent. Therefore, the Groth-Sahai proofs are used in perfect sound mode. In $\mathbf{G}_3$, we change $\mathbf{v}$ into $\mathbf{u} \cdot \mathsf{td} - \left[0\ \mathsf{t}\right]_2$. Thus, only the proof with t is in the perfect zero-knowledge mode, and other proofs are all in the perfect sound mode. The only difference between $\mathbf{G}_2$ and $\mathbf{G}_3$ is that $\mathbf{u}$ and $\mathbf{v}$ is a Diffie-Hellman tuple in $\mathbf{G}_2$. Therefore, we have $|\mathsf{pr}_3 - \mathsf{pr}_2| \leq \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(\lambda)$.

Game $\mathbf{G}_4$: In $\mathbf{G}_4$. we change the generation of proofs in the challenge oracle. Since $\mathbf{u}$ and $\mathbf{v}_\mathsf{t}$ are linearly independent with coefficient td, we can simulate the proofs with the trapdoor td. These changes do not modify the Com_r and π value. Therefore, we have $|\mathsf{pr}_4 - \mathsf{pr}_3| = 0$.

Game $\mathbf{G}_5$: In the previous game, we can notice that we do not need r to generate (Com_r, π). Therefore, we can modify the value of $\mathbf{C}$, such that $\mathbf{C}_1, \mathbf{C}_2$ is not linearly dependent random tuple. More precisely, we generate random values r, r', and compute $\mathbf{C} := \left(\mathbf{A}_0 \cdot r\ \mathbf{A}_1 \cdot r\ \mathbf{A}_2 \cdot (r + r')\right)^\top$. We will show in Lemma 3 that the probability in distinguishing $\mathbf{G}_4$ and $\mathbf{G}_5$ is bounded by the success probability in breaking SXDH assumption in $\mathbb{G}_1$. Therefore, we have $|\mathsf{pr}_5 - \mathsf{pr}_4| =\leq \mathsf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1}(\lambda)$.

Game $\mathbf{G}_6$: In this game, we introduce a bad event BadForg. BadForg happens if the adversary can successfully add a new ciphertext $\mathbf{c}$ different from the challenge

$\underline{\mathbf{Exp}^{\mathsf{IND\text{-}ENC}_b}_{\mathsf{PSigUE}_\mathsf{T},\mathcal{A}}(1^\lambda)}$

$\mathsf{parG} \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$
$\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(\mathsf{parG})$
$\mathbf{u} = \begin{bmatrix} u_1 & u_2 \end{bmatrix}_2^\mathsf{T}$
$\mathsf{td} \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$
$(\mathsf{osk}, \mathsf{ovk}) \xleftarrow{\$} \mathsf{OT.KGen}(1^\lambda)$ // $\mathbf{G}_{2\text{-}7}$
$\mathsf{t} \xleftarrow{\$} \mathsf{H}(\mathsf{ovk}) \in \mathbb{Z}_\mathsf{p}$ // $\mathbf{G}_{2\text{-}7}$
$\mathbf{v} := \mathbf{u} \cdot \mathsf{td} = \begin{bmatrix} u_1 \cdot \mathsf{td} & u_2 \cdot \mathsf{td} \end{bmatrix}_2^\mathsf{T}$ // $\mathbf{G}_{0\text{-}2}$
$\mathbf{v} := \begin{bmatrix} u_1 \cdot \mathsf{td} & u_2 \cdot \mathsf{td} - \mathsf{t} \end{bmatrix}_2^\mathsf{T}$ // $\mathbf{G}_{2\text{-}7}$
$\mathsf{crs} := (\mathbf{u}, \mathbf{v})$
$a_0, a_1, a_2, b, x_1, x_2 \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$
$\mathbf{A} := \begin{bmatrix} a_0 & a_1 & a_2 \end{bmatrix}_1^\mathsf{T}$; $\mathbf{B} := \begin{bmatrix} b \end{bmatrix}_1$
$\mathbf{X} := \begin{pmatrix} 1 & x_1 & x_2 \end{pmatrix} \cdot \mathbf{A}$
$\mathsf{pk}_0 := (\mathbf{A}, \mathbf{B}, \mathbf{X})$; $\mathsf{sk}_0 := (x_1, x_2)$
$(\mathsf{ssk}, \mathsf{svk}) \xleftarrow{\$} \mathsf{SKGen}(\mathsf{pp})$
$\mathcal{L}_{\mathsf{CorrK}}, \mathcal{L}_{\mathsf{Safe}}, \mathcal{L}_{\mathsf{CorrT}} := \emptyset$
$\mathsf{b}' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{IND\text{-}ENC}}}(\mathsf{pp}, \mathsf{pk}_0, \mathsf{svk})$
$(\mathcal{L}^*_{\mathsf{CorrK}}, \mathcal{L}^*_{\mathsf{Safe}})$
  $:= \mathsf{Expand}(\mathcal{L}_{\mathsf{CorrK}}, \mathcal{L}_{\mathsf{Safe}}, \mathcal{L}_{\mathsf{CorrT}})$
if $\mathcal{L}^*_{\mathsf{CorrK}} \cap \mathcal{L}^*_{\mathsf{Safe}} \neq \emptyset$
  $\mathsf{b}' \xleftarrow{\$} \{0, 1\}$
return b'

$\underline{\mathbf{Alg}\ \mathcal{O}^{\mathsf{IND\text{-}ENC}}_{\mathsf{Chall_b}}(\mathsf{e}, \mathsf{m}_0)}$

$\mathsf{m}_1 \xleftarrow{\$} \mathcal{M}$
Check $|\mathsf{m}_0| \neq |\mathsf{m}_1|$
$\mathcal{L}_{\mathsf{Safe}} := \mathcal{L}_{\mathsf{Safe}} \cup \{i\}$
if phase $\neq 1$
  phase $:= 1$
  $(\mathsf{osk}, \mathsf{ovk}) \xleftarrow{\$} \mathsf{OT.KGen}(1^\lambda)$ // $\mathbf{G}_0$
  $\mathsf{t} \xleftarrow{\$} \mathsf{H}(\mathsf{ovk}) \in \mathbb{Z}_\mathsf{p}$ // $\mathbf{G}_0$
  $r \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$
  $\mathbf{C} := \mathbf{A} \cdot r$ // $\mathbf{G}_{0\text{-}4}$
  $r, r' \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$ // $\mathbf{G}_{5\text{-}7}$
  $\mathbf{C} := \begin{pmatrix} \mathbf{A}_0 \cdot r \\ \mathbf{A}_1 \cdot r \\ \mathbf{A}_2 \cdot (r + r') \end{pmatrix}$ // $\mathbf{G}_{5\text{-}7}$
  $\mathbf{D} := \mathbf{B} \cdot r$
  $\mathbf{Y} := \mathbf{X} \cdot r + \mathsf{m}_0$ // $\mathbf{G}_0$
  $\mathbf{Y} := \mathbf{C}_0 + x_1 \mathbf{C}_1 + x_2 \mathbf{C}_2 + \mathsf{m}_0$ // $\mathbf{G}_{1\text{-}6}$
  $\mathbf{Y} := \mathbf{X} \cdot r + \mathsf{m}_1$ // $\mathbf{G}_7$
  $\mathbf{v}_\mathsf{t} := \mathbf{v} + \begin{bmatrix} 0 & \mathsf{t} \end{bmatrix}_2$
  $\mathsf{crs}_\mathsf{t} := (\mathbf{u}, \mathbf{v}_\mathsf{t})$
  $s \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$
  $\mathsf{Com}_r := \mathbf{u} \cdot s + \mathbf{v}_\mathsf{t} \cdot r$ // $\mathbf{G}_{0\text{-}3}$
  $\pi_\mathbf{A} := \begin{pmatrix} \pi_{a_0} \\ \pi_{a_1} \\ \pi_{a_2} \end{pmatrix} := \mathbf{A} \cdot s$ // $\mathbf{G}_{0\text{-}3}$
  $\pi_\mathbf{B} := \mathbf{B} \cdot s$ // $\mathbf{G}_{0\text{-}3}$
  $\mathsf{Com}_r := \mathbf{u} \cdot s$ // $\mathbf{G}_{4\text{-}7}$
  $\pi_\mathbf{A} := \mathbf{A} \cdot s - \mathbf{C} \cdot \mathsf{td}$ // $\mathbf{G}_{4\text{-}7}$
  $\pi_\mathbf{B} := \mathbf{B} \cdot s - \mathbf{C} \cdot \mathsf{td}$ // $\mathbf{G}_{4\text{-}7}$
  $\pi := (\mathsf{Com}_r, \pi_\mathbf{A}, \pi_\mathbf{B})$
  $\sigma_\mathsf{OT} \xleftarrow{\$} \mathsf{OT.Sig}(\mathsf{osk}, (\mathbf{D}, \mathbf{Y}, \mathsf{Com}_r))$
  $\tilde{\mathbf{c}}_\mathsf{e} := (\mathsf{ovk}, \mathbf{C}, \mathbf{D}, \mathbf{Y}, \pi, \sigma_\mathsf{OT})$
  $\tilde{\sigma}_\mathsf{e} \xleftarrow{\$} \mathsf{Sig}(\mathsf{ssk}, \mathsf{pk}_\mathsf{e}, \tilde{\mathbf{c}}_\mathsf{e})$
  return $(\tilde{\mathbf{c}}_\mathsf{e}, \tilde{\sigma}_\mathsf{e})$
else
  return $(\tilde{\mathbf{c}}_\mathsf{e}, \tilde{\sigma}_\mathsf{e})$

Fig. 9. Hybrid security game for $\mathbf{Exp}^{\mathsf{IND\text{-}ENC}}_{\mathsf{PSigUE},\mathcal{A}}$ security game, where $\mathcal{O}_{\mathsf{IND\text{-}ENC}} = (\mathcal{O}_{\mathsf{Sig}}, \mathcal{O}_{\mathsf{Insert}}, \mathcal{O}_{\mathsf{Next}}, \mathcal{O}_{\mathsf{CorrK}}, \mathcal{O}_{\mathsf{CorrT}}, \mathcal{O}_{\mathsf{Upd}}, \mathcal{O}^{\mathsf{IND\text{-}ENC}}_{\mathsf{Chall_b}})$. Here we only change the security game and the oracle $\mathcal{O}^{\mathsf{IND\text{-}ENC}}_{\mathsf{Chall_b}}$. Therefore, we omit other oracles here.

ciphertext but with the same tag t. To analyse the probability of the bad event, we can distinguish the following three cases:

1. $\mathbf{c}$ has the same t as the challenge ciphertext, but ovk is different.
2. $\mathbf{c}$ has the same $(\mathsf{t}, \mathsf{ovk})$ as the challenge ciphertext, but $(\mathbf{D}, \mathbf{Y}, \mathsf{Com}_r, \sigma_\mathsf{OT})$ is different.

3. $\mathbf{c}$ has the same $(\mathsf{t}, \mathsf{ovk}, \mathbf{D}, \mathbf{Y}, \mathsf{Com}_r, \sigma_{\mathsf{OT}})$ as the challenge ciphertext, but $(\mathbf{C}, \pi)$ is different.

We can give the probability of the different cases

1. If this happens, we have found a hash function collision. Therefore, the probability is bounded by $\mathsf{Adv}^{\mathsf{CR}}_{\mathsf{H},\mathcal{B}_1}(1^\lambda)$
2. Since ovk in $\mathbf{c}$ is the same as the one-time verification key in the challenge ciphertext. This leads to a forgery of the underlying one-time signature scheme with probability $\mathsf{Adv}^{\mathsf{Unforg}}_{\mathsf{OT},\mathcal{B}_2}(1^\lambda)$.
3. In this case, t in $\mathbf{c}$ is different from the t of the challenge ciphertext. Therefore, $\mathsf{crs}_{\mathsf{t}}$ is perfectly sound, and given $(\mathsf{Com}_r, \mathsf{crs}_{\mathsf{t}}, \mathbf{A})$, there is only one possible choice of $\mathbf{C}, \pi_{\mathbf{A}}$ pass the verification test. The probability of this case is 0.

Summmering all the above cases, we have

$$\Pr[\mathsf{BadForg}] \leq \mathsf{Adv}^{\mathsf{CR}}_{\mathsf{H},\mathcal{B}_1}(1^\lambda) + \mathsf{Adv}^{\mathsf{Unforg}}_{\mathsf{OT},\mathcal{B}_2}(1^\lambda).$$

By including the aborting probability of the bad event $\mathsf{BadForg}$, we have

$$|\mathsf{pr}_6 - \mathsf{pr}_5| \leq \mathsf{Adv}^{\mathsf{CR}}_{\mathsf{H},\mathcal{B}_1}(1^\lambda) + \mathsf{Adv}^{\mathsf{Unforg}}_{\mathsf{OT},\mathcal{B}_2}(1^\lambda).$$

Game $\mathbf{G}_7$: In this game we change how the challenge ciphertext $\mathbf{Y}$ is generated. Insteed of encrypting m_0, we encrypt a random message m_1 in this game.

To see that the difference between $\mathbf{G}_6$ and $\mathbf{G}_7$ is oblivious for the adversary, we recall that in this game all proofs are perfectly sound except the one in the challenge ciphertext. Moreover, if we decrypt the challenge ciphertext, we have

$$\begin{aligned}\mathbf{Y} &= \mathbf{X} \cdot r + \mathsf{m}_0 \\ &= (\mathbf{A}_0 + \mathbf{A}_1 \cdot x_1 + \mathbf{A}_2 \cdot x_2) \cdot r + \mathsf{m}_0 \\ &= \mathbf{C}_0 + \mathbf{C}_1 \cdot x_1 + \mathbf{C}_2 \cdot x_2 + \mathsf{m}_0 - \mathbf{A}_2 \cdot x_2 \cdot r'\end{aligned}$$

Moreover, we can notice that given $\mathbf{X}$, there is exponentially many pairs (x_1, x_2) such that $\mathbf{X} = \mathbf{A}_0 + \mathbf{A}_1 \cdot x_1 + \mathbf{A}_2 \cdot x_2$. This means that given only $\mathbf{X}, \mathbf{A}$, there is no information of x_2 leaked for the adversary. Moreover, the proofs in every non-challenge ciphertext are in the perfect sound setting. Thus, $\mathbf{C}_1, \mathbf{C}_2$ are linear in all non-challenge ciphertext which does not leak any information of x_2 by decrypting. In summary, x_2 is completely oblivious to the adversary. Thus $\mathbf{A}_2 \cdot r' \cdot x_2$ is uniformly random for the adversary, and m_1 and $\mathsf{m}_0 - \mathbf{A}_2 \cdot r' \cdot x_2$ have the same distribution for the adversary. Therefore, we have

$$|\mathsf{pr}_7 - \mathsf{pr}_6| = 0.$$

We can notice that from the adversary's view, $\mathbf{G}_7$ is identical to the security game with $\mathsf{b} = 1$. Therefore, by taking an union bound over all hybrid games we have

$$\mathsf{Adv}^{\mathsf{IND\text{-}ENC}}_{\mathsf{PSigUE}_{\mathsf{T}},\mathcal{A}}(1^\lambda) \leq \mathsf{Adv}^{\mathsf{CR}}_{\mathsf{H},\mathcal{B}_1}(1^\lambda) + \mathsf{Adv}^{\mathsf{Unforg}}_{\mathsf{OT},\mathcal{B}_2}(1^\lambda) + \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_1}(\lambda) + \mathsf{Adv}^{\mathsf{DDH}}_{\mathbb{G}_2}(\lambda).$$

□

It remains to prove that $\mathbf{G}_4$ and $\mathbf{G}_5$ are computationally indistinguishable under the SXDH assumption.

Lemma 3 ($\mathbf{G}_4 \approx_{\mathsf{SXDH}} \mathbf{G}_5$). *Let $\mathcal{A}$ be a PPT adversary in distinguishing $\boldsymbol{G}_4$ and $\boldsymbol{G}_5$, we can construct an adversary $\mathcal{B}$ that breaks SXDH assumption. Formally, we have*

$$|\mathsf{pr}_4 - \mathsf{pr}_5| \leq \mathsf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathcal{B}}(1^\lambda).$$

Proof. Given an SXDH challenge tuple $\mathsf{CH} = (\left[f\right]_1, \left[g\right]_1, \left[h\right]_1) \in \mathbb{G}_1^3$. The PPT algorithm $\mathcal{B}$ will use the distinguisher $\mathcal{A}$ between $\mathbf{G}_4$ and $\mathbf{G}_5$ decide whether CH is an SXDH tuple.

The adversary $\mathcal{B}$ proceeds exactly as in $\mathbf{G}_4$ except that $\mathbf{A}_2 := \left[a_2 \cdot g\right]_1$, and the challenge ciphertext is generated as

$$\mathbf{C}_0 = \left[a_0 \cdot f\right]_1, \qquad \mathbf{C}_1 = \left[a_1 \cdot f\right]_1, \qquad \mathbf{C}_2 = \left[a_1 \cdot h\right]_1.$$

We can notice that if $h = f \cdot g$ then we have $\log_{\mathbf{A}_1} \mathbf{C_1} = f = \log_{\mathbf{A}_2} \mathbf{C_2}$, and the simulated security game by $\mathcal{B}$ is exactly the same as $\mathbf{G}_4$ for the adversary. On the other hand, if $h \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$, then the simulated security game is the same as $\mathbf{G}_4$ for $\mathcal{A}$. Thus, we have

$$|\mathsf{pr}_4 - \mathsf{pr}_5| \leq \mathsf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathcal{B}}(1^\lambda).$$

□

We give the following theorem of the integrity property

Theorem 6 (Integrity). *Let $\Sigma = (\mathsf{SKGen}, \mathsf{Sig}, \mathsf{Ver})$ be a signature scheme with $\mathsf{UF\text{-}CMA}$ security. For all PPT adversary $\mathcal{A}$ against the integrity security game of $\mathsf{PSigUE}_\mathsf{T}$ described in Figure 8, there exists an PPT adversary $\mathcal{B}$ such that*

$$\mathsf{Adv}^{\mathit{Int}}_{\mathsf{PSigUE}_\mathsf{T},\mathcal{A}}(1^\lambda) \leq \mathsf{Adv}^{\mathsf{UF\text{-}CMA}}_{\Sigma,\mathcal{B}}(1^\lambda).$$

Proof. We will reduce the integrity of $\mathsf{PSigUE}_\mathsf{T}$ scheme given in Figure 8 to the underlying signature scheme Σ. We recall that the core part of $\mathsf{PSigUE}_\mathsf{T}$ ciphertext is of the form $(\mathbf{C}, \mathbf{D}, \mathbf{Y}) = (\mathbf{A} \cdot r, \mathbf{B} \cdot r, \mathbf{X} \cdot r + \mathsf{m})$. Since crs is in the perfect soundness mode for every tag t, for every valid ciphertext with public key $(\mathbf{A}, \mathbf{B}, \mathbf{X})$ we have $\log_\mathbf{A} \mathbf{C} = \log_\mathbf{B} \mathbf{D}$. For any adversary that breaks the integrity of $\mathsf{PSigUE}_\mathsf{T}$, it must also get a forgery of the underlying signature Σ scheme for the new message $(\mathbf{D}', \mathbf{Y}')$. Therefore we have the desired result. □

References

1. Alwen, J., Coretti, S., Dodis, Y., Tselekounis, Y.: Security analysis and improvements for the IETF MLS standard for group messaging. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 248–277. Springer, Heidelberg (Aug 2020)
2. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Heidelberg (May / Jun 2006)
3. Blazy, O., Fuchsbauer, G., Pointcheval, D., Vergnaud, D.: Signatures on randomizable ciphertexts. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 403–422. Springer, Heidelberg (Mar 2011)
4. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (Aug 2004)
5. Boneh, D., Eskandarian, S., Kim, S., Shih, M.: Improving speed and security in updatable encryption schemes. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 559–589. Springer, Heidelberg (Dec 2020)
6. Boneh, D., Lewi, K., Montgomery, H.W., Raghunathan, A.: Key homomorphic PRFs and their applications. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part I. LNCS, vol. 8042, pp. 410–428. Springer, Heidelberg (Aug 2013)
7. Boyd, C., Davies, G.T., Gjøsteen, K., Jiang, Y.: Fast and secure updatable encryption. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 464–493. Springer, Heidelberg (Aug 2020)
8. Chen, L., Li, Y., Tang, Q.: CCA updatable encryption against malicious re-encryption attacks. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 590–620. Springer, Heidelberg (Dec 2020)
9. Cini, V., Ramacher, S., Slamanig, D., Striecks, C., Tairi, E.: Updatable signatures and message authentication codes. In: Garay, J. (ed.) PKC 2021, Part I. LNCS, vol. 12710, pp. 691–723. Springer, Heidelberg (May 2021)
10. Cramer, R., Shoup, V.: A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In: Krawczyk, H. (ed.) CRYPTO'98. LNCS, vol. 1462, pp. 13–25. Springer, Heidelberg (Aug 1998)
11. Davidson, A., Deo, A., Lee, E., Martin, K.: Strong post-compromise secure proxy re-encryption. In: Jang-Jaccard, J., Guo, F. (eds.) ACISP 19. LNCS, vol. 11547, pp. 58–77. Springer, Heidelberg (Jul 2019)
12. Dodis, Y., Karthikeyan, H., Wichs, D.: Updatable public key encryption in the standard model. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part III. LNCS, vol. 13044, pp. 254–285. Springer, Heidelberg (Nov 2021)
13. Fuchsbauer, G., Kamath, C., Klein, K., Pietrzak, K.: Adaptively secure proxy re-encryption. In: Lin, D., Sako, K. (eds.) PKC 2019, Part II. LNCS, vol. 11443, pp. 317–346. Springer, Heidelberg (Apr 2019)
14. Galteland, Y.J., Pan, J.: Backward-leak uni-directional updatable encryption from (homomorphic) public key encryption. In: Public Key Cryptography - PKC 2023 (2023)
15. Groth, J., Sahai, A.: Efficient non-interactive proof systems for bilinear groups. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 415–432. Springer, Heidelberg (Apr 2008)
16. Hofheinz, D., Jager, T.: Tightly secure signatures and public-key encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 590–607. Springer, Heidelberg (Aug 2012)

17. Hofheinz, D., Kiltz, E.: Programmable hash functions and their applications. In: Wagner, D. (ed.) CRYPTO 2008. LNCS, vol. 5157, pp. 21–38. Springer, Heidelberg (Aug 2008)
18. Jiang, Y.: The direction of updatable encryption does not matter much. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part III. LNCS, vol. 12493, pp. 529–558. Springer, Heidelberg (Dec 2020)
19. Jost, D., Maurer, U., Mularczyk, M.: Efficient ratcheting: Almost-optimal guarantees for secure messaging. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 159–188. Springer, Heidelberg (May 2019)
20. Klooß, M., Lehmann, A., Rupp, A.: (R)CCA secure updatable encryption with integrity protection. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part I. LNCS, vol. 11476, pp. 68–99. Springer, Heidelberg (May 2019)
21. Knapp, J., Quaglia, E.A.: Epoch confidentiality in updatable encryption. In: ProvSec. Lecture Notes in Computer Science, vol. 13600, pp. 60–67. Springer (2022)
22. Lehmann, A., Tackmann, B.: Updatable encryption with post-compromise security. In: Nielsen, J.B., Rijmen, V. (eds.) EUROCRYPT 2018, Part III. LNCS, vol. 10822, pp. 685–716. Springer, Heidelberg (Apr / May 2018)
23. Libert, B., Peters, T., Qian, C.: Structure-preserving chosen-ciphertext security with shorter verifiable ciphertexts. In: Fehr, S. (ed.) PKC 2017, Part I. LNCS, vol. 10174, pp. 247–276. Springer, Heidelberg (Mar 2017)
24. Miao, P., Patranabis, S., Watson, G.: Unidirectional updatable encryption and proxy re-encryption from ddh. In: Public Key Cryptography - PKC 2023 (2023)
25. Nishimaki, R.: The direction of updatable encryption does matter. In: Hanaoka, G., Shikata, J., Watanabe, Y. (eds.) PKC 2022, Part II. LNCS, vol. 13178, pp. 194–224. Springer, Heidelberg (Mar 2022)
26. Peikert, C., Waters, B.: Lossy trapdoor functions and their applications. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 187–196. ACM Press (May 2008)
27. Shacham, H.: A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants. IACR Cryptol. ePrint Arch. p. 74 (2007)
28. Slamanig, D., Striecks, C.: Puncture 'em all: Updatable encryption with no-directional key updates and expiring ciphertexts. Cryptology ePrint Archive, Report 2021/268 (2021), https://eprint.iacr.org/2021/268
29. Waters, B.R.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (May 2005)

Supplementary Material

A Instantiation of $\mathcal{T}$-UNIZK from SXDH assumption

We give the explicit construction of $\mathcal{T}$-UNIZK for the relation $\mathsf{R}_{\mathbf{A},\mathbf{h}}$ including the update and re-randomization algorithms in Figure 10.

We can notice that the updatability and the rerandomizability are straightforward.

B Detailed Proofs for PSigUE_2

B.1 wIND-ENC Security of PSigUE_2

The first step of the security proof, is to guess the challenge epochs $(\mathsf{e}_\mathsf{L}, \ldots, \mathsf{e}_\mathsf{R})$. Since in wIND-ENC security game, the adversary $\mathcal{A}$ does not allow to query the key corruption oracle between the epoch e_L and e_R. The success probability in guessing the correct starting and ending challenge epochs e_L and e_R is $1/\mathsf{n}^2$. Then, We prove the wIND-ENC security PSigUE_2 via a sequence of security games.

Then, We prove the wIND-ENC security PSigUE_2 via a sequence of security games.

Game $\mathbf{G}_0$: The $\mathbf{G}_0$ is the wIND-ENC security game with $\mathsf{b} = 0$ while guessing correctly the corruption epochs $\{\mathsf{e}_\mathsf{L}, \ldots, \mathsf{e}_\mathsf{R}\}$. Therefore, we have

$$\mathsf{pr}_0 = \frac{1}{\mathsf{n}^2} \cdot \mathsf{Adv}^{\mathsf{wIND\text{-}ENC}}_{\mathsf{PSigUE}_2,\mathcal{A}}(1^\lambda).$$

Game $\mathbf{G}_1$: In $\mathbf{G}_1$, we change the crs generation of Π. The crs is generated as $(\mathsf{crs}, \mathsf{SimK}) \xleftarrow{\$} \Pi.\mathsf{SimSetup}(1^\lambda)$. Since the only difference between $\mathbf{G}_0$ and $\mathbf{G}_1$ is the fact that all proofs are generated with SimK, by the zero-knowledge property, we have

$$|\mathsf{pr}_1 - \mathsf{pr}_0| \leq \mathsf{Adv}^{\mathsf{ZK}}_{\mathcal{B}_1,\Pi}(1^\lambda).$$

Game $\mathbf{G}_2$: We further modify the public parameter. In $\mathbf{G}_2$, the crs_DM of dual-mode zero-knowledge proof is generated in the perfect sound mode together with an extraction key ExtK. Note that the only difference between $\mathbf{G}_1$ and $\mathbf{G}_2$ is the mode switching for Π_DM. Thus, we have:

$$|\mathsf{pr}_2 - \mathsf{pr}_1| \leq \mathsf{Adv}^{\mathsf{Dual\text{-}Mode}}_{\mathcal{B}_2,\Pi_\mathsf{DM}}(1^\lambda).$$

To analyse the success probability of the adversary $\mathcal{A}$ in $\mathbf{G}_2$, we will constract an adversary $\mathcal{B}_3$ that simulates $\mathbf{G}_2$ to $\mathcal{A}$ while using $\mathcal{A}$ to help us to solve

Alg $\mathsf{Setup}(1^\lambda)$:

$\mathbf{U} \xleftarrow{\$} \mathbb{G}_1^{2\times 2}$
$\hat{\mathbf{U}} \xleftarrow{\$} \mathbb{G}_2^{2\times 2}$
return $\mathsf{crs} = (\mathbf{U}, \hat{\mathbf{U}})$

Alg $\mathsf{Prove}(\mathsf{crs}, (r, \mathsf{m}))$:

parse $\begin{pmatrix}\mathbf{A}_0\\ \mathbf{A}_1\end{pmatrix} =: \mathbf{A}$
$\hat{\mathbf{R}}_r \xleftarrow{\$} \mathbb{Z}_\mathsf{p}^2$
$\hat{\mathcal{C}}_r := \begin{bmatrix}0\\ r\end{bmatrix}_2 + \hat{\mathbf{U}}\hat{\mathbf{R}}_r$
$\pi_r := \mathbf{A}_0 \cdot \hat{\mathbf{R}}_r^\intercal \in \mathbb{G}_1^{1\times 2}$
for $i = 1$ **to** ℓ
  $\mathbf{R}_{\mathsf{m}_i}, \hat{\mathbf{R}}_{\mathsf{m}_i} \xleftarrow{\$} \mathbb{Z}_\mathsf{p}^2$
  $\mathcal{C}_{\mathsf{m}_i} := \begin{bmatrix}0\\ \mathsf{m}_i\end{bmatrix}_1 + \mathbf{U}\mathbf{R}_{\mathsf{m}_i}$
  $\hat{\mathcal{C}}_{\mathsf{m}_i} := \begin{bmatrix}0\\ \mathsf{m}_i\end{bmatrix}_2 + \hat{\mathbf{U}}\hat{\mathbf{R}}_{\mathsf{m}_i}$
  $\mathbf{T}_i, \mathbf{S}_i \xleftarrow{\$} \mathbb{Z}_\mathsf{p}^{2\times 2}$
  $\pi_{\mathsf{m}_i} := \begin{bmatrix}0\\ \mathsf{m}_i\end{bmatrix}_1^\intercal \hat{\mathbf{R}}_{\mathsf{m}_i}^\intercal + \mathbf{U}\mathbf{T}_i$
  $\theta_{\mathsf{m}_i} := \mathbf{R}_{\mathsf{m}_i}\begin{bmatrix}\mathbf{0} & \mathsf{m}_i - 1\end{bmatrix}_2 + \mathbf{R}_{\mathsf{m}_i}\hat{\mathbf{R}}_{\mathsf{m}_i}^\intercal\hat{\mathbf{U}}^\intercal - \mathbf{T}_i\hat{\mathbf{U}}^\intercal$
  $\chi_{\mathsf{m}_i} := -\begin{bmatrix}0\\ 1\end{bmatrix}_1 \hat{\mathbf{R}}_{\mathsf{m}_i}^\intercal + \mathbf{U}\mathbf{S}_i^\intercal$
  $\phi_{\mathsf{m}_i} := \mathbf{R}_{\mathsf{m}_i}\begin{bmatrix}0 & 1\end{bmatrix}_2 - \mathbf{S}_i\hat{\mathbf{U}}^\intercal$
$\pi_\mathbf{C} := \mathbf{A}_1\hat{\mathbf{R}}_r^\intercal + \sum_{i=1}^\ell \mathsf{u}_i\hat{\mathbf{R}}_{\mathsf{m}_i}^\intercal$
return $\pi := (\hat{\mathcal{C}}_r, \{\mathcal{C}_{\mathsf{m}_i}, \hat{\mathcal{C}}_{\mathsf{m}_i}\}_{i=1}^\ell, \pi_r, \{\pi_{\mathsf{m}_i}, \theta_{\mathsf{m}_i}, \chi_{\mathsf{m}_i}, \phi_{\mathsf{m}_i}\}_{i=1}^\ell, \pi_\mathbf{C})$

Alg $\mathsf{Upd}(\pi, \Delta_\mathsf{e}, r')$:

$\pi_r' := \Delta_\mathsf{e} \cdot \pi_r$
$\hat{\mathcal{C}}_r' := \hat{\mathcal{C}}_r + \begin{bmatrix}0\\ r'\end{bmatrix}_2$
return $\pi := (\hat{\mathcal{C}}_r', \{\mathcal{C}_{\mathsf{m}_i}, \hat{\mathcal{C}}_{\mathsf{m}_i}\}_{i=1}^\ell, \pi_r', \{\pi_{\mathsf{m}_i}, \theta_{\mathsf{m}_i}, \chi_{\mathsf{m}_i}, \phi_{\mathsf{m}_i}\}_{i=1}^\ell, \pi_\mathbf{C})$

Alg $\mathsf{Ver}(\mathsf{crs}, \pi)$:

parse $(\pi_\mathbf{C}, \hat{\mathcal{C}}_r, \{\mathcal{C}_{\mathsf{m}_i}, \hat{\mathcal{C}}_{\mathsf{m}_i}\}_{i=1}^\ell, \pi_r, \{\pi_{\mathsf{m}_i}, \theta_{\mathsf{m}_i}, \chi_{\mathsf{m}_i}, \phi_{\mathsf{m}_i}\}_{i=1}^\ell) =: \pi$
Check $\mathbf{A}_0 \bullet \hat{\mathcal{C}}_r^\intercal = \begin{bmatrix}0 & \mathbf{C}_0\end{bmatrix}_1 \bullet \begin{bmatrix}\mathbf{Id}_2\end{bmatrix}_2 + \pi_r \bullet \hat{\mathbf{U}}^\intercal$
for $i = 1$ **to** ℓ
  Check $\mathcal{C}_{\mathsf{m}_i} \bullet (\hat{\mathcal{C}}_{\mathsf{m}_i} - \begin{bmatrix}0\\ 1\end{bmatrix}_2)^\intercal = \mathbf{U} \bullet \theta_{\mathsf{m}_i} + \pi_{\mathsf{m}_i} \bullet \hat{\mathbf{U}}^\intercal$
  Check $\mathcal{C}_{\mathsf{m}_i} \bullet \begin{bmatrix}0\\ 1\end{bmatrix}_2^\intercal - \begin{bmatrix}0\\ 1\end{bmatrix}_1 \bullet \hat{\mathcal{C}}_{\mathsf{m}_i}^\intercal = \mathbf{U} \bullet \phi_{\mathsf{m}_i} + \chi_{\mathsf{m}_i} \bullet \hat{\mathbf{U}}^\intercal$
Check $\mathbf{A}_1 \bullet \hat{\mathcal{C}}_r^\intercal + \sum_{i=0}^\ell \mathbf{h}_i \bullet \hat{\mathcal{C}}_{\mathsf{m}_i}^\intercal = \begin{bmatrix}0 & \mathbf{C}_1\end{bmatrix}_1 \bullet \begin{bmatrix}\mathbf{Id}_2\end{bmatrix}_2 + \pi_\mathbf{C} \bullet \hat{\mathbf{U}}^\intercal$
return 1

Alg $\mathsf{Rand}(\pi)$:

$\hat{\mathbf{R}}_r', \{\mathbf{R}_{\mathsf{m}_i}', \hat{\mathbf{R}}_{\mathsf{m}_i}', \mathbf{T}_i', \mathbf{S}_i'\}_{i=1}^\ell \xleftarrow{\$} \mathbb{Z}_\mathsf{p}^2$
$\hat{\mathcal{C}}_r' := \hat{\mathcal{C}}_r + \hat{\mathbf{U}}\hat{\mathbf{R}}_r'$
$\pi_r' := \pi_r + \mathbf{A}_0 \cdot \hat{\mathbf{R}}_r'^\intercal$
for $i = 1$ **to** ℓ
  $\mathcal{C}_{\mathsf{m}_i}' := \mathcal{C}_{\mathsf{m}_i} + \mathbf{U}\mathbf{R}_{\mathsf{m}_i}$
  $\hat{\mathcal{C}}_{\mathsf{m}_i}' := \hat{\mathcal{C}}_{\mathsf{m}_i} + \hat{\mathbf{U}}\hat{\mathbf{R}}_{\mathsf{m}_i}'$
  $\pi_{\mathsf{m}_i}' := \pi_{\mathsf{m}_i} + \mathcal{C}_{\mathsf{m}_i}\hat{\mathbf{R}}_{\mathsf{m}_i}'^\intercal + \mathbf{U}\mathbf{R}_{\mathsf{m}_i}'\hat{\mathbf{R}}_{\mathsf{m}_i}'^\intercal + \mathbf{U}\mathbf{T}_i'$
  $\theta_{\mathsf{m}_i}' := \theta_{\mathsf{m}_i} + \mathbf{R}_{\mathsf{m}_i}'\hat{\mathcal{C}}_{\mathsf{m}_i}^\intercal - \mathbf{R}_{\mathsf{m}_i}'\begin{bmatrix}0 & 1\end{bmatrix}_2 - \mathbf{T}_i'\hat{\mathbf{U}}^\intercal$
  $\chi_{\mathsf{m}_i}' := \chi_{\mathsf{m}_i} - \begin{bmatrix}0\\ 1\end{bmatrix}_1 \hat{\mathbf{R}}_{\mathsf{m}_i}'^\intercal + \mathbf{U}\mathbf{S}_i'^\intercal$
  $\phi_{\mathsf{m}_i}' := \phi_{\mathsf{m}_i} + \mathbf{R}_{\mathsf{m}_i}'\begin{bmatrix}0 & 1\end{bmatrix}_2 - \mathbf{S}_i'\hat{\mathbf{U}}^\intercal$
$\pi_\mathbf{C}' := \pi_\mathbf{C} + \mathbf{A}_1\hat{\mathbf{R}}_r'^\intercal + \sum_{i=1}^\ell \mathsf{u}_i\hat{\mathbf{R}}_{\mathsf{m}_i}'^\intercal$
return $\pi' := (\hat{\mathcal{C}}_r', \{\mathcal{C}_{\mathsf{m}_i}', \hat{\mathcal{C}}_{\mathsf{m}_i}'\}_{i=1}^\ell, \pi_r', \{\pi_{\mathsf{m}_i}', \theta_{\mathsf{m}_i}', \chi_{\mathsf{m}_i}', \phi_{\mathsf{m}_i}'\}_{i=1}^\ell, \pi_\mathbf{C}')$

Fig. 10. The updatable Groth-Sahai proof $\Pi = (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Ver}, \mathsf{Upd}, \mathsf{Rand})$.

the underlying SXDH assumption. The detailed construction of $\mathcal{B}_3$ is given in Figure 11.

We recall that by the SXDH assumption in $\mathbb{G}_1$ is:

$$(\{[x^\star]_1, [r^\star]_1, [x^\star \cdot r^\star]_1\}_{i=1}^k) \approx_c (\{[x^\star]_1, [r^\star]_1, [s^\star]_1\}_{i=1}^k),$$

where $x^\star, r^\star, s^\star \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$. We define the SXDH challenge set in $\mathbb{G}_1$ as $\mathsf{CH} = (\left[x^\star\right]_1, \left[r^\star\right]_1, \left[s^\star\right]_1)$.

$\mathcal{B}_3(\mathsf{CH}, \mathsf{e_L}, \mathsf{e_R}, \mathcal{A})$:
01 $\mathsf{parG} \xleftarrow{\$} \mathsf{GGen}(1^\lambda)$
02 $(\mathsf{crs}, \mathsf{SimK}) \xleftarrow{\$} \Pi.\mathsf{SimSetup}(1^\lambda)$
03 $(\mathsf{crs_{DM}}, \mathsf{ExtK}) \xleftarrow{\$} \Pi_{\mathsf{DM}}.\mathsf{Setup}_{\mathsf{Snd}}(1^\lambda)$
04 $\mathbf{h} \xleftarrow{\$} \mathbb{G}_1^{\ell+1}$
05 $\mathsf{pp} := (\mathsf{parG}, \mathsf{crs}, \mathsf{crs_{DM}}, \mathbf{h})$
06 $x_0, x_{\mathsf{e_R}+1} \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$
07 $\mathsf{pk}_0 := \left[x_0\right]_1$; $\mathsf{sk}_0 := x_0$
08 $\mathsf{pk}_{\mathsf{e_R}+1} \xleftarrow{\$} \left[x_{\mathsf{e_R}+1}\right]_1$
09 $\mathsf{sk}_{\mathsf{e_R}+1} := x_{\mathsf{e_R}+1}$
10 **for** $i \in \{1, \ldots, \mathsf{e_L} - 1\} \cup \{\mathsf{e_R} + 2, \ldots, \mathsf{n}\}$
11 $\quad x_i \xleftarrow{\$} Zp$
12 $\quad \mathsf{pk}_i := \left[x_i\right]_1$
13 $\quad \mathsf{sk}_i := x_i$; $\Delta_i := x_i \cdot x_{i-1}^{-1}$
14 $\mathsf{pk}_{\mathsf{e_L}} := \left[x_{\mathsf{e_L}}\right]_1$
15 **for** $i = \mathsf{e_L} + 1$ **to** $\mathsf{e_R}$
16 $\quad \Delta_i \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$; $\mathsf{pk}_i = \Delta_i \cdot \mathsf{pk}_{i-1}$
17 $(\mathsf{ssk}, \mathsf{svk}) \xleftarrow{\$} \Sigma.\mathsf{SKGen}(\mathsf{pp})$
18 $\mathsf{b}' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{wIND\text{-}ENC}}}(\mathsf{pp}, \mathsf{pk}_0, \mathsf{svk})$
19 **return** b'

$\mathcal{O}^{\mathsf{wIND\text{-}ENC}}_{\mathsf{Chall_b}}(\mathsf{e}, (\mathsf{m}_0))$
20 $\mathbf{C} := \begin{pmatrix} \left[s^\star\right]_1 \\ \left[r^\star\right]_1 + \mathsf{WH}(\mathsf{m}_0) \end{pmatrix}$
21 $\pi \xleftarrow{\$} \Pi.\mathsf{Sim}(\mathsf{crs}, \mathsf{SimK}, \mathbf{C})$
22 $\tilde{\mathbf{c}}_\mathsf{e} := (\mathbf{C}, \pi)$
23 $\tilde{\sigma}_\mathsf{e} \xleftarrow{\$} \mathsf{Sig}(\mathsf{ssk}, \mathbf{c}_\mathsf{e})$
24 $\mathcal{L}_{\mathsf{Chall}} := \{(\tilde{\mathbf{c}}_\mathsf{e}, \tilde{\sigma}_\mathsf{e})\}$
25 **phase** $:= 1$
26 **return** $(\tilde{\mathbf{c}}_\mathsf{e}, \tilde{\sigma}_\mathsf{e})$

Oracle $\mathcal{O}_{\mathsf{Sig}}(\mathbf{c}, \mathsf{e})$
01 $\sigma \xleftarrow{\$} \mathsf{Sig}(\mathsf{ssk}, \mathbf{c})$
02 **return** σ

Oracle $\mathcal{O}_{\mathsf{Insert}}(\mathbf{c}, \pi_{\mathsf{DM}})$
21 **if** $\Pi_{\mathsf{DM}}.\mathsf{Ver}_r(\mathsf{pk_e}, \mathbf{c}, \pi_{\mathsf{DM}}) = 1$
22 $\quad \sigma \xleftarrow{\$} \mathsf{Sig}(\mathsf{ssk}, \mathbf{c})$
23 $\quad \mathcal{L}_c := \mathcal{L}_c \cup \{(\mathbf{c}, \sigma, \mathsf{e}, \pi)\}$

Oracle $\mathcal{O}_{\mathsf{Next}}()$
24 e++
25 **return** $\mathsf{pk_e}$

Oracle $\mathcal{O}_{\mathsf{CorrK}}(i)$
26 **Check** $0 \leq i \leq \mathsf{e}$
27 **return** sk_i

Oracle $\mathcal{O}_{\mathsf{CorrT}}(i)$
28 **Check** $0 < i \leq \mathsf{e}$
29 **return** Δ_i

Oracle $\mathcal{O}_{\mathsf{Upd}}((\mathbf{c}_{\mathsf{e}_0}, \sigma_{\mathsf{e}_0}), \mathsf{e}_0, \mathsf{e}_1)$
30 **Check** $0 \leq \mathsf{e}_0 < \mathsf{e}_1 \leq \mathsf{e}$
31 **if** $\exists \pi.(\mathbf{c}_{\mathsf{e}_0}, \sigma_{\mathsf{e}_0}, \mathsf{e}_0, \pi) \in \mathcal{L}_c$
32 $\quad \mathsf{m} := \Pi_{\mathsf{DM}}.\mathsf{Ext}(\mathsf{crs}, \mathsf{ExtK}, \pi)$
33 $\quad$ **parse** $\mathbf{A} =: \mathsf{pk}_{\mathsf{e}_1}$
34 $\quad r \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$
35 $\quad \mathbf{C} := \mathbf{A} \cdot r + \begin{pmatrix} 0 & \mathsf{m} \end{pmatrix}^\intercal$
36 $\quad \pi \xleftarrow{\$} \Pi.\mathsf{Sim}(\mathsf{crs}, \mathsf{SimK}, \mathbf{C})$
37 $\quad \mathbf{c}_{\mathsf{e}_1} := (\mathbf{C}, \pi)$
38 $\quad \sigma_{\mathsf{e}_1} := \mathsf{Sig}(\mathsf{ssk}, \mathbf{c})$
39 $\quad \mathcal{L}_c := \mathcal{L}_c \cup \{(\mathbf{c}_{\mathsf{e}_1}, \sigma_{\mathsf{e}_0}, \mathsf{e}_1, \pi)\}$
40 **elseif** $(\mathbf{c}_{\mathsf{e}_0}, \sigma_{\mathsf{e}_0}) \in \mathcal{L}_{\mathsf{Chall}}$
41 $\quad$ **Check** $\mathsf{e_L} \leq \mathsf{e}_0 < \mathsf{e}_1 \leq \mathsf{e_R}$
42 $\quad$ **for** $i = \mathsf{e}_0 + 1$ **to** e_1
43 $\quad\quad (\mathbf{c}_i, \sigma_i) := \mathsf{Upd}(\Delta_i, \mathbf{c}_{i-1}, \sigma_{i-1})$
44 $\quad \mathcal{L}_{\mathsf{Chall}} := \mathcal{L}_{\mathsf{Chall}} \cup \{(\mathbf{c}_{\mathsf{e}_1}, \sigma_{\mathsf{e}_1})\}$
45 **else abort**
46 **return** $(\mathbf{c}_{\mathsf{e}_1}, \sigma_{\mathsf{e}_1})$

Fig. 11. The construction of $\mathcal{B}_3$ that use wIND-ENC adversary $\mathcal{A}$ to break the SXDH assumption. We recall that $\mathsf{e_L}$ and $\mathsf{e_R}$ are the starting and ending epoch for challenge ciphertexts.

We can notice that if CH is a SXDH tuple in $\mathbb{G}_1$, the every challenge ciphertext is an encryption of m_0, otherwise the challenges ciphertexts encrypts a random

message. Therefore, we have

$$\mathsf{pr}_3 \leq \mathsf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathcal{B}_3}(1^\lambda).$$

By an union bound over all the hybrids, we have

$$\mathsf{Adv}^{\mathsf{wIND\text{-}ENC}}_{\mathsf{PSigUE}_2,\mathcal{A}}(1^\lambda) \leq \mathsf{n}^2 \cdot (\mathsf{Adv}^{\mathsf{ZK}}_{\Pi,\mathcal{B}_1}(1^\lambda) + \mathsf{Adv}^{\mathsf{Dual\text{-}Mode}}_{\Pi_{\mathsf{DM}},\mathcal{B}_2}(1^\lambda) + \mathsf{Adv}^{\mathsf{SXDH}}_{\mathbb{G}_1,\mathcal{B}_3}(1^\lambda)).$$

B.2 Proof of Integrity for PSigUE_2

Extended Unforgeable under Chosen-Message Attack: We define the extended version version of signature scheme together with an adapted security notion Ext-UF-CMA.

Definition 11 (Extended Signautre). *An extended signature scheme consists of four PPT algorithms* $\mathsf{ExtSig} = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{Sig}, \mathsf{Ver})$ *with the following syntax:*

- $\mathsf{Setup}(1^\lambda)$ *takes a security parameter* λ *as input, and outputs public parameters* pp.
- $\mathsf{SKGen}(\mathsf{pp})$ *takes public parameters* pp *as input, and outputs signing key* ssk *and verification key* svk.
- $\mathsf{Sig}(\mathsf{ssk}, \mathsf{m}, \mathsf{m}_{\mathsf{aux}})$ *takes a signing key, a message* m *and an auxiliary message* $\mathsf{m}_{\mathsf{aux}}$ *as input, and outputs a signature* σ.
- $\mathsf{Ver}(\mathsf{svk}, \mathsf{m}, \mathsf{m}_{\mathsf{aux}}, \sigma)$ *takes a verification key* svk*, a message* m*, an auxiliary message* $\mathsf{m}_{\mathsf{aux}}$ *and a signature* σ*, and outputs* 0 *or* 1.

We also require correctness and Ext-UF-CMA *for* ExtSig.

- Correctness : ExtSig *is correct if for any security parameter* λ*, any message* m *and any auxiliary message* $\mathsf{m}_{\mathsf{aux}}$*, we have:*

$$\Pr\left[\mathsf{Ver}(\mathsf{svk}, \mathsf{m}, \mathsf{m}_{\mathsf{aux}}, \sigma) = 1 \;\middle|\; \begin{array}{l} \mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda) \\ (\mathsf{svk}, \mathsf{ssk}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{pp}) \\ \sigma \xleftarrow{\$} \mathsf{Sig}(\mathsf{ssk}, \mathsf{m}, \mathsf{m}_{\mathsf{aux}}) \end{array}\right] = 1$$

- Ext-UF-CMA : *For any PPT adversary* $\mathcal{A}$ *and any security parameter* λ*, we define the extended unforgeablbility under chosen-attack security experiment* $\mathbf{Exp}^{\mathsf{Ext\text{-}UF\text{-}CMA}}_{\mathcal{A},\mathsf{ExtSig}}(1^\lambda)$ *as in Figure 12. We say that* ExtSig *is* Ext-UF-CMA *secure if there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that,*

$$\Pr\left[\mathbf{Exp}^{\mathsf{Ext\text{-}UF\text{-}CMA}}_{\mathcal{A},\mathsf{ExtSig}}(1^\lambda) = 1\right] \leq \mathsf{negl}(\cdot)$$

Extended Waters Signature In our construction of PSigUE_2, we use an extended variant of Waters signature [29]. We give the explicit construction $\mathsf{ExtSig} = (\mathsf{Setup}, \mathsf{SKGen}, \mathsf{Sig}, \mathsf{Ver})$ under SXDH assumption in Figure 13.

By using the similar security proof as in [29], we can show that the extended signature ExtSig constructed in Figure 13 is Ext-UF-CMA under the CDH assumption in $\mathbb{G}_1$. We can notice that Ext-UF-CMA tightly implies UF-CMA security. By security analysis for Waters signatures [29,17], we have the following lemma.

$\underline{\mathbf{Exp}^{\mathsf{Ext\text{-}UF\text{-}CMA}}_{\mathcal{A},\mathsf{ExtSig}}(1^\lambda):}$
01 $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
02 $(\mathsf{svk},\mathsf{ssk}) \xleftarrow{\$} \mathsf{SKGen}(\mathsf{pp})$
03 $(\mathsf{m}^\star, \mathsf{m}^\star_{\mathsf{aux}}, \sigma^\star) \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{Sig}}}(\mathsf{pp},\mathsf{svk})$
04 **if** $\mathsf{Ver}(\mathsf{svk},\mathsf{m}^\star,\mathsf{m}^\star_{\mathsf{aux}},\sigma^\star) = 1 \wedge (\mathsf{m}^\star) \notin \mathcal{L}_\sigma$
05 **return** 1
06 **return** 0

$\underline{\mathbf{Oracle}\ \mathcal{O}_{\mathsf{Sig}}(\mathsf{m},\mathsf{m}_{\mathsf{aux}})}$
07 $\sigma \xleftarrow{\$} \mathsf{Sig}(\mathsf{ssk},\mathsf{m},\mathsf{m}_{\mathsf{aux}})$
08 $\mathcal{L}_\sigma := \mathcal{L}_\sigma \cup \{\mathsf{m}\}$
09 **return** σ

Fig. 12. This is the Ext-UF-CMA security notion of ExtSig.

$\underline{\mathbf{Alg}\ \mathsf{Setup}(1^\lambda):}$
01 $\mathsf{u} \xleftarrow{\$} \mathbb{G}_1$
02 $y \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$; $\hat{\mathsf{v}} := \left[y\right]_2 \in \mathbb{G}_2$
03 $\mathsf{w} := y \cdot \mathsf{u}$
04 $\mathsf{svk} := (\mathsf{u},\hat{\mathsf{v}}) \in \mathbb{G}_1 \times \mathbb{G}_2$
05 $\mathsf{ssk} := \mathsf{w} \in \mathbb{G}_1$
06 **return** $(\mathsf{svk},\mathsf{ssk})$

$\underline{\mathbf{Alg}\ \mathsf{Sig}(\mathsf{ssk},\mathsf{m} \in \{0,1\}^\ell, \mathsf{m}_{\mathsf{aux}})}$
07 **parse** $(\mathbf{A},\mathbf{R}) =: \mathsf{m}_{\mathsf{aux}} \in \mathbb{G}_1^2 \times \mathbb{G}_1^2$
08 $s \xleftarrow{\$} \mathbb{Z}_\mathsf{p}$; $\hat{\mathsf{z}} := \left[s\right]_2$
09 $\mathbf{Z} := s \cdot \mathbf{A}$; $\mathbf{Y} := s \cdot \mathbf{R}$
10 $\sigma' := \mathsf{w} + \left(1\ \mathsf{m}_1\ \cdots\ \mathsf{m}_\ell\right) \cdot \mathbf{h} \cdot s$
11 **return** $\sigma := (\sigma', \mathbf{Z}, \mathbf{Y}, \hat{\mathsf{z}})$

$\underline{\mathbf{Alg}\ \mathsf{Ver}(\mathsf{svk},\mathsf{m},\mathsf{m}_{\mathsf{aux}},\sigma)}$
12 **Check** $\mathbf{Z} \bullet \left[1\right]_2 = \mathbf{A} \bullet \hat{\mathsf{z}}$
13 **Check** $\mathbf{Y} \bullet \left[1\right]_2 = \mathbf{R} \bullet \hat{\mathsf{z}}$
14 **Check** $\sigma' \cdot \left[1\right]_2 = \mathsf{u} \bullet \hat{\mathsf{v}} + \left(1\ \mathsf{m}_1\ \cdots\ \mathsf{m}_\ell\right) \cdot \mathbf{h} \bullet \left[s\right]_2$

Fig. 13. This is the extended variant of Waters signature scheme

Lemma 4 (Ext-UF-CMA). *For all adversary $\mathcal{A}$ that, breaks the* Ext-UF-CMA *security game of* ExtSig *constructed as in Figure 13 with $\mathsf{Q}_{\mathsf{Sig}}$ signing queries and success probability $\varepsilon_\mathcal{A}$ in time $\mathsf{T}(\mathcal{A})$, there exits an adversary $\mathcal{B}$ such that*

$$\varepsilon_\mathcal{A} \leq \mathsf{Adv}^{\mathsf{CDH}_1}_{\mathcal{B},\mathbb{G}_1}(1^\lambda) \cdot O(\frac{1}{\sqrt{\ell} \cdot \mathsf{Q}_{\mathsf{Sig}}}), \qquad \mathsf{T}(\mathcal{A}) \approx \mathsf{T}(\mathcal{B}).$$

Theorem 7 (Integrity). *Let PSigUE_2 be the public-key signable updatable encryption described in Figure 7. For any* Int *adversary $\mathcal{A}$ (that makes max $\mathsf{Q}_{\mathsf{Sig}}$ signing queries), there exists an adversary $\mathcal{B}$ with $\mathsf{T}(\mathcal{A}) \approx \mathsf{T}(\mathcal{B})$, such that*

$$\mathsf{Adv}^{\mathit{Int}}_{\mathsf{PSigUE}_2,\mathcal{A}}(1^\lambda) \leq \mathsf{Adv}^{\mathsf{CDH}_1}_{\mathcal{B},\mathbb{G}_1}(1^\lambda) \cdot O(\frac{1}{\sqrt{\ell} \cdot \mathsf{Q}_{\mathsf{Sig}}}).$$

Proof. Let $\mathcal{A}$ be an adversary against the integrity of PSigUE_2 with wining probability $\varepsilon_\mathcal{A}$. To bound $\varepsilon_\mathcal{A}$, we will construct an adversary $\mathcal{B}$ that breaks Ext-UF-CMA security of ExtSig as in Figure 13 with probability $\varepsilon_\mathcal{B}$ and $\mathsf{T}(\mathcal{A}) \approx \mathsf{T}(\mathcal{B})$.

We can notice that if $\mathcal{A}$ win the security game $\mathbf{Exp}^{\mathsf{Int}}_{\mathsf{PSigUE}_2,\mathcal{A}}(1^\lambda)$, then the output $(\mathsf{m},\mathsf{m}_{\mathsf{aux}},\sigma)$ generated by $\mathcal{B}$ is a valid forgery of ExtSig. Combining with Lemma 4, we can conclude our statement. □

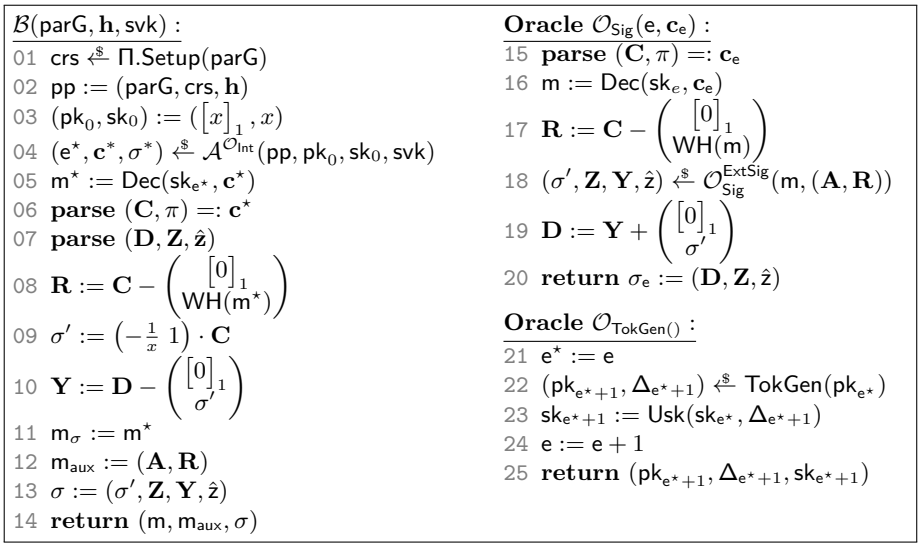

Fig. 14. This is the construction of the adversary $\mathcal{B}$ against Ext-UF-CMA security of ExtSig. We denote by $\mathcal{O}_{\mathsf{Sig}}^{\mathsf{ExtSig}}$ the signing oracle of the experiment $\mathbf{Exp}_{\mathcal{A},\mathsf{ExtSig}}^{\mathsf{Ext\text{-}UF\text{-}CMA}}$.