Vetted Encryption

Martha Norberg Hovd[1,2] and Martijn Stam[1]

[1] Simula UiB
Merkantilen (3rd floor)
Thormøhlensgate 53D
N-5006 Bergen, Norway.
martha,martijn@simula.no
[2] University of Bergen
Høyteknologisenteret i Bergen
Thormøhlensgate 55
N-5008 Bergen, Norway.

Abstract. We introduce Vetted Encryption (VE), a novel cryptographic primitive, which addresses the following scenario: a receiver controls, or vets, who can send them encrypted messages. We model this as a filter publicly checking ciphertext validity, where the overhead does not grow with the number of senders. The filter receives one public key for verification, and every user receives one personal encryption key.
We present three versions: Anonymous, Identifiable, and Opaque VE (AVE, IVE and OVE), and concentrate on formal definitions, security notions and examples of instantiations based on preexisting primitives of the latter two. For IVE, the sender is identifiable both to the filter and the receiver, and we make the comparison with identity-based signcryption. For OVE, a sender is anonymous to the filter, but is identified to the receiver. OVE is comparable to group signatures with message recovery, with the important additional property of confidentiality of messages.

Keywords: Encryption · Group Signatures · Signcryption

1 Introduction

Spam and phishing messages are a bane of modern communication methods, especially email. These days, most email still happens in the clear without end-to-end cryptographic protection. Yet, there are standards, such as S/MIME and OpenPGP, that aim to secure email using a combination of public key and symmetric key confidentiality and authentication primitives. Intuitively, the primitive that best models secure email is signcryption [46, 28]. Although signcryption allows receivers to verify *locally* whether an email was from its purported sender or not, this ability does not immediately lead to an efficient mechanism to filter spam centrally.

A different, though not completely unrelated, scenario arrises with electronic voting systems and eligibility verifiability. This notion informally states that it should be possible to publicly verify that only those with the right to vote have done so. For obvious reasons, voters should still be anonymous, and so whitelisting is not a viable option to prevent ballot stuffing by the bulletin board, for instance.

In this work we propose an alternative primitive called *vetted encryption*, which is closely related to both signcryption and group signatures. Vetted encryption lets a user, the *recipient*, to restrict who can send them encrypted messages by enabling an *outside filter* to detect which users are and are not vetted. The key features of vetted encryption are that a recipient only needs to vet each sender once (with out-of-band communication), yet does *not* need to tell the filter which users they have vetted.

Vetted encryption comes in different flavours, depending on whether senders should be identified and authenticated or, in contrast, should remain anonymous. This choice of authentication versus anonymity can be made with respect to the outside filter and the intended receiver independently of each other, leading to a total of four possible configurations. One configurarion, where the filter would learn the identity of a ciphertext, yet the receiver could not, runs counter to our perspective that the filter is working on behalf of the recipient. Thus, only three settings remain:

1. *Anonymous vetted encryption (AVE)* where the sender remains anonymous to both the filter and the recipient; this scenario can be relevant for a voting system using a bulletin board, on which only eligible users should be able to post, anonymously. For example, the system Belenios [24] applies signatures and credentials to attain eligibility verifiability, which is not too dissimilar from AVE.
2. *Identifiable vetted encryption (IVE)* where the sender is identified for both the filter and the recipient; this scenario is typical for email spam, where the filter gets to see the email-address (or other identifying information) of the sender.

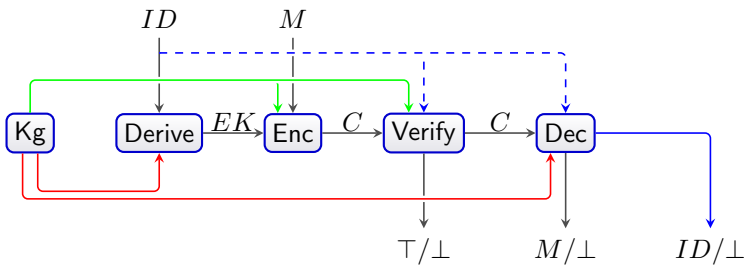

Fig. 1. The algorithms and options involved in vetted encryption for the three options: anonymous includes neither dashed nor solid blue lines; identifiable adds the dashed blue lines only; opaque adds the solid blue lines only and distinguishes between the two red "secret" master keys for derivation resp. decryption.

3. *Opaque vetted encryption (OVE)* where the sender is anonymous to the filter, yet can be identified by the recipient. This primitive is relevant for identifiable communication over an anonymous channel, for example between a trusted anonymous source and a journalist, where the source is anonymous to the newspaper, but identifiable to the reporter. OVE may also be used in an auction setting, where the seller vets who gets to bid. During bidding, the auctioneer may filter the bids, only forwarding bids from vetted participants. However, only the seller knows the identity of the active bidders in the auction.

Our contribution. Fig. 1 provides an overview of the algorithms that constitute a vetted encryption scheme, where we endeavoured to surface all three variants in the picture. The private key material from the key generation (the red lines emanating at the bottom) feeds into two distinct functionalities: firstly to vet users by issuing them an encryption key, and secondly to decrypt ciphertexts. Thus one consideration to make is the possible orthogonality of the corresponding private keys. For both AVE and IVE (Def. 2) we opted for the simplest scenario where both keys are identical, whereas for OVE (Def. 3) we opted for the more challenging scenario where the keys are separate. This choice affects the security definitions and the design space for suitable constructions.

AVE is the simplest variant of vetted encryption and we present it in Appendix B, where we also discuss similarities with signcryption. It turns out all senders can be given the same signing key to encrypt then sign a message. Although this mechanism ensures full anonymity, a malicious sender could make everybody a vetted sender by simply forwarding her key. We therefore focus on IVE and OVE here.

IVE is most closely related to a simplified form of identity-based signcryption with public verifiafibility. As far as we are aware, the related signcryption flavour is virtually unstudied. We give a full comparison of IVE and signcryption in Section 3.3. Our use case for IVE allows us in Section 3 to navigate carefully through the possible definitional choices, esp. quite how much power is available to an adversary trying to break confidentiality, resp. integrity. Our choice allows us to use the novel primitive of "outsider-unique" identity-based signatures, which we show can be constructed by a combination of derandomization and unique signatures (see Appendix A). The unique property is fundamental to provide non-malleability and hence confidentiality against chosen ciphertext attacks for our encrypt-then-IBsign construction (Fig. 6).

OVE bears similarities with group signatures with message recovery, where additionally the message should remain confidential. However, as we will argue in Section 4, our use case allows us to relax security slightly, which in turn enables a slight simplification of the well-known sign-encrypt-proof paradigm for group signatures [13] which we dub *verifiably encrypted certificates* (Fig. 13). We also give a thorough comparison with group signatures in Section 4.3.

1.1 Related Work

Comparison with signcryption. Both AVE and IVE are most closely related to signcryption in its various guises. For the original signcryption concept [46], two users Anna and Bob might want to communicate together in a manner that is simultaneously confidential and authenticated. In a public key setting, if Anna knows Bob's public encryption key and Bob knows Anna's public verification key, then Anna can combine digital signing and public encryption of a message using the signcryption primitive.

Signcryption security is best studied in the multi-user setting, but let us consider just the two user scenario [4]. Confidentiality can be captured by left-or-right indistinguishability under adaptive chosen cipher-

text attacks, where an important modelling choice has to be made with respect to the adversary's control over keys. If the adversary is an outsider, only public key information is available. If the adversary knows private keys (e.g. Anna's signing key when attacking confidentiality or Bob's decryption key when attacking authenticity), we speak of insider security. Insider-secure confidentiality is essential to achieve forward secrecy, that is if Anna's signing key gets compromised, past messages should still remain confidential. Insider-secure authenticity is needed for non-repudiation, where receiver Bob can convince a third party a message really originated from Anna (and wasn't cooked up by Bob himself). Indeed, for most realistic use cases, insider security is required [8].

Clearly insider security is harder to achieve than outsider security. The natural way for Anna and Bob to attempt signcryption would be to combine a public key encryption scheme with a digital signature scheme using generic composition. There are essentially two ways of doing so sequentially: either first encrypt and then sign the ciphertext (encrypt-then-sign) or first sign and then encrypt both message and signature (sign-then-encrypt). The third, parallel alternative of encrypting the message and signing the message is more problematic from a generic composition perspective.

Signcryption with public verifiability [32] could be used as an alternative solution to anonymous vetted encryption, but the precise flavour of signcryption needed is not immediate (see Appendix B.3 for details). Signcryption appears to be unsuitable for OVE (Section 4.3): although it is possible to achieve for instance IB-signcryption with anonymity [20, 22] [10, Section 5.4], crucially these schemes cannot support public verifiability, ruling out the ability to outsource their verification to a filter. Signcryption with public verifiability and explicit whitelisting could be used as a less efficient alternative to IVE, in addition to identity based signcryption with transferable public verifiability, see Section 3.3 for further discussions.

Comparison with group signatures. The setting for group signatures is the following: a group, with a single manager, consist of various members, all with their own secret signing key to sign messages. It may be publicly verified that a signature belongs to a member of the group without revealing *which* member has signed the message. Only the group manager, who possesses a secret opening key, may identify the signer, given a signature on a message.

The classic security notions of group signature schemes encompass both anonymity and traceability [13]. Informally, this means that a signature does not reveal the identity of a signer to anyone who does not possess the opening key, and that one cannot forge someone's signature unless one has their secret key.

With regards to AVE and IVE, the comparison is somewhat natural in the big picture: in both cases, the sender has to verify membership of a group (of vetted senders). The similarities end here, though, as the notion of revealing the identity of the sender runs counter to AVE, and hiding the identity from the filter does not line up with the intention of IVE. However, the setting overlaps to a great extent with desireable features of OVE, with the important exception of confidentiality of messages.

A straightforward, but naive, fix to this would be to simply encrypt the message, then sign the ciphertext using the group signature scheme. Although this seems to add confidentiality to the scheme, it also introduces the following weakness: any group member Eve may intercept a ciphertext, (group)signature pair from Alice, sign the ciphertext using her own key, and thus pass Alice's confidential message off as her own. In particular, if Eve has access to a decryption oracle, she may ask to have the ciphertext decrypted, and by that read the message Alice sent. Thus, we provide a construction of OVE motivated by group signatures, rather than using them as a primitive.

Comparison with matchmaking encryption. Matchmaking encryption (ME) allows, in a sense, for a sender and receiver to vet each other: both may specify policies the other party must satisfy in order for the sent message to be revealed to the recipient. The sender may specify what properties the receiver must have in order to read the message, and the receiver may specify the requirements a sender must meet in order to send the receiver a message. Furthermore, the only information leaked is whether or not a policy match occured, that is: whether or not the recipient received the decrypted message [7].

The set up relies on a trusted authority to generate both encryption and decryption keys for the sender and receiver, respectively, both associated with attributes. In addition, there is a decryption key associated with the policy a sender should satisfy, which is also generated by the trusted authority. Finally, a sender can specify a policy which a reciever must satisfy to be able to decrypt the sent message.

There is an identity based version of ME, which bears some reseblance to OVE. In this version, the more general attributes of the sender and receiver are replaced with a simple identity, so that the sender specifies the identity of the desired recipient, and the identity of the sender is an explicit input of the decryption procedure.

The latter point is an important difference to the OVE primitive, where the identity of the sender is an output of decryption, rather than an input. In other words: we do not assume that the reciever knows who has sent a message before it has been decrypted. Another important difference is that we do not allow the sender to demand any certain attributes of the receiver, though the identity of the receiver is indirectly determined by the sender during encryption, as this involves an encryption key unique to the recipient. We also note that in OVE, the keys are derived and distributed by the recipient, not a third party.

Finally: in ME, determining whether or not a match will occur, that is, if the recipient and sender have vetted each other, is not publicly verifiable. This requires the decryption key of the recipent and the key related to the policy of the sender. This is in contrast with OVE, where determining whether a message has been sent from a vetted sender is possible using only a public key.

Comparison with access control encryption. Access control encryption (ACE) allows for different reading and writing rights to be assigned to different senders and receivers, for example the right to read messages classified as 'Secret', and ensuring a sender with clearance 'Top Secret' cannot send messages classified as 'Public'. This is achieved by introducing a sanitizer into the network, who manipulates every message before it is published on the network, we note in particular that a message is not sent directly to its intended recipient. Now, if a recipient tries to decrypt a message he does not have the right to read, the ciphertext will decrypted into a random string [26].

Although both ACE and OVE in some sense deal with the notion of vetting senders, there are several differences. First of all: an honest filter in OVE does not change the ciphertext in any way, it simply checks whether a sender has been vetted. In particular, if a received ciphertext decrypts to gibberish, it is because the sender intended it so. Furthermore: the sender is always identifiable to the receiver, assuming a message was received. Finally, a sent message is forwarded to the intended recipient, as opposed to published on a network.

We also note that the power dynamic in the two primitives differ. In ACE, the sanitizer enforces a security protocol, typically on behalf of a third party, and determines which subset of a public set of messages anyone is able to read. In OVE, however, all power lies with the recipient, by generating and distributing all the keys. The filter is merely doing the recipient's bidding, as it were, by only allowing messages from vetted senders to reach the recipient.

2 Preliminaries

When defining security, we will use concrete advantages throughout. Moreover, these advantages are defined in terms of an adversary interacting with a game or experiment. While these experiments depend on the schemes at hand, there will be no additional quantifications (e.g. over high entropy sources, simulators, or extractors). We use $\Pr[Code : Event]$ to denote probabilities where the *Code* is used to induce a probability distribution over which *Event* is defined (not to be confused with conditional probabilities). We write $\mathbb{A}^{O}$ for an adversary $\mathbb{A}$ having access to oracle(s) O in security games and reductions.

We use a number of standard primitives and their associated security notions. For completeness, and for the avoidance of any ambiguities in our notation, these are recapitulated below.

- A *public key encryption* scheme PKE consists of a triple of algorithms (Pke.Kg, Pke.Enc, Pke.Dec). The default security notion we consider is single-user multi-query left-or-right indistinguishability under chosen ciphertext attacks (IND-CCA).
- A *signature scheme* SIG consists of a triple of algorithms (Sig.Kg, Sig.Sign, Sig.Verify). The default security notion we consider is single-user strong existential unforgeability under chosen message attacks (EUF-CMA). We often require the SIG to be *unique* (USS), which means that given the verification key, for every message there is only a single signature that verifies.
- A *signcryption* scheme SCR consist of six algorithms (Scr.Kgr, Scr.Kgs, Scr.Signcrypt, Scr.Verify, Scr.Unsigncrypt), where Scr.Kgr generates the receiver's keys and Scr.Kgs the sender's keys.
- An *identity-based signature scheme* IBS consists of the four algorithms (Ibs.Kg, Ibs.Derive, Ibs.Sign, Ibs.Verify), where Ibs.Kg derives a master signing key MSK and a verification key VK. The derivation algorithm

Ibs.Derive takes the master signing key and an identity ID as input, and outputs a user signing key USK. The signing takes a message M, an identity ID and a user signing key USK as input, and outputs a signature σ. Finally, verification takes the verification key VK, a message M, an identity ID and a signature σ as input, and outputs $\top$ or $\bot$. As with signature schemes, we consider EUF-CMA as the default security notion for IBS schemes.

QA-NIZKs. Non-Interactive Zero-Knowledge (NIZK) proofs are defined for families of languages with associated binary relations R, such that for pairs $(\phi, \omega) \in R$ a prover may convince a verifier that the statement ϕ is part of the language, without revealing anything else (such as the witness ω). For the proof to be non-interactive, we require that the only necessary communication between the prover and verifier is the sending of the proof π. This non-interaction requirement disregards the communication required for the set-up of the scheme, which involves the prover and verifier sharing a common reference string (CRS). For Quasi-Adaptive NIZKs (QA-NIZKs) [35], we allow this CRS to depend on the parameters defining the language and its witness relation R. In the following, we let the relation R be given as input to the set-up algorithm and various adversaries, this is to be understood as the parameters defining said relation. Moreover, we let $\mathcal{R}$ be a distribution over the family of languages for which the NIZK is suited.

Definition 1 (Quasi-Adaptive Non-Interactive Zero-Knowledge (QA-NIZK) proofs). *An efficient prover publicly verifiable* Quasi-Adaptive Non-Interactive Zero-Knowledge *(QA-NIZK) for $\mathcal{R}$ is a quadruple of probablilistic algorithms* (Nizk.Setup, Nizk.Prove, Nizk.Verify, Nizk.Sim) *such that*

- Nizk.Setup *produces a CRS σ and a simulation trapdoor τ for the relation R:* $(\sigma, \tau) \leftarrow\!\!\$\ \mathsf{Nizk.Setup}(R)$.
- Nizk.Prove *takes as input a CRS σ and a tuple $(\phi, \omega) \in R$ and returns a proof π:* $\pi \leftarrow\!\!\$\ \mathsf{Nizk.Prove}(\sigma, \phi, \omega)$
- Nizk.Verify *either rejects ($\bot$) or accepts ($\top$) a proof π for a statement ϕ when given these, as well as a CRS σ:* $\top/\bot \leftarrow \mathsf{Nizk.Verify}(\sigma, \phi, \pi)$.
- Nizk.Sim *takes as input a simulation trapdoor τ, and a statement ϕ and returns a proof π:* $\pi \leftarrow\!\!\$\ \mathsf{Nizk.Sim}(\tau, \phi)$.

Completeness. The notion of *perfect completeness* states that, for any true statement ϕ, an honest prover should be able to convince an honest verifier. More formally, we require that for all $R \in \mathcal{R}$ and $(\phi, \omega) \in R$:

$$\Pr[(\sigma, \tau) \leftarrow\!\!\$\ \mathsf{Nizk.Setup}(R); \pi \leftarrow\!\!\$\ \mathsf{Nizk.Prove}(\sigma, \phi, \omega) \ :\ \mathsf{Nizk.Verify}(\sigma, \phi, \omega) \rightarrow \top] = 1\ .$$

Soundness. For a QA-NIZK to achieve *computational soundness*, we require that it is computationally infeasible for an adversary $\mathbb{A}$ given the relation R and the CRS σ, to output a pair (ϕ, π) that satisfy the following conditions: 1) ϕ does not lie in the language defined by R, that is: there does not exist a witness $\bar{\omega}$ such that $(\phi, \bar{\omega}) \in R$, and 2) $\mathsf{Nizk.Verify}(\sigma, \phi, \omega) \rightarrow \top$. Formally, we define the advantage:

$$\mathsf{Adv}^{\text{sound}}_{\text{QANIZK}}(\mathbb{A}) = \Pr\left[\begin{array}{c} R \leftarrow\!\!\$\ \mathcal{R} \\ (\sigma, \tau) \leftarrow\!\!\$\ \mathsf{Nizk.Setup}(R) \ :\ \phi \notin L_R \wedge \mathsf{Nizk.Verify}(\sigma, \phi, \omega) \rightarrow \top \\ (\phi, \pi) \leftarrow\!\!\$\ \mathbb{A}(R, \sigma) \end{array}\right]\ .$$

Zero-knowledge. Informally, a QA-NIZK is *zero-knowledge* if nothing other than the truth of the statement may be inferred by the proof. We formally define the distinguishing advantage using a real and a sim experiment (Fig. 2), and define the advantage of the adversary $\mathbb{A}$ as

$$\mathsf{Adv}^{\text{zk}}_{\text{QANIZK}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\text{zk-real}}_{\text{QANIZK}}(\mathbb{A}) \ :\ \hat{b} = 0\right] - \Pr\left[\mathsf{Exp}^{\text{zk-sim}}_{\text{QANIZK}}(\mathbb{A}) \ :\ \hat{b} = 0\right].$$

We speak of *perfect* zero-knowledge if $\mathsf{Adv}^{\text{zk}}_{\text{QANIZK}}(\mathbb{A}) = 0$ for all adversaries. Perfect zero-knowledge can alternatively be characterized with a single query and a universal quantifier for the choice of language and statement to prove. Many known NIZKs achieve perfect zero-knowledge, facilitating their composability.

$\mathsf{Exp}^{\text{zk-real/sim}}_{\text{QANIZK}}(\mathbb{A})$	prove-real(ϕ, ω)	prove-sim(ϕ, ω)
$R \leftarrow\!\!\$\ \mathcal{R}$	**require** $(\phi, \omega) \in R$	**require** $(\phi, \omega) \in R$
$(\sigma, \tau) \leftarrow\!\!\$\ \mathsf{Nizk.Setup}(R)$	$\pi \leftarrow\!\!\$\ \mathsf{Nizk.Prove}(\sigma, \phi, \omega)$	$\pi \leftarrow\!\!\$\ \mathsf{Nizk.Sim}(\tau, \phi)$
$\hat{b} \leftarrow \mathbb{A}^{O}(R, \sigma)$	**return** π	**return** π

Fig. 2. The real and simulated zero-knowledge experiments.

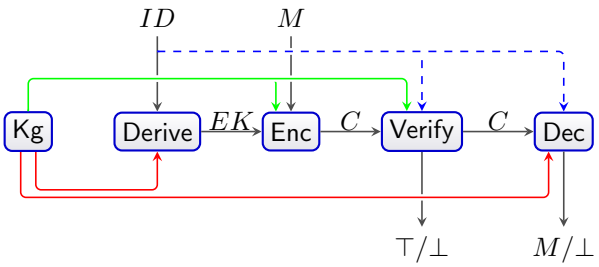

Fig. 3. The algorithms and their inputs/outputs for identifiable vetted encryption.

Unbounded simulation-soundness. A QA-NIZK achieves *unbounded simulation-soundness* if an adversary $\mathbb{A}$ is unable to simulate proofs of any false statement, even after having seen such proofs of arbitrary statements. We define the advantage of the adversary $\mathbb{A}$ as

$$\mathsf{Adv}^{\text{uss}}_{\text{QANIZK}}(\mathbb{A}) = \Pr\left[\begin{array}{c} R \leftarrow\!\!\$\ \mathcal{R} \\ (\sigma, \tau) \leftarrow\!\!\$\ \mathsf{Nizk.Setup} \\ (\phi, \pi) \leftarrow \mathbb{A}^{\mathsf{Nizk.Sim}(\sigma, \tau, \cdot)}(R, \sigma) \end{array} : \begin{array}{c} (\phi, \pi) \notin Q \wedge \phi \notin L_R \\ \wedge \mathsf{Nizk.Verify}(\sigma, \phi, \pi) \rightarrow \top \end{array}\right],$$

where Q is the set of query–response pairs (ϕ, π) to the simulator.

After the introduction of QANIZK protocols [35], a large number of protocols for a large variety of languages (or distributions thereof) has appeared in the literature. They are particularly efficient for linear subspaces, which facilitates pairing based constructions (see [3] and the references contained therein).

3 Identifiable Vetted Encryption (IVE)

3.1 Syntax and Security of IVE

The algorithms. For identifiable vetted encryption, both the filter and the recipient may learn the identity of the sender, which we assume have received the identity via out-of-band communication. An IVE scheme consists of five algorithms, see Def. 2. The identity *ID* is not only an explicit input to the derivation algorithm, but also to both the verification and decryption algorithm, modelling the out-of-band communication. However, encryption does not take *ID* as an input, instead relying on a user's encryption key *EK* implicitly encoding said identity.

We allow encryption to fail, modelled by $\perp$ as output. As we will see, for honestly generated encryption keys, we insist encryption never fails, but for adversarially generated encryption keys, allowing for explicit encryption failure turns out to be useful. One could alternatively introduce a separate algorithm to verify the validity of a private encryption key for a given public encryption/verification key; our approach looks simpler.

Definition 2 (Identifiable Vetted Encryption (IVE)). *An* identifiable vetted encryption *scheme* IVE *consists of a 5-tuple of algorithms* (Ive.Kg, Ive.Derive, Ive.Enc, Ive.Verify, Ive.Dec)*, which behave as follows:*

- Ive.Kg generates *a key pair* (PK, SK)*, where* PK *is the public encryption (and verification) key and* SK *is the private derivation and decryption key. We allow* Ive.Kg *to depend on paramaters param and write* $(PK, SK) \leftarrow\!\!\$\ \mathsf{Ive.Kg}(param)$*. Henceforth, we will assume that* PK *can be uniquely and efficiently computed given* SK*.*
- Ive.Derive derives *an encryption key* EK *based on the private derivation key* SK *and a user's identity* ID*. Thus,* $EK \leftarrow\!\!\$\ \mathsf{Ive.Derive}_{SK}(ID)$*.*

- Ive.Enc encrypts *a message M given the public encryption key PK and using the private encryption key EK, creating a ciphertext C or producing a failed encryption symbol $\perp$. So, $C \leftarrow\!\!\$\ \mathsf{Ive.Enc}_{PK,EK}(M)$ where possibly $C = \perp$.*
- Ive.Verify verifies *the validity of a ciphertext C given the public verification key PK and a user's identity ID. With a slight abuse of notation, $\top/\perp \leftarrow \mathsf{Ive.Verify}^{ID}_{PK}(C)$.*
- Ive.Dec decrypts *a ciphertext C using the private key SK, given the user's identity ID. The result can either be a message M or the invalid-ciphertext symbol $\perp$. In short, $M/\perp \leftarrow \mathsf{Ive.Dec}^{ID}_{SK}(C)$.*

The first three algorithms are probabilistic, the final two deterministic.

Correctness and consistency. For correctness, we require that all honestly generated ciphertexts are received as intended, that is, for all parameters $param$, identities ID and messages M, we have that

$$\Pr\left[\begin{array}{cc} (PK, SK) \leftarrow\!\!\$\ \mathsf{Ive.Kg}(param) & C \neq \perp \\ EK \leftarrow\!\!\$\ \mathsf{Ive.Derive}_{SK}(ID) & : \ \wedge\ \mathsf{Ive.Verify}^{ID}_{PK}(C) = \top \\ C \leftarrow\!\!\$\ \mathsf{Ive.Enc}_{PK,EK}(M) & \wedge\ \mathsf{Ive.Dec}^{ID}_{SK}(C) = M \end{array}\right] = 1\ .$$

Conceptually, a ciphertext may be rejected at two different stages: the filter using Ive.Verify might reject or decryption using Ive.Dec might fail. Thus, we can consider two possible sets of 'valid' ciphertexts: those accepted by verification, and those accepted by decryption. Ideally, these sets coincide, but a priori this cannot be guaranteed. We call a scheme *consistent* if any ciphertext accepted by decryption will also be accepted by the filter verification, whereas we say the scheme is *strict* if any ciphertext that passes the filter, will decrypt to a message.

Formally, we define both strictness and consistency in terms of rejected 'invalid' ciphertexts, thus flipping the order of Ive.Verify and Ive.Dec in the implications below (compared to the intuitive notion described above). That is for all possible keys (PK, SK) output by Ive.Kg and all ciphertexts C, we have

- *Consistency:* $\mathsf{Ive.Verify}^{ID}_{PK}(C) = \perp \Rightarrow \mathsf{Ive.Dec}^{ID}_{SK}(C) = \perp$;
- *Strictness:* $\mathsf{Ive.Dec}^{ID}_{SK}(C) = \perp \Rightarrow \mathsf{Ive.Verify}^{ID}_{PK}(C) = \perp$.

Fortunately, it is relatively easy to guarantee consistency; the trivial transformation that runs verification as part of decryption takes care of this. Henceforth we will concentrate on consistent schemes.

On the other hand, strictness is harder to guarantee a priori. Thus we will allow ciphertexts to pass the filter that are subsequently deemed invalid by decryption. Note that, for *honestly generated* ciphertexts, correctness ensures that decryption will actually succeed, so this scenario can only occur for 'adulterine' ciphertexts.

Security. The security of IVE comprises of two components: *integrity* to ensure the filter cannot be fooled, and *confidentiality* of the messages to outsiders. With reference to the games defined in Fig. 4 and Fig. 5, the advantages are defined as follows:

- *Integrity*:

$$\mathsf{Adv}^{\mathrm{int}}_{\mathrm{IVE}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\mathrm{int}}_{\mathrm{IVE}}(\mathbb{A}) : \begin{array}{c} \hat{ID} \notin \mathcal{E} \wedge (\hat{ID}, \hat{C}) \notin C \\ \wedge\ \mathsf{Ive.Verify}^{\hat{ID}}_{PK}(\hat{C}) = \top \end{array}\right] .$$

- *Confidentiality*:

$$\mathsf{Adv}^{\mathrm{conf}}_{\mathrm{IVE}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\mathrm{conf\text{-}0}}_{\mathrm{IVE}}(\mathbb{A}) : \hat{b} = 0\right] - \Pr\left[\mathsf{Exp}^{\mathrm{conf\text{-}1}}_{\mathrm{IVE}}(\mathbb{A}) : \hat{b} = 0\right] .$$

Integrity. A server running the verification algorithm to filter out invalid ciphertexts should not be easily fooled by an adversary: unless one is in possession of an encryption key (i.e. has been vetted), it should not be possible to construct a valid ciphertext. Even a *vetted* sender should not be able to construct a ciphertext which is considered valid under a different identity. We formally capture integrity in a game (Fig. 4) where we use the output of the verification algorithm as an indication of validity. For consistent schemes this choice is the strongest, as a forgery with respect to decryption will always be a forgery with respect to verification.

$\underline{\mathsf{Exp}^{\mathrm{int}}_{\mathrm{IVE}}(\mathbb{A})}$

$(PK, SK) \leftarrow\!\$\ \mathsf{Ive.Kg}$
$h \leftarrow 0; \mathcal{C} \leftarrow \emptyset; \mathcal{E} \leftarrow \emptyset$
$(\hat{ID}, \hat{C}) \leftarrow \mathbb{A}^{O}(PK)$
winif $\hat{ID} \notin \mathcal{E} \wedge (\hat{ID}, \hat{C}) \notin \mathcal{C}$
$\quad \wedge\ \mathsf{Ive.Verify}^{\hat{ID}}_{PK}(\hat{C}) = \top$

$\underline{\mathrm{derive}(ID)}$

$EK[h] \leftarrow \mathsf{Ive.Derive}_{SK}(ID)$
$h \leftarrow h + 1$
return h

$\underline{\mathrm{corrupt}(H)}$

$\mathcal{E} \leftarrow \mathcal{E} \cup H.ID$
return $EK[H]$

$\underline{\mathrm{encrypt}(H, M)}$

$C \leftarrow\!\$\ \mathsf{Ive.Enc}_{PK, EK[H]}(M)$
$\mathcal{C} \leftarrow \mathcal{C} \cup \{(H.ID, C)\}$
return C

$\underline{\mathrm{decrypt}(ID, C)}$

$M \leftarrow \mathsf{Ive.Dec}^{ID}_{SK}(C)$
return M

Fig. 4. The integrity game for IVE.

The adversary is given the verification key as well as encryptions of messages of her own choosing under honest encryption keys. We use *handles* to grant an adversary control over the encryption keys that are used: an adversary can trigger the creation of an arbitrary number of keys for chosen identities and then indicate which key (by order of creation) should be used for a particular encryption query.

Additionally, an adversary can adaptively ask for encryption keys from a corruption oracle. Obviously, a corrupted encryption key trivially allows for the construction of further valid ciphertexts for the underlying identity, so we exclude corrupted identities from the win condition. Similarly, ciphertexts resulting from an encryption query do not count as a win under the original query's identity.

Finally, an adversary has access to a decryption oracle. This oracle is superfluous for uncorrupted encryption keys, but an adversary could potentially use it to her advantage by querying it with ciphertext created under a corrupted identity. These ciphertexts will, of course, not help her win the integrity game directly, as the corresponding identity is corrupted. Yet, the oracle response might leak information about SK, which could help the adversary construct a valid ciphertext for an *uncorrupted* identity, hence giving an advantage in winning the integrity game. Constructing a non-strict pathological IVE scheme exploiting this loophole is easy: simply allow ciphertexts outside the support of the encryption algorithm to gradually leak the secret key based on their validity under decryption. We stress that in our instantiation of IVE we do *not* face this issue.

Confidentiality. We adopt the CCA security notion for public key encryption to the setting of identifiable vetted encryption (Fig. 5). An adversary can, repeatedly, ask its challenge oracle for the encryption of one of two messages under an adversarially chosen encryption key. We give the adversary an oracle to derive and immediately learn encryption keys; moreover these known honest keys may be fed to the challenge encryption oracle.

We want to avoid the decryption oracle being used by an adversary to win trivially, namely by simply querying a challenge ciphertext under the corresponding identity. But what is this corresponding identity? The encryption *algorithm* only takes as input an encryption key EK that may or may not allow easy extraction of an identity ID. One solution would be to only allow the adversary to ask for challenge encryptions on honestly derived encryption keys (so the game can keep track of the identity when EK is derived). Instead, we opted for a stronger version where the adversary provides the challenge encryption *oracle* with both an encryption key EK *and* a purported identity ID. If verification shows that the freshly generated challenge ciphertext does *not* correspond to ID, which can only happen for dishonestly generated pairs (EK, ID), then the encryption oracle rejects the query by outputting $\perp_G$.

Intuitively, the decryption oracle is mainly relevant for identities that the adversary has previously queried to its derivation oracle: after all, if the decryption oracle would return anything but $\perp$ for a fresh ciphertext under a fresh identity, this would constitute a break of the integrity game.

3.2 Encrypt-then-IBS

An obvious first attempt to create an identifiable vetted encryption scheme is to combine the confidentiality provided by a public key encryption scheme with the authenticity of that of an identity based signature scheme. There are three basic methods for the generic composition: sign-then-encrypt, encrypt-then-sign, and encrypt-and-sign. For the first option, the signature ends up being encrypted, which destroys public verifiability as required for the filter to do its work. The parallel encrypt-and-sign is well-known to be problematic,

$\underline{\mathsf{Exp}^{\text{ind-cca-}b^*}_{\text{IVE}}(\mathbb{A})}$

$(PK, SK) \leftarrow\!\$\ \mathsf{Ive.Kg}$
$C \leftarrow \emptyset$
$\hat{b} \leftarrow \mathbb{A}^{O}(PK)$

$\underline{\text{derive}(ID)}$

$EK \leftarrow \mathsf{Ive.Derive}_{SK}(ID)$
return EK

$\underline{\text{encrypt}(ID, EK, M_0, M_1)}$

$C^* \leftarrow\!\$\ \mathsf{Ive.Enc}_{PK,EK}(M_{b^*})$
if $\mathsf{Ive.Verify}^{ID}_{PK}(C^*) = \perp$ **then**
  return $\perp_G$
$C \leftarrow C \cup \{(ID, C^*)\}$
return C^*

$\underline{\text{decrypt}(ID, C)}$

require $(ID, C) \notin C$
$M \leftarrow \mathsf{Ive.Dec}^{ID}_{SK}(C)$
return M

Fig. 5. The confidentiality game for IVE.

$\underline{\mathsf{Ive.Kg}()}$

$(PK, DK) \leftarrow \mathsf{Pke.Kg}$
$(MVK, MSK) \leftarrow \mathsf{Uibss.Kg}$
return $((PK, MVK), (DK, MSK))$

$\underline{\mathsf{Ive.Enc}_{(PK,MVK),USK,ID}(M)}$

$C \leftarrow \mathsf{Pke.Enc}_{PK}(M \| ID)$
$\sigma \leftarrow \mathsf{Uibss.Sign}_{USK}(C)$
if $\mathsf{Uibss.Verify}^{ID}_{MVK}(C, \sigma) = \perp$ **then**
  return $\perp$
return (C, σ)

$\underline{\mathsf{Ive.Verify}^{ID}_{PK,MVK}(C, \sigma)}$

return $\mathsf{Uibss.Verify}^{ID}_{MVK}(C, \sigma)$

$\underline{\mathsf{Ive.Derive}_{(DK,MSK)}(ID)}$

$USK \leftarrow \mathsf{Uibss.Derive}_{MSK}(ID)$
return USK

$\underline{\mathsf{Ive.Dec}^{ID}_{DK,MSK}(C, \sigma)}$

if $\mathsf{Uibss.Verify}^{ID}_{MVK}(C, \sigma) = \perp$ **then**
  return $\perp$
if $\mathsf{Pke.Dec}_{DK}(C) = \perp$ **then**
  return $\perp$
if $\mathsf{Pke.Dec}_{DK}(C) \to M \| I\bar{D} \wedge I\bar{D} \neq ID$ **then**
  return $\perp$
return M

Fig. 6. Encrypt-then-IBSign (EtIBS): A straightforward composition of public key encryption and an identity-based signature scheme.

as the unencrypted signature directly on the message inevitably leaks information on the message, even when the signatures are confidential [27] (as the signature allows for an easy check whether a given plaintext was encrypted or not). Thus only encrypt-then-sign remains as option, and we specify the construction in Fig. 6.

We show the scheme achieves integrity and confidentiality in Lemmas 1 and 2, respectively. Integrity of the construction follows from the unforgeability of the underlying signature scheme. However, for IVE to inherit the confidentiality of the encryption scheme, we use an identity-based signature scheme with *outsider unique* signatures.

Without unique signatures, an adversary who has received a challenge ciphertext (C, σ) could simply create a new tuple (C, σ') with a secondary valid signature σ'. This tuple will be accepted by a decryption oracle, and hence the adversary will learn the encrypted message, breaking confidentiality. To the best of our knowledge, unique identity-based signatures have not been studied before. It turns out that for our purposes, a computational version of uniqueness suffices (the details are in Appendix A).

Correctness and consistency. Both correctness and consistency follow easily by inspection. The signature verification as part of decryption is needed for consistency, cf. the transformation mentioned previously.

Integrity. Integrity of the Encrypt-then-IBS construction boils down to the unforgeability of the underlying identity-based signature scheme. As the decryption key of the underlying encryption scheme is unrelated to the issuing key of the signature scheme (so an adversary cannot hope to learn any useful information about the issuing key by querying the decryption oracle with ciphertexts of corrupted identities), the reduction is fairly straightforward.

$\underline{\mathsf{Exp}^{\mathrm{int}}_{\mathrm{IVE}}(\mathbb{A})}$

$(PK, DK) \leftarrow\!\!\$\ \mathsf{Pke.Kg}$
$(VK, MSK) \leftarrow\!\!\$\ \mathsf{Uibss.Kg}$
$h \leftarrow 0; \mathcal{C} \leftarrow \emptyset; \mathcal{E} \leftarrow \emptyset$
$(\hat{ID}, \hat{C}) \leftarrow \mathbb{A}^{O}(PK)$
winif $\hat{ID} \notin \mathcal{E} \wedge (\hat{ID}, \hat{C}) \notin \mathcal{C}$
$\quad \wedge\ \mathsf{Ive.Verify}^{\hat{ID}}_{PK}(\hat{C}) = \top$

$\underline{\mathrm{derive}(ID)}$

$ID_h \leftarrow ID$
$h \leftarrow h + 1$
return h

$\underline{\mathrm{corrupt}(H)}$

$EK_H \leftarrow \mathsf{Uibss.Derive}_{MSK}(ID_H)$
$\mathcal{E} \leftarrow \mathcal{E} \cup ID_H$
return EK_H

$\underline{\mathrm{encrypt}(H, M)}$

$C \leftarrow\!\!\$\ \mathsf{Pke.Enc}_{PK}(M \| ID_H)$
$\sigma \leftarrow \mathsf{Uibss.Sign}_{EK_H}(C)$
$\mathcal{C} \leftarrow \mathcal{C} \cup \{(C, \sigma), ID_H\}$
return (C, σ)

$\underline{\mathrm{decrypt}((C, \sigma), ID)}$

if $\mathsf{Uibss.Verify}^{ID}_{VK}(C, \sigma) = \perp$
  return $\perp$
$M \| ID \leftarrow \mathsf{Pke.Dec}_{DK}(C)$
if decryption or parsing fails
  return $\perp$
return M

Fig. 7. The game for the proof of integrity for the Encrypt-then-IBS construction

Lemma 1 (Integrity of Encrypt-then-IBS). *For all adversaries* $\mathbb{A}_{\mathrm{int}}$ *there exists a similarly efficient adversary* $\mathbb{B}_{\text{euf-cma}}$ *such that*

$$\mathsf{Adv}^{\mathrm{int}}_{\mathrm{IVE}}(\mathbb{A}_{\mathrm{int}}) \leq \mathsf{Adv}^{\text{euf-cma}}_{\mathrm{UIBSS}}(\mathbb{B}_{\text{euf-cma}}) \ .$$

Proof. The integrity game defined in Fig. 4 applied to our construction is shown in Fig. 7. Based on this game, we may construct a reduction to a forging game of the underlying identity based signature scheme. An adversary $\mathbb{B}_{\text{euf-cma}}$ is given the verification key VK of the signature scheme. She constructs an encryption scheme and generates the keys $(PK, DK) \leftarrow\!\!\$\ \mathsf{Pke.Kg}$, and sends (PK, VK) to $\mathbb{A}_{\mathrm{int}}$. Whenever $\mathbb{A}_{\mathrm{int}}$ makes a derivation query on an identity ID, $\mathbb{B}_{\text{euf-cma}}$ simply does the administrative work herself, by ascribing the identity with a handle, and returning this. Any encryption queries on a message M under a handle H is managed by $\mathbb{B}_{\text{euf-cma}}$ first producing $C \leftarrow\!\!\$\ \mathsf{Pke.Enc}_{PK}(M \| ID_H)$, and then querying her own signature oracle on (C, ID_H), receiving the signature σ. She then sends (C, σ) to $\mathbb{A}_{\mathrm{int}}$. Any corruption queries on H is answered by $\mathbb{B}_{\text{euf-cma}}$ querying her own corruption oracle on ID_H, and forewarding the given signing key. Finally, all decryption queries from $\mathbb{A}_{\mathrm{int}}$ are handled solely by $\mathbb{B}_{\text{euf-cma}}$, as she can perform all the checks and decryptions herself. When $\mathbb{A}_{\mathrm{int}}$ outputs $((\hat{C}, \hat{\sigma}), \hat{ID})$, $\mathbb{B}_{\text{euf-cma}}$ simply copies this as her own answer. It is clear that $\mathbb{B}_{\text{euf-cma}}$ will win in precisely the same cases as $\mathbb{A}_{\mathrm{int}}$, and so the claim follows.

□

Confidentiality. The confidentiality of the Encrypt-then-IBS hinges on both the confidentiality of the encryption scheme and the computational hardness of finding a signature collision in the IBS scheme. As the IVE adversary does not have access to the master private key of the underlying IBS scheme, it suffices that signatures are unique with respect to individual signing keys (that can be obtained through the derive oracle). That allows us to rule out mauling of a challenge ciphertext (C, σ) through the signature component, leaving the adversary with the only option of breaking the confidentiality of the encryption scheme.

Lemma 2 (Confidentiality of Encrypt-then-IBS). *For all adversaries* $\mathbb{A}_{\mathrm{conf}}$ *there exist similarly efficient adversaries* $\mathbb{B}_{\mathrm{cca}}$ *and* $\mathbb{B}_{\mathrm{ou}}$ *such that*

$$\mathsf{Adv}^{\mathrm{conf}}_{\mathrm{IVE}}(\mathbb{A}_{\mathrm{conf}}) \leq \mathsf{Adv}^{\mathrm{conf}}_{\mathrm{PKE}}(\mathbb{B}_{\mathrm{conf}}) + \mathsf{Adv}^{\mathrm{ou}}_{\mathrm{UIBSS}}(\mathbb{B}_{\mathrm{ou}}) \ .$$

Proof. We introduce a series of games for the adversary $\mathbb{A}_{\mathrm{conf}}$ to play, gradually changing the original game into a distinguishing game against the underlying encryption scheme.

Game $\mathbf{G}^{b^*}_{\mathbf{0}}$: This is the original game applied to our construction, presented in Fig. 8.The advantage of $\mathbb{A}_{\mathrm{conf}}$ may be expressed as $\mathsf{Adv}^{\mathrm{conf}}_{\mathrm{IVE}}(\mathbb{A}_{\mathrm{conf}}) = \Pr\left[\mathbf{G}^{0}_{\mathbf{0}} : \mathbb{A}_{\mathrm{conf}} \rightarrow 1\right] - \Pr\left[\mathbf{G}^{1}_{\mathbf{0}} : \mathbb{A}_{\mathrm{conf}} \rightarrow 1\right]$.

Game $\mathbf{G}^{b^*}_{\mathbf{1}}$: Here, we change the decryption procedure, so that instead of demanding that a query $((C, \sigma), ID) \notin \mathcal{C}$, we require only that C has not been part of a challenge recieved from the encryption oracle. An adversary able to distinguish between these two games would also be able to find two distinct and verifying signatures

Game $\mathbf{G_0}^{b^*}$

$(PK, DK) \leftarrow\$ \mathsf{Pke.Kg}$
$(VK, SK) \leftarrow \mathsf{Uibss.Kg}$
$\mathcal{C} \leftarrow \emptyset$
$\hat{b} \leftarrow \mathbb{A}^{O}((PK, VK)$

encrypt$((M_0, ID, USK), (M_1, ID, USK))$

$C_{b^*} \leftarrow\$ \mathsf{Pke.Enc}_{PK}(M_{b^*} \| ID)$
$\sigma_{b^*} \leftarrow\$ \mathsf{Uibss.Sign}_{USK}(C_{b^*})$
$C^* \leftarrow ((C_{b^*}, \sigma_{b^*}), ID)$
$\mathcal{C} \leftarrow \mathcal{C} \cup \{C^*\}$ **return** C^*

derive(ID)

$USK \leftarrow \mathsf{Uibss.Derive}_{MSK}(ID)$
return USK

decrypt(C, π)

require $(C, \sigma) \notin \mathcal{C}$
if $\mathsf{Uibss.Verify}^{ID}_{VK}(C, \sigma) = \perp$
  return $\perp$
$M \| ID \leftarrow \mathsf{Pke.Dec}_{DK}(C)$
if decryption or parsing fails
  return $\perp$
return M

Fig. 8. Game $\mathbf{G}_1^{b^*}$ for the confidentiality proof of the Encrypt-then-IBS construction.

on the same message. It follows that the difference between $\mathbf{G}_0^{b^*}$ and $\mathbf{G}_1^{b^*}$ may be bounded by the advantage an adversary has of breaking the outsider unicity of the underlying signature scheme.

Given this, we may construct a reduction from $\mathbf{G}_1^{b^*}$ to a standard indistinguishability game of the underlying public key encryption scheme in the following way: an adversary $\mathbb{B}_{\text{cca}}$ given the public key PK of an encryption scheme generates the keys (VK, MSK) for an unique identity based signature scheme, and sends (PK, VK) to the adversary $\mathbb{A}_{\text{conf}}$. Any derivation queries may be answered by $\mathbb{B}_{\text{cca}}$ alone, seeing as she possesses MSK. Whenever $\mathbb{A}_{\text{conf}}$ sends a challenge query (M_0, M_1, ID, USK), $\mathbb{B}_{\text{cca}}$ sends $(M_0 \| ID, M_1 \| ID)$ to her encryption oracle, and when she gets the challenge ciphertext back, she signs it using the user secret key USK before sending the tuple to $\mathbb{A}_{\text{conf}}$. Any decryption query is handled by $\mathbb{B}_{\text{cca}}$ first verifying the signature σ, and sending C to her own decyrption oracle if the signature verifies, and passing on the response from the oracle to $\mathbb{A}_{\text{conf}}$. Once $\mathbb{A}_{\text{conf}}$ guesses $\hat{b}$, $\mathbb{B}_{\text{cca}}$ copies it, and so it follows that $\mathsf{Adv}^{\text{conf}}_{\text{IVE}}(\mathbb{A}_{\text{conf}}) \leq \mathsf{Adv}^{\text{conf}}_{\text{PKE}}(\mathbb{B}_{\text{conf}}) + \mathsf{Adv}^{\text{ou}}_{\text{UIBSS}}(\mathbb{B}_{\text{ou}})$.

□

3.3 Discussion of IVE

IVE resembles identity-based signcryption in many ways, as both primitives offer confidentiality of messages and integrity of communication between two individuals identifiable to each other. In both cases, this concerns insider security: reading the message requires nothing less than the secret key/decryption key of the recipient, and forging the signature of a sender requires the user key/private key of that particular sender. There is also a notion of verification in identity based signcryption, which guarantees that a decrypted message was in fact written by the sender [21].

In addition, it is common for identity based signcryption to satisfy the security notion of ciphertext unlinkability: it is not possible to link a sender to a specific ciphertext, even if the ciphertext decrypts to a message signed by the sender in question. Another security notion relevant for identity based signcryption is insider ciphertext anonymity, which informally means that deducing either the sender or recipient of a given ciphertext requires the private key of the recipient [21].

It is obvious that the two latter security notions do not combine with a central feature of IVE, namely public verification that a sender has in fact been vetted, seeing as the verification algorithm takes the sender identity as input. To filter out messages sent from unvetted individuals is an essential part of IVE, and this does require a public verification algorithm.

There are identity based signcryption schemes that offer such public verification. However, several of the schemes require the receiver to collaborate by supplying the verification algorithm with additional information. For example, in the signcryption scheme proposed by Libert and Quisquater, the receiver has to supply the verifier with an ephemeral key [38]. Again, this runs counter to the idea of IVE, namely that the filter is able to do the filtering without assistance from the recipient. Querying the recipient to check whether a message is sent from a vetted sender renders the filter pointless.

Finally, there does exist identity based signcryption schemes which offer *transferable public verification*. In these schemes, it is possible for a third party to verify that a ciphertext has indeed been signed by the alleged sender, without help from the receiver. To the best of our knowledge, there are only two such schemes, and both of them adopt an encrypt-and-sign approach [42, 44], where the former does not have a proof of security.

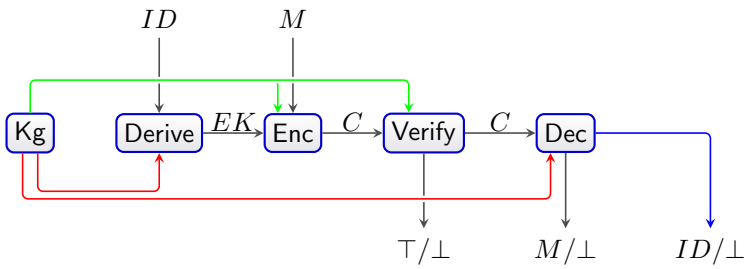

Fig. 9. The algorithms and their inputs/outputs for opaque vetted encryption.

The scheme which is provably secure is based on bilinear pairings, requires very large public parameters, and produces ciphertexts of a large size. We believe that our more general approach might result in a concrete scheme with more favourable sizes, both with regards to parameters and ciphertext size.

4 Opaque Vetted Encryption (OVE)

4.1 Syntax and Security of OVE

The algorithms. For opaque vetted encryption, we are in the most challenging, 'asymmetric' scenario where the filter does *not* learn the identity of the sender, yet the recipient does. In the definition below, we model this change by letting the identity be *output* as part of decryption, in addition to the message of course. Having two outputs also affects how invalid ciphertexts are dealt with: for our syntax and security we deal with the general case where either component can lead to rejection, independently of each other. Thus we allow a large number of error messages, unlike the AVE or IVE case, where only a single error message was modeled.

As we will see, OVE is quite similar to a group signature with message recovery, which seems to be an overlooked primitive. In line with the literature on group signatures, in Definition 3 we split the private key in two: an issuing key IK to derive identity-specific encryption keys and a master decryption key SK to decrypt ciphertexts. Throughout we will also borrow group signature terminology, for instance by referring to the derivation of an encryption key as 'issuing' (of course, in the group signature setting, said key would be a signing key instead), or use 'opening' to extract the identity from a ciphertext as part of decryption.

Definition 3 (Opaque Vetted Encryption (OVE)). *An* opaque vetted encryption *scheme* OVE *is a 5-tuple of algorithms* (Ove.Kg, Ove.Derive, Ove.Enc, Ove.Verify, Ove.Dec) *that satisfy*

- Ove.Kg generates *a key triple* (PK, SK, IK)*, where* PK *is the public encryption (and verification) key,* SK *is the private decryption key, and* IK *is the issuing key. We allow* Ove.Kg *to depend on parameters* $param$ *and write* $(PK, SK, IK) \leftarrow_\$ \mathsf{Ove.Kg}(param)$*. Henceforth, we will assume that* PK *can be uniquely and efficiently computed given either* SK *or* IK*.*
- Ove.Derive issues *an encryption key* EK *based on the issuing key* IK *and a user's identity* ID*. We write* $EK \leftarrow_\$ \mathsf{Ove.Derive}_{IK}(ID)$*.*
- Ove.Enc encrypts *a message* M *given the public encryption key* PK *and private encryption key* EK*, producing a ciphertext* C *or a failed encryption symbol* $\perp$*. So,* $C \leftarrow_\$ \mathsf{Ove.Enc}_{PK,EK}(M)$ *with maybe* $C = \perp$*.*
- Ove.Verify verifies *the validity of a ciphertext* C *given the public verification key* PK*. With a slight abuse of notation,* $\top/\perp \leftarrow \mathsf{Ove.Verify}_{PK}(C)$*.*
- Ove.Dec decrypts *a ciphertext* C *using the private key* SK*, resulting in a message–identity pair* (M, ID)*. Both the message* M *and the identity* ID *may, independently of each other, result in a rejection,* $\perp$*. Again, with a slight abuse of notation,* $(M/\perp, ID/\perp) \leftarrow \mathsf{Ove.Dec}_{SK}(C)$*.*

The first three algorithms are probabilistic, the final two deterministic.

Correctness and consistency. As is the case for AVE and IVE, *correctness* captures that honest usage of the scheme ensures that messages are received as intended, and assigned to the actual sender. For all parameters $param$, identities ID and messages M we have

$\underline{\mathsf{Exp}^{\mathrm{trac}}_{\mathrm{OVE}}(\mathbb{A})}$
$(PK, SK, IK) \leftarrow\!\!\$\ \mathsf{Ove.Kg}$
$h \leftarrow 0; C \leftarrow \emptyset$
$\mathcal{CU} \leftarrow \{\perp\}$
$\hat{C} \leftarrow \mathbb{A}^{O}(PK)$
$(M, ID) \leftarrow \mathsf{Ove.Dec}_{SK}(\hat{C})$
winif $\hat{C} \notin C \wedge ID \notin \mathcal{CU}$
$\quad \wedge\ \mathsf{Ove.Verify}_{PK}(\hat{C}) = \top$

$\underline{\mathrm{derive}(ID)}$
$EK[h] \leftarrow \mathsf{Ove.Derive}_{IK}(ID)$
$h \leftarrow h + 1$
return h

$\underline{\mathrm{corrupt}(H)}$
$\mathcal{CU} \leftarrow \mathcal{CU} \cup \{H.ID\}$
return $EK[H]$

$\underline{\mathrm{encrypt}(H, M)}$
$C \leftarrow\!\!\$\ \mathsf{Ove.Enc}_{PK, EK[H]}(M)$
$C \leftarrow C \cup \{C\}$
return C

$\underline{\mathrm{decrypt}(C)}$
$(M, ID) \leftarrow \mathsf{Ove.Dec}_{SK}(C)$
return (M, ID)

Fig. 10. The traceability game for OVE.

$\underline{\mathsf{Exp}^{\mathrm{int}}_{\mathrm{OVE}}(\mathbb{A})}$
$(PK, SK, IK) \leftarrow\!\!\$\ \mathsf{Ove.Kg}$
$\hat{C} \leftarrow \mathbb{A}^{O}(PK, SK, IK)$
$(M, ID) \leftarrow \mathsf{Ove.Dec}_{SK}(\hat{C})$
winif $\mathsf{Ove.Verify}_{PK}(\hat{C}) = \top \wedge (M = \perp \vee ID = \perp)$

Fig. 11. The integrity game for OVE.

$$\Pr\left[\begin{array}{c|c} (PK, SK, IK) \leftarrow\!\!\$\ \mathsf{Ove.Kg}(param) & C \neq \perp \\ EK \leftarrow\!\!\$\ \mathsf{Ove.Derive}_{IK}(ID) \quad : & \wedge\ \mathsf{Ove.Verify}_{PK}(C) = \top \\ C \leftarrow\!\!\$\ \mathsf{Ove.Enc}_{PK,EK}(M) & \wedge\ \mathsf{Ove.Dec}_{SK}(C) = (M, ID) \end{array}\right] = 1.$$

As with the previously presented schemes, *consistency* means that any ciphertext which decrypts to a valid message and identity, will also pass the filter. Thus we treat any occurrence of $\perp$ in the decryption, as either message or identity, as an invalid ciphertext. Again, we can easily transform a correct scheme into one that is consistent as well: as part of decryption, run the verification, and if verification returns $\perp$, then decryption returns $(\perp, \perp)$.

Security. The security of OVE is an amalgam of the vetted encryption notions we have encountered so far and those for group signatures, primarily the static "BMW" notions [13]. The integrity component we saw earlier now splits into two: on the one hand, we want that ciphertexts that pass the filter (so verify) can be pinned to a user after decryption, yet on the other hand we want to avoid users being falsely suspected of spamming (by an honest recipient). We relabel the first notion integrity and strengthen it slightly, so it becomes essentially a computational equivalent of strictness. The second notion is traceability, known from group signatures. We also require confidentiality of the messages and anonymity of the senders, but it turns out we can fold these two concepts into a single notion, dubbed privacy. Formally, we define the following advantages, with specifications and explanations of the corresponding experiments described below:

- *Traceability*: $\mathsf{Adv}^{\mathrm{trac}}_{\mathrm{OVE}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\mathrm{trac}}_{\mathrm{OVE}}(\mathbb{A}) : \mathbb{A} \text{ wins}\right]$.
- *Integrity*: $\mathsf{Adv}^{\mathrm{int}}_{\mathrm{OVE}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\mathrm{int}}_{\mathrm{OVE}}(\mathbb{A}) : \mathbb{A} \text{ wins}\right]$.
- *Privacy*: $\mathsf{Adv}^{\mathrm{priv}}_{\mathrm{OVE}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\mathrm{priv\text{-}0}}_{\mathrm{OVE}}(\mathbb{A}) : \hat{b} = 0\right] - \Pr\left[\mathsf{Exp}^{\mathrm{priv\text{-}1}}_{\mathrm{OVE}}(\mathbb{A}) : \hat{b} = 0\right]$.

Traceability. This notion (Fig. 10) ensures that a colluding group of vetted users cannot successfully create a ciphertext that opens to the identity of another user (outside the collusion). As we do not incorporate a PKI in our model (cf. the dynamic "BSZ" notions for group signatures [16]), we need to exclude the issuing key IK from the adversary's grasp. Furthermore, in contrast to BMW's traceability, we also do not provide the decryption key DK to the adversary. Our weakening is motivated by the intended use case: the main purpose of the scheme is to trace messages which pass the filter back to an identity and the recipient has no motive to try and create ciphertexts that it will then subsequently open and trace incorrectly. Of course, in order to provide forward security, one could also consider *strong traceability*, where an adversary does have access to the decryption key SK.

$\underline{\mathrm{Exp}^{\text{priv-}b^*}_{\mathrm{OVE}}(\mathbb{A})}$

$(PK, SK, IK) \leftarrow\$ \mathsf{Ove.Kg}$
$\mathcal{C} \leftarrow \emptyset$
$\hat{b} \leftarrow \mathbb{A}^{\mathcal{O}}(PK, IK)$

$\underline{\mathrm{decrypt}(C)}$

require $C \notin \mathcal{C}$
$(M, ID) \leftarrow \mathsf{Ove.Dec}_{SK}(C)$
return (M, ID)

$\underline{\mathrm{encrypt}(EK_0, EK_1, M_0, M_1)}$

$C_0 \leftarrow\$ \mathsf{Ove.Enc}_{PK,EK_0}(M_0)$
$C_1 \leftarrow\$ \mathsf{Ove.Enc}_{PK,EK_1}(M_1)$
if $C_0 \neq \perp \wedge C_1 \neq \perp$ **then**
  $C^* \leftarrow C_{b^*}$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{C^*\}$
else
  $C^* \leftarrow \perp$
return C^*

$\underline{\mathrm{encrypt}_x(EK_0, EK_1, M_0, M_1)}$

$C_0 \leftarrow\$ \mathsf{Ove.Enc}_{PK,EK_0}(M_0)$
$C_1 \leftarrow\$ \mathsf{Ove.Enc}_{PK,EK_1}(M_1)$
$C_x \leftarrow\$ \mathsf{Ove.Enc}_{PK,EK_1}(M_0)$
if $C_0 \neq \perp \wedge C_1 \neq \perp$ **then**
  if $C_x = \perp$ **then** set bad
  $C^* \leftarrow C_x$
  $\mathcal{C} \leftarrow \mathcal{C} \cup \{C^*\}$
else
  $C^* \leftarrow \perp$
return C^*

Fig. 12. The privacy game for OVE (first three columns); the final column is used in the proof of Lemma 3.

Finally, we initialize $\mathcal{CU}$ to contain $\perp$ as we consider the case where the ciphertext opens to an invalid identity, so $\mathsf{Ove.Dec}_{SK}(C) = (M, \perp)$, only as a breach of integrity, not of traceability. Again, this fits the intended use case: an adversary being able to pass the filter without being identified afterwards can effectively "spam" the receiver, who then does not know which sender to have a word with. As the protection against spamming is the raison d'être of our scheme, we will put much stronger guarantees in place to prevent it (as part of integrity).

Integrity. In stark contrast to traceability, *integrity* ensures that even an adversary in possession of all the keys of the scheme cannot create a message which verifies, so $\mathsf{Ove.Verify}_{PK}(C) = \top$, yet does not open to a valid message–identity pair, i.e. leads to $\mathsf{Ove.Dec}_{SK}(C) = (\perp, ID)$, $\mathsf{Ove.Dec}_{SK}(C) = (M, \perp)$ or $\mathsf{Ove.Dec}_{SK}(C) = (\perp, \perp)$. Thus any ciphertext that passes the verification, is opened without a failure message.

We reiterate that we treat $\mathsf{Ove.Dec}_{SK}(C) = (M, \perp)$ as a breach of integrity rather than traceability. One interpretation is that C decrypted successfully to an anonymous message. Yet allowing for anonymous messages would clearly defeat the purpose of opaque vetted encryption, namely that any ciphertext which verifies can be attributed to a vetted sender.

Privacy, confidentiality, and anonymity. Any party not in possession of the decryption key should be unable to determine who is the sender of a ciphertext, and also what the ciphertext decrypts to. Note that we allow an adversary access to the issuing key IK. This is seemingly a contradiction to the discussed honest use case, where the recipient both issues keys and decrypts messages, which was after all the reasoning for denying the adversary the opening key in the traceability case. However, there is a possible separation of authorities, and even though we regard the recpient as the "owner" of the scheme, they may choose to delegate the authority of issuing keys to another authority. We require that even this party should not be able to infer the sender or the content when given a ciphertext.

We formalize this notion as *privacy* (Fig. 12), which we model with a challenge encryption oracle that an adversary can query on two pairs of encryption keys and messages: (EK_0, M_0) and (EK_1, M_1). The oracle either returns an encryption of the left, 0-subscripted or the right, 1-subscripted key–message pair; the adversary should figure out which one. To avoid trivial wins based on faulty encryption keys, we encrypt both pairs, and reject the query if one of the encryptions fail. Privacy should hold even against adversaries knowing the issuing key IK. Our notion of *privacy* encompasses both *anonymity* and *confidentiality* of encryption schemes. We define anonymity as the privacy game with the restriction that for all challenge queries $M_0 = M_1$, and confidentiality as the privacy game where we insist $EK_0 = EK_1$ for all challenge queries.

The resulting anonymity game resembles anonymity known from group signatures. One notable difference is the additional mechanism we put in place by encrypting under both encryption keys and only output the ciphertext if both encryptions are successful. We are not aware of a similar mechanism to define anonymity of group signatures, i.e. where you would sign under both user signing keys and only release the group signature if both are successful: BMW only deal with honestly generated keys and BSZ have a join protocol that alleviates the need for an additional check.

For confidentiality, arguably one could consider a stronger game where one directly encrypts the relevant challenge message under the adversarially chosen key. Yet, this strenghtening is not entirely without gain of generality, as one could concoct a pathological counterexample where for some fake encryption key some messages are more likely to result in an encryption error than others. Henceforth, we will ignore this subtlety.

By definition, privacy obviously implies anonymity and confidentiality (with a small caveat for the latter, as explained above). The converse is true as well, namely that *jointly* anonymity and confidentiality imply privacy. However, in general this is not true, as can be shown by a simple, pathological counterexample.

Consider a scheme that is secure, now modify the scheme so that key derivation prepends keys with a 0-bit. Encryption with a key starting with a 0-bit removes this bit and behaves as before. This fully describes the honest behaviour of the scheme and we proceed to describe behaviour that could only be triggered by an adversary: namely, our modified scheme's encryption with a key starting with a 1-bit outputs the message iff that message equals the key, and rejects otherwise. Essentially, all 1-keys are fake, but it is possible to make each key accept on a single message (and each message can only be used for a single fake key). For the confidentiality and anonymity games, these fake keys cannot be exploited as the reject-filtering mechanism causes the oracle to reject; for the privacy game however it's easy to win exploiting these fake keys.

For schemes that behave nicely however, we show in Lemma 3 that the privacy game is implied by combination of anonymity and confidentiality. Here 'nicely' refers to the property that an encryption key is either always successful on the full message space, or it always rejects.

Lemma 3 (OVE-Anonymity + OVE-Confidentiality implies OVE-Privacy). *Let* OVE *sport encryption keys EK with the property that for all messages M in the message space,* $\mathsf{Ove.Enc}_{PK,EK}(M) = \perp$*, or every message encrypts to a ciphertext with probability 1. Then for any privacy adversary* $\mathbb{A}_{\mathrm{priv}}$ *against an* OVE *scheme, there exist anonymity and confidentiality adversaries* $\mathbb{B}_{\mathrm{conf}}$ *and* $\mathbb{B}_{\mathrm{anon}}$ *of comparable efficiency such that*

$$\mathsf{Adv}^{\mathrm{priv}}_{\mathrm{OVE}}(\mathbb{A}_{\mathrm{priv}}) \leq \mathsf{Adv}^{\mathrm{anon}}_{\mathrm{OVE}}(\mathbb{B}_{\mathrm{anon}}) + \mathsf{Adv}^{\mathrm{conf}}_{\mathrm{OVE}}(\mathbb{B}_{\mathrm{conf}}) \,.$$

Proof. First, we define the games we will use throughout the proof. In all cases, the challenge oracle receives $((EK_0, M_0), (EK_1, M_1))$, but different inputs are selected for encryption as the challenge ciphertext:

- $\mathbf{G_0}$: the challenge oracle chooses (EK_0, M_0);
- $\mathbf{G_1}$: the challenge oracle chooses (EK_1, M_1);
- $\mathbf{G_x}$: the challenge oracle chooses (EK_1, M_0).

Furthermore all three games, including $\mathbf{G_x}$ use the first two cases to decide whether to reject a query (output $\perp$) or not. In the case of $\mathbf{G_x}$, if the encryption itself fails but the check is passed, we set a flag bad. The code for the encryption oracle of $\mathbf{G_x}$ is provided in Fig. 12.

We may express the advantage of $\mathbb{A}_{\mathrm{priv}}$ as:

$$\begin{aligned}\mathsf{Adv}^{\mathrm{priv}}_{\mathrm{OVE}}(\mathbb{A}_{\mathrm{priv}}) &= \Pr\left[\mathbf{G_0} \,:\, \mathbb{A}_{\mathrm{priv}} \to 0\right] - \Pr\left[\mathbf{G_1} \,:\, \mathbb{A}_{\mathrm{priv}} \to 0\right] \\ &= \Pr\left[\mathbf{G_0} \,:\, \mathbb{A}_{\mathrm{priv}} \to 0\right] - \Pr\left[\mathbf{G_x} \,:\, \mathbb{A}_{\mathrm{priv}} \to 0\right] \\ &+ \Pr\left[\mathbf{G_x} \,:\, \mathbb{A}_{\mathrm{priv}} \to 0\right] - \Pr\left[\mathbf{G_1} \,:\, \mathbb{A}_{\mathrm{priv}} \to 0\right] .\end{aligned}$$

We claim existence of $\mathbb{B}_{\mathrm{anon}}$ and $\mathbb{B}_{\mathrm{conf}}$ such that

$$\Pr\left[\mathbf{G_0} \,:\, \mathbb{A}_{\mathrm{priv}} \to 0\right] - \Pr\left[\mathbf{G_x} \,:\, \mathbb{A}_{\mathrm{priv}} \to 0\right] \leq \mathsf{Adv}^{\mathrm{anon}}_{\mathrm{OVE}}(\mathbb{B}_{\mathrm{anon}})$$

as well as

$$\Pr\left[\mathbf{G_x} \,:\, \mathbb{A}_{\mathrm{priv}} \to 0\right] - \Pr\left[\mathbf{G_1} \,:\, \mathbb{A}_{\mathrm{priv}} \to 0\right] \leq \mathsf{Adv}^{\mathrm{conf}}_{\mathrm{OVE}}(\mathbb{B}_{\mathrm{conf}}) \,.$$

We prove the first claim: given a privacy adversary $\mathbb{A}_{\mathrm{priv}}$, we may construct an anonymity adversary in the following way: $\mathbb{B}_{\mathrm{anon}}$ gets input (PK, IK), which she passes along to $\mathbb{A}_{\mathrm{priv}}$. When $\mathbb{A}_{\mathrm{priv}}$ sends her challenge request $((EK_0, M_0), (EK_1, M_1))$, $\mathbb{B}_{\mathrm{anon}}$ first encrypts (EK_1, M_1) herself. If this results in $\perp$, she sends a rejection to $\mathbb{A}_{\mathrm{priv}}$, simulating the response from a privacy encryption oracle. If $\mathsf{Ove.Enc}_{PK,EK_1}(M_1) \neq \perp$, then $\mathbb{B}_{\mathrm{anon}}$ sends the requests $((EK_0, M_0), (EK_1, M_0))$ to her challenge oracle. By the assumption that an encryption key will either encrypt all messages or none, this cannot result in the bad event $\mathsf{Ove.Enc}_{PK,EK_1}(M_0) = \perp$. Thus, if the encryption oracle returns $\perp$, this is caused by (EK_0, M_0), and the rejection is therefore in line with a privacy encryption oracle. Once $\mathbb{B}_{\mathrm{anon}}$ receives the challenge ciphertext, she passes it to $\mathbb{A}_{\mathrm{priv}}$. Any decryption

query made by $\mathbb{A}_{\text{priv}}$ is answered by $\mathbb{B}_{\text{anon}}$'s decryption oracle. When $\mathbb{A}_{\text{priv}}$ outputs a bit b, $\mathbb{B}_{\text{anon}}$ answers the same, and will thus have the same advantage in her game as $\mathbb{A}_{\text{priv}}$ has in hers. The claim follows.

The second claim is proven anologously: given a privacy adversary $\mathbb{A}_{\text{priv}}$, we may construct a confidentiality adversary as follows: $\mathbb{B}_{\text{conf}}$ gets input (PK, IK), which she passes along to $\mathbb{A}_{\text{priv}}$. When $\mathbb{A}_{\text{priv}}$ queries a challenge by sending $((EK_0, M_0), (EK_1, M_1))$, $\mathbb{B}_{\text{conf}}$ encrypts (EK_0, M_0) herself, and rejects the query if the encryption results in $\perp$. This simulates the rejection from a privacy encryption oracle. If she does not reject, $\mathbb{B}_{\text{conf}}$ sends the requests $((EK_1, M_0), (EK_1, M_1)$ to her challenge oracle, and sends the challenge ciphertext she receives to $\mathbb{A}_{\text{priv}}$. Again, if the encryption oracle rejects, this is caused by (EK_1, M_1), and is in line with the behaviour of a privacy encryption oracle. Given the assumption of valid or invalid encryption keys, the bad event $\mathsf{Ove.Enc}_{PK,EK_1}(M_0) = \perp$ does not happen. Any decryption query made by $\mathbb{A}_{\text{priv}}$ is answered by $\mathbb{B}_{\text{conf}}$'s decryption oracle. When $\mathbb{A}_{\text{priv}}$ outputs a bit b, $\mathbb{B}_{\text{conf}}$ answers the same, and will thus have the same advantage in her game as $\mathbb{A}_{\text{priv}}$ has in hers. The claim follows.

Based on these steps, we have:

$$\mathsf{Adv}^{\text{priv}}_{\text{OVE}}(\mathbb{A}_{\text{priv}}) \leq \mathsf{Adv}^{\text{anon}}_{\text{OVE}}(\mathbb{B}_{\text{anon}}) + \mathsf{Adv}^{\text{conf}}_{\text{OVE}}(\mathbb{B}_{\text{conf}}) \ .$$

□

4.2 Generic Construction: Verifiably Encrypted Certificates

Our construction is inspired by the sign-encrypt-proof construction for group signature schemes [13]. This provenance is natural, given the close relationship between OVE and group signatures (albeit with message recovery). The most important difference, aside from having to keep the message confidential, is our weakening of traceability, by not availing the adversary with the decryption key. We reflect on the difference between our scheme and known group signature schemes in Section 4.3.

Our scheme uses an IND-CCA secure PKE, an EUF-CMA secure SIG and a simulation-sound QANIZK; the construction is fleshed out in Fig. 13. The key generation algorithm generates the key pairs (PK, DK), (VK, SK) for the PKE and SIG respectively, as well as the crs σ and trapdoor τ for the QANIZK scheme. The public key for the OVE is the triple (PK, VK, σ), the derivation key is SK, and finally the decryption key is DK. We stress that the trapdoor τ is discarded after derivation: it is used only in the security reductions, not in the actual scheme itself, and accidentally including it in the private derivation or decryption key would actually invalidate integrity!

For a given user with identity ID, the derivation issues a certificate $CERT_{ID}$ by signing ID using the signature scheme. The certificate may then be regarded as the encryption key of the user with identity ID.

To encrypt a message M, on input the public key of the OVE as well as the identity ID and certificate $CERT_{ID}$ of the encryptor, first the validity of the certificate is checked to guard against dishonest certificates. If the certificate passes, the concatenated string $M\|CERT_{ID}\|ID$ is encrypted to C using the underlying encryption scheme. Next, a QANIZK proof π is generated for the statement that the ciphertext is created honestly, specifically that it contains a valid $ID, CERT_{ID}$ pair. The OVE encryption algorithm finally outputs (C, π).

Formally, for the QANIZK proof, the language $L_{(PK,VK)}$ is determined by the public key (PK, VK) and consists of valid ciphertexts, i.e.,

$$L_{(PK,VK)} = \{C : \exists_{M,r,CERT_{ID},ID}\ C = \mathsf{Pke.Enc}_{PK}(M\|CERT_{ID}\|ID; r) \wedge \mathsf{Sig.Verify}_{VK}(ID, CERT_{ID}) = \top\}.$$

Thus the message M and the randomness r used to encrypt are additional witnesses used to create the QANIZK proof π; the full witness is the tuple $(r, M, CERT_{ID}, ID)$.

For the filter to verify a pair (C, π), it simply runs the verification algorithm of the QANIZK scheme, with the public key of the OVE scheme as well as (C, π) as input.

Finally, in order to decrypt an OVE ciphertext (C, π), the receiver first verifies the proof π using the verification algorithm of the QANIZK. If the QANIZK verification fails, the receiver rejects. Otherwise, it decrypts C and attempts to parse the output as $M\|CERT_{ID}\|ID \leftarrow \mathsf{Pke.Dec}_{DK}(C)$. If either decryption or parsing fails, the receiver rejects. If both succeed, it returns (M, ID). There is no need to explicitly run the verification algorithm of the signature scheme on the certificate as its validity is already implicitly checked by the QANIZK verification. Note that we output the rejection symbol $(\perp, \perp)$ in all cases (failure of the verification, decryption, or parsing), and in particular that we do not distinguish between a failure to decrypt the message M or the identity ID, as the syntax (Fig. 9) allows for.

$\underline{\mathsf{Ove.Kg}()}$

$(PK, DK) \leftarrow\!\!\$\ \mathsf{Pke.Kg}$
$(VK, SK) \leftarrow\!\!\$\ \mathsf{Sig.Kg}$
$(\sigma, \tau) \leftarrow\!\!\$\ \mathsf{Nizk.Setup}$
return $((PK, VK, \sigma), DK, SK)$

$\underline{\mathsf{Ove.Derive}_{SK}(ID)}$

$CERT_{ID} \leftarrow\!\!\$\ \mathsf{Sig.Sign}_{SK}(ID)$
return $CERT_{ID}$

$\underline{\mathsf{Ove.Verify}_{PK,VK}(C, \pi)}$

return $\mathsf{Nizk.Verify}_{PK,VK,\sigma}(C, \pi)$

$\underline{\mathsf{Ove.Enc}^{ID}_{PK,VK,\sigma,ID,CERT_{ID}}(M)}$

if $\mathsf{Sig.Verify}_{VK}(ID, CERT_{ID}) = \bot$, **return** $\bot$
$C \leftarrow\!\!\$\ \mathsf{Pke.Enc}_{PK}(M \| CERT_{ID} \| ID; r)$
$\pi \leftarrow \mathsf{Nizk.Prove}_{PK,VK,\sigma}(r, M, CERT_{ID}, ID)$
return (C, π)

$\underline{\mathsf{Ove.Dec}_{DK}(C, \pi)}$

if $\mathsf{Nizk.Verify}_{PK,VK,\sigma}(C, \pi) = \bot$
  return $(\bot, \bot)$
$M \| CERT_{ID} \| ID \leftarrow \mathsf{Pke.Dec}_{DK}(C)$
if decryption or parsing fails
  return $(\bot, \bot)$
return (M, ID)

Fig. 13. Our "Verifiably Encrypted Certificate" construction for OVE.

Game $\mathbf{G_0}$

$(PK, DK) \leftarrow\!\!\$\ \mathsf{Pke.Kg}$
$(VK, SK) \leftarrow \mathsf{Sig.Kg}$
$(\sigma, \tau) \leftarrow\!\!\$\ \mathsf{Nizk.Setup}$
$h \leftarrow 0; C \leftarrow \emptyset$
$\mathcal{CU} \leftarrow \emptyset$
$(\hat{C}, \hat{\pi}) \leftarrow \mathbb{A}^{O}(PK, VK, \sigma)$
$(M, ID) \leftarrow \mathsf{Ove.Dec}_{PK}(\hat{C})$
winif $(\hat{C}, \hat{\pi}) \notin C \wedge ID \notin \mathcal{CU} \wedge$
  $\mathsf{Ove.Verify}_{PK,VK,\sigma}(\hat{C}, \hat{\pi}) = \top$

derive(ID)

$ID_h = ID$
$CERT_{ID_h} \leftarrow \mathsf{Sig.Sign}_{SK}(ID_h)$
$h \leftarrow h + 1$
return h

corrupt(H)

$\mathcal{CU} \leftarrow \mathcal{CU} \cup \{ID_H\}$
return $CERT_{ID_H}$

encrypt(H, M)

$C \leftarrow\!\!\$\ \mathsf{Pke.Enc}_{PK}(M \| CERT_{ID_H} \| ID_H; r)$
$\pi \leftarrow \mathsf{Nizk.Prove}_{PK,VK,\sigma}(r, M, CERT_{ID_H}, ID_H)$
$C \leftarrow C \cup \{(C, \pi)\}$
return (C, π)

decrypt(C, π)

if $\mathsf{Nizk.Verify}_{PK,VK,\sigma}(C, \pi) = \bot$
  return $(\bot, \bot)$
$M \| CERT_{ID} \| ID \leftarrow \mathsf{Pke.Dec}_{DK}(C)$
if decryption or parsing fails
  return $(\bot, \bot)$
return (M, ID)

Fig. 14. The initial traceability game $\mathbf{G_0}$ for our Verifiably Encrypted Certificate construction for OVE.

Correctness and consistency. Correctness follows from the correctness of the underlying PKE and SIG, as well as the completeness of the QANIZK. Consistency is guaranteed by checking the proof in decryption, as this ensures that any ciphertext which decrypts also passes the filter.

Traceability. Intuitively, the traceability of the scheme boils down to the unforgeability of the signature scheme used as a building block. The other properties of the PKE and QANIZK ensure that the encryption oracle is harmless, i.e. that the returned components (C, π) do not leak any information about the valid and potentially honest certificate used.

Lemma 4 (Traceability of OVE). *For all adversaries* $\mathbb{A}_{\text{trac}}$*, there exist similarly efficient adversaries* $\mathbb{B}_{\text{sound}}$*,* $\mathbb{B}_{\text{zk}}$*,* $\mathbb{B}_{\text{cca}}$ *and* $\mathbb{B}_{\text{euf-cma}}$ *such that*

$$\begin{aligned}\mathsf{Adv}^{\text{trac}}_{\text{OVE}}(\mathbb{A}_{\text{trac}}) \leq & \mathsf{Adv}^{\text{sound}}_{\text{QANIZK}}(\mathbb{B}_{\text{sound}}) + \mathsf{Adv}^{\text{zk}}_{\text{QANIZK}}(\mathbb{B}_{\text{zk}}) \\ & + \mathsf{Adv}^{\text{cca}}_{\text{PKE}}(\mathbb{B}_{\text{cca}}) + \mathsf{Adv}^{\text{euf-cma}}_{\text{SIG}}(\mathbb{B}_{\text{euf-cma}}).\end{aligned}$$

Proof. We introduce a series of games which the adversary $\mathbb{A}_{\text{trac}}$ plays, rendering the encryption oracle less and less potent. We bound the advantage between the games using various reductions $\mathbb{B}_{\ldots}$, to finally conclude with a reduction linking the advantage in the final game to the EUF-CMA-advantage against the signature scheme.

Game $\mathbf{G_0}$: This is the original traceability game as presented in Fig. 10, see Fig. 14 for the adaption to our OVE scheme. We note that

$$\begin{aligned}\mathsf{Adv}^{\mathrm{trac}}_{\mathrm{OVE}}(\mathbb{A}_{\mathrm{trac}}) &= \Pr[\,\mathbb{A}_{\mathrm{trac}} \text{ wins } G_0\,] \\ &= \Pr\left[\,\mathbb{A}_{\mathrm{trac}} \text{ wins } G_0 \wedge C \in L_{(PK,VK)}\right] + \Pr\left[\,\mathbb{A}_{\mathrm{trac}} \text{ wins } G_0 \wedge C \notin L_{(PK,VK)}\right]\ ,\end{aligned}$$

where the final probability can be bounded by the advantage of a soundness adversary $\mathbb{B}_{\mathrm{sound}}$ attacking the underlying QANIZK scheme. Henceforth we assume that $\mathbb{A}_{\mathrm{trac}}$ only wins with a valid ciphertext, $C \in L_{(PK,VK)}$.

Game $\mathbf{G_1}$: This is the same as $\mathbf{G_0}$, except for the generation of π during the encryption query. Instead of generating it using Nizk.Prove, the challenger now uses a simulator. The difference in the perception of $\mathbf{G_0}$ and $\mathbf{G_1}$ for the adversary may be bounded by the advantage of a zero-knowledge adversary $\mathbb{B}_{\mathrm{zk}}$ attacking the underlying QANIZK scheme: $\Pr\left[\,\mathbb{A}_{\mathrm{trac}} \text{ wins } G_0 \wedge C \in L_{(PK,VK)}\right] - \Pr[\,\mathbb{A}_{\mathrm{trac}} \text{ wins } G_1\,] \leq \mathsf{Adv}^{\mathrm{zk}}_{\mathrm{QANIZK}}(\mathbb{B}_{\mathrm{zk}})$.

Game $\mathbf{G_2}$: For this game, we change the decryption oracle so that after the Nizk.Verify check is performed, it checks to see whether there is a π' such that $(C, \pi') \in \mathcal{C}$. If so, the oracle also knows which query (H, M) this was a result of, and so outputs (M, ID_H) (without further processing of C). If C is not part of a previous output of the encryption oracle, then decryption proceeds as normal. This modification does not change the adversary's view, so $\Pr[\,\mathbb{A}_{\mathrm{trac}} \text{ wins } G_2\,] = \Pr[\,\mathbb{A}_{\mathrm{trac}} \text{ wins } G_2\,]$.

Game $\mathbf{G_3}$: This game differs from the previous games in the encryption oracle. Instead of encrypting the plaintext $M\|CERT_{ID_H}\|ID_H$, it encrypts a plaintext of the same length drawn at random from the message space. The different views of the adversary in $\mathbf{G_2}$ and $\mathbf{G_3}$ is then bound by $\mathsf{Adv}^{\mathrm{ror-cca}}_{\mathrm{PKE}}(\mathbb{B}_{\mathrm{ror-cca}})$, where ror-cca denotes the real-or-random security notion for public key encryption schemes. Real-or-random security is well-known to be implied by left-or-right indistinguishability [12], namely $\mathsf{Adv}^{\mathrm{ror-cca}}_{\mathrm{PKE}}(\mathbb{B}_{\mathrm{ror-cca}}) \leq \mathsf{Adv}^{\mathrm{cca}}_{\mathrm{PKE}}(\mathbb{B}_{\mathrm{cca}})$. It follows that $\Pr[\,\mathbb{A}_{\mathrm{trac}} \text{ wins } G_2\,] - \Pr[\,\mathbb{A}_{\mathrm{trac}} \text{ wins } G_3\,] \leq \mathsf{Adv}^{\mathrm{cca}}_{\mathrm{PKE}}(\mathbb{B}_{\mathrm{cca}})$.

We may now create a reduction from EUF-CMA to traceability by constructing an adversary $\mathbb{B}_{\text{euf-cma}}$ playing $\mathbf{G_3}$ with $\mathbb{A}_{\mathrm{trac}}$, and using the output to solve her own challenge. $\mathbb{B}_{\text{euf-cma}}$ is given the verification key VK of a signature scheme, and she generates $(PK, DK) \leftarrow\!\!\$\ \mathsf{Pke.Kg}$ and $(\sigma, \tau) \leftarrow\!\!\$\ \mathsf{Nizk.Setup}$ herself, and finally sends (PK, VK, σ) to $\mathbb{A}_{\mathrm{trac}}$. Whenever $\mathbb{A}_{\mathrm{trac}}$ queries the derivation oracle on an identity, $\mathbb{B}_{\text{euf-cma}}$ queries her signing oracle, and forwards the signature to $\mathbb{A}_{\mathrm{trac}}$. Any other query she makes, $\mathbb{B}_{\text{euf-cma}}$ can answer using the decryption key DK and QANIZK trapdoor τ. When $\mathbb{A}_{\mathrm{trac}}$ outputs $(\hat{C}, \hat{\pi})$ as her answer, $\mathbb{B}_{\text{euf-cma}}$ decrypts $\hat{C}$, parses $M\|CERT\|ID \leftarrow \mathsf{Pke.Dec}_{DK}(\hat{C})$, and passes $(CERT, ID)$ as her forgery. Whenever $\mathbb{A}_{\mathrm{trac}}$ wins, so does $\mathbb{B}_{\text{euf-cma}}$.

From all this, it follows that

$$\mathsf{Adv}^{\mathrm{trace}}_{\mathrm{OVE}}(\mathbb{A}_{\mathrm{trace}}) \leq \mathsf{Adv}^{\mathrm{sound}}_{\mathrm{QANIZK}}(\mathbb{B}_{\mathrm{sound}}) + \mathsf{Adv}^{\mathrm{zk}}_{\mathrm{QANIZK}}(\mathbb{B}_{\mathrm{zk}}) + \mathsf{Adv}^{\mathrm{cca}}_{\mathrm{PKE}}(\mathbb{B}_{\mathrm{cca}}) + \mathsf{Adv}^{\text{euf-cma}}_{\mathrm{SIG}}(\mathbb{B}_{\text{euf-cma}}).$$

□

Integrity. The integrity of the OVE scheme follows from the zero-knowledge property of the QANIZK scheme, as well as the correctness of the PKE scheme. Informally, there are only two ways the adversary can win the game: either C has a witness, or it does not. If it does not, the adversary has been able to generate a verifiable proof for an invalid statement, which breaches the soundness of the QANIZK scheme. If C has a witness, it is generated by encrypting a plaintext, and such a ciphertext will decrypt correctly by correctness of PKE, so winning this way is not possible.

Lemma 5 (Integrity of OVE). *For all adversaries $\mathbb{A}_{\mathrm{int}}$, there exist an equally efficient adversary $\mathbb{B}_{\mathrm{sound}}$ such that*

$$\mathsf{Adv}^{\mathrm{int}}_{\mathrm{OVE}}(\mathbb{A}_{\mathrm{int}}) \leq \mathsf{Adv}^{\mathrm{sound}}_{\mathrm{QANIZK}}(\mathbb{B}_{\mathrm{sound}})\ .$$

Proof. We present the integrity game for the OVE scheme in Fig. 15. The advantage of $\mathbb{A}_{\mathrm{int}}$ is

$$\begin{aligned}\Pr\left[\mathsf{Exp}^{\mathrm{int}}_{\mathrm{OVE}}(\mathbb{A}_{\mathrm{int}}) = 1\right] &= \Pr\left[\mathsf{Exp}^{\mathrm{int}}_{\mathrm{OVE}}(\mathbb{A}_{\mathrm{int}}) = 1 \wedge\ \hat{C} \in L_{(PK,VK)}\right] \\ &\quad + \Pr\left[\mathsf{Exp}^{\mathrm{int}}_{\mathrm{OVE}}(\mathbb{A}_{\mathrm{int}}) = 1 \wedge\ \hat{C} \notin L_{(PK,VK)}\right]\ ,\end{aligned}$$

where the latter probability may be bounded by the advantage of a soundness adversary against the QANIZK scheme, as the definition of the two adversaries match.

With regards to the former probability, $\hat{C} \in L_{(PK,VK)}$ implies that, for some $M, CERT_{ID}$, and ID, $\hat{C} = \mathsf{Pke.Enc}_{PK}(M\|CERT_{ID}\|ID; r)$. Correctness of the encryption scheme ensure that decryption will uniquely recover $M, CERT_{ID}$, and ID, and Ove.Dec will not reject. Thus the corresponding probability is zero. □

$\mathsf{Exp}^{\mathrm{int}}_{\mathrm{OVE}}(\mathbb{A})$

$(PK, DK) \leftarrow\!\!\$\ \mathsf{Pke.Kg}$
$(VK, SK) \leftarrow \mathsf{Sig.Kg}$
$(\sigma, \tau) \leftarrow\!\!\$\ \mathsf{Nizk.Setup}$
$h \leftarrow 0; \mathcal{C} \leftarrow \emptyset$
$\mathcal{CU} \leftarrow \emptyset$
$(\hat{C}, \hat{\pi}) \leftarrow \mathbb{A}^{O}((PK, VK, \sigma), SK, DK)$
$\mathsf{Ove.Verify}_{PK}(\hat{C}) = \top \wedge \mathsf{Ove.Dec}_{DK}(\hat{C}, \hat{\pi}) = (\bot, \bot)$

Fig. 15. The integrity game for our Verifiably Encrypted Certificate construction for OVE.

Game $\mathbf{G_1}^{b^*}$	encrypt$((M_0, CERT_{ID_0}, ID_0), (M_1, CERT_{ID_1}, ID_1))$	decrypt(C, π)
$(PK, DK) \leftarrow\!\!\$\ \mathsf{Pke.Kg}$	$C_{b^*} \leftarrow\!\!\$\ \mathsf{Pke.Enc}_{PK}(M_{b^*} \Vert CERT_{ID_{b^*}} \Vert ID_{b^*}; r)$	**require** $(C, \pi) \notin C$
$(VK, SK) \leftarrow \mathsf{Sig.Kg}$	$\pi_{b^*} \leftarrow \mathsf{Nizk.Prove}_{PK,VK,\sigma}(r, M_{b^*}, CERT_{ID_{b^*}}, ID_{b^*})$	**if** $\mathsf{Nizk.Verify}_{PK,VK,\sigma}(C, \pi) = \bot$
$(\sigma, \tau) \leftarrow\!\!\$\ \mathsf{Nizk.Setup}$	$C^* \leftarrow (C_{b^*}, \pi_{b^*})$	**return** $(\bot, \bot)$
$C \leftarrow \emptyset$	$C \leftarrow C \cup \{C^*\}$**return** C^*	$M \Vert CERT_{ID} \Vert ID \leftarrow \mathsf{Pke.Dec}_{DK}(C)$
$\hat{b} \leftarrow \mathbb{A}^{O}((PK, VK, \sigma), SK)$		**if** decryption or parsing fails
		return $(\bot, \bot)$
		return (M, ID)

Fig. 16. Game $\mathbf{G}_1^{b^*}$ for the privacy proof of our Verifiably Encrypted Certificate construction for OVE.

Privacy. The notion of privacy for the OVE rests on the security of the underlying encryption scheme and QANIZK protocol. In essence, the CCA notion of the PKE ensures that the C component does not leak any information about the message or the identity, whilst the zk notion of the QANIZK protocol guards against the proof π revealing anything useful to an adversary. Finally, the simulation soundness of the QANIZK helps guarantee that the adversary cannot forge a proof π', and thus take advantage of a decryption oracle.

Lemma 6 (Privacy of OVE). *For all adversaries $\mathbb{A}_{\mathrm{priv}}$, there exist similarly efficient adversaries $\mathbb{B}_{\mathrm{uss}}$, $\mathbb{B}_{\mathrm{zk}}$ and $\mathbb{B}_{\mathrm{cca}}$ such that*

$$\mathsf{Adv}^{\mathrm{priv}}_{\mathrm{OVE}}(\mathbb{A}_{\mathrm{priv}}) \leq 2\mathsf{Adv}^{\mathrm{zk}}_{\mathrm{QANIZK}}(\mathbb{B}_{\mathrm{zk}}) + 2\mathsf{Adv}^{\mathrm{uss}}_{\mathrm{QANIZK}}(\mathbb{B}_{\mathrm{uss}}) + 3\mathsf{Adv}^{\mathrm{cca}}_{\mathrm{PKE}}(\mathbb{B}_{\mathrm{cca}})\ .$$

Proof. Just as in the traceability game, we introduce a series of games for the adversary $\mathbb{A}_{\mathrm{priv}}$ to play, which gradually changes the original game into a reduction to the CCA game against the underlying encryption scheme.

Game $\mathbf{G}_0^{b^*}$: This is the original game, presented in Fig. 12, applied to our construction. The advantage of $\mathbb{A}_{\mathrm{priv}}$ may be expressed as $\mathsf{Adv}^{\mathrm{priv}}_{\mathrm{OVE}}(\mathbb{A}_{\mathrm{priv}}) = \Pr\left[\mathbf{G}_0^0 : \mathbb{A}_{\mathrm{priv}} \rightarrow 1\right] - \Pr\left[\mathbf{G}_0^1 : \mathbb{A}_{\mathrm{priv}} \rightarrow 1\right]$.

Game $\mathbf{G}_1^{b^*}$: In this game, we assume that the adversary will only forward valid encryption queries, i.e., all queried certificates validates as signatures for identities. We therefore do not need any checks of the validity of signatures in the game and can simplify accordingly, see Fig. 16. The restriction is without loss of generality, as an adversary can check the validity of the certificates. Thus, for $b^* \in \{0, 1\}$, we have $\Pr\left[\mathbf{G}_0^{b^*} : \mathbb{A}_{\mathrm{priv}} \rightarrow 1\right] = \Pr\left[\mathbf{G}_1^{b^*} : \mathbb{A}_{\mathrm{priv}} \rightarrow 1\right]$.

Game $\mathbf{G}_2^{b^*}$: Here, we change the generation of π during the encryption query, so that $\pi \leftarrow \mathsf{Nizk.Sim}_{\tau}(C)$. For both possible values of b^*, the difference in the adversary's view between $\mathbf{G}_1^{b^*}$ and $\mathbf{G}_2^{b^*}$ may be bounded by the advantage of an adversary $\mathbb{B}_{\mathrm{zk}}$ attacking the zero-knowledge property of the underlying QANIZK scheme, i.e., $\Pr\left[\mathbf{G}_1^{b^*} : \mathbb{A}_{\mathrm{priv}} \rightarrow 1\right] - \Pr\left[\mathbf{G}_2^{b^*} : \mathbb{A}_{\mathrm{priv}} \rightarrow 1\right] \leq \mathsf{Adv}^{\mathrm{zk}}_{\mathrm{QANIZK}}(\mathbb{B}_{\mathrm{zk}})$.

Game $\mathbf{G}_3^{b^*}$: In the final game, we replace the decryption procedure, so that any decryption query of the format (C, π) where C has been part of a challenge output, yet π was not, is rejected. In other words: we do not allow the privacy adversary to query challenge ciphertexts with new, valid proofs (obviously invalid proofs would be rejected regardless). The games $\mathbf{G}_2^{b^*}$ and $\mathbf{G}_3^{b^*}$ are therefore identical-until-bad, and we will analyse the probability of the bad event in the final step of the proof.

Given an adversary distinguishing between $\mathbf{G}_3^0$ and $\mathbf{G}_3^1$, we may construct a reduction to the CCA-security of the PKE as follows. An adversary $\mathbb{B}^2_{\mathrm{cca}}$ who is given the public key PK of an encryption scheme PKE sets

up a signature scheme with keys $(VK, SK) \leftarrow_\$ \text{Sig.Kg}$ and a QANIZK with $(\sigma, \tau) \leftarrow_\$ \text{Nizk.Setup}$, and sends $((PK, VK, \sigma), SK)$ to $\mathbb{A}_{\text{priv}}$. Encryption queries for $((M_0, CERT_{ID_0}, ID_0), (M_1, CERT_{ID_1}, ID_1))$ are answered by $\mathbb{B}_{\text{cca}}$ querying her decryption oracle with $(M_0 \| CERT_{ID_0} \| ID_0, M_1 \| CERT_{ID_1} \| ID_1)$ then simulating a proof π on the received challenge ciphertext C, and sending (C, π) to $\mathbb{A}_{\text{priv}}$. For any decryption query of (C', π') by $\mathbb{A}_{\text{priv}}$, $\mathbb{B}^2_{\text{cca}}$ rejects the query if π' does not verify, or $C = C'$. Otherwise, she sends C' to her decryption oracle: if it returns $\perp$, then $\mathbb{B}^2_{\text{cca}}$ returns $(\perp, \perp)$; if not, $\mathbb{B}^2_{\text{cca}}$ parses the received plaintext as $M \| CERT_{ID} \| ID$ and returns (M, ID). When $\mathbb{A}_{\text{priv}}$ outputs $\hat{b}$, $\mathbb{B}^2_{\text{cca}}$ copies it, and thus it follows that $\Pr\left[\mathbf{G}^1_3 : \mathbb{A}_{\text{priv}} \rightarrow 1\right] - \Pr\left[\mathbf{G}^0_3 : \mathbb{A}_{\text{priv}} \rightarrow 1\right] \leq \text{Adv}^{\text{cca}}_{\text{PKE}}(\mathbb{B}^2_{\text{cca}})$.

Finally, we bound the probability of the bad event in game $\mathbf{G}^{b^*}_3$, where the adversary queries the decryption oracle with a tuple consisting of a challenge ciphertext C and a new, valid proof π'. We introduce a new game, $\mathbf{G}^{b^*}_{\mathbf{x}}$ where any encryption query is answered as follows: draw a plaintext at random from the plaintext space, of the same length as a plaintext from an honest query. The plaintext is then encrypted to C, and a proof π for it is simulated, and (C, π) is sent to $\mathbb{A}_{\text{priv}}$. For both values of b^*, we then have $\Pr\left[\mathbf{G}^{b^*}_3 : \text{Bad}\right] - \Pr\left[\mathbf{G}^{b^*}_{\mathbf{x}} : \text{Bad}\right] \leq \text{Adv}^{\text{ror-cca}}_{\text{PKE}}(\mathbb{B}_{\text{ror-cca}})$, where ror-cca denotes the real-or-random security notion for public key encryption schemes. It is well-known that real-or-random security is implied by left-or-right indistinguishability [12]: $\text{Adv}^{\text{ror-cca}}_{\text{PKE}}(\mathbb{B}_{\text{ror-cca}}) \leq \text{Adv}^{\text{cca}}_{\text{PKE}}(\mathbb{B}_{\text{cca}})$. Furthermore, $\Pr\left[\mathbf{G}^{b^*}_{\mathbf{x}} : \text{Bad}\right] \leq \text{Adv}^{\text{uss}}_{\text{QANIZK}}(\mathbb{B}_{\text{uss}})$, and so the following inequality $\Pr\left[\mathbf{G}^{b^*}_3 : \text{Bad}\right] \leq \text{Adv}^{\text{cca}}_{\text{PKE}}(\mathbb{B}^{b^*}_{\text{cca}}) + \text{Adv}^{\text{uss}}_{\text{QANIZK}}(\mathbb{B}_{\text{uss}})$ holds for both values of b^*.

A final detail is combining the three different CCA adversaries from game $\mathbf{G}_3$, $\mathbb{B}^0_{\text{cca}}$, $\mathbb{B}^1_{\text{cca}}$ and $\mathbb{B}^2_{\text{cca}}$ by constructing a 'master' adversary $\mathbb{B}_{\text{cca}}$. This adversary plays the CCA game by uniformly at random picking which sub-reduction to run. We therefore have: $\text{Adv}^{\text{cca}}_{\text{PKE}}(\mathbb{B}_{\text{cca}}) = \frac{1}{3}\text{Adv}^{\text{cca}}_{\text{PKE}}(\mathbb{B}^0_{\text{cca}}) + \frac{1}{3}\text{Adv}^{\text{cca}}_{\text{PKE}}(\mathbb{B}^1_{\text{cca}}) + \frac{1}{3}\text{Adv}^{\text{cca}}_{\text{PKE}}(\mathbb{B}^2_{\text{cca}})$. We finally conclude that:

$$\text{Adv}^{\text{priv}}_{\text{OVE}}(\mathbb{A}_{\text{priv}}) \leq 2\text{Adv}^{\text{zk}}_{\text{QANIZK}}(\mathbb{B}_{\text{zk}}) + 2\text{Adv}^{\text{uss}}_{\text{QANIZK}}(\mathbb{B}_{\text{uss}}) + 3\text{Adv}^{\text{cca}}_{\text{PKE}}(\mathbb{B}_{\text{cca}}).$$

□

We note that our bound is not as tight as the corresponding one for anonymity in BMW. The difference is primarily due to the proof strategy: instead of game hops, Bellare et al. directly provided the code of two CCA adversaries that integrated a bad event and a hop between two games $\mathbf{G}^0$ and $\mathbf{G}^1$, coupled with a refined analysis of the relevant advantages. The integrated approach allowed for some terms in the derivation to cancel, leading to the slighlty tighter bound. We opted for simplicity instead, also as we deal with multi-query games as opposed to the single-query games in the BMW construction. Thus we can potentially avoid a tightness loss as a result of a hybrid argument by plugging in appopriate multi-query secure primitives.

4.3 Discussion of OVE

To the best of our knowledge, OVE schemes offer a combination of functionality and security hitherto unstudied. However, as mentioned before, there are great similarities with group signatures, with the crucial distinction that group signatures do not offer message recovery, nor confidentiality of messages. Our construction was directly inspired by the BMW construction for group signatures [13], with some notable differences. In the following, we explore these differences and also address how ideas from other group signature schemes might apply to OVE. Finally, we briefly compare signcryption to OVE.

A significant difference between our construction and BMW's sign-encrypt-proof is the use of signatures. In the BMW group signature scheme, the user signing key consists of a personal key pair for the signature scheme in addition to a certificate binding the personal verification key to the identifying index. When signing a message, the sender first signs the message using their personal signing key, and then encrypts *this* signature, along with the certificate and personal verification key. This may be regarded as a signature tree of depth two, as the certificate is a signature on the verification key. This indirection enables full traceability, so that even an adversary with access to the group master opening key is unable to forge a signature of an uncompromised group member.

We flattened the construction by removing the personal signature key-pair. The gain in efficiency results in our weaker notion of traceability: an adversary in possession of the secret key of our scheme can readily decrypt a ciphertext to learn the identity and certificate of an honest user, and subsequently send any message in the name of this user. As discussed previously, this weaker notion suits the intended use of the OVE scheme, where the recipient who holds the secret key has no motivation of sending spam to themselves. Our

perspective is that the recipient, holding the decryption key "owns" the system yet might wish to delegate the vetting: thus we introduce separate keys and insist privacy holds against the issuer, but traceability need not hold against the decryptor. A further weakening would completely identify issuer and decryptor as Kiayias and Yung considered for group signatures [36]. If the stronger version of traceability is deemed desirable for OVE, a closer fit with BMW should work.

One difference between OVE and the BMW framework is how the identity of the sender, resp. group member, is treated. For OVE, the identity itself, as input to the key derivation, is retrieved during decryption. For BMW, the identity is linked to an index instead, and it is this index which is part of the various algorithms. In order to get the actual identity of the group member, an additional look up table is required, necessitating further coordination between the issuing of keys and the opening of signatures. With some abuse of naming, we will nevertheless refer to this index i as (part of) the identity in what follows. A side-effect of BMW's use of indices is that they do not model a separate key derivation algorithm, instead generating all user keys as part of the initial key generation. One implication is that, syntactically, users can no longer be added to the group after set-up: this would require regenerating new keys for everyone. Obviously for the construction, it is straightforward to isolate an issuing algorithm, and adding users on the fly is not an issue.

Separate key derivation, or issuing, algorithms are known from dynamic group signature schemes [16, 19], where a useful distinction can be made between partially dynamic schemes where users can join but may never leave, and fully dynamic where a user's credentials may be revoked. A noticeable difference between the dynamic group signatures and OVE is that the former binds signatures to a PKI, providing non-repudiation and requiring the opener to output a proof to demonstrate publicly that the purported identity of signer of the message is correct. These differences render adaptation of the known group signature schemes less immediate as simplifications can likely be made—with the appropriate care. For instance, Groth [33] suggests increasing the depth of the signature tree to three by incorporating an additional one-time secure signature scheme. The advantage of his approach is much more efficient instantiations of the underlying primitives, including the NIZK, resulting in constant size group signatures. Similar ideas might be useful for optimizing OVE.

A more challenging inspiration for OVE arises from a brand new paradigm to construct compact and efficient group signatures based on structure preserving signatures (SPS) and signatures of knowledge (SoK) [2, 37, 29]. Here the signing algorithm does not involve an encryption scheme. Instead, the SPS is used to find a new representative of the user key, which is then signed along with the message using a SoK. Adaption to the OVE setting likely requires some additional tweaking, for example letting the SoK sign an encryption of the desired message, rather than the message itself.

So far we have only looked at the Hotel California situation where users are added dynamically, but they can never leave. The most challenging scenario for OVE is one where senders may become unvetted, such that their ciphertexts no longer pass the filter. This corresponds to fully dynamic group signatures [19], which can be achieved based on an accountable ring signature scheme (the signing of the message is simply applying the signing algorithm of said ring signature scheme). Adding unvetting would be a useful feature to OVE, but ideally without incurring the overhead of ring signatures: black listing at the filter is probably easier to achieve than the white listing at the senders (implicit when using ring signatures).

Finally, we note that generic transforms from either group signatures or signcryption to OVE are less obvious. For signcryption schemes, as we observed before, the combination of hiding the sender while still allowing for public verification appear mutually exclusive. On the other hand, a simple encrypt-then-groupsign transform fails privacy, as user Eve can simply intercept user Anna's ciphertext and supplant the group signature with one of her own, and ask for it to be decrypted. Where for IVE, unicity of signatures prevented such an attack, here no such protection is possible. Also a group signature's implicit encryption capacity [1, 31] appears hard to unlock generically to serve OVE.

5 Conclusion

We introduced vetted encryption, which allows a recipient to specify who is allowed to send them messages and outsource the filtering to any third party. We concentrated on only a single receiver in two distinct scenarios: the filter would or would not learn the identity of the sender. Either way, the sender would remain identifiable to the recipient. OVE has the potential to facilitate confidential communication with whistleblowers, sources for journalists and other scenarios for anonymous communication where an organization wants to filter the anonymous traffic, yet the individual needs to be identified to the recipient in a way that is convincing to the recipient while allowing repudiation by the sender.

When considering multiple receivers, a possible extension would be to allow a single filter in such a way that the intended recipient remains anonymous to the filter as well. Such an extension could be relevant for all three types of vetted encryption, though it is possibly more natural in the AVE and OVE setting. We have, after all, already lifted anonymity from the filter altogether in the IVE setting, which at the very least opens up the possibility to use the recipient's identity.

For identifiable vetted encryption we made the link with signcryption; one could further try to extend this link by considering an alternative multi-recipient scenario where a single sender wants to transmit the same message to multiple recipients simultaneously. This is quite common in email applications and one expects some performance benefits due to amortization (though the security definitions might become more complex, cf. multi-user signcryption).

Finally, for our constructions we concentrated on proofs of concepts. For both IVE and OVE we leave open the challenge of designing the most efficient scheme, either by suitably instantiating our generic construction or by taking further inspiration from, respectively, signcryption and group signatures, and beyond. Another possible feature for either primitive would be to revoke the right to send.

References

1. Michel Abdalla and Bogdan Warinschi. On the minimal assumptions of group signature schemes. In Javier López, Sihan Qing, and Eiji Okamoto, editors, *ICICS 04*, volume 3269 of *LNCS*, pages 1–13. Springer, Heidelberg, October 2004.
2. Masayuki Abe, Georg Fuchsbauer, Jens Groth, Kristiyan Haralambiev, and Miyako Ohkubo. Structure-preserving signatures and commitments to group elements. In Tal Rabin, editor, *CRYPTO 2010*, volume 6223 of *LNCS*, pages 209–236. Springer, Heidelberg, August 2010.
3. Masayuki Abe, Charanjit S. Jutla, Miyako Ohkubo, Jiaxin Pan, Arnab Roy, and Yuyu Wang. Shorter QA-NIZK and SPS with tighter security. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part III*, volume 11923 of *LNCS*, pages 669–699. Springer, Heidelberg, December 2019.
4. Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 83–107. Springer, Heidelberg, April / May 2002.
5. Jee Hea An and Tal Rabin. Security for signcryption: The two-user model. In Dent and Zheng [28], pages 21–42.
6. Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to securely release unverified plaintext in authenticated encryption. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, volume 8873 of *LNCS*, pages 105–125. Springer, Heidelberg, December 2014.
7. Giuseppe Ateniese, Danilo Francati, David Nuñez, and Daniele Venturi. Match me if you can: Matchmaking encryption and its applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 701–731. Springer, Heidelberg, August 2019.
8. Christian Badertscher, Fabio Banfi, and Ueli Maurer. A constructive perspective on signcryption security. In Dario Catalano and Roberto De Prisco, editors, *SCN 18*, volume 11035 of *LNCS*, pages 102–120. Springer, Heidelberg, September 2018.
9. Feng Bao and Robert H. Deng. A signcryption scheme with signature directly verifiable by public key. In Hideki Imai and Yuliang Zheng, editors, *PKC'98*, volume 1431 of *LNCS*, pages 55–59. Springer, Heidelberg, February 1998.
10. Paulo S. L. M. Barreto, Benoît Libert, Noel McCullagh, and Jean-Jacques Quisquater. Signcryption schemes based on bilinear maps. In Dent and Zheng [28], pages 71–97.
11. Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 566–582. Springer, Heidelberg, December 2001.
12. Mihir Bellare, Anand Desai, Eric Jokipii, and Phillip Rogaway. A concrete security treatment of symmetric encryption. In *38th FOCS*, pages 394–403. IEEE Computer Society Press, October 1997.
13. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, Heidelberg, May 2003.
14. Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, volume 3027 of *LNCS*, pages 268–286. Springer, Heidelberg, May 2004.
15. Mihir Bellare and Phillip Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 399–416. Springer, Heidelberg, May 1996.
16. Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, Heidelberg, February 2005.
17. Tor E. Bjørstad. Hybrid signcryption. In Dent and Zheng [28], pages 121–147.
18. Tor E. Bjørstad and Alexander W. Dent. Building better signcryption schemes with tag-KEMs. In Moti Yung, Yevgeniy Dodis, Aggelos Kiayias, and Tal Malkin, editors, *PKC 2006*, volume 3958 of *LNCS*, pages 491–507. Springer, Heidelberg, April 2006.
19. Jonathan Bootle, Andrea Cerulli, Pyrros Chaidos, Essam Ghadafi, and Jens Groth. Foundations of fully dynamic group signatures. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 16*, volume 9696 of *LNCS*, pages 117–136. Springer, Heidelberg, June 2016.
20. Xavier Boyen. Multipurpose identity-based signcryption (a swiss army knife for identity-based cryptography). In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 383–399. Springer, Heidelberg, August 2003.
21. Xavier Boyen. Identity-based signcryption. In Dent and Zheng [28], pages 195–216.
22. Liqun Chen and John Malone-Lee. Improved identity-based signcryption. In Serge Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 362–379. Springer, Heidelberg, January 2005.
23. Jean-Sébastien Coron. On the exact security of full domain hash. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 229–235. Springer, Heidelberg, August 2000.
24. Véronique Cortier, Pierrick Gaudry, and Stéphane Glondu. Belenios: A simple private and verifiable electronic voting system. In *Foundations of Security, Protocols, and Equational Reasoning*, volume 11565 of *Lecture Notes in Computer Science*, pages 214–238. Springer, 2019.
25. Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.
26. Ivan Damgård, Helene Haagh, and Claudio Orlandi. Access control encryption: Enforcing information flow with cryptography. In Martin Hirt and Adam D. Smith, editors, *TCC 2016-B, Part II*, volume 9986 of *LNCS*, pages 547–576. Springer, Heidelberg, October / November 2016.
27. Alexander W. Dent, Marc Fischlin, Mark Manulis, Martijn Stam, and Dominique Schröder. Confidential signatures and deterministic signcryption. In Phong Q. Nguyen and David Pointcheval, editors, *PKC 2010*, volume 6056 of *LNCS*, pages 462–479. Springer, Heidelberg, May 2010.
28. Alexander W. Dent and Yuliang Zheng, editors. *Practical Signcryption*. ISC. Springer, Heidelberg, 2010.
29. David Derler and Daniel Slamanig. Highly-efficient fully-anonymous dynamic group signatures. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *ASIACCS 18*, pages 551–565. ACM Press, April 2018.
30. Yevgeniy Dodis and Aleksandr Yampolskiy. A verifiable random function with short proofs and keys. In Serge Vaudenay, editor, *PKC 2005*, volume 3386 of *LNCS*, pages 416–431. Springer, Heidelberg, January 2005.

31. Keita Emura, Goichiro Hanaoka, and Yusuke Sakai. Group signature implies PKE with non-interactive opening and threshold PKE. In Isao Echizen, Noboru Kunihiro, and Ryôichi Sasaki, editors, *IWSEC 10*, volume 6434 of *LNCS*, pages 181–198. Springer, Heidelberg, November 2010.
32. Chandana Gamage, Jussipekka Leiwo, and Yuliang Zheng. Encrypted message authentication by firewalls. In Hideki Imai and Yuliang Zheng, editors, *PKC'99*, volume 1560 of *LNCS*, pages 69–81. Springer, Heidelberg, March 1999.
33. Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In Xuejia Lai and Kefei Chen, editors, *ASIACRYPT 2006*, volume 4284 of *LNCS*, pages 444–459. Springer, Heidelberg, December 2006.
34. Ik Rae Jeong, Hee Yun Jeong, Hyun Sook Rhee, Dong Hoon Lee, and Jong In Lim. Provably secure encrypt-then-sign composition in hybrid signcryption. In Pil Joong Lee and Chae Hoon Lim, editors, *ICISC 02*, volume 2587 of *LNCS*, pages 16–34. Springer, Heidelberg, November 2003.
35. Charanjit S. Jutla and Arnab Roy. Shorter quasi-adaptive NIZK proofs for linear subspaces. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part I*, volume 8269 of *LNCS*, pages 1–20. Springer, Heidelberg, December 2013.
36. Aggelos Kiayias and Moti Yung. Group signatures: Provable security, efficient constructions and anonymity from trapdoor-holders. Cryptology ePrint Archive, Report 2004/076, 2004. http://eprint.iacr.org/2004/076.
37. Benoît Libert, Thomas Peters, and Moti Yung. Short group signatures via structure-preserving signatures: Standard model security from simple assumptions. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 296–316. Springer, Heidelberg, August 2015.
38. Benoît Libert and Jean-Jacques Quisquater. New identity based signcryption schemes from pairings. Cryptology ePrint Archive, Report 2003/023, 2003. http://eprint.iacr.org/2003/023.
39. Benoît Libert and Jean-Jacques Quisquater. Efficient signcryption with key privacy from gap Diffie-Hellman groups. In Feng Bao, Robert Deng, and Jianying Zhou, editors, *PKC 2004*, volume 2947 of *LNCS*, pages 187–200. Springer, Heidelberg, March 2004.
40. Benoît Libert and Jean-Jacques Quisquater. Improved signcryption from q-Diffie-Hellman problems. In Carlo Blundo and Stelvio Cimato, editors, *SCN 04*, volume 3352 of *LNCS*, pages 220–234. Springer, Heidelberg, September 2005.
41. Anna Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 597–612. Springer, Heidelberg, August 2002.
42. Noel McCullagh and Paulo S. L. M. Barreto. Efficient and forward-secure identity-based signcryption. Cryptology ePrint Archive, Report 2004/117, 2004. http://eprint.iacr.org/2004/117.
43. Kenneth G. Paterson and Jacob C. N. Schuldt. Efficient identity-based signatures secure in the standard model. In Lynn Margaret Batten and Reihaneh Safavi-Naini, editors, *ACISP 06*, volume 4058 of *LNCS*, pages 207–222. Springer, Heidelberg, July 2006.
44. S. Sharmila Deva Selvi, S. Sree Vivek, Dhinakaran Vinayagamurthy, and C. Pandu Rangan. ID based signcryption scheme in standard model. In Tsuyoshi Takagi, Guilin Wang, Zhiguang Qin, Shaoquan Jiang, and Yong Yu, editors, *ProvSec 2012*, volume 7496 of *LNCS*, pages 35–52. Springer, Heidelberg, September 2012.
45. Shiuan-Tzuo Shen, Amir Rezapour, and Wen-Guey Tzeng. Unique signature with short output from CDH assumption. In Man Ho Au and Atsuko Miyaji, editors, *ProvSec 2015*, volume 9451 of *LNCS*, pages 475–488. Springer, Heidelberg, November 2015.
46. Yuliang Zheng. Digital signcryption or how to achieve cost(signature & encryption) ≪ cost(signature) + cost(encryption). In Burton S. Kaliski Jr., editor, *CRYPTO'97*, volume 1294 of *LNCS*, pages 165–179. Springer, Heidelberg, August 1997.

A Unique Identity Based Signature Scheme

We construct an *identity based signature scheme with unique signatures* (UIBSS) by using the known certificate based transformation on an unique signature scheme [14]. We generate a master signing and verification key for a USS scheme, as well as set up a PRF. Given an identity ID we use the PRF to derive randomness, which is then fed into the key generation algorithm of a USS scheme. In other words: the key generation of the USS is derandomised, with the given randomness depending on a given identity. The resulting key pair (SK, VK) are components of the user key USK of ID. The key USK also includes a certificate, which is $VK \| ID$ signed under the master signing key. A signature of a message M in the UIBSS scheme is simply $(\sigma, VK, CERT_{ID})$, where $\sigma \leftarrow \mathsf{Uss.Sign}(SK, M)$. Finally, verification requires that both signatures σ and $CERT_{ID}$ verifies on M and $VK \| ID$, respectively. We present the construction of our UIBSS in Fig. 17, see 2 for the syntax of IBS schemes.

We adopt the security notion of existential unforgeability of identity based signature schemes to our scheme [43]. Informally, the notion states that given access to a signing oracle and a corruption oracle, an adversary should not be able to find a tuple (M, ID, σ) which passes the verification algorithm, where she has not asked to corrupt ID, and not asked for a signature on (M, ID). Since the general certificate construction has been proven to produce identity-based signature schemes that satisfy this notion of security, it follows that our scheme is secure with respect to existential unforgeability [14].

For unique signature schemes $\mathsf{Uss.Verify}_{VK}(M, \sigma) = \mathsf{Uss.Verify}_{VK}(M, \sigma')$ implies $\sigma = \sigma'$ [45]. However, we will relax this requirement, and rather require that it is computationally hard for an adversary to win the following game: given the verification key, and access to a derivation oracle, find a tuple $(M, ID, \varsigma, \varsigma')$ such that $\mathsf{Uibss.Verify}_{VK,ID}(M, \varsigma) = \top = \mathsf{Uibss.Verify}_{VK,ID}(M, \varsigma')$, yet $\varsigma \neq \varsigma'$. We define this security notion as *outsider unicity*, with the game formally defined in Fig. 18. As always, the advantage of the adversary is her probability of winning the game.

$\underline{\mathsf{Uibss.Kg}()}$

$(MSK, MVK) \leftarrow \mathsf{Uss.Kg}()$
$k \leftarrow \mathsf{Prf.Kg}()$
return $((MSK, k), MVK)$

$\underline{\mathsf{Uibss.Derive}_{(MSK,k)}(ID)}$

$R \leftarrow \mathsf{PRF}(k, ID)$
$(SK, VK) \leftarrow \mathsf{Uss.Kg}(;R)$
$CERT_{ID} \leftarrow \mathsf{Uss.Sign}(MSK, VK\|ID)$
return $USK \leftarrow (SK, VK, CERT_{ID})$

$\underline{\mathsf{Uibss.Sign}_{USK,ID}(M)}$

$\sigma \leftarrow \mathsf{Uss.Sign}(SK, M)$
return $\varsigma \leftarrow (\sigma, VK, CERT_{ID})$

$\underline{\mathsf{Uibss.Verify}_{MVK,ID}(M, \varsigma)}$

if $\mathsf{Uss.Verify}_{MVK}(VK\|ID, CERT_{ID}) = \bot$
  return $\bot$
if $\mathsf{Uss.Verify}_{VK}(M, \sigma) = \bot$
  return $\bot$
else
  return $\top$

Fig. 17. The construction of an identity based signature scheme with unique signatures using a unique signature scheme (USS) and a psuedo random function (PRF). Note that we denote the signing and verification key generated by Uss.Kg during Uibss.Kg as (MSK, MVK) solely to distinguish these keys from the signing and verification keys that constitute the USK of a particular ID.

$\underline{\mathsf{Exp}^{\mathrm{ou}}_{\mathrm{UIBSS}}(\mathbb{A})}$

$(MSK, VK) \leftarrow \mathsf{Uibss.Kg}$
$(\hat{ID}, \hat{M}, \hat{\sigma}, \hat{\sigma}') \leftarrow \mathbb{A}^{O}(PK)$
winif $\mathsf{Uibss.Verify}_{VK}(\hat{ID}, \hat{M}, \hat{\sigma}) = \top \wedge$
  $\mathsf{Uibss.Verify}_{VK}(\hat{ID}, \hat{M}, \hat{\sigma}') = \top \wedge \hat{\sigma} \neq \hat{\sigma}'$

$\underline{\mathrm{derive}_{MSK}(ID)}$

return $USK \leftarrow \mathsf{Uibss.Derive}_{MSK}(ID)$

Fig. 18. The outsider unicity game for unique identity based signature schemes.

Our certificate based UIBSS scheme achieves outsider unicity due to the unicity property of the underlying unique signature scheme, as well as it's notion of unforgeability. Informally, the unicity of signatures forces an adversary to find a forgery on $VK\|ID$.

Lemma 7 (Outsider unicity of UIBSS construction). *For all adversaries $\mathbb{A}_{\mathrm{ou}}$, there exists an adversary $\mathbb{B}_{\text{euf-cma}}$ such that*

$$\mathsf{Adv}^{\mathrm{ou}}_{\mathrm{UIBSS}}(\mathbb{A}_{\mathrm{ou}}) \leq \mathsf{Adv}^{\text{euf-cma}}_{\mathrm{USS}}(\mathbb{B}_{\text{euf-cma}}).$$

Proof. The adversary $\mathbb{B}_{\text{euf-cma}}$ is given a verification key MVK, which she passes on to $\mathbb{A}_{\mathrm{ou}}$, and creates a key k from Prf.Kg. Whenever $\mathbb{A}_{\mathrm{ou}}$ sends a derivation query for an identity ID, $\mathbb{B}_{\text{euf-cma}}$ generates $R \leftarrow \mathsf{PRF}(k, ID)$, which she uses to derive $(SK, VK) \leftarrow \mathsf{Uss.Derive}(;R)$. She then queries her signing oracle with the message $VK\|ID$, and uses the received signature as $CERT_{ID}$. She then sends $(SK, VK, CERT_{ID})$ to $\mathbb{A}_{\mathrm{ou}}$. Eventually, $\mathbb{A}_{\mathrm{ou}}$ will output a tuple $(M, ID, \varsigma, \varsigma')$, where $\varsigma = (\sigma, VK, CERT_{ID})$. Assuming $\varsigma \neq \varsigma'$, at least one of the three components must differ. Due to the unicity of signatures in USS, we cannot have that $\varsigma = (\sigma, VK, CERT_{ID})$, $\varsigma' = (\sigma', VK, CERT_{ID})$. Similarly, we cannot have $\varsigma = (\sigma, VK, CERT_{ID})$, $\varsigma' = (\sigma, VK, CERT'_{ID})$, as this would mean $VK\|ID$ has two distinct signatures. It must therefore be the case that there are two different verification keys VK and VK', and that *at most one of them* has been issued by $\mathbb{B}_{\text{euf-cma}}$, meaning she has queried her signing oracle at most one of $VK\|ID, VK'\|ID$. Assuming she queried $VK\|ID$, she outputs $(VK'\|ID, CERT'_{ID})$ as the answer to her challenge. It is clear that $\mathbb{B}_{\text{euf-cma}}$ wins with the same probability as $\mathbb{A}_{\mathrm{ou}}$. □

B Anonymous Vetted Encryption (AVE)

B.1 Syntax and Security of AVE

The algorithms. For anonymous vetted encryption, neither the filter nor the recipient should be able to identify who encrypted a message. An AVE scheme consists of five algorithms, as listed in Definition 4 below. We remark on two slightly less obvious definitional choices.

Firstly, the input of the identity ID to Ave.Derive is not really needed, and any decent anonymous system would simply ignore this input. However, for full generality and ease of comparison with IVE and OVE

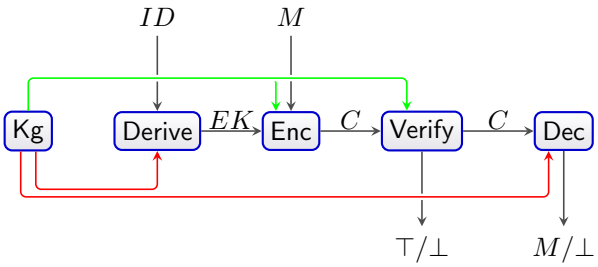

Fig. 19. The algorithms and their inputs/outputs for anonymous vetted encryption.

that do require ID as input in order to derive an encryption key, we allow Ave.Derive to depend on a user's identity ID.

Secondly, we allow encryption to fail, as captured by the $\perp$ output. As we will see, for honestly generated encryption keys, we insist encryption never fails, but for adversarially generated encryption keys, it turns out useful to allow for explicit encryption failure. Of course, one could alternatively introduce a separate algorithm to verify the validity of a private encryption key for a given public encryption/verification key, but our approach appears simpler.

Definition 4 (Anonymous Vetted Encryption (AVE)). *An* anonymous vetted encryption *scheme* AVE *consists of a 5-tuple of algorithms* (Ave.Kg, Ave.Derive, Ave.Enc, Ave.Verify, Ave.Dec) *that satisfy*

- Ave.Kg generates *a key pair* (PK, SK)*, where* PK *is the public encryption (and verification) key and* SK *is the private derivation and decryption key. We allow* Ave.Kg *to depend on parameters* $param$ *and write* $(PK, SK) \leftarrow\!\!\$\ \mathsf{Ave.Kg}(param)$*. Henceforth, we will assume that* PK *can be uniquely and efficiently computed given* SK*.*
- Ave.Derive derives *an encryption key* EK *based on the private derivation key* SK *and a user's identity* ID*. We write* $EK \leftarrow\!\!\$\ \mathsf{Ave.Derive}_{SK}(ID)$*.*
- Ave.Enc encrypts *a message* M *given the public encryption key* PK *and using the private encryption key* EK*, creating a ciphertext* C *or producing a failed encryption symbol* $\perp$*. In other words,* $C \leftarrow\!\!\$\ \mathsf{Ave.Enc}_{PK,EK}(M)$ *where possibly* $C = \perp$*.*
- Ave.Verify verifies *the validity of a ciphertext* C *given the public verification key* PK*, leading to either accept '*$\top$*' or reject '*$\perp$*'. With a slight abuse of notation,* $\top/\perp \leftarrow \mathsf{Ave.Verify}_{PK}(C)$*.*
- Ave.Dec decrypts *a ciphertext* C *using the private key* SK*. The result can either be a message* M *or the invalid-ciphertext symbol* $\perp$*. In short,* $M/\perp \leftarrow \mathsf{Ave.Dec}_{SK}(C)$*.*

The first three algorithms are probabilistic, whereas we assume that the final two algorithms are deterministic.

Correctness and consistency. *Correctness* captures that honest usage results in messages being received as intended. That is, for all parameters $param$, identities ID and messages M, we have that

$$\Pr\left[\begin{array}{l}(PK, SK) \leftarrow\!\!\$\ \mathsf{Ave.Kg}(param) \\ EK \leftarrow\!\!\$\ \mathsf{Ave.Derive}_{SK}(ID) \\ C \leftarrow\!\!\$\ \mathsf{Ave.Enc}_{PK,EK}(M)\end{array} : C \neq \perp \wedge \mathsf{Ave.Verify}_{PK}(C) = \top \wedge \mathsf{Ave.Dec}_{SK}(C) = M\right] = 1$$

As with IVE and OVE, *consistency* ensures that any ciphertext which decrypts to a valid message also passes the filter. We guarantee consistency of an AVE scheme by running verification as part of decryption, which is the same transformation we applied to IVE. We note here as well that correctness ensures that all honestly generated ciphertexts will decrypt to a valid message

Security. The security of AVE comprises three components: integrity to ensure the filter cannot be fooled, confidentiality of the message to outsiders, and finally sender anonymity even from the recipient. With reference to the games defined in Figures 20, 21, and 22, the relevant advantages are defined as follows:

- *Integrity*

$$\mathsf{Adv}^{\mathrm{int}}_{\mathrm{AVE}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\mathrm{int}}_{\mathrm{AVE}}(\mathbb{A}) : \hat{C} \notin C \wedge \mathsf{Ave.Verify}_{PK}(\hat{C}) = \top\right].$$

$\underline{\mathsf{Exp}^{\mathrm{int}}_{\mathrm{AVE}}(\mathbb{A})}$

$(PK, SK) \leftarrow\!\$\ \mathsf{Ave.Kg}$
$h \leftarrow 0; C \leftarrow \emptyset$
$\hat{C} \leftarrow \mathbb{A}^{O}(PK)$
winif $\hat{C} \notin C \wedge \mathsf{Ave.Verify}_{PK}(\hat{C}) = \top$

$\underline{\mathrm{derive}(ID)}$

$EK[h] \leftarrow \mathsf{Ave.Derive}_{SK}(ID)$
$h \leftarrow h + 1$
return h

$\underline{\mathrm{encrypt}(H, M)}$

$C \leftarrow\!\$\ \mathsf{Ave.Enc}_{PK, EK[H]}(M)$
$C \leftarrow C \cup \{C\}$
return C

$\underline{\mathrm{decrypt}(C)}$

$M \leftarrow \mathsf{Ave.Dec}_{SK}(C)$
return M

Fig. 20. The integrity game for AVE.

$\underline{\mathsf{Exp}^{\text{ind-cca-}b^*}_{\mathrm{AVE}}(\mathbb{A})}$

$(PK, SK) \leftarrow\!\$\ \mathsf{Ave.Kg}$
$C \leftarrow \emptyset$
$\hat{b} \leftarrow \mathbb{A}^{O}(PK)$

$\underline{\mathrm{derive}(ID)}$

$EK \leftarrow \mathsf{Ave.Derive}_{SK}(ID)$
return EK

$\underline{\mathrm{encrypt}(EK, M_0, M_1)}$

$C^* \leftarrow\!\$\ \mathsf{Ave.Enc}_{PK, EK}(M_{b^*})$
$C \leftarrow C \cup \{C^*\}$
return C^*

$\underline{\mathrm{decrypt}(C)}$

require $C \notin C$
$M \leftarrow \mathsf{Ave.Dec}_{SK}(C)$
return M

Fig. 21. The confidentiality game for AVE.

– *Confidentiality*

$$\mathsf{Adv}^{\mathrm{conf}}_{\mathrm{AVE}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\text{conf-}0}_{\mathrm{AVE}}(\mathbb{A}) \,:\, \hat{b} = 0\right] - \Pr\left[\mathsf{Exp}^{\text{conf-}1}_{\mathrm{AVE}}(\mathbb{A}) \,:\, \hat{b} = 0\right].$$

– *Anonymity*

$$\mathsf{Adv}^{\mathrm{anon}}_{\mathrm{AVE}}(\mathbb{A}) = \Pr\left[\mathsf{Exp}^{\text{anon-}0}_{\mathrm{AVE}}(\mathbb{A}) \,:\, \hat{b} = 0\right] - \Pr\left[\mathsf{Exp}^{\text{anon-}1}_{\mathrm{AVE}}(\mathbb{A}) \,:\, \hat{b} = 0\right].$$

Integrity. Integrity may informally be stated as: unless one has been vetted and is in possession of an encryption key, it should not be possible to furnish a valid ciphertext. We capture this integrity security notion in a game (Fig. 20), where the goal of the adversary is to create a valid ciphertext. As with IVE, we use the output of the verification algorithm as the indicator of validity. We note that for consistent AVE schemes, this choice of integrity is the strongest, as a forgery w.r.t. decryption will always be a forgery w.r.t. verification.

The adversary is given the verification key and additionally can ask for encryptions of messages of its choosing. We use the same *handle* mechanism as previously, so the adversary has some control over the encryption keys that are used: an adversary can trigger the game into the creation of an arbitrary number of keys (given an identity) and then indicate which key (by order of creation) to use for a particular encryption query. Obviously, the adversary does *not* receive any encryption keys themselves.

For full generality, we also grant access to a decryption oracle. One could also consider a weaker flavour of integrity without this oracle access. For consistent schemes, the decryption oracle is essentially pointless: if querying it on a fresh ciphertext C were to result in some message (so not $\perp$), then $\mathsf{Ave.Verify}_{PK}(C) = \top$ would already constitute a valid forgery.

Note that if decryption would somehow leak information—say when there are multiple possible decryption failures [25, Remark 14] or unverified plaintext is released early [6]—the decryption oracle would increase an adversary's power. As became evident in the treatment of IVE and OVE, the introduction of identities for the decryption algorithm also renders the corresponding decryption oracle more powerful and relevant.

Confidentiality. In Fig. 21, we adapt the well-trodden IND–CCA notion for public key encryption to the setting of anonymmous vetted encryption. An adversary can (repeatedly) ask its challenge oracle for the encryption of one of two messages. However, our new syntax requires an encryption key in addition to the verification key. We allow the adversary to specify the encryption key to use, which may or may not be honestly generated. In contrast to the integrity game, the key derivation oracle here does provide the adversary with encryption keys for its chosen identities.

Weaker definitions are possible by insisting an adversary can only query the challenge encryption oracle on honest encryption keys (as provided by the derivation oracle), or even hiding said keys using a similar handle-based mechanism as for the integrity game. We believe the stronger notion with adversarially chosen

keys is easier to deal with and, as we will see in B.2, still relatively easy to achieve based on standard public key primitives. For IVE and especially OVE, the stronger notion is the more natural one as well.

Anonymity. The idea of anonymity is that a recipient has no clue from which of the vetted people a ciphertext originated, as anonymity towards the recipient implies anonymity towards the filter. Here anonymity extends beyond not being able to extract or link a specific identity ID to a ciphertext: we also want to ensure that ciphertexts created using the same encryption key EK remain unlinkable.

We model our notion of anonymity using a distinguishing game, where the adversary knows the private key SK and gets to choose the encryption keys EK to use. If it cannot tell apart which encryption key EK was used (by the challenge encryption oracle), we deem the scheme anonymous. There is one caveat though: the game is likely winnable by deriving one true encryption key EK_0 (using knowledge of SK) and creating one fake EK_1. Assuming integrity, the challenge ciphertext will verify iff $b^* = 0$. To avoid these trivial wins, we only output a challenge ciphertext if it is valid irrespective of the challenge bit. Our game implements this mechanic by creating possible ciphertexts for both challenge bits and, rather than check based on verification, we put the onus on the encryption itself.

Anonymity is reminiscent of key privacy for public key encryption schemes [11] or its "ciphertext anonymity" adaptation to signcryption (which we will discuss later in B.3).

B.2 Generic Composition: Encrypt-then-Sign

As with IVE, an obvious first attempt to create an anonymous vetted encryption scheme is to combine the confidentiality provided by a public key encryption scheme with the authenticity of a signature scheme. Based on the reasoning as for IVE, we opt for the encrypt-then-sign approach.

The general construction is described in Fig. 23. First, the receiving party generates two key pairs: one for a PKE scheme and one for a signature scheme. It hands out the same signing key to whomever it wants to vet, so any vetted party can use the public encryption key and the received signing key to first encrypt, then sign. Verification by the filter consists of a simple signature verification.

The scheme inherits its authenticity from the signature scheme and its confidentiality from the encryption scheme. The latter inheritance only works when the signature scheme is *unique*. The signature is also verified as part of encryption and decryptions, which at first sight might appear superfluous. However, signature verification at decryption time is required for consistency (in line with the generic transform to achieve consistency), whereas the signature verification at encryption time is required to ensure anonymity even against malicious receivers.

Correctness and consistency. Both correctness and consistency follow easily by inspection. The signature verification as part of decryption is needed for consistency, in line with the transformation from before.

Integrity. Integrity of the scheme follows from the unforgeability of the underlying signature scheme. The proof is by a simple black-box reduction $\mathbb{B}_{\text{euf-cma}}$ where the PKE-ciphertexts in the int-game become messages in the EUF-CMA game. The overhead of $\mathbb{B}_{\text{euf-cma}}$ is running Pke.Kg once, plus one public-key encryption per encryption query posed by $\mathbb{A}_{\text{int}}$. As EtS is consistent, without loss of generality we assume $\mathbb{A}_{\text{int}}$ does not make any decryption queries.

$\mathsf{Exp}^{\text{anon-}b^*}_{\text{AVE}}(\mathbb{A})$	$\text{encrypt}(EK_0, EK_1, M)$
$(PK, SK) \leftarrow\!\!\$\ \mathsf{Ave.Kg}$	$C_0 \leftarrow\!\!\$\ \mathsf{Ave.Enc}_{PK,EK_0}(M)$
$C \leftarrow \emptyset$	$C_1 \leftarrow\!\!\$\ \mathsf{Ave.Enc}_{PK,EK_1}(M)$
$\hat{b} \leftarrow \mathbb{A}^{O}(PK, SK)$	**if** $C_0 \neq \perp \wedge C_1 \neq \perp$ **then**
	$\quad C^* \leftarrow C_{b^*}$
	else
	$\quad C^* \leftarrow \perp$
	return C^*

Fig. 22. The anonymity game for AVE.

Lemma 8 (Integrity of EtS). *For all adversaries* $\mathbb{A}_{\text{int}}$*, there exists an equallly efficient adversary* $\mathbb{B}_{\text{euf-cma}}$ *such that*

$$\mathsf{Adv}^{\text{int}}_{\text{AVE}}(\mathbb{A}_{\text{int}}) \le \mathsf{Adv}^{\text{euf-cma}}_{\text{SIG}}(\mathbb{B}_{\text{euf-cma}}) .$$

Proof. Upon receiving a verification key VK, $\mathbb{B}_{\text{euf-cma}}$ generates a key pair $(PK, DK) \leftarrow\!\!\$\ \mathsf{Pke.Kg}$ and runs $\mathbb{A}_{\text{int}}$ on input (PK, VK). Whenever $\mathbb{A}_{\text{int}}$ makes an encryption query, $\mathbb{B}_{\text{euf-cma}}$ performs the public-key encryption to obtain a ciphertext on which it uses its own signing oracle to obtain a signature. When $\mathbb{A}_{\text{int}}$ manages to create a forgery, then it has to create a valid PKE-ciphertext–signature pair that has not been returned by its encryption oracle. From $\mathbb{B}_{\text{euf-cma}}$'s perspective, this means a valid message–signature pair that has not been returned by its signature oracle. The claim follows. □

Confidentiality. Confidentiality of the scheme follows from that of the public key encryption scheme, provided the signature scheme has unique signatures. Unicity of the signature scheme appears necessary. After all, an adversary $\mathbb{A}_{\text{conf}}$ knows the signing key and thus if there are multiple valid signatures, it will be able to generate these and knowing a second signature for a challenge ciphertext would lead to a valid ciphertext to the decryption oracle, learning M_{b^*} and thus b^*, breaking confidentiality. In particular, derandomising a probabilistic signature scheme would be insufficient as $\mathbb{A}_{\text{conf}}$ could simply ignore the derandomisation and generate a signature using fresh randomness, exploiting that verification does not—and usually cannot—check whether the randomness used was constructed deterministically as prescribed by the derandomisation.

With unique signatures in place, the proof is by a simple black-box reduction $\mathbb{B}_{\text{ind-cca}}$, whose overhead is running Sig.Kg once, plus one signature per challenge encryption query and one signature verification per decryption query posed by $\mathbb{A}_{\text{conf}}$.

Lemma 9 (Confidentiality of EtS). *Let* SIG *be a unique signature scheme. Then for all adversaries* $\mathbb{A}_{\text{conf}}$*, there exists an equallly efficient adversary* $\mathbb{B}_{\text{ind-cca}}$ *such that*

$$\mathsf{Adv}^{\text{conf}}_{\text{AVE}}(\mathbb{A}_{\text{conf}}) \le \mathsf{Adv}^{\text{ind-cca}}_{\text{PKE}}(\mathbb{B}_{\text{ind-cca}}) .$$

Proof. Upon receiving a public key PK, $\mathbb{B}_{\text{ind-cca}}$ generates a key pair $(VK, SK) \leftarrow\!\!\$\ \mathsf{Sig.Kg}$ and runs $\mathbb{A}_{\text{conf}}$ on input (PK, VK). Whenever $\mathbb{A}_{\text{conf}}$ makes a challenge encryption query, $\mathbb{B}_{\text{ind-cca}}$ uses its own challenge encryption query to get a PKE-ciphertext, which it subsequently signs using SK. Thus by design, the challenge bits in the PKE and AVE games coincide.

The only potential complication is answering decryption queries (C, σ) by $\mathbb{A}_{\text{conf}}$. If (C, σ) was returned by $\mathbb{B}_{\text{ind-cca}}$ to $\mathbb{A}_{\text{conf}}$ as a previous challenge ciphertext, then the query may be ignored. So let us assume that (C, σ) is fresh. Then $\mathbb{B}_{\text{ind-cca}}$ first verifies whether σ is a valid signature on C. If not, return $\perp$, otherwise there are two possibilities: either C itself if fresh, i.e. it has not been returned by the game as a challenge ciphertext to $\mathbb{B}_{\text{ind-cca}}$, or it is not fresh, meaning it *is* a challenge ciphertext. In the former case, $\mathbb{B}_{\text{ind-cca}}$ can forward C to its own decryption oracle, receive a message M as result, and forward M to $\mathbb{A}_{\text{conf}}$. In the latter case, $\mathbb{B}_{\text{ind-cca}}$ cannot realistically forward C to its own decryption oracle (as it would be rejected). Luckily, the latter case cannot actually occur due to the unicity of signatures. □

Anonymity. Intuitively, anonymity follows from all encryption keys being the same, so independent of any identity or any additional randomness. However, in our security model an adversary is allowed to provide

$\mathsf{Ave.Kg}()$

$(PK, DK) \leftarrow\!\!\$\ \mathsf{Pke.Kg}$
$(VK, SK) \leftarrow\!\!\$\ \mathsf{Uss.Kg}$
return $((PK, VK), (DK, SK))$

$\mathsf{Ave.Derive}_{(DK,SK)}(ID)$

return SK

$\mathsf{Ave.Enc}_{(PK,VK),SK}(M)$

$C \leftarrow\!\!\$\ \mathsf{Pke.Enc}_{PK}(M)$
$\sigma \leftarrow \mathsf{Uss.Sign}_{SK}(C)$
if $\mathsf{Uss.Verify}_{VK}(C, \sigma) = \perp$ **then**
  return $\perp$
return (C, σ)

$\mathsf{Ave.Verify}_{PK,VK}(C, \sigma)$

return $\mathsf{Uss.Verify}_{VK}(C, \sigma)$

$\mathsf{Ave.Dec}_{DK,SK}(C, \sigma)$

if $\mathsf{Uss.Verify}_{VK}(C, \sigma) = \perp$ **then**
  return $\perp$
return $\mathsf{Pke.Dec}_{DK}(C)$

Fig. 23. Encrypt-then-Sign (EtS): A straightforward composition of public key encryption and signature scheme.

the encryption keys and thus deviate from honestly generated ones. Luckily, the unique-signatures property coupled with signature verification as part of the encryption routine, ensures an adversary can gain no benefit from such deviations, allowing us to show that anonymity of the scheme holds unconditionally.

Lemma 10 (Anonymity of EtS). *Let* SIG *be a unique signature scheme. Then for all adversaries* $\mathbb{A}$*,*

$$\mathsf{Adv}^{\mathrm{anon}}_{\mathrm{AVE}}(\mathbb{A}) = 0 \ .$$

Proof. The standard anonymity game for AVE allows multiple queries, but by a straightforward hybrid argument we can consider a single query only; as we target advantage 0 anyway, this hybrid will not incur a tightness loss. So consider $\mathbb{A}$'s single query SK_0, SK_1, M. The first part of EtS encryption calculates $C \leftarrow \mathsf{Pke.Enc}_{PK}(M)$. The resulting random variable C is clearly indepedent of the challenge bit. Next, a signature on C is produced, using either the signing key SK_0 or SK_1. If either of the signatures produced is invalid, the verification step as part of the encryption will notice and the game ensures the challenge oracle will output $\perp$ (making distinguishing impossible). Thus assume that both signatures pass the verification step. Then unicity of the signature scheme implies the signatures are in fact the same, thus the output of the challenge encryption oracle is independent of the challenge bit b^*: even an information theoretic adversary $\mathbb{A}$ cannot do better than random guessing. □

Instantiations. There is an abundance of efficient IND-CCA-secure PKE schemes available, based on a wide variety of cryptographic hardness assumptions. Unique signatures are rarer, especially in the standard model [41, 30]. In the random oracle model, an obvious candidate would be RSA-FDH [15, 23].

Remark 1. In practice a sender could of course "precompute" the signature verification by checking whether the signing key received as part of of the *EK*-derivation routine is valid for the public verification key. Such a precomputation is not entirely without loss of generality as it requires a signing-key checking algorithm that cannot be fooled (namely that once a signing key is accepted, acceptance of the resulting signatures is guaranteed for *all* messages).

B.3 Alternative Approaches

Signcryption. In many ways, AVE is reminiscent of signcryption, thus a natural question is whether one can turn a signcryption scheme into an AVE scheme. In order to answer this question, we need to zoom in on the right kind of signcryption scheme: which functionality does it need to support and which security does it need to provide?

From a functional perspective, the main restriction is the need for public verifiability [9, 32] as for AVE the filter needs to be able to verify ciphertexts without access to private key material. Thus, we consider a signcryption scheme to consist of six algorithms (Scr.Kgr, Scr.Kgs, Scr.Signcrypt, Scr.Verify, Scr.Unsigncrypt), where Scr.Kgr generates the receiver's keys and Scr.Kgs the sender's keys. We can transform such a signcryption scheme into an AVE scheme by simply letting Ave.Kg run both $(PK, DK) \leftarrow\!\!\$\ \mathsf{Scr.Kgr}$ and $(VK, SK) \leftarrow\!\!\$\ \mathsf{Scr.Kgs}$, and setting (PK, VK) to the public key of the AVE, keeping (DK, SK) private. The sender private key SK will serve as the encryption key and is therefore returned by Ave.Derive (irrespective of the identity). For the final three algorithms, there is a clean correspondence:

- $\mathsf{Ave.Enc}_{(PK,VK),SK}(M) = \mathsf{Scr.Signcrypt}_{(PK,VK),SK}(M)$;
- $\mathsf{Ave.Verify}_{PK,VK}(C) = \mathsf{Scr.Verify}_{VK}(C)$;
- $\mathsf{Ave.Dec}_{DK,SK}(C) = \mathsf{Scr.Unsigncrypt}_{DK}(C)$.

For the resulting AVE scheme, *correctness* is directly inherited from that of the signcryption scheme and *consistency* is satisfied provided the signcryption satisfies a similar notion (between verification and unsigncryption). For the security notions, our AVE setting only has two users, which implies the two-user model for signcryption suffices [5, Section 2.2]. In that case, *integrity* is a consequence of strong outsider secure unforgeability under chosen message attacks [5, Section 2.2.1.2]. Here outside security suffices as, in the AVE integrity game, an adversary does not have access to the derived encryption keys (which would correspond to a sender's private signcryption key). In contrast, for confidentiality we do need *insider* secure indistinguishability under chosen ciphertext attacks, as the sender's private signcryption key will be readily available to an adversary in the corresponding AVE confidentiality game (through the derive oracle).

$\underline{\mathsf{Exp}^{\text{anon-}b^*}_{\text{SCR}}(\mathbb{A})}$

$(PK_r, SK_r) \leftarrow\!\!\$\ \mathsf{Scr.Kgr}$
$C \leftarrow \emptyset$
$\hat{b} \leftarrow \mathbb{A}^{O}(PK_r, SK_r)$

$\underline{\text{signcrypt}(SK_{s0}, SK_{s1}, M)}$

$C_0 \leftarrow\!\!\$\ \mathsf{Scr.Signcrypt}_{PK, SK_{s0}}(M)$
$C_1 \leftarrow\!\!\$\ \mathsf{Scr.Signcrypt}_{PK, SK_{s1}}(M)$
if $C_0 \neq \perp \wedge C_1 \neq \perp$ **then**
  $C^* \leftarrow C_{b^*}$
else
  $C^* \leftarrow \perp$
return C^*

Fig. 24. The anonymity game needed when constructing AVE from signcryption.

Finally, *anonymity* is hardest to place, so let's look at anonymity notions for signcryption. The original ciphertext anonymity [20] captures only indistinguishability for honestly generated keys; moreover it attempts to hide both sender and receiver (the latter is irrelevant for us). Later incarnations of ciphertext anonymity [39, 40], do consider adversarially generated sender-keys. However, the syntax does not explicitly allow signcryption failure (even though signcryption can fail for some constructions). Moreover, the corresponding security game does not seem to care if challenge signcryption fails for only one of the two "left-or-right" adversarially provided sender signcryption keys (cf. [10, Section 5.6.2]), as we do in our anonymity game. It is relatively straighforward to derive the matching anonymity game for signcryption needed for the signcryption-to-AVE transform to work (see Fig. 24).

As an aside, although encrypt-then-sign has been studied in the signcryption literature, we are not aware of the potential of using unique signatures in order to achieve insider IND-CCA security (cf. [5, Theorem 2.2]).

Hybrid encryption. The typical operations associated with public key primitives are typically considerably more expensive than their symmetric counterparts. Hybrid encryption allows one to leverage the speed of symmetric cryptography, while maintaining the functionality and security of public key cryptography. A natural question is how applicable the concepts of hybrid encryption are for AVE.

Obviously, in the EtS transform it is possible to use a hybrid PKE. A natural question is whether our transform could then deal with distinct decryption failures from the KEM, resp. the DEM [25, Remark 14]. As we mentioned, for the integrity game, the decryption oracle might come into play, but for the EtS construction they do not cause any trouble (the reduction knows the PKE decryption key and the signing key isn't use by the AVE decryption). For the other two security properties, the proofs go through as is.

Potentially even more relevant and potent is the idea of hybrid signcryption [34, 17]. Here the signcryption KEM takes as input the receiver's public signcryption key and the sender's private signcryption key, returning a signcryptext as well as an ephemeral key for the (standard) DEM. The problem of the resulting construction is that it does not provide insider security and it is easy to see how the resulting AVE fails to provide proper integrity: after observing a valid pair (C, C') where C is the signcryptext and C' the DEM-ciphertext, simply substitute the second component. An alternative is the use of signcryption tag-KEMs [18], which do provide insider security. The signcryption scheme with public verifiability [32] can be cast this way and thus would be a good candidate for AVE (not entirely surprising given the original design goal).