Simpler and Stronger Models for Deniable Authentication

Chen-Da Liu-Zhang[1], Christopher Portmann[2], and Guilherme Rito[3]

[1] Lucerne University of Applied Sciences and Arts & Web3 Foundation
chendaliu@gmail.com
[2] Concordium
cp@concordium.com
[3] Ruhr-Universität Bochum
guilherme.teixeirarito@rub.de

Abstract. Deniable Authentication is a highly desirable guarantee for secure messaging: it allows Alice to authentically send a message m to a designated receiver Bob in a *Plausibly Deniable* manner. Concretely, while Bob is guaranteed Alice sent m, he cannot convince a judge Judy that Alice really sent this message—even if he gives Judy his secret keys—because Judy knows Bob *can* make things up. This paper models the security of Multi-Designated Verifier Signatures (MDVS) and Multi-Designated Receiver Signed Public Key Encryption (MDRS-PKE)—two (related) types of schemes that provide such guarantees—in the Constructive Cryptography (CC) framework (Maurer and Renner, ICS '11).
The only work modeling dishonest parties' ability of "making things up" was by Maurer et al. (ASIACRYPT '21), who modeled the security of MDVS, also in CC. Their security model has two fundamental limitations:

1. deniability is not guaranteed when honest receivers read;
2. it relies on the CC-specific concept of specifications.

We solve both problems. We give a standard simulator-based model that guarantees deniability when honest receivers read. Interestingly, our composable treatment allowed to identify a new property, Forgery Invalidity, without which we do not know how to prove the deniability of neither MDVS nor MDRS-PKE when honest receivers read. Finally, we prove that Chakraborty et al.'s MDVS (EUROCRYPT '23) has this property, and that Maurer et al.'s MDRS-PKE (EUROCRYPT '22) preserves it from the underlying MDVS.

1 Introduction

Messaging apps allow for asynchronous conversations between users who are apart. Naturally, it is desirable these apps provide the same security guarantees of *in-person* conversations.

Authenticity and plausible deniability. When Alice and Bob have an in-person conversation *authenticity* is naturally guaranteed because they know each other and can both identify their voices and who speaks. For example when Alice

says "Hello!", Bob knows it was Alice because he could see her talking. Although in some cases Alice and Bob may want their communication to be authentic in a publicly verifiable manner—e.g. by signing a contract that can later be verified by a judge in case of dispute—the authenticity of in-person conversations is of the exact opposite nature: it is exclusive to the parties having the conversation. This exclusiveness of authenticity—commonly referred to as *deniable authentication* or *off-the-record* in the cryptographic literature [2, 6, 9, 11–14, 25–27]—is particularly useful for protecting the privacy of personal communication. In the case above, it gives Alice *plausible deniability*: even if Bob later tries to convince Charlie—who did not participate in the conversation—that Alice came up to him to say "Hello", Charlie has no reason to believe Bob[4] because Charlie knows Bob could be making it up. (This deniability can become particularly challenging in messaging apps because users often keep their conversations stored in their devices.)

Group conversations. Now suppose Alice, Bob and Charlie participate in a three-party group chat. A natural guarantee they want is that every party gets the same messages (*consistency*). For example even if Alice wants to create confusion among Bob and Charlie, she cannot send malformed ciphertexts so Bob and Charlie obtain different messages.

MDVS. When translating these three guarantees into a cryptographic scheme one obtains so-called Multi-Designated Verifier Signature (MDVS). These schemes were introduced in [9], and have received significant attention for their security guarantees [4, 6, 18]; they allow Alice to select a set of designated receivers, say Bob and Charlie, and sign a message that only they can verify. Receivers are guaranteed consistency: if honest Bob successfully validates that Alice signed some message m to him and Charlie, then Bob is guaranteed that Charlie can also successfully validate the same signature if he is honest. These schemes also provide the type of authenticity of in-person conversations: if Alice and Bob are honest then Bob can only (successfully) validate messages sent by Alice; in particular, even if Charlie is dishonest, Charlie cannot fool Bob into believe Alice sent a message to him (Bob) and Charlie, unless she actually did.

Confidentiality and anonymity. Another natural guarantee from in-person conversations is *confidentiality*: if Alice wants to tell Bob and Charlie a secret, she can whisper so only they hear; of course that is only possible if both agree to keep the secret. Finally, Alice, Bob and Charlie may also want to hide whether they even had a conversation; indeed one often does not want third parties to know with whom one communicates.

MDRS-PKE. Introduced in [18], Multi-Designated Receiver Signed Public Key Encryption (MDRS-PKE) schemes provide all the guarantees of MDVS plus confidentiality and anonymity. These schemes were essentially tailor-made for so-called "Off-the-record" messaging ([18, Section 1.4]) and work similarly to

[4] Other than her trust in Bob.

MDVS schemes: the main difference is that decryption does not require receivers to *a priori* know the sender, the set of designated receivers or the message; instead they obtain this information via the decryption algorithm, which only takes as input the ciphertext to decrypt and the receiver's secret key. (In contrast, the verification of an MDVS signature requires knowledge of who the sender is, who the set of verifiers is, of the message to verify, and of the sender's and verifiers' public keys.)

Recent progress. A recent interest in these schemes has provided us with

- a better conceptual understanding of their guarantees via game-based notions capturing 1. stronger authenticity [27]; 2. stronger plausible deniability (Any-Subset Off-The-Record [6]); and 3. consistency [6] guarantees;
- the first application semantics that capture dishonest parties' ability of making things up (the composable treatment of MDVS from [17]);
- new abstractions that allow for conceptually simple constructions of these primitives—e.g. Provably-Simulatable Designated Verifier Signatures [6] and Public Key Encryption for Broadcast schemes [18];
- efficient constructions from building blocks that are known to have instantiations with tight security reductions to standard assumptions [4, 18];
- stronger deniability guarantees to capture scenarios where a judge could obtain the secret keys of honest senders[5] and respective schemes providing these guarantees [4].

State of affairs. Despite this progress some important questions remain unanswered, in particular regarding to their application-level guarantees:

No Application Semantics for MDRS-PKE. It is not know whether the existing game-based security models for MDRS-PKE [4, 18] imply the natural application semantics one would expect from these schemes. This means, in particular, that we do not know if the existing constructions can be used for the applications that motivated their introduction. Furthermore, while these schemes are similar MDVS, their game-based security notions are more involved due to the additional anonymity guarantees and different decryption syntax. (More involved notions typically increase the chance of not identifying important security properties and ending up with schemes that cannot be used for the applications they were developed for.) Concretely, the MDVS game-based notions [4, 6, 17, 18] provide adversaries with access to a signature verification oracle that requires a sender and a set of receivers to be specified (by means the oracle's input), but the (analogous) decryption oracle provided by MDRS-PKE game-based security notions from [4, 18] is not provided with that information.

Limited Deniability Guarantees I. The only existing composable treatment of MDVS schemes [17] provides surprisingly weak deniability guarantees: the

[5] Some countries may legally require citizens who are under investigation to provide authorities with their passwords/secret keys.

application semantics only give a sender plausible deniability on messages that are not read by honest receivers. This contrasts with the game-based models for MDVS and MDRS-PKE schemes which provide deniability when honest receivers read—this is reflected on the Off-The-Record game-based notions which provide adversaries with an oracle to verify signatures (for the case of MDVS) or to decrypt ciphertexts (for the case of MDRS-PKE).

Limited Deniability Guarantees II. [17]'s MDVS composable treatment only considers the weaker setting where the secret keys of honest senders do not leak. This means, in particular, that the MDVS and MDRS-PKE constructions from [4] are not actually known to allow for the application that motivated their introduction.

CC-Specific Security Model. The MDVS application semantics from [17] are defined using the concept of specifications—a concept which, to the best of our knowledge, only exists in the Constructive Cryptography (CC) framework [16, 20]. This is the only work capturing that dishonest parties must have a capability—in their case of forging MDVS signatures, which is crucial for plausible deniability.

Limited Unforgeability Guarantees. Replay protection is desirable for messaging apps: dishonest outsiders should not be able to come up with valid replays of previously sent messages. However, existing MDVS and MDRS-PKE security models [4, 6, 17, 18] (both game-based and composable ones) do not provide protection against replay attacks: [17]'s application semantics explicitly defines a copy operation that dishonest parties can use to replay previously sent messages; at a high level, the unforgeability notions from [4, 6, 18] only require the infeasibility of forging valid signatures on previously unsigned messages, but do not require the infeasibility of coming up with new signatures on previously signed messages.

Our contribution. We give a comprehensive composable treatment of MDVS and MDRS-PKE schemes in CC that addresses each of these limitations:

Stronger Deniability Guarantees. Senders are guaranteed plausible deniability on messages that have already been read by honest receivers even if they are forced to give away their secret keys.

Standard Simulator-Based Model. Our composable treatment is more easily digestible by readers unfamiliar with CC and gets closer to being translatable into other composable frameworks like UC [3].

Replay-Unforgeability. Our application semantics disallow replay attacks.

Our composable treatment of MDVS and MDRS-PKE schemes unveiled issues in their current game-based security models. (This paper fixes every issue we identified.) Concretely, we identified the following problems:

Forgery Invalidity. We identified a new property, Forgery Invalidity, that went unnoticed in prior work and which is crucial to our composable treatment of these schemes. Roughly, it captures that forged MDVS signatures (or ciphertexts, for the case of MDRS-PKE) must be invalid, i.e. their verification

by honest receivers fails (respectively, their decryption by honest receivers fails, for the case of MDRS-PKE). (This is not captured by the standard unforgeability notion because the adversary is given the sender's secret key—which it cannot obtain when playing the unforgeability game.) This property is key in our composable treatment because we do not know how to prove that neither existing MDVS nor MDRS-PKE game-based security models imply the application semantics we define if we do not assume the underlying scheme provides this extra guarantee.

Stronger confidentiality and anonymity. We define $\{\mathsf{IND}, \mathsf{IK}\}\text{-}\mathsf{CCA}_S$: a stronger MDRS-PKE game-based notion without which we do not know how to prove the composable security of MDRS-PKE schemes *in the weaker setting where the secret keys of honest senders do not leak.* Our difficulty with the weaker IND-CCA and IK-CCA notions from [18] is that a reduction 1. cannot obtain honest senders' secret keys via oracle $\mathcal{O}_{SK}$; and 2. cannot issue challenge queries $\mathcal{O}_E((A_{i,0}, \vec{V_0}, m_0), (A_{i,1}, \vec{V_1}, m_1))$ if one of the receivers is dishonest (i.e. for which there is a query $\mathcal{O}_{RK}(B_j \in \mathrm{Set}(\vec{V_0}) \cup \mathrm{Set}(\vec{V_1}))$) even if $(A_{i,0}, \vec{V_0}, m_0) = (A_{i,1}, \vec{V_1}, m_1)$. For these reasons we do not know how a reduction can generate encryptions of messages from honest senders to vectors of receivers that include dishonest ones without accessing honest senders' secret keys.

We also introduce game-based notions that capture replay unforgeability; our notions are just sEUF-CMA type of notions for MDVS and MDRS-PKE.

Finally, we prove that Chakraborty et al.'s MDVS [4] and Maurer et al.'s MDRS-PKE [4, 18] satisfy our composable notions—by showing their constructions satisfy the game-based notions we assume in our composable treatments of these schemes. (For the case of $\{\mathsf{IND}, \mathsf{IK}\}\text{-}\mathsf{CCA}_S$ and replay unforgeability, the same arguments used to prove the security of these schemes with respect to the corresponding weaker notions also imply security with respect to the stronger notions we introduce.).

We illustrate (part of) our contributions in Figure 1.

2 Preliminaries

For a set/alphabet S, we denote the set of non-empty vectors/strings over S by S^+. We denote the arity of a vector $\vec{x}$ by $|\vec{x}|$ and its i-th element by x_i. We write $\mathrm{Set}(\vec{x})$ to denote the set induced by $\vec{x}$: $\mathrm{Set}(\vec{x}) \coloneqq \{x_i \mid x_i \in \vec{x}\}$. For a vector of parties $\vec{V}$, we denote by $\vec{v}$ the corresponding vector of public keys, and for $i \in \{1, \ldots, |\vec{V}|\}$, v_i is party V_i's public key. We will denote the set of all parties by $\mathcal{P}$. For any subset of parties $\mathcal{S} \subseteq \mathcal{P}$, we denote by $\mathcal{S}^H$ and $\overline{\mathcal{S}^H}$ the partitions of $\mathcal{S}$ corresponding to honest and dishonest parties, respectively (with $\mathcal{S} = \mathcal{S}^H \uplus \overline{\mathcal{S}^H}$).

Throughout the paper, for an event-based security notion X (e.g. Unforgeability) we define an adversary $\mathbf{A}$'s advantage as its probability of winning the security game defined by X: $Adv^{\mathsf{X}}(\mathbf{A}) \coloneqq \Pr[\mathbf{AG}^{\mathsf{X}} = \mathtt{win}]$. For a distinguishing-type of notion Y (e.g. IND-CCA) defining games $\mathbf{G}^{\mathsf{Y}}_{\mathbf{0}}$ and $\mathbf{G}^{\mathsf{Y}}_{\mathbf{1}}$ we define an adversary $\mathbf{A}$'s advantage as $Adv^{\mathsf{Y}}(\mathbf{A}) \coloneqq \left|\Pr[\mathbf{AG}^{\mathsf{Y}}_{\mathbf{0}} = \mathtt{win}] + \Pr[\mathbf{AG}^{\mathsf{Y}}_{\mathbf{1}} = \mathtt{win}] - 1\right|$.

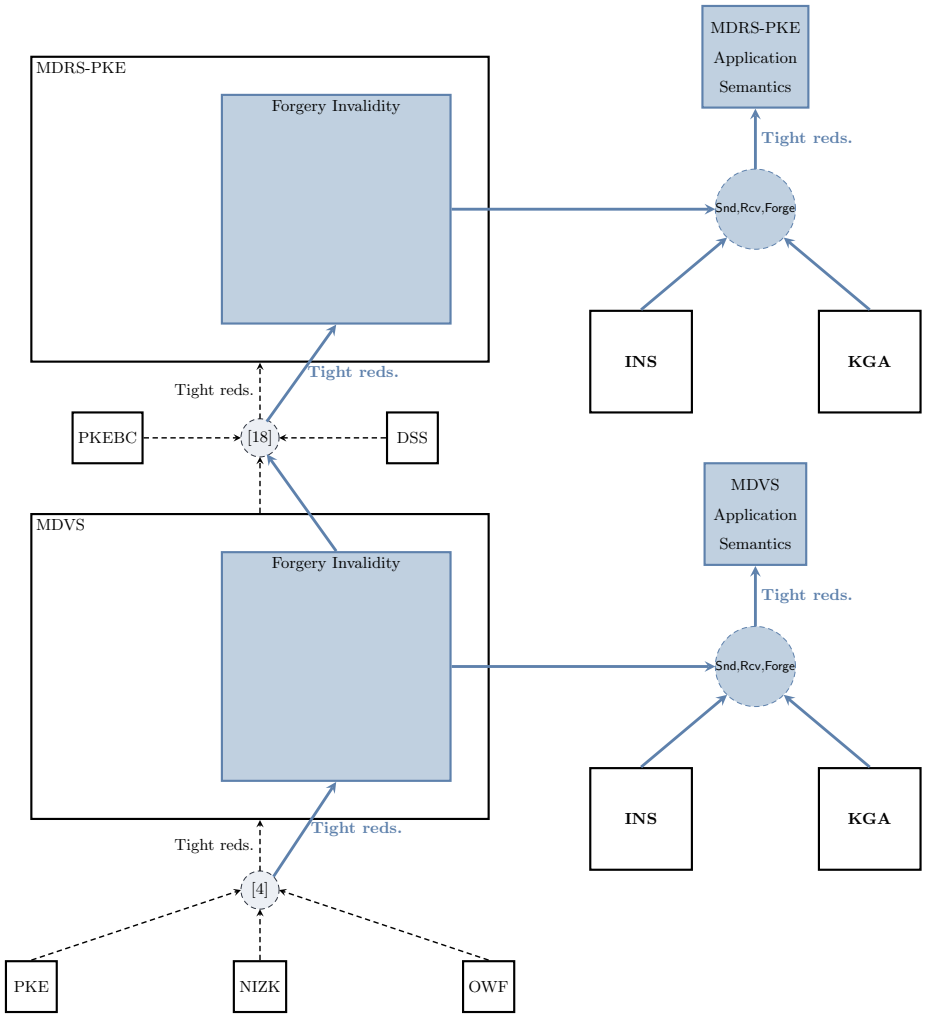


Fig. 1: Illustration of contributions. In blue are the new security notions, constructions and results.

2.1 (Simplified) Constructive Cryptography

Our paper's statements are phrased in a (rather) simplified version of the Constructive Cryptography (CC) framework [16, 20] which allows for standard simulator-based type of composable notions and requires little to no familiarity with CC. All construction statements trivially carry to CC.

CC views cryptography as a resource theory: protocols construct new resources from existing (assumed) ones. The notion of resource construction is inherently composable: if a protocol π_1 constructs $\mathbf{S}$ from $\mathbf{R}$ and π_2 constructs $\mathbf{T}$ from $\mathbf{S}$, then running both protocols $(\pi_2 \cdot \pi_1)$ constructs $\mathbf{T}$ from $\mathbf{R}$.

Resources. Resources are interactive systems akin to functionalities in UC [3]. Similarly to a function $f : X \to Y$, a resource also has input and output domains; if a resource $\mathbf{R}$ has input domain $\mathcal{X}$ and output (co-)domain $\mathcal{Y}$, we say $\mathbf{R}$ is an $(\mathcal{X}, \mathcal{Y})$ resource. One interacts with a $(\mathcal{X}, \mathcal{Y})$ resource by providing an input $x \in \mathcal{X}$ and receiving an output $y \in \mathcal{Y}$. Formally, resources are random systems [21, 22]; in turn, a random system is defined as a sequence of conditional probability distributions [22, Definition 2]. If two $(\mathcal{X}, \mathcal{Y})$-resources $\mathbf{R}$ and $\mathbf{S}$ are the same sequence of conditional probability distributions, we say they are equivalent and write $\mathbf{R} \equiv \mathbf{S}$ [22, Definition 3]. For simplicity, we describe resources by pseudo-code.

We often attach resources together; for (compatible) resources $\mathbf{R}$ and $\mathbf{S}$, we denote by $\mathbf{R} \cdot \mathbf{S}$ the resource resulting from attaching $\mathbf{R}$ and $\mathbf{S}$. (Resources $\mathbf{R}$ and $\mathbf{S}$ can only be attached together if their composition results in a well-defined sequence of conditional probability distributions—see, e.g. [15, Definition 7]; this is not the case for all pairs of resources.) For n resources $\{\mathbf{R}_i\}_{i=1}^n$, where each $\mathbf{R}_i$ is an $(\mathcal{X}_i, \mathcal{Y}_i)$-resource, if for all distinct $i, j \in [n]$, both $\mathcal{X}_i$ and $\mathcal{Y}_i$ are disjoint from $\mathcal{Y}_j$, then we denote the combined resource—corresponding to attaching $\mathbf{R}_1, \ldots, \mathbf{R}_n$ together—by $\mathbf{R} \coloneqq [\mathbf{R}_1, \ldots, \mathbf{R}_n]$, and call $\mathbf{R}$ the parallel composition of $\{\mathbf{R}_i\}_{i=1}^n$.

Interfaces. For an $(\mathcal{X}, \mathcal{Y})$-resource $\mathbf{R}$, an interface $I = (I_\mathcal{X}, I_\mathcal{Y})$ is a pair of subsets of $\mathbf{R}$'s input and output domains, i.e. $I_\mathcal{X} \subseteq \mathcal{X}$ and $I_\mathcal{Y} \subseteq \mathcal{Y}$; we call $I_\mathcal{X}$ and $I_\mathcal{Y}$ input and output interfaces of $\mathbf{R}$, respectively. For two interfaces $I_1 = (I_{1,\mathcal{X}}, I_{1,\mathcal{Y}})$ and $I_2 = (I_{2,\mathcal{X}}, I_{2,\mathcal{Y}})$, we say that I_1 is a subset of I_2—or write $I_1 \subseteq I_2$—to mean $I_{1,\mathcal{X}} \subseteq I_{2,\mathcal{X}}$ and $I_{1,\mathcal{Y}} \subseteq I_{2,\mathcal{Y}}$. Similarly, we say I_1 and I_2 are disjoint—or write $I_1 \cap I_2 = \emptyset$—to mean $I_{1,\mathcal{X}} \cap I_{2,\mathcal{X}} = \emptyset$ and $I_{1,\mathcal{Y}} \cap I_{2,\mathcal{Y}} = \emptyset$. We define the union of interfaces I_1 and I_2 as $I_1 \cup I_2 \coloneqq (I_{1,\mathcal{X}} \cup I_{2,\mathcal{X}}, I_{1,\mathcal{Y}} \cup I_{2,\mathcal{Y}})$.

A set of interfaces $\mathcal{I}$ of an $(\mathcal{X}, \mathcal{Y})$-resource $\mathbf{R}$ is one such that any distinct interfaces $I_1, I_2 \in \mathcal{I}$ are disjoint, and the union of all interfaces in $\mathcal{I}$ is $\mathbf{R}$'s input and output domains, i.e. $(\mathcal{X}, \mathcal{Y}) = \bigcup_{I \in \mathcal{I}} I$.

When considering (simulator-based) security notions it is often helpful to have the notion of a party. For a set of n parties $\mathcal{P} \coloneqq (P_1, \ldots, P_n)$, one considers a set of interfaces $\mathcal{I}$ where for each party $P \in \mathcal{P}$ there is an interface $I_P = (I_{P,\mathcal{X}} \coloneqq (\{P\} \times \mathcal{X}_P), I_{P,\mathcal{Y}} \coloneqq (\{P\} \times \mathcal{Y}_P))$. We say that $I_{P,\mathcal{X}}$ and $I_{P,\mathcal{Y}}$ are P's input and output interfaces for $\mathbf{R}$, respectively.

Converters. A *converter* is an $(\mathcal{X}, \mathcal{Y})$-resource that is executed either locally by a single party or cooperatively by multiple parties. The inside interface connects to (a subset of those parties' interfaces of) the available resources, resulting in a new resource. For instance, connecting a converter α to Alice's interface A of a resource $\mathbf{R}$ results in a new resource denoted $\alpha^A\mathbf{R}$; we denote the inside interface of α by $\alpha.in$. The outside interface of α, denoted $\alpha.out$, is the new A-interface of $\alpha^A\mathbf{R}$. This means resource $\mathbf{R}$'s A interface is no longer present in the new resource $\alpha^A\mathbf{R}$: it is covered by converter α. Converters applied at different interfaces commute [10, Proposition 1]: $\beta^B\alpha^A\mathbf{R} \equiv \alpha^A\beta^B\mathbf{R}$.

A protocol is given by a tuple of converters $\pi = (\pi_{P_i})_{P_i \in \mathcal{P}}$, one for each party $P_i \in \mathcal{P}$. Simulators are also given by converters. For a party set $\mathcal{S}$, $\pi^{\mathcal{S}}\mathbf{R}$ denotes $(\pi_{P_i})_{P_i \in \mathcal{S}}\mathbf{R}$. When clear from context, we omit the interfaces π connects to, writing simply $\pi\mathbf{R}$.

Distinguishers. Analogous to a UC environment [3], a distinguisher is an interactive system $\mathbf{D}$ which interacts with a resource at all its interfaces and outputs a bit 0 or 1. The distinguishing advantage for distinguisher $\mathbf{D}$ is defined as

$$\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S}) := |\Pr[\mathbf{DS} = 1] - \Pr[\mathbf{DR} = 1]|$$

where $\mathbf{DR}$ and $\mathbf{DS}$ are the probability distributions induced by $\mathbf{D}$'s output when it interacts with $\mathbf{R}$ and $\mathbf{S}$, respectively.

Reductions. Typically one proves that the ability to distinguish between two resources is bounded by some function of the distinguisher, e.g. for any $\mathbf{D}$,

$$\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{S}) \leq |\varepsilon(\mathbf{D})|$$

where $\varepsilon(\mathbf{D})$ might be the probability that $\mathbf{D}$ can win a game or solves some problem believed to be hard.

Security Statements. We now have all the elements needed to define a cryptographic construction.

Definition 1 (Simulation-based construction). *Let $\mathbf{R}$ and $\mathbf{S}$ be two resources with a free interface I_F, and π a protocol for $\mathbf{R}$. We say π ε-constructs $\mathbf{S}$ from $\mathbf{R}$ if there is a simulator sim such that for any distinguisher $\mathbf{D}$, $\Delta^{\mathbf{D}}(\pi\mathbf{R}, \mathsf{sim}\mathbf{S}) \leq \varepsilon(\mathbf{D})$ and the interfaces of sim, of π and I_F are all pairwise disjoint.*

2.2 Modeling Access Control via Repositories

Similarly to [17], we model access control via repositories. A repository contains a set of registers and a corresponding set of register identifiers IdSet; a register is a pair $\mathtt{reg} = (\mathtt{id}, m)$, where m is a message and $\mathtt{id}$ is the register's identifier, which uniquely identifies it among all repositories. We consider two types of repository access rights: *read access* and *write access*. (This is in contrast to [17], which additionally considers *copy access*.) We denote by $\mathcal{W}$ and $\mathcal{R}$ the sets of parties with write and read access to a repository $\mathbf{rep}$, respectively; to make the access permissions explicit we write $\mathbf{rep}^{\mathcal{W}}_{\mathcal{R}}$, but otherwise simply write $\mathbf{rep}$. For example, consider a three party setting with a sender Alice, a receiver Bob and a dishonest third-party Eve—so $\mathcal{P} = \{A, B, E\}$. An insecure repository—which allows everyone to read and write—is given by $\mathbf{INS}^{\mathcal{P}}_{\mathcal{P}}$; a (replay-protected) authentic repository from Alice to Bob is given by $\mathbf{AUT}^{\{A\}}_{\{B,E\}}$. The semantics of atomic repositories is defined in Algorithm 1.[6]

[6] As needed to capture the Off-The-Record guarantee, the repository semantics capture sender anonymity: for a repository $\mathbf{rep}^{\mathcal{W}}_{\mathcal{R}}$, readers do not learn who wrote (among the parties in $\mathcal{W}$) each of the repository's messages.

Algorithm 1 Atomic repository $\mathbf{rep}_{\mathcal{R}}^{\mathcal{W}}$.

◇ Initialization
  IdSet $\leftarrow \emptyset$

▷ $(P \in \mathcal{W})$-Write(m)
  id $\leftarrow$ NewRegister(m)
  IdSet $\leftarrow$ IdSet $\cup$ {id}
  Output(id)

▷ $(P \in \mathcal{R})$-Read
  list $\leftarrow \emptyset$
  for id $\in$ IdSet **:**
    list $\leftarrow$ list $\cup$ {(id, GetMessage(id))}
  Output(list)

Algorithm 2 Repository $\mathbf{REP} = [\mathbf{rep_1}_{\mathcal{R}_1}^{\mathcal{W}_1}, \ldots, \mathbf{rep_n}_{\mathcal{R}_n}^{\mathcal{W}_n}]$.

▷ $(P \in \mathcal{P})$-Write$(\mathbf{rep_i}_{\mathcal{R}_i}^{\mathcal{W}_i}, m)$
Require: $(P \in \mathcal{W}_i)$
  Output($\mathbf{rep_i}$-Write(m))

▷ $(P \in \mathcal{P})$-Read
  list $\leftarrow \emptyset$
  for $\mathbf{rep_i} \in \mathbf{REP}$ **with** $P \in \mathcal{R}_i$ **:**
    for $(\mathtt{id}, m) \in \mathbf{rep_i}$-Read **:**
      list $\leftarrow$ list $\cup$ (id, $(\mathbf{rep_i}, m)$)
  Output(list)

Following [17], to model that parties may have access to multiple repositories—say $\mathbf{rep_1}_{\mathcal{R}_1}^{\mathcal{W}_1}, \ldots, \mathbf{rep_n}_{\mathcal{R}_n}^{\mathcal{W}_n}$—we define a new type of repository denoted $\mathbf{REP} = [\mathbf{rep_1}_{\mathcal{R}_1}^{\mathcal{W}_1}, \ldots, \mathbf{rep_n}_{\mathcal{R}_n}^{\mathcal{W}_n}]$, which is essentially a parallel composition of atomic repositories equipped with a single read operation that allows parties to (efficiently) read all their incoming messages at once (instead of having to read from each atomic repository $\mathbf{rep_i}$ they have access to). The exact semantics of **REP** is defined in Algorithm 2.

2.2.1 Modeling an Asynchronous Network To model an asynchronous network we define a network converter Net (in Algorithm 3), which provides an interface for message delivery and ensures honest receivers only read delivered messages.

Algorithm 3 Semantics of Net for a repository $\mathbf{REP} = [\mathbf{rep_1}, \ldots, \mathbf{rep_n}]$.

◇ Initialization
  for $P_i \in \mathcal{P}$ **:**
    Received$[P_i] \leftarrow \emptyset$

▷ $(P \in \mathcal{P}^H)$-Read
  list $\leftarrow \emptyset$
  for $(\mathtt{id}, (\mathbf{rep_i}, m)) \in$ Read **:**
    if id $\in$ Received$[P]$ **:**
      list $\leftarrow$ list $\cup$ (id, $(\mathbf{rep_i}, m)$)
  Output(list)

▷ $(P \in \mathcal{P})$-Write$(\mathbf{rep_i}, m)$
  Output(Write$(\mathbf{rep_i}, m)$)

▷ $(P \in \overline{\mathcal{P}^H})$-Read
  Output(Read)

▷ Deliver$(P \in \mathcal{P}^H, \mathtt{id})$
  Received$[P] \leftarrow$ Received$[P] \cup$ {id}

2.3 Multi-Designated Verifier Signatures

Following [4], an MDVS scheme Π for a message space $\mathcal{M}$ is a 6-tuple of PPTs $\Pi = (S, G_S, G_V, Sig, Vfy, Forge)$, where:

$S(1^k)$**:** generates public parameters $\mathtt{pp}$;
$G_S(\mathtt{pp})$**:** generates a signer key-pair ($\mathtt{spk}$, $\mathtt{ssk}$);
$G_V(\mathtt{pp})$**:** generates a verifier key-pair ($\mathtt{vpk}$, $\mathtt{vsk}$);
$Sig(\mathtt{pp}, \mathtt{ssk}, \vec{v}, m)$**:** generates a signature σ, where $\mathtt{ssk}$ is the signer's secret key, $\vec{v}$ is the vector of public verifier keys of the designated verifiers and $m \in \mathcal{M}$ is the message;
$Vfy(\mathtt{pp}, \mathtt{spk}, \mathtt{vsk}, \vec{v}, m, \sigma)$**:** outputs a bit indicating whether σ is a valid signature on message m with respect to signer's public key $\mathtt{spk}$ and vector of verifier public keys $\vec{v}$, where $\mathtt{vsk}$ is a verifier's secret key;
$Forge(\mathtt{pp}, \mathtt{spk}, \vec{v}, m, \vec{s})$**:** generates a forged signature σ, where $\mathtt{spk}$, $\vec{v}$ and m are as before, and $\vec{s}$ is a vector of designated verifiers' secret keys—with $|\vec{s}| = |\vec{v}|$ and where for $i \in \{1, \ldots, |\vec{v}|\}$, either $s_i = \perp$ or s_i is the secret key corresponding to the i-th public key of $\vec{v}$, i.e. v_i.

2.3.1 Security Notions The security games below have an implicitly defined security parameter k and provide adversaries with access to a set of oracles which, for an MDVS scheme $\Pi = (S, G_S, G_V, Sig, Vfy, Forge)$ are defined as follows:

$\mathcal{O}_{PP}$**:** On the first query, compute $\mathtt{pp} \leftarrow S(1^k)$; output $\mathtt{pp}$.
$\mathcal{O}_{SK}(A_i)$**:** On the first query $\mathcal{O}_{SK}(A_i)$, compute $(\mathtt{spk}_i, \mathtt{ssk}_i) \leftarrow G_S(\mathtt{pp})$; output $(\mathtt{spk}_i, \mathtt{ssk}_i)$.
$\mathcal{O}_{VK}(B_j)$**:** Analogous to $\mathcal{O}_{SK}(A_i)$.
$\mathcal{O}_{SPK}(A_i)$**:** for $(\mathtt{spk}_i, \mathtt{ssk}_i) \leftarrow \mathcal{O}_{SK}(A_i)$; output $\mathtt{spk}_i$.
$\mathcal{O}_{VPK}(B_j)$**:** Analogous to $\mathcal{O}_{SPK}(A_i)$.
$\mathcal{O}_S(A_i, \vec{V}, m)$**:** for $(\mathtt{spk}_i, \mathtt{ssk}_i) \leftarrow \mathcal{O}_{SK}(A_i)$, $\vec{v} = (\mathcal{O}_{VPK}(V_1), \ldots, \mathcal{O}_{VPK}(V_{|\vec{V}|}))$, output $\sigma \leftarrow Sig_{\mathtt{pp}}(\mathtt{ssk}_i, \vec{v}, m)$.
$\mathcal{O}_V(A_i, B_j \in \mathrm{Set}(\vec{V}), \vec{V}, m, \sigma)$**:** for $\mathtt{spk}_i$, $\vec{v}$ as above, output $Vfy_{\mathtt{pp}}(\mathtt{spk}_i, \mathtt{vsk}_j, \vec{v}, m, \sigma)$.

The notions ahead—Correctness, Consistency, Unforgeability, Off-The-Record and Message-Bound Validity—give adversaries access to all the oracles above (for Off-The-Record, oracles $\mathcal{O}_S$ and $\mathcal{O}_V$ behave differently, as we will explain). For conciseness, we simply omit the oracles in these notions' definitions.

Definition 2 (Correctness: $\mathbf{G}^{\mathsf{Corr}}$). *An adversary* $\mathbf{A}$ *wins if there are two queries* q_S *and* q_V *to* $\mathcal{O}_S$ *and* $\mathcal{O}_V$*, respectively, where* q_S *has input* $(A_i, \vec{V}, m)$ *and* q_V *has input* $(A_i{}', B_j, \vec{V}', m', \sigma)$*, satisfying* $(A_i, \vec{V}, m) = (A_i{}', \vec{V}', m')$*,* $B_j \in \vec{V}$*, the input* σ *in* q_V *is the output of the oracle* $\mathcal{O}_S$ *on query* q_S*, and the output of the oracle* $\mathcal{O}_V$ *on the query* q_V *is* 0*.*

Definition 3 (Consistency: $\mathbf{G}^{\mathsf{Cons}}$). *An adversary* $\mathbf{A}$ *wins if it makes two queries* $\mathcal{O}_V(A_i, B_j, \vec{V}, m, \sigma)$ *and* $\mathcal{O}_V(A_i{}', B_j{}', \vec{V}', m', \sigma')$ *such that* $(A_i, \vec{V}, m, \sigma) = (A_i{}', \vec{V}', m', \sigma')$, $\{B_j, B_j{}'\} \subseteq \vec{V}$, *the outputs of the two queries differ, and there is no query* $\mathcal{O}_{VK}(B_j)$ *prior to query* $\mathcal{O}_V(A_i, B_j, \vec{V}, m, \sigma)$, *and no query* $\mathcal{O}_{VK}(B_j{}')$ *prior to query* $\mathcal{O}_V(A_i{}', B_j{}', \vec{V}', m', \sigma')$.

Definition 4 (Unforgeability: $\mathbf{G}^{\mathsf{Unforg}}$). *An adversary* $\mathbf{A}$ *wins if it makes a query* $\mathcal{O}_V(A_i{}^*, B_j{}^*, \vec{V}^*, m^*, \sigma^*)$ *with* $B_j{}^* \in \vec{V}^*$ *that outputs* 1, *for every query* $\mathcal{O}_S(A_i{}', \vec{V}', m')$, $(A_i{}^*, \vec{V}^*, m^*) \neq (A_i{}', \vec{V}', m')$ *and there is no* $\mathcal{O}_{SK}$ *query on* $A_i{}^*$ *nor* $\mathcal{O}_{VK}$ *query on* $B_j{}^*$.

The games defined by the Off-The-Record notion give adversaries access to the oracles from before as well as to (modified) oracles $\mathcal{O}_S$ and $\mathcal{O}_V$. For $\mathbf{b} \in \{0, 1\}$, game $\mathbf{G}_{\mathbf{b}}^{\mathsf{OTR}}$'s these oracles behave as follows:

$\mathcal{O}_S(\mathtt{type} \in \{\mathtt{sig}, \mathtt{sim}\}, A_i, \vec{V}, m, \mathcal{C} \subseteq \mathrm{Set}(\vec{V}))$**:** 1. $(\mathtt{spk}_i, \mathtt{ssk}_i) \leftarrow \mathcal{O}_{SK}(A_i)$;
2. Let $\vec{v} = (v_1, \ldots, v_{|\vec{V}|})$ and $\vec{s} = (s_1, \ldots, s_{|\vec{V}|})$ where for $i \in [|\vec{V}|]$:
   - $(v_i, s_i) = \begin{cases} \mathcal{O}_{VK}(V_i) & \text{if } V_i \in \mathcal{C} \\ (\mathcal{O}_{VPK}(V_i), \perp) & \text{otherwise;} \end{cases}$
3. $(\sigma_{\mathbf{0}}, \sigma_{\mathbf{1}}) \leftarrow (\Pi.Sig_{\mathtt{pp}}(\mathtt{ssk}_i, \vec{v}, m), \Pi.Forge_{\mathtt{pp}}(\mathtt{spk}_i, \vec{v}, m, \vec{s}))$;
4. If $\mathbf{b} = 0$, output $\sigma_{\mathbf{0}}$ if $\mathtt{type} = \mathtt{sig}$ and $\sigma_{\mathbf{1}}$ if $\mathtt{type} = \mathtt{sim}$; otherwise, if $\mathbf{b} = 1$, output $\sigma_{\mathbf{1}}$.

$\mathcal{O}_V(A_i, B_j \in \mathrm{Set}(\vec{V}), \vec{V}, m, \sigma)$**:** 1. If σ was output by a query to $\mathcal{O}_S$ on an input $(\cdot, A_i{}', \vec{V}', m', \mathcal{C})$ such that $(A_i{}', \vec{V}', m') = (A_i, \vec{V}, m)$ and with $B_j \in \vec{V}$, output $\mathtt{test}$;
2. Otherwise, compute $b \leftarrow Vfy_{\mathtt{pp}}(\mathtt{spk}_i, \mathtt{vsk}_j, \vec{v}, m, \sigma)$; output b.

Definition 5 (Off-The-Record: $\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}$ and $\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}$). *An adversary* $\mathbf{A}$ *wins if it outputs a guess bit* $b' = \mathbf{b}$, *and for every query* $\mathcal{O}_S(\mathtt{type}, A_i, \vec{V}, m, \mathcal{C})$ *there is no query* $\mathcal{O}_{VK}(B_j)$ *with* $B_j \in Set(\vec{V}) \setminus \mathcal{C}$.

Definition 6 (Message-Bound Validity: $\mathbf{G}^{\mathsf{Bound\text{-}Val}}$). *An adversary* $\mathbf{A}$ *wins if there are two queries* q_S *and* q_V *to* $\mathcal{O}_S$ *and* $\mathcal{O}_V$, *respectively, where* q_S *has input* $(A_i, \vec{V}, m)$ *and* q_V *has input* $(A_i{}', B_j, \vec{V}', m', \sigma)$, *satisfying 1.* $(A_i, \vec{V}) = (A_i{}', \vec{V}')$; *2.* $B_j \in \vec{V}$; *3.* $m \neq m'$; *4. the input* σ *in* q_V *is* $\mathcal{O}_S$*'s output on query* q_S; *and 5. the output of* $\mathcal{O}_V$ *on query* q_V *is* 1.

2.4 Multi-Designated Receiver Signed Public Key Encryption

Introduced in [18], an MDRS-PKE scheme Π for a message space $\mathcal{M}$ is a 6-tuple of PPTs $\Pi = (S, G_S, G_R, E, D, Forge)$, where S, G_S, G_R, E and *Forge* are analogous to an MDVS's S, G_S, G_V, *Sig* and *Forge* PPTs, respectively (for an MDRS-PKE, E and *Forge* output ciphertexts instead of signatures), and

$D(\mathtt{pp}, \mathtt{rsk}_j, c)$**:** outputs a triple $(\mathtt{spk}, \vec{v}, m)$—where $\mathtt{spk}$ is a sender's public key, $\vec{v}$ a vector of receiver public keys, and m a message—or $\perp$ if decryption fails.

2.4.1 Security Notions [4, 18] Let $\Pi = (S, G_S, G_R, E, D, Forge)$ be an MDRS-PKE scheme with message space $\mathcal{M}$. The games defined by the notions below give adversaries access to oracles $\mathcal{O}_{PP}$, $\mathcal{O}_{SK}$, $\mathcal{O}_{SPK}$, $\mathcal{O}_{RK}$, $\mathcal{O}_{RPK}$ and $\mathcal{O}_E$ that are analogous to MDVS's oracles $\mathcal{O}_{PP}$, $\mathcal{O}_{SK}$, $\mathcal{O}_{SPK}$, $\mathcal{O}_{VK}$, $\mathcal{O}_{VPK}$ and $\mathcal{O}_S$, respectively, plus to the following oracle:

$\mathcal{O}_D(B_j, c)$**:** 1. $(\cdot, \mathtt{rsk}_j) \leftarrow \mathcal{O}_{RK}(B_j)$;
2. $(\mathtt{spk}_i, \vec{v}, m) \leftarrow D_{\mathtt{pp}}(\mathtt{rsk}_j, c)$;
3. if, for each party A_i previously input to either $\mathcal{O}_{SK}$, $\mathcal{O}_{SPK}$ or $\mathcal{O}_E$, $\mathtt{spk}_i \neq \mathcal{O}_{SPK}(A_i)$, then output $\perp$;
4. if, for some $l \in \{1, \ldots, |\vec{V}|\}$, there is no party B_j that was previously input to either $\mathcal{O}_{RK}$, $\mathcal{O}_{RPK}$, $\mathcal{O}_E$ or $\mathcal{O}_D$ such that $v_l = \mathcal{O}_{RPK}(V_l)$, then output $\perp$;
5. output $(\mathtt{spk}, \vec{v}, m)$.

As for MDVS, the notions ahead give adversaries access to all oracles above, so for succinctness we omit them in these notions' definitions. Below, we define the Correctness, Consistency and Off-The-Record notions for MDRS-PKE schemes; we do not introduce unforgeability and CCA notions here because we do not know how to prove that the notions considered in the literature imply our composable notions; we will define strengthenings of these notions later in the paper.

Definition 7 (Correctness: $\mathbf{G}^{\mathsf{Corr}}$). *An adversary* **A** *wins the game if there is a query q_E to $\mathcal{O}_E$ and a later query q_D to $\mathcal{O}_D$ such that q_E has input $(A_i, \vec{V}, m)$ and q_D has input (B_j, c) with $B_j \in \vec{V}$ and c being the output of q_E, the output of q_D is $(\mathtt{spk}_i{}', \vec{v}', m')$ with $(\mathtt{spk}_i{}', \vec{v}', m') \neq (\mathtt{spk}_i, \vec{v}, m)$—where $\mathtt{spk}_i$ is A_i's public key and $\vec{v}$ is the vector of public keys corresponding to $\vec{V}$.*

The Consistency notion below slightly differs from the one given in [4]: it additionally captures the (natural) property that if the decryption of a ciphertext c by a party B_j outputs some valid triple $(\mathtt{spk}, \vec{v}, m) \neq \perp$, then B_j's public key $\mathtt{rpk}_j$ must be part of the vector $\vec{v}$ output by decryption (i.e. $\mathtt{rpk}_j \in \vec{v}$).[7] As we will see, this is useful because it eliminates the need that a receivers' protocol makes this additional check.

Definition 8 (Consistency: $\mathbf{G}^{\mathsf{Cons}}$). *An adversary* **A** *wins if (at least) one of the following events (ξ_1 or ξ_2) occurs:*

Event ξ_1: *there is a query $\mathcal{O}_D(B_i, c)$ that outputs some triple $(\mathtt{spk}, \vec{v}, m)$ with $(\mathtt{spk}, \vec{v}, m) \neq \perp$ and $\mathtt{pk}_i \notin \vec{v}$, where $\mathtt{pk}_i$ is B_i's public key;*

Event ξ_2: *there are two $\mathcal{O}_D$ queries, say q_{D_i} and q_{D_j}, on inputs, respectively, (B_i, c) and (B_j, c') with $c = c'$ such that: 1. the outputs of q_{D_i} and q_{D_j} differ; 2. either the receiver public key $\mathtt{rpk}_j$ of B_j is part of the vector of receiver public keys output by q_{D_i}, or the receiver public key $\mathtt{rpk}_i$ of B_i is part of*

[7] Maurer et al.'s MDRS-PKE construction [18] satisfies this modified notion as decryption checks if the public key of the receiver decrypting the ciphertext is part of the public key vector to be output.

the vector of public keys output by q_{D_j}; 3. there is no query $\mathcal{O}_{RK}(B_i)$ (resp. $\mathcal{O}_{RK}(B_j)$) prior to q_{D_i} (resp. q_{D_j}); and 4. there is no sender A (resp. no receiver B) which had not been input to a query $\mathcal{O}_{SPK}$, $\mathcal{O}_{SK}$ or $\mathcal{O}_E$ (resp. $\mathcal{O}_{RPK}$, $\mathcal{O}_{RK}$ or $\mathcal{O}_E$) prior to both q_{D_i} and q_{D_j} and whose public key is output by one of these queries.

For $\mathbf{b} \in \{0,1\}$, the $\mathcal{O}_E$ and $\mathcal{O}_D$ oracles that game $\mathbf{G}_{\mathbf{b}}^{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}}$ provides to an adversary are as follows:

$\mathcal{O}_E\big((A_{i,0},\vec{V}_0,m_0),(A_{i,1},\vec{V}_1,m_1)\big)$**:** output $c \leftarrow \Pi.E_{\mathtt{pp}}(\mathtt{ssk}_{i,\mathbf{b}},\vec{v_{\mathbf{b}}},m_{\mathbf{b}})$.
$\mathcal{O}_D(B_j,c)$**:** If c was output by an $\mathcal{O}_E$ query, output `test`; otherwise proceed as in the default $\mathcal{O}_D$ oracle.

Definition 9 ($\{\mathsf{IND},\mathsf{IK}\}$-CCA Security [4]: $\mathbf{G}_{\mathbf{0}}^{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}}$ and $\mathbf{G}_{\mathbf{1}}^{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}}$). *An adversary $\mathbf{A}$ wins if it outputs a guess bit b' with $b' = \mathbf{b}$ and for every query $\mathcal{O}_E\big((A_{i,0},\vec{V}_0,m_0),(A_{i,1},\vec{V}_1,m_1)\big)$: 1. $|m_0| = |m_1|$; 2. $|\vec{V}_0| = |\vec{V}_1|$; and 3. there is no query to $\mathcal{O}_{RK}$ on any $B_j \in Set(\vec{V}_0) \cup Set(\vec{V}_1)$.*

Off-The-Record defines games $\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}$ and $\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}$ which give adversaries access to the oracles from before and to modified $\mathcal{O}_E$ and $\mathcal{O}_D$ oracles. Oracle $\mathcal{O}_E$ is defined analogously to MDVS' OTR games' oracle $\mathcal{O}_S$, whereas $\mathcal{O}_D$ is as follows:

$\mathcal{O}_D(B_j,c)$**:** 1. If c was the output of some query to $\mathcal{O}_E$, output `test`; 2. Otherwise, proceed as in the original $\mathcal{O}_D$ oracle.

Definition 10 (Off-The-Record [4]: $\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}$ and $\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}$). *Adversary $\mathbf{A}$ wins if it outputs a guess bit b' with $b' = \mathbf{b}$ and for every query $\mathcal{O}_E(\mathtt{type},A_i,\vec{V},m,\mathcal{C})$ and every query $\mathcal{O}_{VK}(B_j)$, we have $B_j \notin Set(\vec{V}) \setminus \mathcal{C}$.*

3 New and Stronger Game-Based Notions for MDVS

In this section we introduce a stronger unforgeability notion—which captures security against replay attacks—plus Forgery Invalidity: a new guarantee that we identified thanks to our composable treatment of MDVS schemes. We introduce them because we assume them in our composable treatment of MDVS schemes (and do not know how to prove the existing MDVS game-based notions imply our composable notions without requiring them). Lastly, we prove that Chakraborty et al.'s MDVS [4] satisfies all these notions.

3.1 (New) Security Notions

The security definitions given in Section 2.3 do not give any guarantee on whether a signature forgery on messages picked by an adversary who can access the secret key of the sender may not verify as valid by honest receivers; this is not captured by Unforgeability (Definition 4) because the adversary could choose messages to be forged depending on the signer's secret key (which it does not have access

to in the Unforgeability game). Forgery Invalidity captures this guarantee: that forged signatures are not valid even when messages are picked by adversaries who know senders' secret keys.

In addition to the oracles from Section 2.3.1 (which, as before, for simplicity we omit in notions below), game $\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$ also gives adversaries access to the following new oracle:

$\mathcal{O}_{Forge}(A_i, \vec{V}, m, \mathcal{C} \subseteq \mathrm{Set}(\vec{V}))$**:** let $\mathtt{spk}_i \leftarrow \mathcal{O}_{SPK}(A_i)$, and for $i \in [|\vec{V}|]$, let $(v_i, s_i) = \mathcal{O}_{VK}(V_i)$ for $V_i \in \mathcal{C}$, and $(v_i, s_i) = (\mathcal{O}_{VPK}(V_i), \perp)$ for $V_i \notin \mathcal{C}$; output $\Pi.Forge_{\mathtt{pp}}(\mathtt{spk}_i, \vec{v}, m, \vec{s})$, where $\vec{v} = (v_1, \ldots, v_{|\vec{V}|})$ and $\vec{s} = (s_1, \ldots, s_{|\vec{V}|})$.

Definition 11 (Forgery Invalidity: $\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$). *An adversary* $\mathbf{A}$ *wins the game if there are two queries* q_{Forge} *and* q_V *to* $\mathcal{O}_{Forge}$ *and* $\mathcal{O}_V$*, respectively, where* q_{Forge} *has input* $(A_i, \vec{V}, m, \mathcal{C})$ *and* q_V *has input* $(A_i{}', B_j, \vec{V}', m', \sigma)$*, satisfying 1.* $(A_i, \vec{V}, m) = (A_i{}', \vec{V}', m')$*; 2.* $B_j \in \vec{V}$*; 3.* $B_j \notin \mathcal{C}$*; 4. the input* σ *in* q_V *is the output of* $\mathcal{O}_{Forge}$ *on query* q_{Forge}*; and 5. the output of the oracle* $\mathcal{O}_V$ *on the query* q_V *is* 1*.*

An adversary $\mathbf{A}$ (ε, t)-breaks the $(n_S, n_V, d_S, d_F, q_S, q_V, q_F)$-Forgery Invalidity of Π if $\mathbf{A}$ runs in time at most t, queries $\mathcal{O}_{SK}$, $\mathcal{O}_{SPK}$, $\mathcal{O}_S$, $\mathcal{O}_V$ and $\mathcal{O}_{Forge}$ on at most n_S different signers, $\mathcal{O}_{VK}$, $\mathcal{O}_{VPK}$, $\mathcal{O}_S$, $\mathcal{O}_V$ and $\mathcal{O}_{Forge}$ on at most n_V different verifiers, makes at most q_S, q_V and q_F queries to $\mathcal{O}_S$, $\mathcal{O}_V$ and $\mathcal{O}_{Forge}$, respectively, with the sum of the verifier vectors' lengths input to $\mathcal{O}_S$ and $\mathcal{O}_{Forge}$ being at most d_S and d_F, respectively, and satisfies $Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{A}) \geq \varepsilon$.

Definition 12 (Unforgeability against Replays: $\mathbf{G}^{\mathsf{R\text{-}Unforg}}$). *An adversary* $\mathbf{A}$ *wins if it makes a query* $\mathcal{O}_V({A_i}^*, {B_j}^*, \vec{V}^*, m^*, \sigma^*)$ *with* ${B_j}^* \in \vec{V}^*$ *that outputs* 1*, for every query* $\mathcal{O}_S(A_i{}', \vec{V}', m')$*,* $({A_i}^*, \vec{V}^*, m^*, \sigma^*) \neq (A_i{}', \vec{V}', m', \sigma')$*—*$\sigma'$ *being the output of query* $\mathcal{O}_S(A_i{}', \vec{V}', m')$*—and there is no* $\mathcal{O}_{SK}$ *query on* ${A_i}^*$ *nor* $\mathcal{O}_{VK}$ *query on* ${B_j}^*$*.*

We use the notion below to prove Maurer et al.'s MDRS-PKE construction satisfies an analogous MDRS-PKE notion, which in turn significantly simplifies our composable treatment of MDRS-PKE schemes. (Concretely, this trivial property allows keeping our composable proofs much simpler because by assuming it we avoid having to prove it does hold in the composable proofs—which would require dealing with extra artifacts inherent from composable security notions.)

Definition 13 (Public-Key Collision Resistance). *An* MDVS *scheme* $\Pi = (S, G_S, G_V, Sig, Vfy, Forge)$ *is* (n, ℓ)*-Party* ε*-Public-Key Collision Resistant if*

$$\Pr\left[\begin{array}{l|l} \left|\{\mathtt{spk}_1, \ldots, \mathtt{spk}_n, \right. & \mathtt{pp} \leftarrow S(1^k), \\ \left. \mathtt{vpk}_1, \ldots, \mathtt{vpk}_\ell\}\right| & (\mathtt{spk}_i, \cdot) \leftarrow G_S(\mathtt{pp}), i \in [n] \\ < n + \ell & (\mathtt{vpk}_j, \cdot) \leftarrow G_V(\mathtt{pp}), j \in [\ell] \end{array}\right] \leq \varepsilon.$$

3.2 Security of Chakraborty et al.'s MDVS [4]

We prove the security of Chakraborty et al.'s MDVS construction [4]—denoted $\Pi^{\mathsf{adap}}_{\mathrm{MDVS}}$ and defined in Algorithm 11—with respect to the new security notions. Its building blocks are a Public Key Encryption scheme Π_{PKE}, a One Way Function Π_{OWF} and a Non Interactive Zero Knowledge Π_{NIZK}; the informal theorem below summarizes our results regarding $\Pi^{\mathsf{adap}}_{\mathrm{MDVS}}$'s additional security guarantees. (Regarding replay unforgeability, we the original argument from [4], establishing the unforgeability of their construction, also implies the stronger notion we consider [5, Proof of Theorem 6].)

Theorem 1 (Informal). *If 1. Π_{PKE} is correct and tightly multi-user and multi-challenge* IND-CPA *secure under non-adaptive corruptions; 2. Π_{NIZK} is tightly multi-statement adaptive zero-knowledge and tightly multi-statement simulation–sound; and 3. Π_{OWF} is tightly multi-instance secure under non-adaptive corruptions, then $\Pi^{\mathsf{adap}}_{\mathrm{MDVS}}$ is tightly Forgery Invalidity secure (Theorem 6), tightly Unforgeability against Replays ([4, Theorem 6], Theorem 5 [8]) and is Public-Key Collision Resistant (Corollary 1).*

Remark 1. We note that our Forgery Invalidity proof of Chakraborty et al.'s MDVS does not require any additional guarantees from its underlying building blocks and our reductions to prove the Forgery Invalidity security of their scheme are also tight. Overall, this means their construction can still be instantiated from building blocks that are known to have (compatible) structure preserving instantiations with tight security reductions to standard assumptions [4].

4 New and Stronger Game-Based Notions for MDRS-PKE

In this section we introduce a stronger unforgeability notion for MDRS-PKE schemes—analogous to the one we introduced for MDVS schemes—and a new Forgery Invalidity notion—also analogous to the one we introduced for MDVS. As for the stronger MDVS notions, we introduce these notions because our composable treatment of MDVS assumes them, and furthermore we do not know how to prove the MDRS-PKE composable semantics (from Section 6) are implied by the MDRS-PKE game-based notions without relying on the stronger notions we now present. For completeness, we also present a stronger MDRS-PKE {IND, IK}-CCA notion: without this notion we do not know how to prove the composable semantics of the MDRS-PKE game-based in the setting where *senders secret keys do not leak*. (The original notion from [18]—where honest senders' secret keys cannot be obtained via oracle $\mathcal{O}_{SK}$—does not allow for challenge queries $\mathcal{O}_E((A_{i,0}, \vec{V}_0, m_0), (A_{i,1}, \vec{V}_1, m_1))$ for which there may be some query $\mathcal{O}_{RK}(B_j \in \mathrm{Set}(\vec{V}_0) \cup \mathrm{Set}(\vec{V}_1))$; we do not know how to make a reduction to that weaker notion because it is not clear to us how to generate encryptions of messages from honest senders to vectors of receivers that include dishonest ones

[8] The proof of [4, Theorem 6] already implies replay unforgeability of their construction.

without having access to honest senders' secret keys.) Finally, we show Maurer et al.'s MDRS-PKE [18,19] satisfies all these new notions (our Forgery Invalidity reduction assumes the analogous guarantee from the underlying MDVS). Put together with Theorem 1 this means Chakraborty et al.'s construction can be used as the MDVS underlying Maurer et al.'s MDRS-PKE.

4.1 (New) Security Notions

Game $\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$ defined by the Forgery Invalidity notion provides adversaries with access to the oracles from above plus oracle $\mathcal{O}_{Forge}$ below:

$\mathcal{O}_{Forge}(A_i, \vec{V}, m, \mathcal{C} \subseteq \mathrm{Set}(\vec{V}))$**:** 1. $\mathtt{spk}_i \leftarrow \mathcal{O}_{SPK}(A_i)$; 2. for $i = 1, \ldots, |\vec{V}|$, if $V_i \in \mathcal{C}$ let $(v_i, s_i) = \mathcal{O}_{RK}(V_i)$, and otherwise let $(v_i, s_i) = (\mathcal{O}_{RPK}(V_i), \perp)$; 3. output $\Pi.Forge_{\mathtt{pp}}(\mathtt{spk}_i, \vec{v}, m, \vec{s})$, where $\vec{v} = (v_1, \ldots, v_{|\vec{V}|})$, $\vec{s} = (s_1, \ldots, s_{|\vec{V}|})$.

Definition 14 (Forgery Invalidity: $\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$). *An adversary* $\mathbf{A}$ *wins if there is a query* $\mathcal{O}_{Forge}(A_i, \vec{V}, m, \mathcal{C})$ *and a later query* $\mathcal{O}_D(B_j, c)$ *such that: 1.* $B_j \in \vec{V}$*; 2.* $B_j \notin \mathcal{C}$*; 3. the input* c *to* $\mathcal{O}_D$ *is the output of* $\mathcal{O}_{Forge}$*; and 4. the output of* $\mathcal{O}_D$ *is not* $\perp$*.*

Definition 15 (Replay Unforgeability: $\mathbf{G}^{\mathsf{R\text{-}Unforg}}$). *An adversary* $\mathbf{A}$ *wins if it makes a query* $\mathcal{O}_D(B_j, c)$ *that outputs* $(\mathtt{spk}_i, \vec{v}, m) \neq \perp$*, there is a sender* A_i *and a vector of receivers* $\vec{V}$ *such that* $\mathtt{spk}_i$ *is* A_i*'s sender public key (i.e.* $\mathcal{O}_{SPK}(A_i) = \mathtt{spk}_i$*) and* $\vec{v}$ *is the vector of receiver public keys corresponding to* $\vec{V}$ *(i.e.* $|\vec{V}| = |\vec{v}|$ *and for each* $l \in \{1, \ldots, |\vec{v}|\}$*,* $\mathcal{O}_{RPK}(V_l) = v_l$*), there was no query* $\mathcal{O}_E(A_i', \vec{V}', m')$ *with* $(A_i, \vec{V}, m) = (A_i', \vec{V}', m')$ *that output the same ciphertext* c *that was input to* $\mathcal{O}_D$*, and neither* $\mathcal{O}_{SK}$ *was queried on input* A_i *nor* $\mathcal{O}_{RK}$ *was queried on input* B_j*.*

Definition 16 (Public-Key Collision Resistance). MDRS-PKE $\Pi = (S, G_S, G_R, E, D, Forge)$ *is* (n, ℓ)*-Party* ε*-Public-Key Collision Resistant if*

$$\Pr\left[\begin{array}{l|c} \big|\{\mathtt{spk}_1, \ldots, \mathtt{spk}_n, & \mathtt{pp} \leftarrow S(1^k), \\ \quad \mathtt{rpk}_1, \ldots, \mathtt{rpk}_\ell\}\big| & (\mathtt{spk}_i, \cdot) \leftarrow G_S(\mathtt{pp}), i \in [n] \\ \quad < n + \ell & (\mathtt{rpk}_j, \cdot) \leftarrow G_R(\mathtt{pp}), j \in [\ell] \end{array}\right] \leq \varepsilon.$$

For $\mathbf{b} \in \{0, 1\}$, the $\mathcal{O}_E$ and $\mathcal{O}_D$ oracles that game $\mathbf{G}_{\mathbf{b}}^{\{\mathsf{IND},\,\mathsf{IK}\}\text{-}\mathsf{CCA}_S}$ provides to an adversary are as follows:

$\mathcal{O}_E\big((A_{i,0}, \vec{V_0}, m_0), (A_{i,1}, \vec{V_1}, m_1)\big)$**:** output $c \leftarrow \Pi.E_{\mathtt{pp}}(\mathtt{ssk}_{i,\mathbf{b}}, \vec{v_{\mathbf{b}}}, m_{\mathbf{b}})$.
$\mathcal{O}_D(B_j, c)$**:** If c was output by an $\mathcal{O}_E$ query, output $\mathtt{test}$; otherwise proceed as in the default $\mathcal{O}_D$ oracle.

Definition 17 ($\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}_S$ Security: $\mathbf{G}_{\mathbf{0}}^{\{\mathsf{IND},\,\mathsf{IK}\}\text{-}\mathsf{CCA}_S}$ and $\mathbf{G}_{\mathbf{1}}^{\{\mathsf{IND},\,\mathsf{IK}\}\text{-}\mathsf{CCA}_S}$). *An adversary* $\mathbf{A}$ *wins if it outputs guess bit* $b' = \mathbf{b}$ *and for every oracle* $\mathcal{O}_E$ *query* $\mathcal{O}_E\big((A_{i,0}, \vec{V_0}, m_0), (A_{i,1}, \vec{V_1}, m_1)\big)$*: 1.* $(|m_0|, |\vec{V_0}|) = (|m_1|, |\vec{V_1}|)$*; and 2. if* $(A_{i,0}, \vec{V_0}, m_0) \neq (A_{i,1}, \vec{V_1}, m_1)$*, there is no query* $\mathcal{O}_{SK}(A \in \{A_{i,0}, A_{i,1}\})$ *nor* $\mathcal{O}_{RK}(B_j \in Set(\vec{V_0}) \cup Set(\vec{V_1}))$*.*

4.2 Security of Maurer et al.'s MDRS-PKE Construction [18]

As already mentioned, the building blocks for Maurer et al.'s MDRS-PKE construction are an MDVS scheme Π_{MDVS}, a Public Key Encryption for Broadcast scheme Π_{PKEBC} and a Digital Signature Scheme Π_{DSS} [18,19]. The informal theorem below gives an overview of our results regarding the additional guarantees given by Maurer et al.'s MDRS-PKE construction $\Pi_{\text{MDRS-PKE}}$ [18,19]:

Theorem 2 (Informal). *If 1.* Π_{PKEBC} *is tightly correct, robust, consistent and* $\{\mathsf{IND},\mathsf{IK}\}$-$\mathsf{CCA}$ *secure under adaptive corruptions; 2.* Π_{MDVS} *is tightly consistent, unforgeable, message-bound validity and forgery invalidity secure (all under adaptive corruptions) and is public-key collision resistant; and 3.* Π_{DSS} *is tightly* 1-$\mathsf{sEUF\text{-}CMA}$ *secure then* $\Pi_{\text{MDRS-PKE}}$ *is tightly:*

1. *consistent under adaptive corruptions ([18, Theorem 7], Theorem 7);*
2. *replay unforgeable under adaptive corruptions (Theorem 8);*
3. $\{\mathsf{IND},\mathsf{IK}\}$-$\mathsf{CCA}_S$ *secure under adaptive corruptions ([4, Theorem 13], Theorem 9*[9]*);*
4. *forgery invalidity secure (Theorem 10); and*
5. *public-key collision resistant (Corollary 2).*

It follows from Theorems 1 and 2 that a remark analogous to Remark 1 also applies for MDRS-PKE [4].

5 Strong Application Semantics for MDVS

In this section we introduce a new security model (i.e. composable notions) for MDVS, and prove that the existing game-based notions (Section 2.3 [4,6]) together with our new notions (Section 3.1) imply these new composable notions. Our new model gives significantly stronger guarantees than [17].

Throughout this section, $\mathcal{S} = \{A_1,\ldots,A_l\}$ is the set of senders, $\mathcal{R} = \{B_1,\ldots,B_n\}$ the set of receivers, and $\mathcal{F} \coloneqq \mathcal{S} \cup \mathcal{R}$; we assume $\mathcal{R}^H$, $\overline{\mathcal{R}^H}$, $\mathcal{S}^H$ and $\overline{\mathcal{S}^H}$ are non-empty. We also consider a judge J(-udy) who is not a sender nor a receiver. The set of parties is $\mathcal{P} = \{A_1,\ldots,A_l,B_1,\ldots,B_n,J\}$.

5.1 The Real World: Assumed Resources and Protocol

A composable security notion involves defining a set of assumed resources (the building blocks), a protocol (which in our case specifies how the MDVS is used) and an ideal resource that captures the application semantics one is trying to model (e.g. the semantics one expects from an MDVS scheme). The set of assumed resources together with the protocol form the so-called real-world resource.

For the case of MDVS we use the same assumed resources and protocol (i.e. converter tuple) as [17]. Apart from the differences arising from considering an asynchronous network setting—which, as explained in Section 2.2.1 is modeled

[9] Our proof is essentially the same as [4, Proof of Theorem 13].

via converter Net (Algorithm 3) that allows controlling message delivery—a simple but crucial difference from [17] is that we duplicate certain dishonest party interfaces. This duplication is key to our composable notions: it allows having dishonest parties run the signature forgery protocol while preserving their access to the assumed resources. (Recall, from Section 2.1, that when a converter α is attached to an interface $I = (I_{\mathcal{X}}, I_{\mathcal{Y}})$ of a resource $\mathbf{R}$, the resulting resource $\alpha^I\mathbf{R}$ no longer includes interface I because it becomes covered by α.) This means, e.g., they still retain access to parties' public keys and dishonest senders' and receivers' secret keys, which is crucial to capture a strong Off-The-Record guarantee.

Algorithm 4 The **KGA** resource for MDVS $\Pi = (S, G_S, G_V, \mathit{Sig}, \mathit{Vfy}, \mathit{Forge})$.

◇ Initialization
  $\mathtt{pp} \leftarrow \Pi.S(1^k)$
  for $A_i \in \mathcal{S}$: $(\mathtt{spk}_i, \mathtt{ssk}_i) \leftarrow \Pi.G_S(\mathtt{pp})$
  for $B_j \in \mathcal{R}$: $(\mathtt{vpk}_j, \mathtt{vsk}_j) \leftarrow \Pi.G_V(\mathtt{pp})$

▷ $(P \in \mathcal{P})$-PublicParameters
  Output($\mathtt{pp}$)

▷ $(A_i \in \mathcal{S}^H)$-SenderKeyPair
  Output($\mathtt{spk}_i, \mathtt{ssk}_i$)

▷ (J)-SenderKeyPair$(A_i \in \mathcal{S}^H)$
  Output($\mathtt{spk}_i, \mathtt{ssk}_i$)

▷ $(P \in \overline{\mathcal{P}^H})$-SenderKeyPair$(A_i \in \overline{\mathcal{S}^H})$
  Output($\mathtt{spk}_i, \mathtt{ssk}_i$)

▷ $(P \in \mathcal{P})$-SenderPublicKey$(A_i \in \mathcal{S})$
  Output($\mathtt{spk}_i$)

▷ $(B_j \in \mathcal{R}^H)$-ReceiverKeyPair
  Output($\mathtt{vpk}_j, \mathtt{vsk}_j$)

▷ $(P \in \overline{\mathcal{P}^H})$-ReceiverKeyPair$(B_j \in \overline{\mathcal{R}^H})$
  Output($\mathtt{vpk}_j, \mathtt{vsk}_j$)

▷ $(P \in \mathcal{P})$-ReceiverPublicKey$(B_j \in \mathcal{R})$
  Output($\mathtt{vpk}_j$)

Assumed Resources. Parties have access to an asynchronous and anonymous insecure repository Net·**INS**—to which everyone can write to and read from—and to a *Key Generation Authority* (**KGA**) resource [17], which generates and stores parties' key pairs (Algorithm 4). The **KGA** guarantees not only that dishonest receivers' key-pairs are "well-formed" but also that the dishonest receivers actually have access to their secret keys—which allows them to come up with forged ciphertexts; being able to come up with forged ciphertexts that look like real ones is necessary for the Off-The-Record guarantee [6, 7, 17]. Since we are considering the setting introduced in [4]—where judge J(-udy) has access to the secret keys of honest senders—the **KGA** gives Judy access to the secret keys of honest senders. The **KGA** resource—which is implicitly parameterized by a security parameter k—runs setup algorithm S to obtain the MDVS's public parameters.Then, it generates key-pairs for all senders and receivers—using G_S and G_V, respectively. Every honest party can query their own key-pair, the public parameters and everyone's public keys at their own interface. Dishonest parties can additionally obtain the key-pairs of any dishonest senders or receivers; finally, the judge J can additionally obtain the secret keys of honest senders.

Protocol. Each honest sender $A_i \in \mathcal{S}^H$ and each honest receiver $B_j \in \mathcal{R}^H$ locally runs a converter Snd and Rcv, respectively (see Algorithm 5); both these

Algorithm 5 Converters Snd, Rcv and Forge for MDVS $\Pi = (S, G_S, G_V, Sig, Vfy, Forge)$.

▷ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V}\rangle, m)$ // Conv. Snd
  $\mathtt{pp} \leftarrow$PUBLICPARAMETERS
  $(\mathtt{spk}_i, \mathtt{ssk}_i) \leftarrow$SENDERKEYPAIR
  for $l \in [|\vec{V}|]$ **:**
    $\mathtt{vpk}_l \leftarrow$RECEIVERPUBLICKEY$(V_l)$
  $\vec{v} \coloneqq (\mathtt{vpk}_1, \ldots, \mathtt{vpk}_{|\vec{V}|})$
  $\sigma \leftarrow \Pi.Sig_{\mathtt{pp}}(\mathtt{ssk}_i, \vec{v}, m)$
  OUTPUT(WRITE$(m, \sigma, (A_i, \vec{V}))$)

▷ $(B_j \in \mathcal{R}^H)$-READ // Conv. Rcv
  $(\mathtt{vpk}_j, \mathtt{vsk}_j) \leftarrow$RECEIVERKEYPAIR
  $\mathtt{pp} \leftarrow$PUBLICPARAMETERS
  list, readSet $\leftarrow \emptyset$
  for $(\mathtt{id}, tup \coloneqq (m, \sigma, (A_i, \vec{V}))) \in$ READ **with** $(tup \notin \text{readSet}) \wedge (B_j \in \text{Set}(\vec{V}))$ **:**
    readSet $\leftarrow$ readSet $\cup \{tup\}$
    $\mathtt{spk}_i \leftarrow$SENDERPUBLICKEY$(A_i)$
    for $l \in [|\vec{V}|]$ **:**
      $\mathtt{vpk}_l \leftarrow$RECEIVERPUBLICKEY$(V_l)$
    $\vec{v} \coloneqq (\mathtt{vpk}_1, \ldots, \mathtt{vpk}_{|\vec{V}|})$
    if $\Pi.Vfy_{\mathtt{pp}}(\mathtt{spk}_i, \mathtt{vsk}_j, \vec{v}, m, \sigma)$ **:**
      list $\leftarrow$ list $\cup \{(\langle A_i \to \vec{V}\rangle, \mathtt{id}, m)\}$
  OUTPUT(list)

▷ $(P \in \mathcal{F})$-WRITE$(\langle [\text{Forge}] A_i \to \vec{V}\rangle, m)$
  // Conv. Forge
  $\mathtt{pp} \leftarrow$PUBLICPARAMETERS
  $\mathtt{spk}_i \leftarrow$SENDERPUBLICKEY$(A_i)$
  for $l \in [|\vec{V}|]$ **with** $V_l \in \mathcal{R}^H$ **:**
    $\mathtt{vpk}_l \leftarrow$ RECEIVERPUBLICKEY(V_l)
    $\mathtt{vsk}_l \leftarrow \perp$
  for $l \in [|\vec{V}|]$ **with** $V_l \in \overline{\mathcal{R}^H}$ **:**
    $(\mathtt{vpk}_l, \mathtt{vsk}_l) \leftarrow$RECEIVERKEYPAIR$(V_l)$
  $\vec{v} \coloneqq (\mathtt{vpk}_1, \ldots, \mathtt{vpk}_{|\vec{V}|})$
  $\vec{s} \coloneqq (\mathtt{vsk}_1, \ldots, \mathtt{vsk}_{|\vec{V}|})$
  $\sigma \leftarrow \Pi.Forge_{\mathtt{pp}}(\mathtt{spk}_i, \vec{v}, m, \vec{s})$
  OUTPUT(WRITE$(m, \sigma, (A_i, \vec{V}))$)

converters connect to the **KGA** and Net · **INS**; they provide the same outer interfaces as a repository for a party who is a writer (Snd) and a reader (Rcv), respectively. A party A_i's Snd converter provides a procedure WRITE which takes as input a label $\langle A_i \to \vec{V}\rangle$ (defining the vector of receivers $\vec{V} = (V_1, \ldots, V_{|\vec{V}|})$) and a message m; upon such input, Snd signs the input message, writes tuple $(m, \sigma, (A_i, \vec{V}))$ to Net · **INS**—where σ is the resulting signature—and outputs the id output by writing to Net · **INS**. A party B_j's Rcv converter reads all (received) tuples from Net · **INS**—filtering out duplicates—and verifies the ones for which B_j is part of the receivers; if verification succeeds, it adds a triple with the sender/receiver-vector label, id and message to the output set.

In addition to converters Snd and Rcv, every party in $\mathcal{F} \coloneqq \mathcal{S} \cup \mathcal{R}$—*including dishonest ones*—runs a converter Forge (Algorithm 5) that allows to forge messages, mimicking (honest) senders' WRITE operations towards dishonest parties. Each party's Forge converter connects to the **KGA** and **INS** resources—but, *crucially*, is not given access to senders' secret keys. Having dishonest parties run converter Forge is what captures their ability of forging real-looking ciphertexts (i.e. which gives the sender *plausible deniability*). However, simply attaching a converter to dishonest parties' interfaces eliminates their access to the **KGA** and **INS** resources (Section 2.1)—i.e. dishonest parties cannot access parties' public keys, dishonest parties' secret keys, nor READ and WRITE from/to the **INS** repository—resulting in a rather weak Off-The-Record guarantee (illustrated

in Figure 2).[10] To ensure that running Forge does not restrict dishonest parties' capabilities, we duplicate some of the **KGA** and **INS** interfaces and have converter Forge connect only to these redundant interfaces. Concretely, we extend the **KGA** resource with additional interfaces that allows a party's Forge converter to obtain the public and secret keys necessary to forge signatures (Algorithm 6), and extend the **INS** repository to provide these converters with write access, so they can write forged signatures to **INS**. This is achieved by defining **INS** as $\mathbf{INS}_{\mathcal{P}}^{\mathcal{P}\cup(\mathcal{F}\times\{\mathsf{Forge}\})}$.

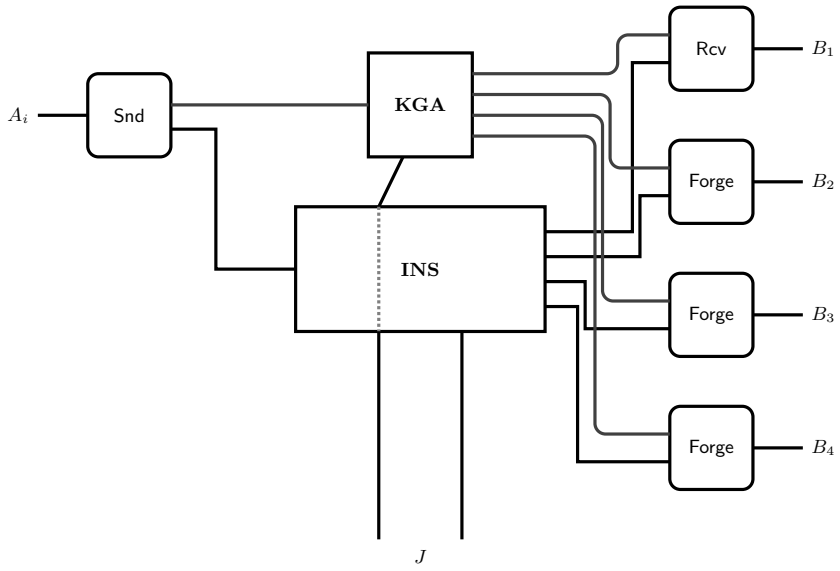


Fig. 2: A real world resource that captures weak deniability guarantees. The only sender depicted is A_i, who is honest. From the four receivers depicted—B_1 through B_4—only B_1 is honest. Judge J is dishonest.

Algorithm 6 Additional **KGA** interfaces for the Forge converters.

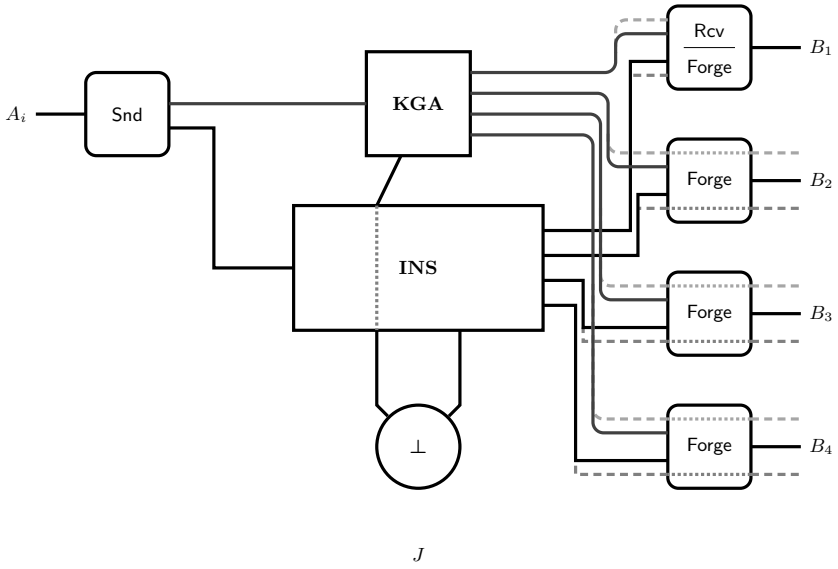
▷ ($P \in \mathcal{F}$, Forge)-PublicParameters
Output($\mathtt{pp}, \mathtt{rpk}_{\mathtt{pp}}$)

▷ ($P \in \mathcal{F}$, Forge)-SenderPublicKey($A_i \in \mathcal{S}$)
Output($\mathtt{spk}_i$)

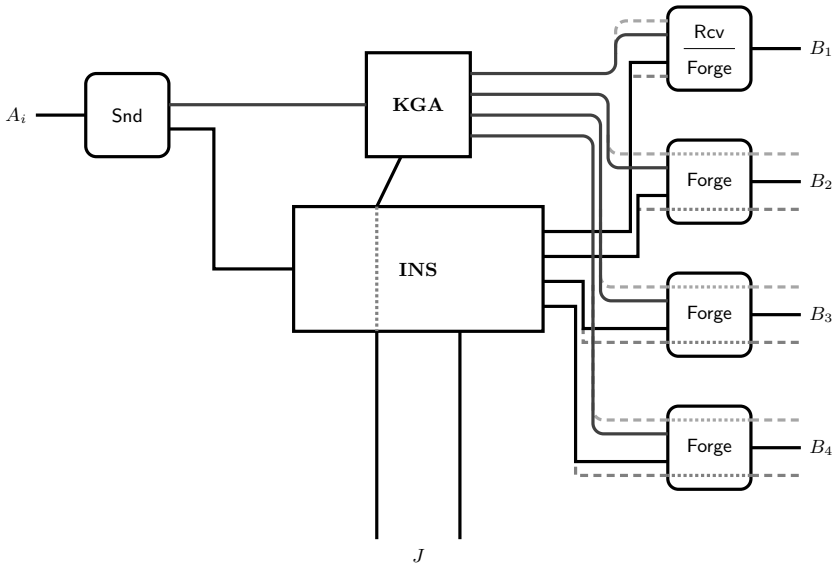
▷ ($P \in \mathcal{F}$, Forge)-ReceiverPublicKey($B_j \in \mathcal{R}^H$)
Output($\mathtt{vpk}_j$)

▷ ($P \in \mathcal{F}$, Forge)-ReceiverKeyPair($B_j \in \overline{\mathcal{R}^H}$)
Output($\mathtt{vpk}_j, \mathtt{vsk}_j$)

[10] In fact, for such notion one would need to redefine the set $\mathcal{F}$ of parties running the Forge converter to include only dishonest ones.



(a) Real world resource when judge J is honest.



(b) Real world resource for dishonest judge J.

Fig. 3: In both (sub-)figures: A_i is the only sender depicted, and is honest; four receivers are depicted—B_1 through B_4—among whom only B_1 is honest; B_1 runs two converters—Rcv and Forge—which are agglomerated into the same box.

Real World System. One of the guarantees one expects from an MDVS (and from an MDRS-PKE) is authenticity. However, since judge J has access to the secret keys of honest senders it is not possible to guarantee authenticity if J is dishonest. We then consider two cases: honest J and dishonest J. The real world resource for the first case (illustrated in Figure 3b) is given by

$$\mathsf{Snd}^{\mathcal{S}^H}\,\mathsf{Rcv}^{\mathcal{R}^H}\,\mathsf{Forge}^{(\mathcal{F}\times\{\mathsf{Forge}\})}\,[\mathbf{KGA}, \mathsf{Net}\cdot\mathbf{INS}]. \tag{5.1}$$

For the case of honest J, it is instead given by

$$\mathsf{Snd}^{\mathcal{S}^H}\,\mathsf{Rcv}^{\mathcal{R}^H}\,\mathsf{Forge}^{(\mathcal{F}\times\{\mathsf{Forge}\})}\,\bot^{J}\,[\mathbf{KGA}, \mathsf{Net}\cdot\mathbf{INS}]. \tag{5.2}$$

where $\bot$ is a (dummy) converter that covers all of J's interfaces and provides no outside interface (see Figure 3b).

5.2 Application Semantics

The guarantees one expects from an MDVS scheme depend on the honesty of the judge J(-udy): if she is dishonest, we expect consistency and plausible deniability (i.e. Off-The-Record); if she is honest, we additionally expect authenticity. The ideal resources we now define give stronger guarantees than [17].

5.2.1 Dishonest Judy For each sender $A_i \in \mathcal{S}$ and receiver-vector $\vec{V} \in \mathcal{R}^+$, the ideal system includes a repository $\langle A_i \to \vec{V}\rangle$ to which A_i and any dishonest party can write to, and from which dishonest parties and the ones in $\vec{V}$ can read, i.e.

$$\langle A_i \to \vec{V}\rangle \coloneqq \langle A_i \to \vec{V}\rangle^{\{A_i\}\cup\overline{\mathcal{P}^H}}_{\mathrm{Set}(\vec{V})\cup\overline{\mathcal{P}^H}}.$$

These repositories naturally capture consistency: for each atomic repository $\langle A_i \to \vec{V}\rangle$, either there is a register with identifier `id`—in which case *all honest receivers* $B_j \in \vec{V}$ to whom `id` was delivered get the *same unique tuple* $(\mathtt{id}, (\langle A_i \to \vec{V}\rangle, m))$ as part of the output of a Read operation—or there is not—in which case *no honest receiver* $B_j \in \vec{V}$ gets a *tuple with a matching identifier* `id` (as part of the output of a Read operation).

To capture Off-The-Record, for each sender A_i and vector of designated receivers $\vec{V}$ we consider an additional repository to which "fake" messages are written: $\langle[\mathrm{Forge}]A_i \to \vec{V}\rangle$. The Write interface of this new repository matches the joint Write interfaces of the Forge converters, i.e. $\mathcal{F}$; on the other hand, while dishonest parties should not be able to tell apart "real" messages from "fake" ones, honest parties should, which means repository $\langle[\mathrm{Forge}]A_i \to \vec{V}\rangle$'s set of readers is $\overline{\mathcal{P}^H}$; i.e. $\langle[\mathrm{Forge}]A_i \to \vec{V}\rangle \coloneqq \langle[\mathrm{Forge}]A_i \to \vec{V}\rangle^{\mathcal{F}}_{\overline{\mathcal{P}^H}}$. The ideal system includes a repository with all these (atomic) repositories put together with the Net converter attached (see Equation 5.3).

$$\left[\begin{array}{c} \mathsf{Net}\ \cdot\ \left[\langle A_i \to \vec{V}\rangle^{\{A_i\}\cup\overline{\mathcal{P}^H}}_{\mathrm{Set}(\vec{V})\cup\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+} \\ \left[\langle[\mathrm{Forge}]A_i \to \vec{V}\rangle^{\mathcal{F}}_{\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+} \end{array}\right]. \tag{5.3}$$

(Converter Net need not be attached to $\langle[\text{Forge}]A_i \to \vec{V}\rangle$: Net only controls message delivery to honest parties, and all readers of $\langle[\text{Forge}]A_i \to \vec{V}\rangle$ are dishonest.)

Algorithm 7 Converter Otr.

```
▷ (P ∈ overline{P^H})-READ
  list ← ∅
  for (id, (rep_i, m)) ∈ READ :
    if rep_i = ⟨[Forge]A_i → V⃗⟩ :
      list ← list ∪ {(id, (⟨A_i → V⃗⟩, m))}
    else // rep_i = ⟨A_i → V⃗⟩.
      list ← list ∪ {(id, (rep_i, m))}
  OUTPUT(list)
```

The repositories in Equation 5.3 do not capture Off-The-Record, and we do not know how to model these only using the repository resources from before: READ operations leak the atomic repository from which each of the tuples output was read from, and so a party can distinguish real messages—which would be paired with repositories labels of the form $\langle A_i \to \vec{V}\rangle$—from forged ones—which would be paired with labels of the form $\langle[\text{Forge}]A_i \to \vec{V}\rangle$. We model Off-The-Record via a converter Otr that is attached to dishonest parties' READ interfaces and limits their (reading) capabilities. Otr—formally defined in Algorithm 7—captures Off-The-Record because it ensures (dishonest) parties do not know whether they are reading real messages—written to $\langle A_i \to \vec{V}\rangle$—or forged ones—written to $\langle[\text{Forge}]A_i \to \vec{V}\rangle$. The resulting ideal resource $\mathbf{S}$ is then

$$\mathbf{S} \coloneqq \mathsf{Otr}^{\overline{\mathcal{P}^H}} \cdot \left[\begin{array}{c} \mathsf{Net} \;\cdot\; \left[\langle A_i \to \vec{V}\rangle^{\{A_i\}\cup\overline{\mathcal{P}^H}}_{\text{Set}(\vec{V})\cup\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+} \\ \left[\langle[\text{Forge}]A_i \to \vec{V}\rangle^{\mathcal{F}}_{\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+} \end{array}\right]. \tag{5.4}$$

Figure 4a illustrates a simpler variant of $\mathbf{S}$—corresponding to the case where J is honest (so honest receivers cannot be impersonated).

5.2.2 Honest Judy When Judy is honest we also expect authenticity; a natural way to capture this is having the ideal resource include, for each honest sender $A_i \in \mathcal{S}^H$, an atomic repository $\langle A_i \to \vec{V}\rangle$ to which only A_i can write:

$$\left[\langle A_i \to \vec{V}\rangle^{\{A_i\}}_{\text{Set}(\vec{V})\cup\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S}^H,\vec{V}\in\mathcal{R}^+}.$$

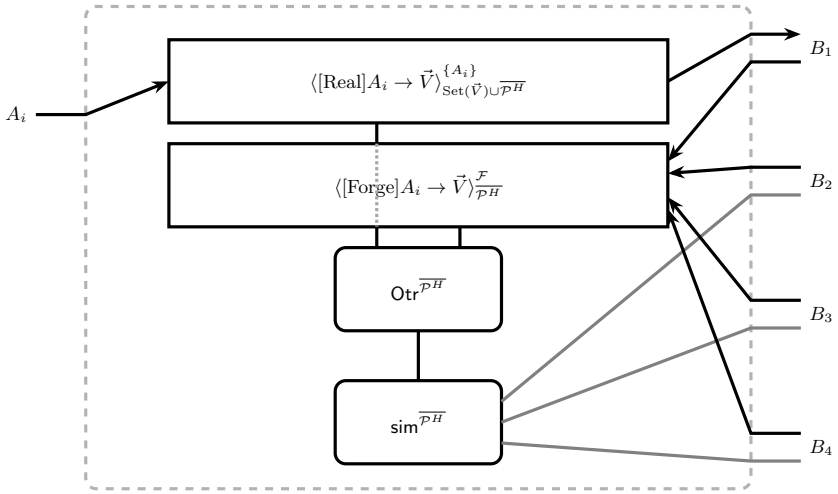
This is the same we did for modeling Off-The-Record: we are restricting dishonest parties' writing capabilities by eliminating their WRITE sub-interfaces on input labels $\langle A_i \to \vec{V}\rangle$ such that $A_i \in \mathcal{S}^H$. Denoting these (sub-)interfaces by $\mathsf{Auth\text{-}Intf} \coloneqq \overline{\mathcal{P}^H}\text{-WRITE}(\langle \mathcal{S}^H \to \mathcal{R}^+\rangle, \cdot)$, we can equivalently connect the

(dummy) converter $\perp$ so it disables these WRITE interfaces, i.e.

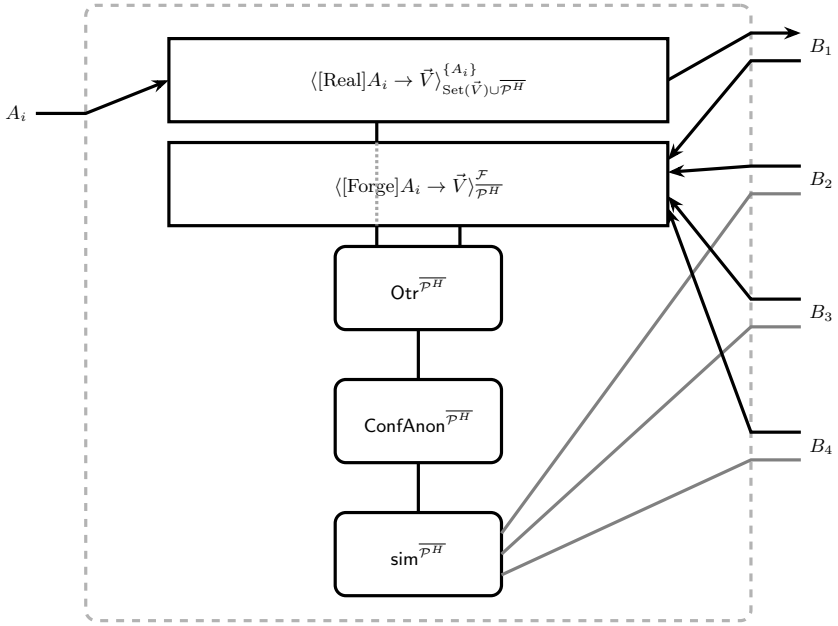
$$\begin{aligned} \perp^{\mathsf{Auth\text{-}Intf}} \cdot \left[\langle A_i \to \vec{V}\rangle^{\{A_i\}\cup\overline{\mathcal{P}^H}}_{\mathrm{Set}(\vec{V})\cup\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S}^H,\vec{V}\in\mathcal{R}^+} \\ \equiv \left[\langle A_i \to \vec{V}\rangle^{\{A_i\}}_{\mathrm{Set}(\vec{V})\cup\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S}^H,\vec{V}\in\mathcal{R}^+}. \end{aligned} \tag{5.5}$$

We can then define the ideal resource $\mathbf{T}$ as:

$$\mathbf{T} := \mathsf{Otr}^{\overline{\mathcal{P}^H}} \cdot \left[\begin{array}{c} \mathsf{Net}\cdot \perp^{\mathsf{Auth\text{-}Intf}} \cdot \left[\langle A_i \to \vec{V}\rangle^{\{A_i\}\cup\overline{\mathcal{P}^H}}_{\mathrm{Set}(\vec{V})\cup\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+} \\ \left[\langle [\mathrm{Forge}]A_i \to \vec{V}\rangle^{\mathcal{F}}_{\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+} \end{array}\right]. \tag{5.6}$$



(a) Ideal MDVS resource.



(b) Ideal MDRS-PKE resource.

Fig. 4: Only honest sender A_i is depicted. From the four depicted receivers—B_1 through B_4—only B_1 is honest. Judge J is honest. In the figure we use arrows to make clear that dishonest parties can only write to repositories $\langle[\text{Forge}]A_i \to \vec{V}\rangle$, but not read.

5.3 Application Semantics of Game-Based Notions

The informal theorem below establishes the composable semantics of MDVS's game-based notions (Section 2.3 [4, 6], Section 3.1). For the formal theorem statements and respective full proofs, see Section E.

Theorem 3 (Informal). *Suppose an* MDVS *scheme Π is correct, consistent, replay-unforgeable, Off-The-Record and satisfies forgery invalidity. If Π is used as the* MDVS *scheme underlying the real world systems defined in Section 5.1 then there are poly-time simulators $\mathsf{sim}_{\mathbf{S}}$ and $\mathsf{sim}_{\mathbf{T}}$ and there are negligible functions $\varepsilon_{\mathbf{S}}$ and $\varepsilon_{\mathbf{T}}$ such that, for any (suitable) poly-time distinguishers $\mathbf{D}_{\mathbf{S}}, \mathbf{D}_{\mathbf{T}}$, the two statements below hold:*
$\Delta^{\mathbf{D_T}}\Big(\mathsf{Snd}^{\mathcal{S}^H}\,\mathsf{Rcv}^{\mathcal{R}^H}\,\mathsf{Forge}^{\mathcal{F}}\,\bot^{J}\,[\mathbf{KGA}, \mathsf{Net}\cdot\mathbf{INS}],\ \mathsf{sim}_{\mathbf{T}}\cdot\mathbf{T}\Big) \leq \varepsilon_{\mathbf{T}}$ *(Theorem 13);*
$\Delta^{\mathbf{D_S}}\Big(\mathsf{Snd}^{\mathcal{S}^H}\,\mathsf{Rcv}^{\mathcal{R}^H}\,\mathsf{Forge}^{\mathcal{F}}\,[\mathbf{KGA}, \mathsf{Net}\cdot\mathbf{INS}],\ \mathsf{sim}_{\mathbf{S}}\cdot\mathbf{S}\Big) \leq \varepsilon_{\mathbf{S}}$ *(Theorem 14).*

Remark 2. The proof of Theorem 13—which corresponds to the weaker setting where honest senders' secret keys do not leak—also assumes the underlying MDVS to satisfy Forgery Invalidity.

6 Application Semantics for MDRS-PKE Schemes

In this section we introduce the first composable security notions for MDRS-PKE schemes; as we will see, these notions are surprisingly similar to the ones for MDVS. Finally, we prove that the existing game-based notions (Section 2.4 [4,6]) together with our new notions (Section 4.1) do imply these new composable notions.

As before, $\mathcal{S}$ are the senders, $\mathcal{R}$ the receivers, and $\mathcal{F} \coloneqq \mathcal{S} \cup \mathcal{R}$; we assume $\mathcal{R}^H$, $\overline{\mathcal{R}^H}$, $\mathcal{S}^H$ and $\overline{\mathcal{S}^H}$ are non-empty. Together with the judge J(-udy), the set of parties is then is $\mathcal{P} = \mathcal{S} \cup \mathcal{R} \cup \{J\}$.

6.1 The Real World: Assumed Resources and Protocols

Assumed Resources. The assumed resources for our MDRS-PKE composable notions are essentially the same as the ones for our MDVS notions: an asynchronous and anonymous insecure repository $\mathsf{Net}\cdot\mathbf{INS}$ and a **KGA** (Algorithm 8). The **KGA** resource is implicitly parameterized by a security parameter k; it first runs setup algorithm S and then samples an MDRS-PKE receiver public key $\mathtt{rpk}_{\mathtt{pp}}$ which it attaches to the public parameters. (The additional public key allows for simpler reductions, as one can rely on the corresponding secret key for decryption, and eliminates the need for the MDRS-PKE scheme to satisfy a robustness type of notion.) It then generates key-pairs for all senders and receivers—using G_S and G_R, respectively. For simplicity, the MDRS-PKE **KGA** resource supports an additional helper operation, GetLabel, which given an sender's public key and a vector of receiver public keys outputs the unique corresponding label $\langle A_i \to \vec{V}\rangle$, or $\bot$ if the label either does not exist or is not unique.

Algorithm 8 The **KGA** resource for MDRS-PKE $\Pi = (S, G_S, G_R, E, D, Forge)$. Below we only show the differences relative to Algorithm 4, i.e. the **KGA** resource defined for the MDVS composable notions.

◇ Initialization
  $\mathtt{pp} \leftarrow \Pi.S(1^k)$
  $(\mathtt{rpk}_{\mathtt{pp}}, \cdot) \leftarrow \Pi.G_R(\mathtt{pp})$
  for $A_i \in \mathcal{S}$: $(\mathtt{spk}_i, \mathtt{ssk}_i) \leftarrow \Pi.G_S(\mathtt{pp})$
  for $B_j \in \mathcal{R}$: $(\mathtt{rpk}_j, \mathtt{rsk}_j) \leftarrow \Pi.G_R(\mathtt{pp})$

▷ $(P \in \mathcal{P})$-PublicParameters
  Output$(\mathtt{pp}, \mathtt{rpk}_{\mathtt{pp}})$

▷ $(A_i \in \mathcal{S}^H)$-SenderKeyPair
▷ (J)-SenderKeyPair$(A_i \in \mathcal{S}^H)$
▷ $(P \in \overline{\mathcal{P}^H})$-SenderKeyPair$(A_i \in \overline{\mathcal{S}^H})$
▷ $(P \in \mathcal{P})$-SenderPublicKey$(A_i \in \mathcal{S})$
▷ $(B_j \in \mathcal{R}^H)$-ReceiverKeyPair
▷ $(P \in \overline{\mathcal{P}^H})$-ReceiverKeyPair$(B_j \in \overline{\mathcal{R}^H})$
▷ $(P \in \mathcal{P})$-ReceiverPublicKey$(B_j \in \mathcal{R})$

▷ $(B_j \in \mathcal{R}^H)$-GetLabel$(\mathtt{spk}, \vec{v}')$
  $\mathcal{S}_{\mathtt{spk}} \coloneqq \{A_i \mid \mathtt{spk} = \mathtt{spk}_i\}$
  if $|\mathcal{S}_{\mathtt{spk}}| \neq 1 \vee {v_1}' \neq \mathtt{rpk}_{\mathtt{pp}}$ **:**
    Output$(\perp)$
  for $l \in \{2, \ldots, |\vec{v}'|\}$ **:**
    $\mathcal{R}_{{v_l}'} \coloneqq \{B_k \mid {v_l}' = \mathtt{rpk}_k\}$
    if $|\mathcal{R}_{{v_l}'}| \neq 1$ **:**
      Output$(\perp)$
    else
      Let B_k be the element of $\mathcal{R}_{{v_l}'}$
      $V_{l-1} = B_k$
  Let A_i be the element of $\mathcal{S}_{\mathtt{spk}}$
  Let $\vec{V} \coloneqq (V_1, \ldots, V_{|\vec{v}'|-1})$
  Output$(\langle A_i \rightarrow \vec{V} \rangle)$

Protocol. As for MDVS, honest senders receivers locally run converter Snd and Rcv, respectively (see Algorithm 9), which are attached to **KGA** and Net · **INS**. These converters are mostly analogous to their MDVS counterparts, the main differences being:

1. Converter Snd no longer writes the sender's and vector of receivers' identities nor the input message to **INS** (as required to guarantee anonymity and confidentiality);
2. The first recipient of (valid) ciphertexts is the public parameters public key;
3. For each ciphertext that decrypts correctly, Rcv uses the **KGA**'s GetLabel operation to obtain a label—i.e. looks up the sender/vector of receivers with public keys matching the ones obtained from decryption—and then outputs a set of triples, each triple corresponding to a ciphertext that decrypted correctly and for which the GetLabel operation returned a valid label (i.e. not $\perp$);
4. The Forge converter—which connects to extended **KGA** and Net · **INS** similarly to the analogous MDVS converter—on inputs $(\langle [\text{Forge}]A_i \rightarrow \vec{V} \rangle, m)$ such that $\vec{V} \in \mathcal{R}^{H^+}$ forges messages differently (see Algorithm 9 for details).

The real world resource is defined similarly to the MDVS real world resource—the only difference being the different (but analogous) **KGA** and converters.

6.2 MDRS-PKE Application Semantics

Apart from the additional anonymity and confidentiality guarantees—for messages sent by honest senders to vectors of all-honest receivers—the properties we expect from an MDRS-PKE are the same we expect from an MDVS.

Algorithm 9 Converters Snd, Rcv and Forge.

▷ $(A_i \in \mathcal{S}^H)$-Write$(\langle A_i \to \vec{V}\rangle, m)$
// Conv. Snd
$(\mathtt{pp}, \mathtt{rpk}_{\mathtt{pp}}) \leftarrow$PublicParameters
$(\mathtt{spk}_i, \mathtt{ssk}_i) \leftarrow$SenderKeyPair
for $l \in \{1, \ldots, |\vec{V}|\}$ **:**
$\mathtt{rpk}_l \leftarrow$ReceiverPublicKey$(V_l)$
$\vec{v}' \coloneqq (\mathtt{rpk}_{\mathtt{pp}}, \mathtt{rpk}_1, \ldots, \mathtt{rpk}_{|\vec{V}|})$
$c \leftarrow \Pi.E_{\mathtt{pp}}(\mathtt{ssk}_i, \vec{v}', m)$
Output(Write(c))

▷ $(B_j \in \mathcal{R}^H)$-Read // Conv. Rcv
$(\mathtt{rpk}_j, \mathtt{rsk}_j) \leftarrow$ReceiverKeyPair
$(\mathtt{pp}, \cdot) \leftarrow$PublicParameters
list, ctxtSet $\leftarrow \emptyset$
for $(\mathtt{id}, c) \in$ Read **with** $c \notin$ ctxtSet **:**
ctxtSet $\leftarrow$ ctxtSet $\cup \{c\}$
$(\mathtt{spk}_i, \vec{v}', m) \leftarrow \Pi.D_{\mathtt{pp}}(\mathtt{rsk}_j, c)$
if $(\mathtt{spk}_i, \vec{v}', m) \neq \perp$ **:**
$\langle A_i \to \vec{V}\rangle \leftarrow$GetLabel$(\mathtt{spk}_i, \vec{v}')$
if $\langle A_i \to \vec{V}\rangle \neq \perp$ **:**
list $\leftarrow$ list $\cup \{(\mathtt{id}, (\langle A_i \to \vec{V}\rangle, m))\}$
Output(list)

▷ $(P \in \mathcal{F})$-Write$(\langle [\text{Forge}]A_i \to \vec{V}\rangle, m)$ // Conv. Forge
$(\mathtt{pp}, \mathtt{rpk}_{\mathtt{pp}}) \leftarrow$PublicParameters
if $\vec{V} \in \mathcal{R}^{H+}$ **:**
$\mathtt{spk}_1 \leftarrow$SenderPublicKey$(A_1)$
$c \leftarrow \Pi.Forge_{\mathtt{pp}}(\mathtt{spk}_1, \mathtt{rpk}_{\mathtt{pp}}{}^{|\vec{V}|+1}, 0^{|m|}, \perp^{|\vec{V}|+1})$
else
$\mathtt{spk}_i \leftarrow$SenderPublicKey$(A_i)$
for $l \in \{1, \ldots, |\vec{V}|\}$ **:**
if $V_l \in \mathcal{R}^H$ **:**
$(\mathtt{rpk}_l, \mathtt{rsk}_l) \leftarrow$ (ReceiverPublicKey$(V_l), \perp)$
else // $V_l \in \overline{\mathcal{R}^H}$
$(\mathtt{rpk}_l, \mathtt{rsk}_l) \leftarrow$ReceiverKeyPair$(V_l)$
$(\vec{v}', \vec{s}) \coloneqq \big((\mathtt{rpk}_{\mathtt{pp}}, \mathtt{rpk}_1, \ldots, \mathtt{rpk}_{|\vec{V}|}), \quad (\perp, \mathtt{rsk}_1, \ldots, \mathtt{rsk}_{|\vec{V}|})\big)$
Output(Write$(c \leftarrow \Pi.Forge_{\mathtt{pp}}(\mathtt{spk}_i, \vec{v}', m, \vec{s}))$)

6.2.1 Dishonest Judy The ideal MDVS resource **S** defined in Equation 5.4 captures all the guarantees we expect from an MDRS-PKE, except for anonymity and confidentiality.

Algorithm 10 Converter ConfAnon.

▷ $(P \in \overline{\mathcal{P}^H})$-Read
list $\leftarrow \emptyset$
for $(\mathtt{id}, (\langle A_i \to \vec{V}\rangle, m)) \in$ Read **:**
if $(A_i, \vec{V}) \in \mathcal{S}^H \times (\mathcal{R}^H)^+$ **:**
list $\leftarrow$ list $\cup \{(\mathtt{id}, (|\vec{V}|, |m|))\}$
else
list $\leftarrow$ list $\cup \{(\mathtt{id}, (\langle A_i \to \vec{V}\rangle, m))\}$
Output(list)

As for Off-The-Record, it is not clear how to capture these additional guarantees from the repository resources alone: Read operations output the atomic repository $\langle A_i \to \vec{V}\rangle$—which identifies the sender and vector of receivers—associated with each tuple that is output; regarding confidentiality, either a party has read access to an atomic repository—in which case Read operations output all of the repository's messages in plain—or the party has no read access to the atomic

repository—in which case the party does not learn anything about the messages written in that repository: not their length nor how many there are. Fortunately, we can follow the same idea we used to model Off-The-Record: by defining an appropriate converter—ConfAnon, Algorithm 10—that is attached to dishonest parties' READ interfaces and limits their (reading) capabilities: if an honest sender $A_i \in \mathcal{S}^H$ sends a message m to a vector of receivers $\vec{V}$ all of whom are honest (i.e. $\vec{V} \in \mathcal{R}^{H^+}$), then ConfAnon only leaks $|\vec{V}|$ and $|m|$ to dishonest parties, instead of $\langle A_i \to \vec{V} \rangle$ and m. The resulting ideal resource is then

$$\mathbf{S} \coloneqq \left(\mathsf{ConfAnon}^{\overline{\mathcal{P}^H}} \cdot \ \mathsf{Otr}^{\overline{\mathcal{P}^H}}\right) \cdot \left[\begin{array}{c} \mathsf{Net} \ \cdot \ \left[\langle A_i \to \vec{V} \rangle^{\{A_i\} \cup \overline{\mathcal{P}^H}}_{\mathrm{Set}(\vec{V}) \cup \overline{\mathcal{P}^H}}\right]_{A_i \in \mathcal{S}, \vec{V} \in \mathcal{R}^+} \\ \left[\langle [\mathrm{Forge}] A_i \to \vec{V} \rangle^{\mathcal{F}}_{\overline{\mathcal{P}^H}}\right]_{A_i \in \mathcal{S}, \vec{V} \in \mathcal{R}^+} \end{array}\right]. \quad (6.1)$$

6.2.2 Honest Judy This case is analogous; the ideal resource $\mathbf{T}$ is defined as:

$$\mathbf{T} \coloneqq \left(\begin{array}{c} \mathsf{ConfAnon}^{\overline{\mathcal{P}^H}} \\ \cdot \, \mathsf{Otr}^{\overline{\mathcal{P}^H}} \end{array}\right) \cdot \left[\begin{array}{c} \mathsf{Net} \cdot \ \perp^{\mathsf{Auth\text{-}Intf}} \cdot \left[\langle A_i \to \vec{V} \rangle^{\{A_i\} \cup \overline{\mathcal{P}^H}}_{\mathrm{Set}(\vec{V}) \cup \overline{\mathcal{P}^H}}\right]_{A_i \in \mathcal{S}, \vec{V} \in \mathcal{R}^+} \\ \left[\langle [\mathrm{Forge}] A_i \to \vec{V} \rangle^{\mathcal{F}}_{\overline{\mathcal{P}^H}}\right]_{A_i \in \mathcal{S}, \vec{V} \in \mathcal{R}^+} \end{array}\right]. \quad (6.2)$$

6.3 Application Semantics of Game-Based Notions

The informal theorem below summarizes our claims regarding the application semantics of the MDRS-PKE security notions we introduced in this section. For the formal theorem statements and respective full proofs, see Section D.

Theorem 4 (Informal). *Suppose an* MDRS-PKE *scheme Π is correct, consistent, replay-unforgeable,* $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}$ *secure, Off-The-Record, satisfies forgery invalidity and is public-key collision resistant. If Π is used as the* MDRS-PKE *scheme underlying the real world systems defined in Section 6.1 then there are poly-time simulators $\mathsf{sim}_{\mathbf{S}}$ and $\mathsf{sim}_{\mathbf{T}}$ and there are negligible functions $\varepsilon_{\mathbf{S}}$ and $\varepsilon_{\mathbf{T}}$ such that, for any (suitable) poly-time distinguishers $\mathbf{D}_{\mathbf{S}}, \mathbf{D}_{\mathbf{T}}$, the two statements below hold:*
$\Delta^{\mathbf{D_S}}\Big(\mathsf{Snd}^{\mathcal{S}^H}\,\mathsf{Rcv}^{\mathcal{R}^H}\,\mathsf{Forge}^{\mathcal{F}}\,[\mathbf{KGA}, \mathsf{Net} \cdot \mathbf{INS}],\ \mathsf{sim}_{\mathbf{S}} \cdot \mathbf{S}\Big) \leq \varepsilon_{\mathbf{S}}$ *(Theorem 12);*
$\Delta^{\mathbf{D_T}}\Big(\mathsf{Snd}^{\mathcal{S}^H}\,\mathsf{Rcv}^{\mathcal{R}^H}\,\mathsf{Forge}^{\mathcal{F}}\perp^{J}\,[\mathbf{KGA}, \mathsf{Net} \cdot \mathbf{INS}],\ \mathsf{sim}_{\mathbf{T}} \cdot \mathbf{T}\Big) \leq \varepsilon_{\mathbf{T}}$ *(Theorem 11).*

Remark 3. As already mentioned, for the setting where honest sender secret keys do not leak to the judge one needs the MDRS-PKE to satisfy $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}_S$ (Definition 17). While this notion is implied by the $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}$ notion (Definition 9) [4], it is not (known to be) implied by the $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}$ notion introduced in [18].

A remark analogous to Remark 2 also applies for the case of MDRS-PKE.

References

1. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 409–426. Springer, Berlin, Heidelberg (May / Jun 2006). https://doi.org/10.1007/11761679_25
2. Borisov, N., Goldberg, I., Brewer, E.A.: Off-the-record communication, or, why not to use PGP. In: Atluri, V., Syverson, P.F., di Vimercati, S.D.C. (eds.) WPES 2004. pp. 77–84. ACM (2004), https://doi.org/10.1145/1029179.1029200
3. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: 42nd FOCS. pp. 136–145. IEEE Computer Society Press (Oct 2001). https://doi.org/10.1109/SFCS.2001.959888
4. Chakraborty, S., Hofheinz, D., Maurer, U., Rito, G.: Deniable authentication when signing keys leak. In: Hazay, C., Stam, M. (eds.) EUROCRYPT 2023, Part III. LNCS, vol. 14006, pp. 69–100. Springer, Cham (Apr 2023). https://doi.org/10.1007/978-3-031-30620-4_3
5. Chakraborty, S., Hofheinz, D., Maurer, U., Rito, G.: Deniable authentication when signing keys leak. Cryptology ePrint Archive, Report 2023/213 (2023), https://eprint.iacr.org/2023/213
6. Damgård, I., Haagh, H., Mercer, R., Nitulescu, A., Orlandi, C., Yakoubov, S.: Stronger security and constructions of multi-designated verifier signatures. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 229–260. Springer, Cham (Nov 2020). https://doi.org/10.1007/978-3-030-64378-2_9
7. Dodis, Y., Katz, J., Smith, A., Walfish, S.: Composability and on-line deniability of authentication. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 146–162. Springer, Berlin, Heidelberg (Mar 2009). https://doi.org/10.1007/978-3-642-00457-5_10
8. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences **28**(2), 270–299 (1984)
9. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: Maurer, U.M. (ed.) EUROCRYPT'96. LNCS, vol. 1070, pp. 143–154. Springer, Berlin, Heidelberg (May 1996). https://doi.org/10.1007/3-540-68339-9_13
10. Jost, D., Maurer, U.: Overcoming impossibility results in composable security using interval-wise guarantees. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 33–62. Springer, Cham (Aug 2020). https://doi.org/10.1007/978-3-030-56784-2_2
11. Laguillaumie, F., Vergnaud, D.: Multi-designated verifiers signatures. In: López, J., Qing, S., Okamoto, E. (eds.) ICICS 04. LNCS, vol. 3269, pp. 495–507. Springer, Berlin, Heidelberg (Oct 2004). https://doi.org/10.1007/978-3-540-30191-2_38
12. Laguillaumie, F., Vergnaud, D.: Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In: Blundo, C., Cimato, S. (eds.) SCN 04. LNCS, vol. 3352, pp. 105–119. Springer, Berlin, Heidelberg (Sep 2005). https://doi.org/10.1007/978-3-540-30598-9_8
13. Li, Y., Susilo, W., Mu, Y., Pei, D.: Designated verifier signature: Definition, framework and new constructions. In: Indulska, J., Ma, J., Yang, L.T., Ungerer, T., Cao, J. (eds.) UIC 2007. LNCS, vol. 4611, pp. 1191–1200. Springer (2007), https://doi.org/10.1007/978-3-540-73549-6_116
14. Lipmaa, H., Wang, G., Bao, F.: Designated verifier signature schemes: Attacks, new security notions and a new construction. In: Caires, L., Italiano, G.F., Monteiro,

L., Palamidessi, C., Yung, M. (eds.) ICALP 2005. LNCS, vol. 3580, pp. 459–471. Springer, Berlin, Heidelberg (Jul 2005). https://doi.org/10.1007/11523468_38
15. Liu-Zhang, C.D., Maurer, U.: Synchronous constructive cryptography. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 439–472. Springer, Cham (Nov 2020). https://doi.org/10.1007/978-3-030-64378-2_16
16. Maurer, U.: Constructive cryptography—a new paradigm for security definitions and proofs. In: Proceedings of Theory of Security and Applications, TOSCA 2011. Lecture Notes in Computer Science, vol. 6993, pp. 33–56. Springer (2012). https://doi.org/10.1007/978-3-642-27375-9_3
17. Maurer, U., Portmann, C., Rito, G.: Giving an adversary guarantees (or: How to model designated verifier signatures in a composable framework). In: Tibouchi, M., Wang, H. (eds.) ASIACRYPT 2021, Part III. LNCS, vol. 13092, pp. 189–219. Springer, Cham (Dec 2021). https://doi.org/10.1007/978-3-030-92078-4_7
18. Maurer, U., Portmann, C., Rito, G.: Multi-designated receiver signed public key encryption. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 644–673. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3_22
19. Maurer, U., Portmann, C., Rito, G.: Multi-designated receiver signed public key encryption. Cryptology ePrint Archive, Report 2022/256 (2022), https://eprint.iacr.org/2022/256
20. Maurer, U., Renner, R.: Abstract cryptography. In: Chazelle, B. (ed.) ICS 2011. pp. 1–21. Tsinghua University Press (Jan 2011)
21. Maurer, U.M.: Indistinguishability of random systems. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 110–132. Springer, Berlin, Heidelberg (Apr / May 2002). https://doi.org/10.1007/3-540-46035-7_8
22. Maurer, U.M., Pietrzak, K., Renner, R.: Indistinguishability amplification. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 130–149. Springer, Berlin, Heidelberg (Aug 2007). https://doi.org/10.1007/978-3-540-74143-5_8
23. Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th FOCS. pp. 543–553. IEEE Computer Society Press (Oct 1999). https://doi.org/10.1109/SFFCS.1999.814628
24. Shoup, V.: Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332 (2004), https://eprint.iacr.org/2004/332
25. Steinfeld, R., Bull, L., Wang, H., Pieprzyk, J.: Universal designated-verifier signatures. In: Laih, C.S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 523–542. Springer, Berlin, Heidelberg (Nov / Dec 2003). https://doi.org/10.1007/978-3-540-40061-5_33
26. Steinfeld, R., Wang, H., Pieprzyk, J.: Efficient extension of standard Schnorr/RSA signatures into universal designated-verifier signatures. In: Bao, F., Deng, R., Zhou, J. (eds.) PKC 2004. LNCS, vol. 2947, pp. 86–100. Springer, Berlin, Heidelberg (Mar 2004). https://doi.org/10.1007/978-3-540-24632-9_7
27. Zhang, Y., Au, M.H., Yang, G., Susilo, W.: (strong) multi-designated verifiers signatures secure against rogue key attack. In: Xu, L., Bertino, E., Mu, Y. (eds.) NSS 2012. LNCS, vol. 7645, pp. 334–347. Springer (2012), https://doi.org/10.1007/978-3-642-34601-9_25

Appendix

A Game-Based Security Definitions

In this section we introduce game-based notions that we use to prove the security of Maurer et al.'s MDRS-PKE construction [18]. We only introduce notions that are strictly necessary for such security proofs.

A.1 One Way Function

A One Way Function (OWF) is a pair $\Pi = (S, F)$, where S is a PPT and F a PT.

Consider an OWF $\Pi = (S, F)$; the game system of Definition 18 has (an implicitly defined) security parameter k and provides adversaries with access to oracles $\mathcal{O}_Y$ and $\mathcal{O}_S$ defined below:

$\mathcal{O}_Y(i \in \mathbb{N})$**:** 1. On the first call on index $i \in \mathbb{N}$, compute $x \leftarrow S(1^k)$ and store $(i, x, y \coloneqq F(x))$; output y;
2. On subsequent calls, simply output y.

$\mathcal{O}_S(i \in \mathbb{N}, x)$**:** 1. On the first call on i (to either this oracle or to $\mathcal{O}_Y$), compute $x \leftarrow S(1^k)$ and store $(i, x, y \coloneqq F(x))$; the oracle does not give any output;
2. On subsequent calls, the oracle simply does not perform any action nor give any output.

Definition 18. *Game* $\mathbf{G}^{\text{OWF}}$ *gives an adversary* $\mathbf{A}$ *access to oracles* $\mathcal{O}_Y$ *and* $\mathcal{O}_S$*.* $\mathbf{A}$ *wins if it makes a query* $\mathcal{O}_S(i, x)$ *such that* $F(x) = \mathcal{O}_Y(i)$*.*

An adversary $\mathbf{A}$ (ε, t)-breaks the (n)-One-Wayness of OWF Π if it runs in time t, queries oracles $\mathcal{O}_Y$ and $\mathcal{O}_S$ on at most n different indices $i \in \mathbb{N}$, and satisfies $Adv^{\text{OWF}}(\mathbf{A}) \geq \varepsilon$.

A.2 Public Key Encryption

A Public Key Encryption (PKE) scheme Π with message space $\mathcal{M}$ is a triple of PPTs $\Pi = (G, E, D)$. Below we state the multi-user multi-challenge variants of Correctness (from [5]) and IND-CPA security for PKE schemes (first introduced in [8]).

Let $\Pi = (G, E, D)$ be a PKE scheme with message space $\mathcal{M}$; as before, we assume the game system of the following definition has (an implicitly defined) security parameter k. Definition 19 provides adversaries with access to the following oracles:

$\mathcal{O}_{SK}(B_j)$**:** 1. On the first call on B_j, compute and store $(\mathtt{pk}_j, \mathtt{sk}_j) \leftarrow G(1^k)$; output $(\mathtt{pk}_j, \mathtt{sk}_j)$; 2. On subsequent calls, simply output $(\mathtt{pk}_j, \mathtt{sk}_j)$.

$\mathcal{O}_{PK}(B_j)$**:** 1. $(\mathtt{pk}_j, \mathtt{sk}_j) \leftarrow \mathcal{O}_{SK}(B_j)$; 2. Output $\mathtt{pk}_j$.

$\mathcal{O}_E(B_j, m; r)$**:** 1. If r is given as input, encrypt m under $\mathtt{pk}_j$ (B_j's public key, as generated by $\mathcal{O}_{PK}$) using r as random tape; if r is not given as input create a fresh encryption of m under $\mathtt{pk}_j$; 2. Output the resulting ciphertext back to the adversary.

$\mathcal{O}_D(B_j, c)$**:** 1. Decrypt c using $\mathtt{sk}_j$ (B_j's secret key, as generated by $\mathcal{O}_{PK}$); 2. Output the resulting plaintext back to the adversary (or $\perp$ if decryption failed).

Definition 19 (Correctness: $\mathbf{G}^{\mathsf{Corr}}$). *Game $\mathbf{G}^{\mathsf{Corr}}$ provides an adversary $\mathbf{A}$ with access to oracles $\mathcal{O}_{SK}$, $\mathcal{O}_{PK}$, $\mathcal{O}_E$ and $\mathcal{O}_D$. $\mathbf{A}$ wins the game if there are two queries q_E and q_D to $\mathcal{O}_E$ and $\mathcal{O}_D$, respectively, where q_E has input $(B_j, m; r)$ and q_D has input $(B_j{}', c)$, the input c in q_D is the output of q_E, $B_j = B_j{}'$, and the output of q_D is not m.*

A (computationally unbounded) adversary $\mathbf{A}$ (ε)-breaks the (n)-Correctness of a PKE scheme Π if $\mathbf{A}$ queries $\mathcal{O}_{SK}$, $\mathcal{O}_E$ and $\mathcal{O}_D$ on at most n different parties and satisfies $Adv^{\mathsf{Corr}}(\mathbf{A}) \geq \varepsilon$.

The IND-CPA game systems provide adversaries with access to oracle $\mathcal{O}_{PK}$ described above, and to an additional oracle $\mathcal{O}_E$ which behaves as follows:

$\mathcal{O}_E(B_j, m_0, m_1)$**:** 1. For game system $\mathbf{G}_{\mathbf{b}}^{\mathsf{IND\text{-}CPA}}$, the oracle encrypts $m_{\mathbf{b}}$ under B_j's public key, $\mathtt{pk}_j$, creating a fresh ciphertext c; 2. The oracle outputs the resulting ciphertext c back to the adversary.

Definition 20. *For $\mathbf{b} \in \{0, 1\}$, game $\mathbf{G}_{\mathbf{b}}^{\mathsf{IND\text{-}CPA}}$ gives an adversary $\mathbf{A}$ access to oracles $\mathcal{O}_{PK}$ and $\mathcal{O}_E$. $\mathbf{A}$ wins if $b' = \mathbf{b}$ and for every query $\mathcal{O}_E(B_j, m_0, m_1)$, $|m_0| = |m_1|$.*

We say $\mathbf{A}$ (ε, t)-breaks the (n, q_E)-IND-CPA security of a PKE scheme Π if $\mathbf{A}$ runs in time at most t, queries the oracles it has access to on at most n different parties, makes at most q_E queries to oracle $\mathcal{O}_E$, and satisfies $Adv^{\mathsf{IND\text{-}CPA}}(\mathbf{A}) \geq \varepsilon$. Finally, Π is $(\varepsilon_{\mathsf{Corr}}, \varepsilon_{\mathsf{IND\text{-}CPA}}, t, n, q_E)$-secure if no adversary $\mathbf{A}$ $(\varepsilon_{\mathsf{IND\text{-}CPA}}, t)$-breaks the (n, q_E)-IND-CPA security of Π and no (possibly computationally unbounded) adversary $(\varepsilon_{\mathsf{Corr}})$-breaks the (n)-Correctness of Π.

A.3 Digital Signature Scheme

A Digital Signature Scheme (DSS) for a message space $\mathcal{M}$ is a triple $\Pi = (G, Sig, Vfy)$ of PPTs. Below we state the definition of (One-Time) Strong Existential Unforgeability for DSS. The notion has an implicitly defined security parameter k and makes use of oracles $\mathcal{O}_{VK}$, $\mathcal{O}_S$ and $\mathcal{O}_V$, which, for a DSS $\Pi = (G, Sig, Vfy)$, are defined as:

$\mathcal{O}_{VK}(i \in \mathbb{N})$**:** 1. On the first query on i, compute and store $(\mathtt{vk}_i, \mathtt{sk}_i) \leftarrow G(1^k)$; 2. Output $\mathtt{vk}_i$.

$\mathcal{O}_S(i, m)$**:** 1. Compute $\sigma \leftarrow Sig_{\mathtt{sk}_i}(m)$, where $\mathtt{sk}_i$ is the signing key associated with i; output σ.

$\mathcal{O}_V(i, m, \sigma)$**:** 1. Compute $d \leftarrow Vfy_{\mathtt{vk}_i}(m, \sigma)$, where $\mathtt{vk}_i$ is the verification key associated with i; output d.

Definition 21. *Game* $\mathbf{G}^{1\text{-sEUF-CMA}}$ *provides an adversary with access to oracles* $\mathcal{O}_{VK}$, $\mathcal{O}_S$ *and* $\mathcal{O}_V$. $\mathbf{A}$ *wins the game if there is a query to* $\mathcal{O}_V$ *on some input* (i^*, m^*, σ^*) *that outputs* 1*, there is no query to* $\mathcal{O}_S$ *on input* (i^*, m^*) *that output* σ^**, and for each* $i \in \mathbb{N}$ *there is only at most one query to* $\mathcal{O}_S$ *with input* i*.*

$\mathbf{A}$*'s winning advantage is* $Adv^{1\text{-sEUF-CMA}}(\mathbf{A}) \coloneqq \Pr[\mathbf{AG}^{1\text{-sEUF-CMA}} = \mathtt{win}]$.

An adversary $\mathbf{A}$ $(\varepsilon_{1\text{-sEUF-CMA}}, t)$-breaks the (n, q_S, q_V)-1-sEUF-CMA security of Π if $\mathbf{A}$ runs in time at most t, queries $\mathcal{O}_{VK}$, $\mathcal{O}_S$ and $\mathcal{O}_V$ on at most n different indices, makes at most q_S and q_V queries to, respectively, $\mathcal{O}_S$ and $\mathcal{O}_V$, and satisfies $Adv^{1\text{-sEUF-CMA}}(\mathbf{A}) \geq \varepsilon_{1\text{-sEUF-CMA}}$.

A.4 Non Interactive Zero Knowledge

For a binary relation R, let L_R be the language $L_R \coloneqq \{x \mid \exists w, (x, w) \in R\}$ induced by R. A *Non Interactive Proof System* (NIPS) for L_R is a triple of PPT algorithms $\Pi = (G, P, V)$ where:

- $G(1^k)$: given security parameter 1^k, outputs a common reference string $\mathtt{crs}$;
- $P_{\mathtt{crs}}(x, w)$: given a common reference string $\mathtt{crs}$ and a statement-witness pair $(x, w) \in R$, outputs a proof p;
- $V_{\mathtt{crs}}(x, p)$: given a common reference string $\mathtt{crs}$, a statement x and a proof p, either accepts, outputting $\mathtt{valid}$ $(= 1)$ or rejects, outputting $\mathtt{invalid}$ $(= 0)$.

A NIZK scheme $\Pi = (G, P, V, S = (S_G, S_P))$ for a relation R consists of a NIPS scheme $\Pi' = (G, P, V)$ for R and a simulator $S = (S_G, S_P)$, where:

- $S_G(1^k)$: given security parameter 1^k, outputs a pair $(\mathtt{crs}, \tau)$;
- $S_{P(\mathtt{crs},\tau)}(x)$: given a pair $(\mathtt{crs}, \tau)$ and a statement x, outputs a proof p.

Consider a NIZK scheme $\Pi = (G, P, V, S = (S_G, S_P))$. The following security notion, which defines game systems $\mathbf{G}_\mathbf{0}^{\mathsf{ZK}}$ and $\mathbf{G}_\mathbf{1}^{\mathsf{ZK}}$, provides adversaries with access to two oracles, $\mathcal{O}_S$ and $\mathcal{O}_P$, whose behavior depends on the underlying game system. For $\mathbf{G}_\mathbf{b}^{\mathsf{ZK}}$ (with $\mathbf{b} \in \{0, 1\}$):

$\mathcal{O}_S$**:** On the first call, compute and store $\mathtt{crs} \leftarrow G(1^k)$ if $\mathbf{b} = \mathbf{0}$, and $(\mathtt{crs}, \tau) \leftarrow S_G(1^k)$ if $\mathbf{b} = \mathbf{1}$; output $\mathtt{crs}$.

$\mathcal{O}_P(x, w)$**:** If $\mathbf{b} = \mathbf{0}$, output $\pi \leftarrow P_{\mathtt{crs}}(x, w)$; if $\mathbf{b} = \mathbf{1}$, output $\pi \leftarrow S_{P(\mathtt{crs},\tau)}(x)$.

Definition 22 (Zero-Knowledge: $\mathbf{G}_\mathbf{0}^{\mathsf{ZK}}$ and $\mathbf{G}_\mathbf{1}^{\mathsf{ZK}}$). *For* $\mathbf{b} \in \{0, 1\}$*, game* $\mathbf{G}_\mathbf{b}^{\mathsf{ZK}}$ *gives an adversary* $\mathbf{A}$ *access to oracles* $\mathcal{O}_S$ *and* $\mathcal{O}_P$. $\mathbf{A}$ *wins the game if* $b' = \mathbf{b}$ *and every query* $\mathcal{O}_P(x, w)$ *satisfies* $(x, w) \in R$*.*

We say that an adversary $\mathbf{A}$ $(\varepsilon_{\mathsf{ZK}}, t)$-breaks the (q_P)-ZK security of a NIZK scheme Π if it makes at most q_P queries to $\mathcal{O}_P$ and satisfies $Adv^{\mathsf{ZK}}(\mathbf{A}) \geq \varepsilon_{\mathsf{ZK}}$.

We now introduce Simulation Soundness for NIZK [23]. The game system defined by this notion provides adversaries with access to oracles $\mathcal{O}_S$, $\mathcal{O}_P$ and $\mathcal{O}_V$ defined as:

$\mathcal{O}_S$ on the first call sample $(\mathtt{crs}, \tau) \leftarrow S_G(1^k)$; output $\mathtt{crs}$.

$\mathcal{O}_P(x)$**:** Output $S_{P(\mathtt{crs},\tau)}(x)$.
$\mathcal{O}_V(x,p)$**:** Output $V_{\mathtt{crs}}(x,p)$.

Definition 23 (Simulation Soundness: $\mathbf{G}^{\mathsf{SS}}$). *Game* $\mathbf{G}^{\mathsf{SS}}$ *gives an adversary* $\mathbf{A}$ *access to oracles* $\mathcal{O}_S$, $\mathcal{O}_P$ *and* $\mathcal{O}_V$. $\mathbf{A}$ *wins if it makes a query* $\mathcal{O}_V(x,p)$ *with* $x \notin L_R$ *that outputs* `valid` *and no query* $\mathcal{O}_P(x)$ *output* p.

An adversary $\mathbf{A}$ $(\varepsilon_{\mathsf{SS}}, t)$-breaks the (q_P, q_V)-Simulation Soundness of a NIZK scheme Π if it makes at most q_P and q_V queries to $\mathcal{O}_P$ and $\mathcal{O}_V$, respectively, and satisfies $Adv^{\mathsf{SS}}(\mathbf{A}) \geq \varepsilon_{\mathsf{SS}}$. Finally, we say that a NIZK scheme Π is $(\varepsilon_{\mathsf{ZK}}, \varepsilon_{\mathsf{SS}}, t, q_P, q_V)$-secure if no adversary $\mathbf{A}$ $(\varepsilon_{\mathsf{ZK}}, t)$-breaks the (q_P)-Zero-Knowledge of Π or $(\varepsilon_{\mathsf{SS}}, t)$-breaks the (q_P, q_V)-Simulation Soundness of Π.

A.5 Public Key Encryption for Broadcast

A Public Key Encryption for Broadcast (PKEBC) scheme Π with message space $\mathcal{M}$ is a quadruple $\Pi = (S, G, E, D)$ of PPTs. Below we state the Correctness, Robustness, Consistency and $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}$ security notions from [5].

Consider a PKEBC $\Pi = (S, G, E, D)$ with message space $\mathcal{M}$. The game systems defined by the security notions ahead have an implicitly defined security parameter k and provide adversaries with access to the following oracles:

$\mathcal{O}_{PP}$**:** 1. On the first call, compute and store $\mathtt{pp} \leftarrow S(1^k)$; output $\mathtt{pp}$; 2. On subsequent calls, output the previously generated $\mathtt{pp}$.
$\mathcal{O}_{SK}(B_j)$**:** 1. If $\mathcal{O}_{SK}$ was queried on B_j before, simply look up and return the previously generated key for B_j; 2. Otherwise, store $(\mathtt{pk}_j, \mathtt{sk}_j) \leftarrow G(\mathtt{pp})$ as B_j's key-pair, and output $(\mathtt{pk}_j, \mathtt{sk}_j)$.
$\mathcal{O}_{PK}(B_j)$**:** 1. $(\mathtt{pk}_j, \mathtt{sk}_j) \leftarrow \mathcal{O}_{SK}(B_j)$; 2. Output $\mathtt{pk}_j$.
$\mathcal{O}_E(\vec{V}, m)$**:** 1. $\vec{v} \leftarrow (\mathcal{O}_{PK}(V_1), \ldots, \mathcal{O}_{PK}(V_{|\vec{V}|}))$; 2. Create and output a fresh encryption $c \leftarrow E_{\mathtt{pp},\vec{v}}(m)$.
$\mathcal{O}_D(B_j, c)$**:** 1. Query $\mathcal{O}_{SK}(B_j)$ to obtain the corresponding secret-key $\mathtt{sk}_j$; 2. Decrypt c using $\mathtt{sk}_j$, $(\vec{v}, m) \leftarrow D_{\mathtt{pp},\mathtt{sk}_j}(c)$, and then output the resulting receiver-s-message pair $(\vec{v}, m)$, or $\perp$ (if $(\vec{v}, m) = \perp$, i.e. the ciphertext is not valid with respect to B_j's secret key).

Definition 24 (Correctness). *Game* $\mathbf{G}^{\mathsf{Corr}}$ *provides an adversary* $\mathbf{A}$ *with access to oracles* $\mathcal{O}_{PP}$, $\mathcal{O}_{SK}$, $\mathcal{O}_E$ *and* $\mathcal{O}_D$. $\mathbf{A}$ *wins the game if there are two queries* q_E *and* q_D *to* $\mathcal{O}_E$ *and* $\mathcal{O}_D$, *respectively, where* q_E *has input* $(\vec{V}, m)$ *and* q_D *has input* (B_j, c), *satisfying* $B_j \in \vec{V}$, *the input* c *in* q_D *is the output of* q_E, *and the output of* q_D *is either* $\perp$ *or* $(\vec{v}', m')$ *with* $(\vec{v}, m) \neq (\vec{v}', m')$.

The advantage of $\mathbf{A}$ *in winning the Correctness game is the probability that* $\mathbf{A}$ *wins game* $\mathbf{G}^{\mathsf{Corr}}$ *as described above, and is denoted* $Adv^{\mathsf{Corr}}(\mathbf{A})$.

Definition 25 (Robustness). *Game* $\mathbf{G}^{\mathsf{Rob}}$ *provides an adversary* $\mathbf{A}$ *with access to oracles* $\mathcal{O}_{PP}$, $\mathcal{O}_{SK}$, $\mathcal{O}_{PK}$, $\mathcal{O}_E$ *and* $\mathcal{O}_D$. $\mathbf{A}$ *wins the game if there are two queries* q_E *and* q_D *to* $\mathcal{O}_E$ *and* $\mathcal{O}_D$, *respectively, where* q_E *has input* $(\vec{V}, m)$ *and* q_D *has input* (B_j, c), *satisfying* $B_j \notin \vec{V}$, *the input* c *in* q_D *is the output of* q_E, *and*

the output of q_D is $(\vec{v}', m')$ with $(\vec{v}', m') \neq \perp$. The advantage of $\mathbf{A}$ *in winning the Robustness game is the probability that* $\mathbf{A}$ *wins game* $\mathbf{G}^{\mathsf{Rob}}$ *as described above, and is denoted* $Adv^{\mathsf{Rob}}(\mathbf{A})$.

Definition 26 (Consistency). $\mathbf{G}^{\mathsf{Cons}}$ *provides an adversary* $\mathbf{A}$ *with access to oracles $\mathcal{O}_{PP}$, $\mathcal{O}_{SK}$, $\mathcal{O}_E$ and $\mathcal{O}_D$.* $\mathbf{A}$ *wins the game if there are two queries $\mathcal{O}_D(B_i, c)$ and $\mathcal{O}_D(B_j, c)$ for some B_i and B_j (possibly with $B_i = B_j$) on the same ciphertext c such that query $\mathcal{O}_D(B_i, c)$ outputs some pair $(\vec{v}, m) \neq \perp$ with $\mathtt{pk}_j \in \vec{v}$ (where $\mathtt{pk}_j$ is B_j's public key), and query $\mathcal{O}_D(B_j, c)$ does not output $(\vec{v}, m)$.*

$\mathbf{A}$*'s advantage in winning the Consistency game is denoted* $Adv^{\mathsf{Cons}}(\mathbf{A})$ *and corresponds to the probability that* $\mathbf{A}$ *wins game* $\mathbf{G}^{\mathsf{Cons}}$.

Below we present the definition of $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}$ security from [4]. The games defined by this definition provide adversaries with access to the oracles $\mathcal{O}_{PP}$, $\mathcal{O}_{SK}$ and $\mathcal{O}_{PK}$ defined above, as well as to the following modified $\mathcal{O}_E$ and $\mathcal{O}_D$ oracles:

$\mathcal{O}_E\big((\vec{V}_0, m_0), (\vec{V}_1, m_1)\big)$**:** 1. For game system $\mathbf{G}_{\mathbf{b}}^{\{\mathsf{IND},\,\mathsf{IK}\}\text{-}\mathsf{CCA}}$, encrypt $m_{\mathbf{b}}$ under $\vec{v}_{\mathbf{b}}$, the vector of public keys corresponding to $\vec{V}_{\mathbf{b}}$; output c.

$\mathcal{O}_D(B_j, c)$**:** 1. If c was the output of some query to $\mathcal{O}_E$, output $\mathtt{test}$;
2. Otherwise, compute and output $(\vec{v}, m) \leftarrow D_{\mathtt{pp},\mathtt{sk}_j}(c)$, where $\mathtt{sk}_j$ is B_j's secret key.

Definition 27 ($\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}$ Security). *For* $\mathbf{b} \in \{0, 1\}$*, game system* $\mathbf{G}_{\mathbf{b}}^{\{\mathsf{IND},\,\mathsf{IK}\}\text{-}\mathsf{CCA}}$ *provides an adversary* $\mathbf{A}$ *with access to oracles $\mathcal{O}_{PP}$, $\mathcal{O}_{SK}$, $\mathcal{O}_{PK}$, $\mathcal{O}_E$ and $\mathcal{O}_D$.* $\mathbf{A}$ *wins the game if it outputs a guess bit b' satisfying $b' = \mathbf{b}$ and for every query $\mathcal{O}_E\big((\vec{V}_0, m_0), (\vec{V}_1, m_1)\big)$: 1. $|\vec{V}_0| = |\vec{V}_1|$; 2. $|m_0| = |m_1|$; and 3. there is no query to $\mathcal{O}_{SK}$ on any $B_j \in Set(\vec{V}_0) \cup Set(\vec{V}_1)$ at any point during the game. We define the advantage of* $\mathbf{A}$ *in winning the* $\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}$ *game as*

$$Adv^{\{\mathsf{IND},\,\mathsf{IK}\}\text{-}\mathsf{CCA}}(\mathbf{A}) := \Big|\Pr[\mathbf{AG}_{\mathbf{0}}^{\{\mathsf{IND},\,\mathsf{IK}\}\text{-}\mathsf{CCA}} = \mathtt{win}] + \Pr[\mathbf{AG}_{\mathbf{1}}^{\{\mathsf{IND},\,\mathsf{IK}\}\text{-}\mathsf{CCA}} = \mathtt{win}] - 1\Big|.$$

We say an adversary $\mathbf{A}$ (ε, t)-breaks the (n, d_E, q_E, q_D)-Correctness (resp. -Robustness, -Consistency, -$\{\mathsf{IND}, \mathsf{IK}\}$-$\mathsf{CCA}$ security) of a PKEBC scheme Π if $\mathbf{A}$ runs in time at most t, queries $\mathcal{O}_{SK}$, $\mathcal{O}_E$ and $\mathcal{O}_D$ on at most n different parties, makes at most q_E and q_D queries to $\mathcal{O}_E$ and $\mathcal{O}_D$, respectively, with the sum of lengths of the party vectors input to $\mathcal{O}_E$ being at most d_E, and satisfies $Adv^{\mathsf{Corr}}(\mathbf{A}) \geq \varepsilon$ (resp. $Adv^{\mathsf{Rob}}(\mathbf{A}) \geq \varepsilon$, $Adv^{\mathsf{Cons}}(\mathbf{A}) \geq \varepsilon$, $Adv^{\{\mathsf{IND},\,\mathsf{IK}\}\text{-}\mathsf{CCA}}(\mathbf{A}) \geq \varepsilon$).

B Security of Chakraborty et al.'s MDVS [4]

We now prove Chakraborty et al.'s MDVS construction (defined in Algorithm 11) satisfies Forgery Invalidity.

B.1 Unforgeability against Replays

Theorem 5. *If* Π_{PKE} *is*

$$(\varepsilon_{\text{PKE-Corr}}, \varepsilon_{\text{PKE-IND-CPA}}, t_{\text{PKE}}, n_{\text{PKE}}, q_{E\,\text{PKE}})\text{-}secure, \tag{B.1}$$

with $n_{\text{PKE}} \geq 1$, Π_{NIZK} *is*

$$(\varepsilon_{\text{NIZK-ZK}}, \varepsilon_{\text{NIZK-SS}}, t_{\text{NIZK}}, q_{P\,\text{NIZK}}, q_{V\,\text{NIZK}})\text{-}secure, \tag{B.2}$$

and Π_{OWF} *is*

$$(\varepsilon_{\text{OWF}}, t_{\text{OWF}}, n_{\text{OWF}})\text{-}secure, \tag{B.3}$$

with $t_{\text{OWF}} \gtrapprox n_{\text{OWF}} \cdot (t_S + t_F)$*—where* t_S *and* t_F *are, respectively, the times to run* $\Pi_{\text{OWF}}.S$ *and* $\Pi_{\text{OWF}}.F$*—and with* $n_{\text{OWF}} \geq 1$*, then no adversary* $\mathbf{A}$ (ε, t)*-breaks* Π*'s*

$$(n_S \coloneqq \max(n_{\text{OWF}} - n_V, 0), n_V \coloneqq \min(n_{\text{PKE}}, \max(n_{\text{OWF}} - n_S, 0)),$$
$$q_S \coloneqq \min(q_{P\,\text{NIZK}}, q_{E\,\text{PKE}}), q_V \coloneqq q_{V\,\text{NIZK}})\text{-}Unforgeability\ against\ Replays,$$

with $\varepsilon > (3 \cdot \varepsilon_{\text{PKE-Corr}} + \varepsilon_{\text{PKE-IND-CPA}}) + \varepsilon_{\text{NIZK-ZK}} + \varepsilon_{\text{NIZK-SS}} + 4 \cdot \varepsilon_{\text{OWF}}$*, with* $t_{\text{PKE}}, t_{\text{OWF}} \approx t + t_{\text{R-Unforg}} + q_S \cdot t_{S_{Sim}} + t_{S_{CRS}}$ *and with* $t_{\text{NIZK}} \approx t + t_{\text{R-Unforg}}$*, where* $t_{\text{R-Unforg}}$ *is the time to run* Π*'s* $\mathbf{G}^{\text{R-Unforg}}$ *game and* $t_{S_{Sim}}$ *and* $t_{S_{CRS}}$ *are, respectively, the runtime of* Π_{NIZK}*'s* S_{Sim} *and* S_{CRS} *algorithms.*

The original proof from [4] establishing the unforgeability of their MDVS construction also implies this stronger result (see [5, Proof of Theorem 6]).

B.2 Forgery Invalidity

Theorem 6. *If no adversary* $\mathbf{A}$ $(\varepsilon_{\text{PKE}})$*-breaks the* (n_{PKE})*-Correctness of* Π_{PKE} *then no (computationally unbounded) adversary* (ε)*-breaks* $\Pi^{\text{adap}}_{\text{MDVS}}$*'s*

$$(n_V \coloneqq n_{\text{PKE}})\text{-}Forgery\ Invalidity,$$

with $\varepsilon > 2 \cdot \varepsilon_{\text{PKE}}$.

Proof. This proof proceeds in a sequence of games [1, 24].

$\mathbf{G}^{\text{Forge-Invalid}} \rightsquigarrow \mathbf{G}^1$**:** $\mathbf{G}^1$ is just like the original game $\mathbf{G}^{\text{Forge-Invalid}}$, except that in $\mathbf{G}^1$ the Π_{PKE} key-pair $(\mathtt{pk}_0, \mathtt{sk}_0)$ sampled for each party B_j is assumed to be correct.

One can reduce distinguishing these two games to breaking Π_{PKE}'s correctness because the reduction holds all secret keys (and so it can handle any oracle queries). If an adversary $\mathbf{A}$ only queries for the verifier keys of up to $n_V \leq n_{\text{PKE}}$ parties, and given the reduction only has to use Π_{PKE}-$\mathcal{O}_{SK}$ oracle to generate at most one key-pair per party—namely, $(\mathtt{pk}_0, \mathtt{sk}_0)$—since by assumption no computationally unbounded adversary $(\varepsilon_{\text{PKE}})$-breaks the (n_{PKE})-Correctness of Π_{PKE}, it follows

$$\left|\Pr[\mathbf{AG}^1 = \mathtt{win}] - \Pr[\mathbf{AG}^{\text{Forge-Invalid}} = \mathtt{win}]\right| \leq \varepsilon_{\text{PKE}}.$$

$\mathbf{G}^1 \rightsquigarrow \mathbf{G}^2$**.** This game hop is just like the previous one (i.e. $\mathbf{G}^{\mathsf{Forge\text{-}Invalid}} \rightsquigarrow \mathbf{G}^1$), the only difference being that the key-pair which is assumed to be a correct one is now $(\mathtt{pk}_1, \mathtt{sk}_1)$. It follows

$$\left|\Pr[\mathbf{AG}^2 = \mathtt{win}] - \Pr[\mathbf{AG}^1 = \mathtt{win}]\right| \leq \varepsilon_{\mathrm{PKE}}.$$

To finish the proof we now prove the following claim:

Claim. For any adversary $\mathbf{A}$

$$\Pr[\mathbf{AG}^2 = \mathtt{win}] = 0.$$

Proof. Recall that an adversary can only win game $\mathbf{G}^2$ if it makes a query to $\mathcal{O}_V$ on some input $(A_i, B_j, \vec{V}, m, \sigma \coloneqq (p, \vec{c}, c_{\mathtt{pp}}))$ such that σ was output by a query to $\mathcal{O}_{Forge}$ on input $(A_i, \vec{V}, m, \mathcal{C})$ satisfying $B_j \in \vec{V}$, $B_j \notin \mathcal{C}$, and $\mathcal{O}_V$ outputs 1. First, note that, by the definition of $\mathcal{O}_{Forge}$, this means that B_j's secret verifier key is not given as input when the oracle is forging the signature using algorithm *Forge* of $\Pi^{\mathsf{adap}}_{\mathrm{MDVS}}$. Furthermore, by the definition of $\Pi^{\mathsf{adap}}_{\mathrm{MDVS}}$'s *Forge* algorithm (Algorithm 11), it follows that for every $l \in \{1, \ldots, |\vec{V}|\}$ such that $V_l = B_j$, the two ciphertexts in c_l (i.e. $c_{l,0}$ and $c_{l,1}$) are encryptions of 0. Finally, by the assumption that the two PKE key-pairs of B_j—i.e. $(\mathtt{pk}_0, \mathtt{sk}_0)$ and $(\mathtt{pk}_1, \mathtt{sk}_1)$—are correct, the decryption of either $c_{l,0}$ or $c_{l,1}$ will result in 0 being output, and so the signature will not verify as valid by $\mathcal{O}_V$. □

□

B.3 Public-Key Collision Resistance

Definition 28 (n-Instance ε-Image Collision Resistance [5]). *A* OWF $\Pi = (S, F)$ *is n-Instance ε-Image Collision Resistant if*

$$\Pr\left[\left|\{\Pi.F(x_1), \ldots, \Pi.F(x_n)\}\right| < n \;\middle|\; \begin{array}{c} x_1 \leftarrow \Pi.S(1^k) \\ \ldots \\ x_n \leftarrow \Pi.S(1^k) \end{array}\right] \leq \varepsilon.$$

Lemma 1 ([5, Lemma 1]). *If no adversary (ε, t)-breaks the (n)-One-Wayness of a* OWF $\Pi = (S, F)$*, with $t \gtrapprox n \cdot (t_S + t_F)$—where t_S and t_F are, respectively, the times to run S and F—then Π is n-Instance ε'-Image Collision-Resistant, with $\varepsilon' \leq 2 \cdot \varepsilon$.*

Corollary 1. *If no adversary $(\varepsilon_{\mathrm{OWF}}, t_{\mathrm{OWF}})$-breaks the (n_{OWF})-One-Wayness of $\Pi_{\mathrm{OWF}} = (S, F)$, with $t_{\mathrm{OWF}} \gtrapprox n_{\mathrm{OWF}} \cdot (t_S + t_F)$—where t_S and t_F are, respectively, the times to run $\Pi_{\mathrm{OWF}}.S$ and $\Pi_{\mathrm{OWF}}.F$—then $\Pi^{\mathsf{adap}}_{\mathrm{MDVS}}$ is*

$$(n \coloneqq \max(n_{\mathrm{OWF}} - n_V, 0), \ell \coloneqq \max(n_{\mathrm{OWF}} - n_S, 0))\text{-}\textit{Party}$$
$$\varepsilon\text{-}\textit{Public-Key Collision-Resistant}$$

with $\varepsilon \geq 2 \cdot \varepsilon_{\mathrm{OWF}}$.

Proof. Follows from the definition of $\Pi^{\mathsf{adap}}_{\mathrm{MDVS}}$ (Algorithm 11), from Lemma 1 and from the assumption on Π_{OWF}. □

C Security of Maurer et al.'s MDRS-PKE [18]

In this section we prove the security of Maurer et al.'s MDRS-PKE construction [18] (see Algorithm 12) with respect to our new security notions (see Section 4.1).

C.1 Consistency

Theorem 7. *If no adversary* $(\varepsilon_{\text{PKEBC}}, t_{\text{PKEBC}})$*-breaks the* $(n_{\text{PKEBC}}, d_{E\text{PKEBC}}, q_{E\text{PKEBC}}, q_{D\text{PKEBC}})$*-Consistency of* Π_{PKEBC}*, no adversary* $(\varepsilon_{\text{MDVS}}, t_{\text{MDVS}})$*-breaks the* $(n_{S\text{MDVS}}, n_{V\text{MDVS}}, d_{S\text{MDVS}}, d_{F\text{MDVS}}, q_{S\text{MDVS}}, q_{V\text{MDVS}}, q_{F\text{MDVS}})$*-Consistency of* Π_{MDVS} *and* $\Pi_{\text{DSS}}.\mathit{Vfy}$ *is a deterministic algorithm, then no adversary* (ε, t)*-breaks* $\Pi_{\text{MDRS-PKE}}$*'s*

$$\begin{aligned}&(n_S \coloneqq n_{S\text{MDVS}}, n_R \coloneqq \min(n_{\text{PKEBC}}, n_{V\text{MDVS}}),\\ &d_E \coloneqq \min(d_{E\text{PKEBC}}, d_{S\text{MDVS}}), q_E \coloneqq \min(q_{E\text{PKEBC}}, q_{S\text{MDVS}}),\\ &q_D \coloneqq \min(q_{D\text{PKEBC}}, q_{V\text{MDVS}}))\text{-}\mathit{Consistency},\end{aligned}$$

with $\varepsilon > \varepsilon_{\text{PKEBC}} + \varepsilon_{\text{MDVS}}$*, and* $t_{\text{PKEBC}}, t_{\text{MDVS}} \approx t + t_{\mathsf{Cons}}$*, where* t_{Cons} *is the time to run* $\Pi_{\text{MDRS-PKE}}$*'s* $\mathbf{G}^{\mathsf{Cons}}$ *game.*

Proof. Follows from the proof of [18, Theorem 7] and the definition of $\Pi_{\text{MDRS-PKE}}$'s decryption (see Algorithm 12). □

C.2 Replay Unforgeability

Theorem 8. *If no adversary* $(\varepsilon_{\text{MDVS}}, t_{\text{MDVS}})$*-breaks the* $(n_{S\text{MDVS}}, n_{V\text{MDVS}}, d_{S\text{MDVS}}, q_{S\text{MDVS}}, q_{V\text{MDVS}})$*-Unforgeability of* Π_{MDVS} *and no adversary* $(\varepsilon_{\text{DSS}}, t_{\text{DSS}})$*-breaks the* $(n_{\text{DSS}}, q_{S\text{DSS}}, q_{V\text{DSS}})$*-1-*$\mathsf{sEUF\text{-}CMA}$ *security of* Π_{DSS}*, then no adversary* $\mathbf{A}$ (ε, t)*-breaks* $\Pi_{\text{MDRS-PKE}}$*'s*

$$\begin{aligned}&(n_S \coloneqq n_{S\text{MDVS}}, n_R \coloneqq n_{V\text{MDVS}},\\ &d_E \coloneqq d_{S\text{MDVS}}, q_E \coloneqq \min(q_{S\text{MDVS}}, n_{\text{DSS}}, q_{S\text{DSS}}),\\ &q_D \coloneqq \min(q_{V\text{MDVS}}, q_{V\text{DSS}}))\text{-}\mathit{Replay\ Unforgeability},\end{aligned}$$

with $\varepsilon > \varepsilon_{\text{DSS}} + \varepsilon_{\text{MDVS}}$*, and* $t_{\text{DSS}}, t_{\text{MDVS}} \approx t + t_{\mathsf{R\text{-}Unforg}}$*, where* $t_{\mathsf{R\text{-}Unforg}}$ *is the time to run* $\Pi_{\text{MDRS-PKE}}$*'s* $\mathbf{G}^{\mathsf{R\text{-}Unforg}}$ *game.*

Proof. This proof proceeds via a sequence of games.

$\mathbf{G}^{\mathsf{R\text{-}Unforg}} \rightsquigarrow \mathbf{G}^1$**:** The difference between $\mathbf{G}^1$ and $\mathbf{G}^{\mathsf{R\text{-}Unforg}}$ is that in $\mathbf{G}^1$, when $\mathcal{O}_D$ is queried on an input $(B_j, c \coloneqq (\mathtt{vk}, \sigma', c'))$ such that there is a query $\mathcal{O}_E(A_i, \vec{V}, m)$ that output $c^* \coloneqq (\mathtt{vk}^*, \sigma'^*, c'^*)$ with $c \neq c^*$ and $\mathtt{vk} = \mathtt{vk}^*$, $\mathcal{O}_D$ simply outputs $\perp$.

One can reduce distinguishing the two games to breaking the 1-$\mathsf{sEUF\text{-}CMA}$ security of Π_{DSS}: the reduction holds all MDVS and PKEBC secret keys and

can sign ciphertexts using the $\mathcal{O}_S$ oracle from Π_{DSS}'s $\mathbf{G}^{1\text{-sEUF-CMA}}$ game so it can handle any oracle queries. If $\mathbf{A}$ only makes at most $q_E \leq \min(n_{\text{DSS}}, q_{S\,\text{DSS}})$ and $q_D \leq q_{V\,\text{DSS}}$ queries to $\mathcal{O}_E$ and $\mathcal{O}_D$, respectively, since by assumption no adversary $(t_{\text{DSS}}, \varepsilon_{\text{DSS}})$-breaks the

$$(n_{\text{DSS}}, q_{S\,\text{DSS}}, q_{V\,\text{DSS}})\text{-1-sEUF-CMA}$$

of Π_{DSS}, it follows

$$\left|\Pr[\mathbf{AG}^1 = \texttt{win}] - \Pr[\mathbf{AG}^{\text{R-Unforg}} = \texttt{win}]\right| \leq \varepsilon_{\text{DSS}}.$$

We can now directly reduce to the Unforgeability game of Π_{MDVS}. To see why, note that $\mathbf{G}^1$ already outputs $\perp$ for any query $\mathcal{O}_D(B_j, c \coloneqq (\texttt{vk}, \sigma', c'))$ such that there was a query $\mathcal{O}_E(A_i, \vec{V}, m)$ that output $c^* \coloneqq (\texttt{vk}^*, \sigma'^*, c'^*)$ with $c \neq c^*$ and $\texttt{vk} = \texttt{vk}^*$. This then means that we only have to make sure that no decryption query $\mathcal{O}_D(B_j, c \coloneqq (\texttt{vk}, \sigma', c'))$ such that there was no query $\mathcal{O}_E(A_i, \vec{V}, m)$ that output $c^* \coloneqq (\texttt{vk}^*, \sigma'^*, c'^*)$ with $c \neq c^*$ and $\texttt{vk} = \texttt{vk}^*$ allows the adversary to win the game. On one hand, if $c = c^*$ then the adversary does not win the game (see Definition 15); on the other hand, if some query $\mathcal{O}_D(B_j, c \coloneqq (\texttt{vk}, \sigma', c'))$ (where $\texttt{vk}$ was not output as part of any challenge ciphertext) outputs something other than $\perp$, then the MDVS signature encrypted by c' actually verified as being a valid signature on a triple $(\vec{v}_{\text{PKEBC}}, m, \texttt{vk})$ which was never signed (since $\texttt{vk}$ was not output as part of any $\mathcal{O}_E$ ciphertext).

Since by assumption no adversary $(\varepsilon_{\text{MDVS}}, t_{\text{MDVS}})$-breaks the

$$(n_{S\,\text{MDVS}}, n_{V\,\text{MDVS}}, d_{S\,\text{MDVS}}, q_{S\,\text{MDVS}}, q_{V\,\text{MDVS}})\text{-Unforgeability}$$

of Π_{MDVS}, if $\mathbf{A}$ only queries for at most $n_S \leq n_{S\,\text{MDVS}}$ (resp. $n_R \leq n_{V\,\text{MDVS}}$) different sender keys (resp. different receiver keys), makes up to $q_E \leq q_{S\,\text{MDVS}}$ queries to $\mathcal{O}_E$ and up to $q_D \leq q_{V\,\text{MDVS}}$ queries to $\mathcal{O}_D$, and the sum of lengths of the party vectors input to $\mathcal{O}_E$ is at most $d_E \leq d_{S\,\text{MDVS}}$, it follows

$$\Pr[\mathbf{AG}^1 = \texttt{win}] \leq \varepsilon_{\text{MDVS}}.$$

□

C.3 {IND, IK}-CCA$_S$ Security

Theorem 9. *If no adversary $(\varepsilon_{\text{MDVS}}, t_{\text{MDVS}})$-breaks the $(n_{S\,\text{MDVS}}, n_{V\,\text{MDVS}}, d_{S\,\text{MDVS}}, q_{S\,\text{MDVS}}, q_{V\,\text{MDVS}})$-Message-Bound Validity of Π_{MDVS}, no adversary $(\varepsilon_{\text{DSS}}, t_{\text{DSS}})$-breaks the $(n_{\text{DSS}}, q_{S\,\text{DSS}}, q_{V\,\text{DSS}})$-1-sEUF-CMA security of Π_{DSS}, no adversary $(\varepsilon_{\text{PKEBC}}, t_{\text{PKEBC}})$-breaks the $(n_{\text{PKEBC}}, d_{E\,\text{PKEBC}}, q_{E\,\text{PKEBC}}, q_{D\,\text{PKEBC}})$-Robustness of Π_{PKEBC}, and no adversary $(\varepsilon_{\text{PKEBC}}, t_{\text{PKEBC}})$-breaks the $(n_{\text{PKEBC}}, d_{E\,\text{PKEBC}}, q_{E\,\text{PKEBC}}, q_{D\,\text{PKEBC}})$-{IND, IK}-CCA$_S$ security of Π_{PKEBC}, then no adversary $\mathbf{A}$ (ε, t)-breaks $\Pi_{\text{MDRS-PKE}}$'s*

$$\Big(n_S \coloneqq n_{S\,\text{MDVS}}, n_R \coloneqq \min(n_{\text{PKEBC}}, n_{V\,\text{MDVS}}), d_E \coloneqq \min(d_{E\,\text{PKEBC}}, d_{S\,\text{MDVS}}),$$
$$q_E \coloneqq \min(q_{E\,\text{PKEBC}}, q_{S\,\text{MDVS}}, n_{\text{DSS}}, q_{S\,\text{DSS}}),$$
$$q_D \coloneqq \min(q_{D\,\text{PKEBC}}, q_{V\,\text{MDVS}}, q_{V\,\text{DSS}})\Big)\text{-}\{\mathsf{IND}, \mathsf{IK}\}\text{-}\mathsf{CCA}_S \text{ security,}$$

with

$$\varepsilon > 2 \cdot (\varepsilon_{\text{DSS-1-EUF-CMA}} + \varepsilon_{\text{MDVS-Bound-Val}} + \varepsilon_{\text{PKEBC-Rob}}) + 4 \cdot \varepsilon_{\text{PKEBC-Corr}} + \varepsilon_{\text{PKEBC-\{IND,IK\}-CCA}},$$

and with $t_{\text{DSS}}, t_{\text{MDVS}}, t_{\text{PKEBC}} \approx t + t_{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}_S}$*, where* $t_{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}_S}$ *is the time to run* Π*'s* $\mathbf{G}^{\{\mathsf{IND},\,\mathsf{IK}\}\text{-}\mathsf{CCA}_S}$ *games.*

Proof. One can prove Theorem 9 by following the same sequence of hybrids (and the same arguments) from [4, Proof of Theorem 13], with only minor differences. Below we only explain the differences from [4, Proof of Theorem 13], i.e. we explain how to handle queries $\mathcal{O}_E\big((A_{i,0}, \vec{V}_0, m_0), (A_{i,1}, \vec{V}_1, m_1)\big)$ for the case where $(A_{i,0}, \vec{V}_0, m_0) = (A_{i,1}, \vec{V}_1, m_1)$—which are not considered in the $\{\mathsf{IND},\mathsf{IK}\}$-$\mathsf{CCA}$ notion from [4] for the case where either not all receivers in $\vec{V}_0$ and $\vec{V}_1$ are honest (i.e. the adversary may query for secret keys), vectors $\vec{V}_0$ and $\vec{V}_1$ are of different lengths, or m_0 and m_1 have different sizes.

First, note that regardless of whether an adversary is interacting with game $\mathbf{G}_{\mathbf{0}}^{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}_S}$ or $\mathbf{G}_{\mathbf{1}}^{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}_S}$, the ciphertexts generated by the oracle in such queries have exactly the same distribution, and therefore we only need to ensure the reductions have everything needed to produce such ciphertexts.

For $\beta \in \{0,1\}$, [4, Proof of Theorem 13] proceeds via a sequence of hybrids $\mathbf{G}_\beta^{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}} \rightsquigarrow \mathbf{G}_\beta^1$, $\mathbf{G}_\beta^1 \rightsquigarrow \mathbf{G}_\beta^2$, $\mathbf{G}_\beta^2 \rightsquigarrow \mathbf{G}_\beta^3$, $\mathbf{G}_\beta^3 \rightsquigarrow \mathbf{G}_\beta^4$ and $\mathbf{G}_\beta^4 \rightsquigarrow \mathbf{G}_\beta^5$, and gives reductions from distinguishing each of these pairs to breaking a security property of one of the underlying building blocks. Concretely, [4, Proof of Theorem 13] reduces distinguishing

- **S.1** $\mathbf{G}_\beta^{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}}$ and $\mathbf{G}_\beta^1$ to breaking the 1-$\mathsf{sEUF\text{-}CMA}$ security of the underlying Π_{DSS};
- **S.2** $\mathbf{G}_\beta^1$ and $\mathbf{G}_\beta^2$ to breaking the robustness of the underlying Π_{PKEBC};
- **S.3** $\mathbf{G}_\beta^2$ and $\mathbf{G}_\beta^3$ to breaking the correctness of the underlying Π_{PKEBC};
- **S.4** $\mathbf{G}_\beta^3$ and $\mathbf{G}_\beta^4$ to breaking the message-bound validity of the underlying Π_{MDVS};
- **S.5** $\mathbf{G}_\beta^4$ and $\mathbf{G}_\beta^5$ to breaking the correctness of the underlying Π_{PKEBC}; and
- **S.6** $\mathbf{G}_{\mathbf{0}}^5$ and $\mathbf{G}_{\mathbf{1}}^5$ to breaking the $\{\mathsf{IND},\mathsf{IK}\}$-$\mathsf{CCA}$ security of the underlying Π_{PKEBC}.

The reductions corresponding to **S.2**, **S.3**, **S.5** and **S.6** are to PKEBC notions and therefore have access to all the DSS and MDVS secret keys; since only public keys are needed to generate PKEBC ciphertexts, and the reductions have access to these, they can handle the additional queries. (Note that, for the $\{\mathsf{IND},\mathsf{IK}\}$-$\mathsf{CCA}$ reduction, since the reduction generates the additional ciphertexts without relying on the $\mathcal{O}_E$ oracle provided by games, it can then still use the $\mathcal{O}_D$ oracle provided by the games to handle decryption queries even if the PKEBC component of such ciphertexts is left unchanged by the adversary).

The reduction corresponding to **S.4** is to MDVS message-bound validity, which does allow adversaries to query for MDVS secret keys of senders. Since the reduction is to an MDVS notion, it has access to all PKEBC and DSS secret keys;

as it also has access to the secret keys of senders, it can generate any ciphertexts for the additional queries too.[11]

The reduction corresponding to **S.1** is to the 1-sEUF-CMA security of Π_{DSS} and therefore has access to all the PKEBC and MDVS secret keys. Noting that the MDRS-PKE encryption algorithm samples a fresh Π_{DSS} key-pair for encryption, for these additional queries one can have the reduction simply sample the Π_{DSS} key-pair itself, and therefore can generate the ciphertext as intended. (Alternatively, one could rely on the 1-sEUF-CMA game of Π_{DSS} to sample the key-pairs, and then use the $\mathcal{O}_S$ oracle provided by the game to generate the signatures, but this is not necessary for this reduction.) □

C.4 Forgery Invalidity

Theorem 10. *If no adversary* $(\varepsilon_{\text{PKEBC}}, t_{\text{PKEBC}})$*-breaks the* $(n_{\text{PKEBC}}, d_{E\,\text{PKEBC}}, q_{E\,\text{PKEBC}}, q_{D\,\text{PKEBC}})$*-Correctness of* Π_{PKEBC} *and no adversary* $(\varepsilon_{\text{MDVS}}, t_{\text{MDVS}})$*-breaks the* $(n_S, n_V, d_S, d_F, q_S, q_V, q_F)$*-Forgery Invalidity of* Π_{MDVS} *then no adversary* (ε, t)*-breaks* $\Pi_{\text{MDRS-PKE}}$*'s*

$$
\begin{aligned}
(n_S &\coloneqq n_{S\,\text{MDVS}},\\
n_R &\coloneqq \min(n_{\text{PKEBC}}, n_{V\,\text{MDVS}}),\\
d_F &\coloneqq \min(d_{E\,\text{PKEBC}}, d_{S\,\text{MDVS}}),\\
q_F &\coloneqq \min(q_{E\,\text{PKEBC}}, q_{F\,\text{MDVS}}),\\
q_D &\coloneqq \min(q_{D\,\text{PKEBC}}, q_{V\,\text{MDVS}}))\text{-}\textit{Forgery Invalidity,}
\end{aligned}
$$

with $\varepsilon > \varepsilon_{\text{PKEBC}} + \varepsilon_{\text{MDVS}}$ *and* $t_{\text{PKEBC}}, t_{\text{MDVS}} \approx t + t_{\mathsf{Forge\text{-}Invalid}}$*, where* $t_{\mathsf{Forge\text{-}Invalid}}$ *is the time to run* $\Pi_{\text{MDRS-PKE}}$*'s* $\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$ *game.*

Proof. We prove this result via game hopping.

$\mathbf{G}^{\mathsf{Forge\text{-}Invalid}} \rightsquigarrow \mathbf{G}^1$: The only difference between games $\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$ and $\mathbf{G}^1$ is that in $\mathbf{G}^1$ some decryption queries are handled differently. More concretely, when $\mathcal{O}_D$ is queried on an input $(B_j, c \coloneqq (\mathtt{vk}, \sigma', c'))$ where c was output by a query $\mathcal{O}_{Forge}(A_i, \vec{V}, m, \mathcal{C})$ such that $B_j \in \text{Set}(\vec{V}) \setminus \mathcal{C}$, oracle $\mathcal{O}_D$ works as follows: let $(\mathtt{spk}_i, \vec{v}_{\text{MDVS}}, m, \sigma)$ be the plaintext that was encrypted by $\Pi_{\text{PKEBC}}.E$ under $\vec{v}_{\text{PKEBC}}$ (which resulted in ciphertext c'), where $\mathtt{spk}_i$ is A_i's public key,

$$
\begin{aligned}
\vec{v}_{\text{MDVS}} &\coloneqq (\mathtt{vpk}_{\text{MDVS}_1}, \ldots, \mathtt{vpk}_{\text{MDVS}_{|\vec{v}|}}), \text{ and}\\
\vec{v}_{\text{PKEBC}} &\coloneqq (\mathtt{pk}_{\text{PKEBC}_1}, \ldots, \mathtt{pk}_{\text{PKEBC}_{|\vec{v}|}})
\end{aligned}
$$

[11] We note that one can also adapt the analogous reduction to the unforgeability of Π_{MDVS} of [18, 19, Proof of Theorem 10]—which captures the setting where the secret keys of honest senders do not leak—to handle the additional encryption queries, because the MDVS unforgeability game provides a signing oracle which the reduction could use to generate the necessary MDVS signatures.

are, respectively, the vectors of public MDVS verifier keys and public PKEBC receiver keys corresponding to $\vec{V}$, and where

$$\sigma \leftarrow \Pi_{\text{MDVS}}.\mathit{Forge}_{\mathtt{pp}_{\text{MDVS}}}(\mathtt{spk}_{\text{MDVS}i}, \vec{v}_{\text{MDVS}}, (\vec{v}_{\text{PKEBC}}, m, \mathtt{vk}), \vec{s}_{\text{MDVS}}),$$

is a forged MDVS signature on $(\vec{v}_{\text{PKEBC}}, m, \mathtt{vk})$ using vector of secret keys $\vec{s}_{\text{MDVS}}$ (as defined by oracle $\mathcal{O}_{Forge}$), and $\mathtt{vk}$ being the DSS verification key in c; oracle $\mathcal{O}_D$ no longer decrypts c' using $\Pi_{\text{PKEBC}}.D$ with B_j's PKEBC secret key, and instead simply assumes decryption outputs $(\vec{v}_{\text{PKEBC}}, (\mathtt{spk}_i, \vec{v}_{\text{MDVS}}, m, \sigma))$.

It is easy to see that one can reduce distinguishing the two games to breaking the correctness of Π_{PKEBC}: since the reduction holds all secret keys, it can handle any oracle queries. If $\mathbf{A}$ only queries for at most $n_R \leq n_{\text{PKEBC}}$ different receivers, the sum of lengths of the vectors input to $\mathcal{O}_{Forge}$ is at most $d_F \leq d_{E\text{PKEBC}}$, and makes at most $q_F \leq q_{E\text{PKEBC}}$ and $q_D \leq q_{D\text{PKEBC}}$ queries to oracles $\mathcal{O}_{Forge}$ and $\mathcal{O}_D$, respectively, since by assumption no adversary $(t_{\text{PKEBC}}, \varepsilon_{\text{PKEBC}})$-breaks the

$$(n_{\text{PKEBC}}, d_{E\text{PKEBC}}, q_{E\text{PKEBC}}, q_{D\text{PKEBC}})\text{-Correctness}$$

of Π_{PKEBC}, it follows

$$\left|\Pr[\mathbf{AG}^1 = \mathtt{win}] - \Pr[\mathbf{AG}^{\mathsf{Forge\text{-}Invalid}} = \mathtt{win}]\right| \leq \varepsilon_{\text{PKEBC}}.$$

$\mathbf{G}^1 \rightsquigarrow \mathbf{G}^2$: Game $\mathbf{G}^2$ is just like $\mathbf{G}^1$, except that once again some decryption queries are handled differently. In contrast to the previous hop—where $\mathbf{G}^1$ differed from $\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$ in that it assumed the ciphertext c' in each ciphertext $c \coloneqq (\mathtt{vk}, \sigma', c')$ output by a query $\mathcal{O}_{Forge}(A_i, \vec{V}, m, \mathcal{C})$ decrypted correctly when $\mathcal{O}_D$ was queried on (B_j, c), with $B_j \in \text{Set}(\vec{V}) \setminus \mathcal{C}$—game $\mathbf{G}^2$ differs from $\mathbf{G}^1$ in that it now assumes that each MDVS signature σ generated by $\mathcal{O}_{Forge}$ using $\Pi_{\text{MDVS}}.\mathit{Forge}$ does not verify as being valid when $\mathcal{O}_D$ is queried on a matching input. To be more precise, for a query $\mathcal{O}_{Forge}(A_i, \vec{V}, m, \mathcal{C})$: let $(\vec{v}_{\text{PKEBC}}, m, \mathtt{vk})$ be the plaintext on which an MDVS signature was forged with respect to $\mathtt{spk}_i$ and $\vec{v}_{\text{MDVS}}$ using $\Pi_{\text{MDVS}}.\mathit{Forge}$, where $\mathtt{spk}_i$ is A_i's public key,

$$\begin{aligned}\vec{v}_{\text{MDVS}} &\coloneqq (\mathtt{vpk}_{\text{MDVS}1}, \ldots, \mathtt{vpk}_{\text{MDVS}|\vec{v}|}), \text{ and}\\ \vec{v}_{\text{PKEBC}} &\coloneqq (\mathtt{pk}_{\text{PKEBC}1}, \ldots, \mathtt{pk}_{\text{PKEBC}|\vec{v}|})\end{aligned}$$

are, respectively, the vectors of public MDVS verifier keys and public PKEBC receiver keys corresponding to $\vec{V}$; let σ be the resulting forged signature

$$\sigma \leftarrow \Pi_{\text{MDVS}}.\mathit{Forge}_{\mathtt{pp}_{\text{MDVS}}}(\mathtt{spk}_{\text{MDVS}i}, \vec{v}_{\text{MDVS}}, (\vec{v}_{\text{PKEBC}}, m, \mathtt{vk}), \vec{s}_{\text{MDVS}}),$$

where $\vec{s}_{\text{MDVS}}$ is as defined by $\mathcal{O}_{Forge}$; and let c be the ciphertext output by the $\mathcal{O}_{Forge}$ query. Then, when queried on input (B_j, c) such that $B_j \in \text{Set}(\vec{V}) \setminus \mathcal{C}$, $\mathcal{O}_D$ no longer verifies if σ is valid by running

$$\Pi_{\text{MDVS}}.\mathit{Vfy}(\mathtt{pp}, \mathtt{spk}_i, \mathtt{vsk}_j, \vec{v}_{\text{MDVS}}, (\vec{v}_{\text{PKEBC}}, m, \mathtt{vk}))$$

and instead simply assumes the MDVS signature verification outputs 0—implying $\mathcal{O}_D$ outputs $\perp$.

It is easy to see that one can reduce distinguishing $\mathbf{G}^1$ and $\mathbf{G}^2$ to breaking the Forgery Invalidity of Π_{MDVS}: since the reduction holds all secret keys, it can handle any oracle queries. If $\mathbf{A}$ only queries for at most $n_S \leq n_{S\text{MDVS}}$ different senders and $n_R \leq n_{V\text{MDVS}}$ different receivers, the sum of lengths of the vectors input to $\mathcal{O}_{Forge}$ is at most $d_F \leq d_{F\text{MDVS}}$, and makes at most $q_F \leq q_{F\text{MDVS}}$ and $q_D \leq q_{D\text{MDVS}}$ queries to oracles $\mathcal{O}_{Forge}$ and $\mathcal{O}_D$, respectively, since by assumption no adversary $(t_{\text{MDVS}}, \varepsilon_{\text{MDVS}})$-breaks Π_{MDVS}'s

$$(n_{S\text{MDVS}}, n_{V\text{MDVS}}, d_{S\text{MDVS}}, d_{F\text{MDVS}}, q_{S\text{MDVS}}, q_{V\text{MDVS}}, q_{F\text{MDVS}})\text{-}$$

Forgery Invalidity, it follows

$$\left|\Pr[\mathbf{AG}^2 = \texttt{win}] - \Pr[\mathbf{AG}^1 = \texttt{win}]\right| \leq \varepsilon_{\text{MDVS}}.$$

To conclude the proof note that no adversary can win $\mathbf{G}^2$, implying

$$\Pr[\mathbf{AG}^2 = \texttt{win}] = 0.$$

□

C.5 Public-Key Collision Resistance

Corollary 2. *If Π_{MDVS} is $(n_{\text{MDVS}}, \ell_{\text{MDVS}})$-Party ε-Public-Key Collision Resistant then $\Pi_{\text{MDRS-PKE}}$ is*

$$(n \coloneqq n_{\text{MDVS}}, \ell \coloneqq \ell_{\text{MDVS}})\text{-}\textit{Party}$$
$$\varepsilon\text{-}\textit{Public-Key Collision-Resistant.}$$

Proof. Follows from the definition of $\Pi_{\text{MDRS-PKE}}$ (Algorithm 12) and the assumption on Π_{MDVS}. □

D Application Semantics of MDRS-PKE Game Notions

As in Section 6, we consider a set of parties $\mathcal{F}$ consisting of all senders and receivers, i.e. $\mathcal{F} \coloneqq \mathcal{S} \cup \mathcal{R}$. The theorems below establish composable semantics for the MDRS-PKE game-based notions we introduced in Section 4 together with the ones from [4,18].

Theorem 11. *Consider simulator* sim *defined in Algorithms 13 and 16, reductions* $\mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}$, $\mathbf{C}^{\mathsf{Cons}\text{-}H}$, $\mathbf{C}^{\mathsf{Cons}}$, $\mathbf{C}^{\mathsf{Corr}}$, $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}$, $\mathbf{C}^{\mathsf{R\text{-}Unforg}}$, $\mathbf{C}^{\mathsf{CCA}}$ *and* $\mathbf{C}^{\mathsf{OTR}}$ *(defined, respectively, in Algorithms 14, 15 and 17, Algorithms 14, 15 and 18, Algorithms 14, 15 and 19, Algorithms 14, 15 and 20, Algorithms 14, 15 and 21, Algorithms 14, 15 and 22, Algorithms 14, 15 and 23, and Algorithms 14, 15 and 24), and reductions* $\perp^J \cdot \mathbf{C}^{\mathsf{OTR}\text{-}\xi_1}$, $\perp^J \cdot \mathbf{C}^{\mathsf{Corr}\text{-}\xi_1}$, $\perp^J \cdot \mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1}$, $\perp^J \cdot \mathbf{C}^{\mathsf{OTR}\text{-}\xi_2}$,

$\perp^J \cdot \mathbf{C}^{\mathsf{Cons}\text{-}0\text{-}\xi_2}$, $\perp^J \cdot \mathbf{C}^{\mathsf{Cons}\text{-}1\text{-}\xi_2}$, $\perp^J \cdot \mathbf{C}^{\mathsf{Corr}\text{-}\xi_2}$ *and* $\perp^J \cdot \mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2}$ *(where* $\mathbf{C}^{\mathsf{OTR}\text{-}\xi_1}$, $\mathbf{C}^{\mathsf{Corr}\text{-}\xi_1}$, $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1}$, $\mathbf{C}^{\mathsf{OTR}\text{-}\xi_2}$, $\mathbf{C}^{\mathsf{Cons}\text{-}0\text{-}\xi_2}$, $\mathbf{C}^{\mathsf{Cons}\text{-}1\text{-}\xi_2}$, $\mathbf{C}^{\mathsf{Corr}\text{-}\xi_2}$ *and* $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2}$ *are defined, respectively, in Algorithms 14 and 26, Algorithms 14 and 27, Algorithms 14 and 28, Algorithms 14 and 29, Algorithms 14 and 30, Algorithms 14 and 31, Algorithms 14 and 32, and Algorithms 14 and 33). If the* MDRS-PKE *scheme is* (m, n)*-Party* ε*-Public Key Collision Resistant, then for any distinguisher* $\mathbf{D}$*,*

$$
\begin{aligned}
&\Delta^{\mathbf{D}}\Bigg(\mathsf{Snd}^{\mathcal{S}^H}\,\mathsf{Rcv}^{\mathcal{R}^H}\,\mathsf{Forge}^{\mathcal{F}}\perp^J[\mathbf{KGA}, \mathsf{Net}\cdot\mathbf{INS}],\\
&\quad \mathsf{sim}^{\overline{\mathcal{P}^H}}\cdot\mathsf{ConfAnon}^{\overline{\mathcal{P}^H}}\cdot\mathsf{Otr}^{\overline{\mathcal{P}^H}}\cdot\\
&\qquad\left[\begin{array}{l}\mathsf{Net}\cdot\perp^{\mathsf{Auth\text{-}Intf}}\cdot\left[\langle A_i\to\vec{V}\rangle^{\{A_i\}\cup\overline{\mathcal{P}^H}}_{Set(\vec{V})\cup\overline{\mathcal{P}^H}}\right]_{\substack{A_i\in\mathcal{S}\\ \vec{V}\in\mathcal{R}^+}}\\ \qquad\left[\langle[Forge]A_i\to\vec{V}\rangle^{\mathcal{F}}_{\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+}\end{array}\right)\\
&\quad\leq\varepsilon+4\cdot\Big(Adv^{\mathsf{OTR}}(\mathbf{D}\perp^J\mathbf{C}^{\mathsf{OTR}\text{-}\xi_1})+Adv^{\mathsf{Corr}}(\mathbf{D}\perp^J\mathbf{C}^{\mathsf{Corr}\text{-}\xi_1})\\
&\qquad\quad+Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{D}\perp^J\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1})\Big)\\
&\qquad+Adv^{\mathsf{OTR}}(\mathbf{D}\perp^J\mathbf{C}^{\mathsf{OTR}\text{-}\xi_2})+Adv^{\mathsf{Cons}}(\mathbf{D}\perp^J\mathbf{C}^{\mathsf{Cons}\text{-}0\text{-}\xi_2})\\
&\qquad+Adv^{\mathsf{Cons}}(\mathbf{D}\perp^J\mathbf{C}^{\mathsf{Cons}\text{-}1\text{-}\xi_2})+Adv^{\mathsf{Corr}}(\mathbf{D}\perp^J\mathbf{C}^{\mathsf{Corr}\text{-}\xi_2})\\
&\qquad+Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{D}\perp^J\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2})+Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons}})\\
&\qquad+Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}})+Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}})+Adv^{\mathsf{R\text{-}Unforg}}(\mathbf{DC}^{\mathsf{R\text{-}Unforg}})\\
&\qquad+Adv^{\{\mathsf{IND},\,\mathsf{IK}\}\text{-}\mathsf{CCA}}(\mathbf{DC}^{\mathsf{CCA}})+Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}}).
\end{aligned}
$$

Theorem 12. *Consider simulator* sim *defined in Algorithms 13 and 25, reductions* $\mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}$, $\mathbf{C}^{\mathsf{Cons}\text{-}H}$, $\mathbf{C}^{\mathsf{Cons}}$, $\mathbf{C}^{\mathsf{Corr}}$, $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}$, $\mathbf{C}^{\mathsf{CCA}}$ *and* $\mathbf{C}^{\mathsf{OTR}}$ *(analogous to Theorem 12's reductions), and reductions* $\mathbf{C}^{\mathsf{OTR}\text{-}\xi_1}$, $\mathbf{C}^{\mathsf{Corr}\text{-}\xi_1}$, $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1}$, $\mathbf{C}^{\mathsf{OTR}\text{-}\xi_2}$, $\mathbf{C}^{\mathsf{Cons}\text{-}0\text{-}\xi_2}$, $\mathbf{C}^{\mathsf{Cons}\text{-}1\text{-}\xi_2}$, $\mathbf{C}^{\mathsf{Corr}\text{-}\xi_2}$ *and* $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2}$ *(i.e. the reductions from Lemmata 2 and 3). If the* MDRS-PKE *scheme is* (m, n)*-Party* ε*-Public Key*

Collision Resistant, then for any distinguisher $\mathbf{D}$*,*

$$\Delta^{\mathbf{D}}(\mathsf{Snd}^{\mathcal{S}^H}\mathsf{Rcv}^{\mathcal{R}^H}\mathsf{Forge}^{\mathcal{F}}[\mathbf{KGA}, \mathsf{Net}\cdot\mathbf{INS}],$$
$$\mathsf{sim}^{\overline{\mathcal{P}^H}}\cdot\ \mathsf{ConfAnon}^{\overline{\mathcal{P}^H}}\cdot\ \mathsf{Otr}^{\overline{\mathcal{P}^H}}\cdot\ \begin{bmatrix}\mathsf{Net}\ \cdot\ \left[\langle A_i \to \vec{V}\rangle^{\overline{\mathcal{P}^H}}_{Set(\vec{V})\cup\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+}\\ \left[\langle[Forge]A_i \to \vec{V}\rangle^{\mathcal{F}}_{\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+}\end{bmatrix})$$
$$\leq \varepsilon + 4\cdot\Big(Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}) + Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}\text{-}\xi_1})$$
$$+ Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1})\Big)$$
$$+ Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_2}) + Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons}\text{-}0\text{-}\xi_2}) + Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons}\text{-}1\text{-}\xi_2})$$
$$+ Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}\text{-}\xi_2}) + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2})$$
$$+ Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons}}) + Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}}) + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}})$$
$$+ Adv^{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}}(\mathbf{DC}^{\mathsf{CCA}}) + Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}}).$$

D.1 Proofs

For simplicity, in Algorithm 13 we describe the behavior of the simulators we will consider in the proofs of Theorems 11 and 12 for the (sub-)interfaces of dishonest parties that correspond to an interface of the **KGA** resource in the real world system. Similarly, in Algorithm 14 we describe the behavior of the reductions we will consider in these proofs for the same (sub-)interfaces.

D.1.1 Helper Claims Below we state two useful results that help in simplifying the proofs of Theorems 11 and 12. (See Sections D.1.4 and D.1.5 for their proofs.) Consider the following events:

Event ξ_1 There are two WRITE queries at the interface of an honest party $A_i \in \mathcal{S}^H$ that output $\mathtt{id}$ and $\mathtt{id}'$ with $\mathtt{id} \neq \mathtt{id}'$, such that the contents of the registers with these identifiers (i.e. $\mathtt{id}$ and $\mathtt{id}'$) are the same.

Event ξ_2 There is a WRITE query at a dishonest party's interface with input ciphertext c that outputs a register identifier $\mathtt{id}$ and there is a later WRITE query at the interface of an honest party $A_i \in \mathcal{S}^H$ that outputs a register identifier $\mathtt{id}'$ such that the contents of the registers with identifiers $\mathtt{id}$ and $\mathtt{id}'$ are the same.

At a high level, Lemmata 2 and 3 bound the probability of events ξ_1 and ξ_2 to occur. The reason why these results are necessary is that in the real world duplicate ciphertexts are filtered out (to protect against replay attacks), and so if either event would occur one would be able to distinguish the real and ideal worlds. In the following $\mathbf{R}$ is defined as in Section 6.1, i.e.

$$\mathbf{R} \coloneqq \mathsf{Snd}^{\mathcal{S}^H}\mathsf{Rcv}^{\mathcal{R}^H}\mathsf{Forge}^{(\mathcal{F}\times\{\mathsf{Forge}\})}[\mathbf{KGA}, \mathsf{Net}\cdot\mathbf{INS}]. \tag{D.1}$$

Lemma 2. *For any distinguisher* $\mathbf{D}$*, the probability that event* ξ_1 *occurs when it interacts with the real world system* $\mathbf{R}$ *(Equation D.1) is upper bounded by*

$$4 \cdot \Big(Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}) + Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}\text{-}\xi_1}) + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1}) \Big).$$

where $\mathbf{C}^{\mathsf{OTR}\text{-}\xi_1}$*,* $\mathbf{C}^{\mathsf{Corr}\text{-}\xi_1}$ *and* $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1}$ *are the reductions given in Algorithms 14 and 26, Algorithms 14 and 27, and Algorithms 14 and 28.*

A proof of Lemma 2 is given in Section D.1.4.

Lemma 3. *For any distinguisher* $\mathbf{D}$*, the probability that event* ξ_2 *occurs when it interacts with the real world system* $\mathbf{R}$ *(Equation D.1) is upper bounded by*

$$Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_2}) + Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons}\text{-}0\text{-}\xi_2}) + Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons}\text{-}1\text{-}\xi_2}) + Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}\text{-}\xi_2}) + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2}).$$

where $\mathbf{C}^{\mathsf{OTR}\text{-}\xi_2}$*,* $\mathbf{C}^{\mathsf{Cons}\text{-}0\text{-}\xi_2}$*,* $\mathbf{C}^{\mathsf{Cons}\text{-}1\text{-}\xi_2}$*,* $\mathbf{C}^{\mathsf{Corr}\text{-}\xi_2}$ *and* $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2}$ *are the reductions given in Algorithms 14 and 29, Algorithms 14 and 30, Algorithms 14 and 31, Algorithms 14 and 32, and Algorithms 14 and 33.*

A proof of Lemma 3 is given in Section D.1.5.

Remark 4. Lemmata 2 and 3 rely on the Forgery Invalidity of the MDRS-PKE scheme. One can alternatively rely on the Unforgeability against Replays if the secret keys of honest senders do not leak.

D.1.2 Proof of Theorem 11

Algorithm 11 MDVS construction $\Pi^{\mathsf{adap}}_{\mathrm{MDVS}}$ from [4]. The building blocks are a PKE scheme $\Pi_{\mathrm{PKE}} = (G, E, D)$, a One Way Function $\Pi_{\mathrm{OWF}} = (S, F)$, and a Non Interactive Zero Knowledge scheme $\Pi_{\mathrm{NIZK}} = (G, P, V, S \coloneqq (S_G, S_P))$.

$S(1^k)$
  $(\mathtt{pk}, \mathtt{sk}) \leftarrow \Pi_{\mathrm{PKE}}.G(1^k)$
  return $\mathtt{pp} \coloneqq (1^k, \mathtt{crs} \leftarrow \Pi_{\mathrm{NIZK}}.G(1^k), \mathtt{pk})$

$G_S(\mathtt{pp})$
  $(x_0, x_1) \leftarrow (\Pi_{\mathrm{OWF}}.S(1^k), \Pi_{\mathrm{OWF}}.S(1^k))$
  $(y_0, y_1) \leftarrow (\Pi_{\mathrm{OWF}}.F(x_0), \Pi_{\mathrm{OWF}}.F(x_1))$
  $b \leftarrow RandomCoin$
  return $(\mathtt{spk} \coloneqq (y_0, y_1), \mathtt{ssk} \coloneqq (\mathtt{spk}, x \coloneqq x_b))$

$G_V(\mathtt{pp})$
  $((\mathtt{pk}_0, \mathtt{sk}_0), (\mathtt{pk}_1, \mathtt{sk}_1)) \leftarrow (\Pi_{\mathrm{PKE}}.G(1^k), \Pi_{\mathrm{PKE}}.G(1^k))$
  $(x_0, x_1) \leftarrow (\Pi_{\mathrm{OWF}}.S(1^k), \Pi_{\mathrm{OWF}}.S(1^k))$
  $(y_0, y_1) \leftarrow (\Pi_{\mathrm{OWF}}.F(x_0), \Pi_{\mathrm{OWF}}.F(x_1))$
  $b \leftarrow RandomCoin$
  return $(\mathtt{vpk} \coloneqq (\mathtt{pk}_0, y_0, \mathtt{pk}_1, y_1), \mathtt{vsk} \coloneqq (\mathtt{vpk}, b, \mathtt{sk} \coloneqq \mathtt{sk}_b, x \coloneqq x_b))$

$Sig_{\mathtt{pp}}(\mathtt{ssk}, \vec{v} \coloneqq (\mathtt{vpk}_1, \ldots, \mathtt{vpk}_{|\vec{v}|}), m)$
  for $i \in \{1, \ldots, |\vec{v}|\}$ **:**
    $(c_{i,0}, c_{i,1}) \leftarrow (\Pi_{\mathrm{PKE}}.E_{\mathtt{vpk}_i.\mathtt{pk}_0}(1; r_{i,0}), \Pi_{\mathrm{PKE}}.E_{\mathtt{vpk}_i.\mathtt{pk}_1}(1; r_{i,1}))$
  $(\vec{c}, \vec{r}) \leftarrow (((c_{1,0}, c_{1,1}), \ldots, (c_{|\vec{v}|,0}, c_{|\vec{v}|,1})), ((r_{1,0}, r_{1,1}), \ldots, (r_{|\vec{v}|,0}, r_{|\vec{v}|,1})))$
  $\vec{\alpha} \leftarrow (\alpha_1 \coloneqq (1, \mathtt{ssk}.x), \ldots, \alpha_{|\vec{v}|} \coloneqq (1, \mathtt{ssk}.x))$
  $c_{\mathtt{pp}} \leftarrow \Pi_{\mathrm{PKE}}.E_{\mathtt{pp}.\mathtt{pk}}((m, 1, \vec{\alpha}); r_{\mathtt{pp}})$
  $p \leftarrow \Pi_{\mathrm{NIZK}}.P_{\mathtt{crs}}\big((\mathtt{pp}.\mathtt{pk}, \mathtt{spk}, \vec{v}, m, \vec{c}, c_{\mathtt{pp}}) \in L_{\mathrm{MDVS}^{\mathsf{adap}}}, (\vec{\alpha}, \vec{r}, r_{\mathtt{pp}}, 1)\big)$
  return $\sigma \coloneqq (p, \vec{c}, c_{\mathtt{pp}})$

$Vfy_{\mathtt{pp}}(\mathtt{spk}, \mathtt{vsk}, \vec{v}, m, \sigma \coloneqq (p, \vec{c}, c_{\mathtt{pp}}))$
  if $\Pi_{\mathrm{NIZK}}.V_{\mathtt{crs}}\big((\mathtt{pp}.\mathtt{pk}, \mathtt{spk}, \vec{v}, m, \vec{c}, c_{\mathtt{pp}}) \in L_{\mathrm{MDVS}^{\mathsf{adap}}}, p\big) = 1$ **:**
    for $i = 1, \ldots, |\vec{v}|$ **do**
      if $\mathtt{vsk}.\mathtt{vpk} = v_i$ **:**
        return $\Pi_{\mathrm{PKE}}.D_{\mathtt{vsk}.\mathtt{sk}}(c_{i,\mathtt{vsk}.b})$
  return 0

$Forge_{\mathtt{pp}}(\mathtt{spk}, \vec{v} \coloneqq (\mathtt{vpk}_1, \ldots, \mathtt{vpk}_{|\vec{v}|}), m, \vec{s} \coloneqq (\mathtt{vsk}_1, \ldots, \mathtt{vsk}_{|\vec{s}|}))$ // Assumed: $|\vec{v}| = |\vec{s}|$
  for $i \in \{1, \ldots, |\vec{v}|\}$ **:**
    if $s_i \neq \perp$ **:**
      $(c_{i,0}, c_{i,1}) \leftarrow (\Pi_{\mathrm{PKE}}.E_{\mathtt{vpk}_i.\mathtt{pk}_0}(1; r_{i,0}), \Pi_{\mathrm{PKE}}.E_{\mathtt{vpk}_i.\mathtt{pk}_1}(1; r_{i,1}))$
      $\alpha_i \coloneqq (1, \mathtt{vsk}_i.x)$
    else
      $(c_{i,0}, c_{i,1}) \leftarrow (\Pi_{\mathrm{PKE}}.E_{\mathtt{vpk}_i.\mathtt{pk}_0}(0; r_{i,0}), \Pi_{\mathrm{PKE}}.E_{\mathtt{vpk}_i.\mathtt{pk}_1}(0; r_{i,1}))$
      $\alpha_i \coloneqq (0, 0)$
  $(\vec{c}, \vec{r}) \leftarrow (((c_{1,0}, c_{1,1}), \ldots, (c_{|\vec{v}|,0}, c_{|\vec{v}|,1})), ((r_{1,0}, r_{1,1}), \ldots, (r_{|\vec{v}|,0}, r_{|\vec{v}|,1})))$
  $\vec{\alpha} \leftarrow (\alpha_1, \ldots, \alpha_{|\vec{v}|})$
  $c_{\mathtt{pp}} \leftarrow \Pi_{\mathrm{PKE}}.E_{\mathtt{pp}.\mathtt{pk}}((m, 0, \vec{\alpha}); r_{\mathtt{pp}})$
  $p \leftarrow \Pi_{\mathrm{NIZK}}.P_{\mathtt{crs}}\big((\mathtt{pp}.\mathtt{pk}, \mathtt{spk}, \vec{v}, m, \vec{c}, c_{\mathtt{pp}}) \in L_{\mathrm{MDVS}^{\mathsf{adap}}}, (\vec{\alpha}, \vec{r}, r_{\mathtt{pp}}, 0)\big)$
  return $\sigma \coloneqq (p, \vec{c}, c_{\mathtt{pp}})$

Algorithm 12 MDRS-PKE construction $\Pi_{\text{MDRS-PKE}}$ from [18]. The building blocks are a PKEBC scheme $\Pi_{\text{PKEBC}} = (S, G, E, D)$, an MDVS scheme $\Pi_{\text{MDVS}} = (S, G_S, G_V, Sig, Vfy, Forge)$ and a DSS $\Pi_{\text{DSS}} = (G, Sig, Vfy)$.

$S(1^k)$
  $\mathtt{pp}_{\text{MDVS}} \leftarrow \Pi_{\text{MDVS}}.S(1^k)$
  $\mathtt{pp}_{\text{PKEBC}} \leftarrow \Pi_{\text{PKEBC}}.S(1^k)$
  return $\mathtt{pp} \coloneqq (\mathtt{pp}_{\text{MDVS}}, \mathtt{pp}_{\text{PKEBC}}, 1^k)$

$G_S(\mathtt{pp})$
  $(\mathtt{spk}_{\text{MDVS}}, \mathtt{ssk}_{\text{MDVS}}) \leftarrow \Pi_{\text{MDVS}}.G_S(\mathtt{pp}_{\text{MDVS}})$
  return $(\mathtt{spk} \coloneqq \mathtt{spk}_{\text{MDVS}}, \mathtt{ssk} \coloneqq (\mathtt{spk}, \mathtt{ssk}_{\text{MDVS}}))$

$G_R(\mathtt{pp})$
  $(\mathtt{vpk}_{\text{MDVS}}, \mathtt{vsk}_{\text{MDVS}}) \leftarrow \Pi_{\text{MDVS}}.G_V(\mathtt{pp}_{\text{MDVS}})$
  $(\mathtt{pk}_{\text{PKEBC}}, \mathtt{sk}_{\text{PKEBC}}) \leftarrow \Pi_{\text{PKEBC}}.G(\mathtt{pp}_{\text{PKEBC}})$
  return $(\mathtt{rpk} \coloneqq (\mathtt{vpk}_{\text{MDVS}}, \mathtt{pk}_{\text{PKEBC}}), \mathtt{rsk} \coloneqq (\mathtt{rpk}, (\mathtt{vsk}_{\text{MDVS}}, \mathtt{sk}_{\text{PKEBC}})))$

$E_{\mathtt{pp}}(\mathtt{ssk}, \vec{v} \coloneqq (\mathtt{rpk}_1, \ldots, \mathtt{rpk}_{|\vec{v}|}), m)$
  $\vec{v}_{\text{PKEBC}} \coloneqq (\mathtt{rpk}_1.\mathtt{pk}_{\text{PKEBC}}, \ldots, \mathtt{rpk}_{|\vec{v}|}.\mathtt{pk}_{\text{PKEBC}})$
  $\vec{v}_{\text{MDVS}} \coloneqq (\mathtt{rpk}_1.\mathtt{vpk}_{\text{MDVS}}, \ldots, \mathtt{rpk}_{|\vec{v}|}.\mathtt{vpk}_{\text{MDVS}})$
  $(\mathtt{vk}, \mathtt{sk}) \leftarrow \Pi_{\text{DSS}}.G(\mathtt{pp}.1^k)$
  $\sigma \leftarrow \Pi_{\text{MDVS}}.Sig_{\mathtt{pp}_{\text{MDVS}}}(\mathtt{ssk}_{\text{MDVS}}, \vec{v}_{\text{MDVS}}, (\vec{v}_{\text{PKEBC}}, m, \mathtt{vk}))$
  $c \leftarrow \Pi_{\text{PKEBC}}.E_{\mathtt{pp}_{\text{PKEBC}}}(\vec{v}_{\text{PKEBC}}, (\mathtt{spk}_{\text{MDVS}}, \vec{v}_{\text{MDVS}}, m, \sigma))$
  $\sigma' \leftarrow \Pi_{\text{DSS}}.Sig_{\mathtt{sk}}(c)$
  return $(\mathtt{vk}, \sigma', c)$

$D_{\mathtt{pp}}(\mathtt{rsk}, c \coloneqq (\mathtt{vk}, \sigma', c'))$
  if $\Pi_{\text{DSS}}.Vfy_{\mathtt{vk}}(c', \sigma') = 0$ **:**
    return $\perp$
  $(\vec{v}_{\text{PKEBC}}, (\mathtt{spk} \coloneqq \mathtt{spk}_{\text{MDVS}}, \vec{v}_{\text{MDVS}}, m, \sigma)) \leftarrow \Pi_{\text{PKEBC}}.D_{\mathtt{pp}_{\text{PKEBC}}}(\mathtt{rsk}.\mathtt{sk}_{\text{PKEBC}}, c')$
  if $(\vec{v}_{\text{PKEBC}}, (\mathtt{spk}, \vec{v}_{\text{MDVS}}, m, \sigma)) = \perp \ \vee \ |\vec{v}_{\text{PKEBC}}| \neq |\vec{v}_{\text{MDVS}}|$ **:**
    return $\perp$
  $\vec{v} \coloneqq ((v_{\text{MDVS}1}, v_{\text{PKEBC}1}), \ldots, (v_{\text{MDVS}|\vec{v}_{\text{PKEBC}}|}, v_{\text{PKEBC}|\vec{v}_{\text{PKEBC}}|}))$
  if $\mathtt{rsk}.\mathtt{rpk} \notin \vec{v}$ **:**
    return $\perp$
  if $\Pi_{\text{MDVS}}.Vfy_{\mathtt{pp}_{\text{MDVS}}}(\mathtt{spk}, \mathtt{vsk}_{\text{MDVS}}, \vec{v}_{\text{MDVS}}, (\vec{v}_{\text{PKEBC}}, m, \mathtt{vk}), \sigma) \neq \mathtt{valid}$ **:**
    return $\perp$
  return $(\mathtt{spk}, \vec{v}, m)$

$Forge_{\mathtt{pp}}(\mathtt{spk}, \vec{v} \coloneqq (\mathtt{rpk}_1, \ldots, \mathtt{rpk}_{|\vec{v}|}), m, \vec{s} \coloneqq (\mathtt{rsk}_1, \ldots, \mathtt{rsk}_{|\vec{s}|}))$
  $\vec{v}_{\text{PKEBC}} \coloneqq (\mathtt{rpk}_1.\mathtt{pk}_{\text{PKEBC}}, \ldots, \mathtt{rpk}_{|\vec{v}|}.\mathtt{pk}_{\text{PKEBC}})$
  $\vec{v}_{\text{MDVS}} \coloneqq (\mathtt{rpk}_1.\mathtt{vpk}_{\text{MDVS}}, \ldots, \mathtt{rpk}_{|\vec{v}|}.\mathtt{vpk}_{\text{MDVS}})$
  $\vec{s}_{\text{MDVS}} \coloneqq (\mathtt{rsk}_1.\mathtt{vsk}_{\text{MDVS}}, \ldots, \mathtt{rsk}_{|\vec{s}|}.\mathtt{vsk}_{\text{MDVS}})$
  $(\mathtt{vk}, \mathtt{sk}) \leftarrow \Pi_{\text{DSS}}.G(\mathtt{pp}.1^k)$
  $\sigma \leftarrow \Pi_{\text{MDVS}}.Forge_{\mathtt{pp}_{\text{MDVS}}}(\mathtt{spk}_{\text{MDVS}}, \vec{v}_{\text{MDVS}}, (\vec{v}_{\text{PKEBC}}, m, \mathtt{vk}), \vec{s}_{\text{MDVS}})$
  $c \leftarrow \Pi_{\text{PKEBC}}.E_{\mathtt{pp}_{\text{PKEBC}}}(\vec{v}_{\text{PKEBC}}, (\mathtt{spk}_{\text{MDVS}}, \vec{v}_{\text{MDVS}}, m, \sigma))$
  $\sigma' \leftarrow \Pi_{\text{DSS}}.Sig_{\mathtt{sk}}(c)$
  return $(\mathtt{vk}, \sigma', c)$

Algorithm 13 Description of (part of) the simulators considered in the proofs of Theorems 11 and 12 for the (sub-)interfaces of (dishonest) parties that correspond to an interface of the **KGA** resource in the real world system. In the following, $k \in \mathbb{N}$ is the (implicitly defined) security parameter.

◇ Initialization
INS-Initialization
(Ctxts,CtxtSet) $\leftarrow (\emptyset, \emptyset)$
$\mathtt{pp} \leftarrow \Pi.S(1^k)$
$(\mathtt{rpk}_{\mathtt{pp}}, \mathtt{rsk}_{\mathtt{pp}}) \leftarrow \Pi.G_R(\mathtt{pp})$
for $A_i \in \mathcal{S}$ **:** $(\mathtt{spk}_i, \mathtt{ssk}_i) \leftarrow \Pi.G_S(\mathtt{pp})$
for $B_j \in \mathcal{R}$ **:** $(\mathtt{rpk}_j, \mathtt{rsk}_j) \leftarrow \Pi.G_R(\mathtt{pp})$
if $\left|\{\mathtt{spk}_i\}_{A_i \in \mathcal{S}} \cup \{\mathtt{rpk}_j\}_{B_j \in \mathcal{R}}\right| \neq |\mathcal{S}| + |\mathcal{R}|$ **:**
  Abort

◇ GetLabel($\mathtt{spk}, \vec{v}'$) // Local procedure.
$\mathcal{S}_{\mathtt{spk}} \coloneqq \{A_i \mid \mathtt{spk} = \mathtt{spk}_i\}$
if $|\mathcal{S}_{\mathtt{spk}}| \neq 1 \vee {v_1}' \neq \mathtt{rpk}_{\mathtt{pp}}$ **: return** $\perp$
for $l \in \{2, \ldots, |\vec{v}'|\}$ **:**
  $\mathcal{R}_{{v_l}'} \coloneqq \{B_k \mid {v_l}' = \mathtt{rpk}_k\}$
  if $|\mathcal{R}_{{v_l}'}| \neq 1$ **: return** $\perp$
  else
    Let B_k be the element of $\mathcal{R}_{{v_l}'}$
    $V_{l-1} = B_k$
Let A_i be the element of $\mathcal{S}_{\mathtt{spk}}$
Let $\vec{V} \coloneqq (V_1, \ldots, V_{|\vec{v}'|-1})$
return $\langle A_i \rightarrow \vec{V} \rangle$

▷ $(P \in \overline{\mathcal{P}^H})$-PublicParameters
Output($\mathtt{pp}, \mathtt{rpk}_{\mathtt{pp}}$)

▷ $(P \in \overline{\mathcal{P}^H})$-SenderKeyPair($A_i \in \overline{\mathcal{S}^H}$)
Output($\mathtt{spk}_i, \mathtt{ssk}_i$)

▷ $(P \in \overline{\mathcal{P}^H})$-SenderPublicKey($A_i \in \mathcal{S}$)
Output($\mathtt{spk}_i$)

▷ $(P \in \overline{\mathcal{P}^H})$-ReceiverKeyPair($B_j \in \overline{\mathcal{R}^H}$)
Output($\mathtt{rpk}_j, \mathtt{rsk}_j$)

▷ $(P \in \overline{\mathcal{P}^H})$-ReceiverPublicKey($B_j \in \mathcal{R}$)
Output($\mathtt{rpk}_j$)

Algorithm 14 Description of the reductions considered in the proofs of Lemmata 2 and 3 and Theorems 11 and 12 for the (sub-)interfaces of (dishonest) parties that correspond to **KGA** interface in the real world system, plus the DELIVER interface.

◇ INITIALIZATION
INS-INITIALIZATION
CtxtDec $\leftarrow \emptyset$
Dec $\leftarrow \emptyset$ // Used in reductions for Lemmata 2 and 3.
(CtxtHon,CtxtForge,CtxtDis) $\leftarrow (\emptyset, \emptyset, \emptyset)$
$(\mathtt{pp}, \mathtt{rpk}_{\mathtt{pp}}) \leftarrow (\mathcal{O}_{PP}, \mathcal{O}_{RPK}(B_{\mathtt{pp}}))$
for $A_i \in \mathcal{S}$ **:** $\mathcal{O}_{SPK}(A_i)$
for $B_j \in \mathcal{R}$ **:** $\mathcal{O}_{RPK}(B_j)$
for $B_j \in \mathcal{R}$ **:** Received$[B_j] \leftarrow \emptyset$
if $\left|\{\mathcal{O}_{SPK}(A_i)\}_{A_i \in \mathcal{S}} \cup \{\mathcal{O}_{RPK}(B_j)\}_{B_j \in \mathcal{R}}\right| \neq |\mathcal{S}| + |\mathcal{R}|$ **:**
  Abort

◇ GETLABEL$(\mathtt{spk}, \vec{v}')$ // Local procedure.
$\mathcal{S}_{\mathtt{spk}} \coloneqq \{A_i \mid \mathtt{spk} = \mathtt{spk}_i\}$
if $|\mathcal{S}_{\mathtt{spk}}| \neq 1 \vee {v_1}' \neq \mathtt{rpk}_{\mathtt{pp}}$ **: return** $\perp$
for $l \in \{2, \ldots, |\vec{v}'|\}$ **:**
  $\mathcal{R}_{{v_l}'} \coloneqq \{B_k \mid {v_l}' = \mathtt{rpk}_k\}$
  if $|\mathcal{R}_{{v_l}'}| \neq 1$ **: return** $\perp$
  else
    Let B_k be the element of $\mathcal{R}_{{v_l}'}$
    $V_{l-1} = B_k$
Let A_i be the element of $\mathcal{S}_{\mathtt{spk}}$
Let $\vec{V} \coloneqq (V_1, \ldots, V_{|\vec{v}'|-1})$
return $\langle A_i \rightarrow \vec{V} \rangle$

▷ $(P \in \overline{\mathcal{P}^H})$-PUBLICPARAMETERS
OUTPUT$(\mathtt{pp}, \mathtt{rpk}_{\mathtt{pp}})$

▷ $(P \in \overline{\mathcal{P}^H})$-SENDERKEYPAIR$(A_i \in \overline{\mathcal{S}^H})$
OUTPUT$(\mathcal{O}_{SK}(A_i))$

▷ $(P \in \overline{\mathcal{P}^H})$-SENDERPUBLICKEY$(A_i \in \mathcal{S})$
OUTPUT$(\mathcal{O}_{SPK}(A_i))$

▷ $(P \in \overline{\mathcal{P}^H})$-RECEIVERKEYPAIR$(B_j \in \overline{\mathcal{R}^H})$
OUTPUT$(\mathcal{O}_{RK}(B_j))$

▷ $(P \in \overline{\mathcal{P}^H})$-RECEIVERPUBLICKEY$(B_j \in \mathcal{R})$
OUTPUT$(\mathcal{O}_{RPK}(B_j))$

▷ DELIVER$(P, \mathtt{id})$
Received$[P] \leftarrow$ Received$[P] \cup \{\mathtt{id}\}$

Algorithm 15 Helper functions used in the reductions considered in the proofs of Lemmata 2 and 3 and Theorems 11 and 12.

◇ DECRYPTION(B, c) // Local procedure.
$(\mathtt{spk}, \vec{v}', m) \leftarrow \mathcal{O}_D(B, c)$
if $(\mathtt{spk}, \vec{v}', m) \neq \perp$ **:**
  $\langle A_i \rightarrow \vec{V} \rangle \leftarrow$ GETLABEL$(\mathtt{spk}, \vec{v}')$
  if $\langle A_i \rightarrow \vec{V} \rangle \neq \perp$ **:**
    return $(\langle A_i \rightarrow \vec{V} \rangle, m)$
return $\perp$

◇ GETDELIVERED(P, list) // Local procedure.
filteredList $\leftarrow \emptyset$
for $(\mathtt{id}, x) \in$ list **with** $\mathtt{id} \in$ Received$[P]$ **:**
  filteredList $\leftarrow$ filteredList $\cup \{(\mathtt{id}, x)\}$
return filteredList

◇ FORGE$(A_i, \vec{V}, m, \mathcal{C})$ // Local procedure.
if $\vec{V} \in (\mathcal{R}^H)^+$ **:**
  return $\Pi.Forge_{\mathtt{pp}}(\mathtt{spk}_1, {\mathtt{rpk}_{\mathtt{pp}}}^{|\vec{V}|+1}, 0^{|m|}, \perp^{|\vec{V}|+1})$
$\vec{v}' \coloneqq (\mathtt{rpk}_{\mathtt{pp}}, v_1, \ldots, v_{|\vec{V}|})$
$\vec{s} \coloneqq (\perp \quad, \mathtt{rsk}_1, \ldots, \mathtt{rsk}_{|\vec{V}|})$ // For each V_l: if $V_l \in \mathcal{C}$, s_{l+1} is V_l's secret key; else it is $\perp$.
return $\Pi.Forge_{\mathtt{pp}}(\mathtt{spk}_i, \vec{v}', m, \vec{s})$

Algorithm 16 Description of the behavior of the simulator considered in the proof of Theorem 11 for the (sub-)interfaces of dishonest parties that correspond to an interface of $\mathsf{Net} \cdot \mathbf{INS}$ in the real world system. In the following, $\mathbf{T}$ is as defined in Equations 6.2 and D.2.

▷ $(P \in \overline{\mathcal{P}^H})$-WRITE$(c)$
  if $c \notin$ CtxtSet **:**
    CtxtSet $\leftarrow$ CtxtSet $\cup \{c\}$
    $(\mathtt{spk}, \vec{v}', m) \leftarrow \Pi.D_{\mathtt{pp}}(\mathtt{rsk}_{\mathtt{pp}}, c)$
    if $(\mathtt{spk}, \vec{v}', m) \neq \bot$ **:**
      $\langle A_i \to \vec{V} \rangle \leftarrow$ GETLABEL$(\mathtt{spk}, \vec{v}')$
      if $\langle A_i \to \vec{V} \rangle \neq \bot \wedge A_i \in \overline{\mathcal{S}^H}$ **:**
        $\mathtt{id} \leftarrow \mathbf{T}$-WRITE$(\langle A_i \to \vec{V} \rangle, m)$
        Ctxts[id] $\leftarrow c$ // Add entry to map Ctxts.
        OUTPUT(id)
  OUTPUT(**INS**-WRITE(c))

▷ $(P \in \overline{\mathcal{P}^H})$-READ
  outputList $\leftarrow \emptyset$
  for $(\langle A_i \to \vec{V} \rangle, \mathtt{id}, m) \in \mathbf{T}$-READ **:**
    if id $\notin$ Ctxts **:** // Check existence of entry with given key in map Ctxts.
      $\vec{s} \coloneqq (\bot, \mathtt{rsk}_1, \ldots, \mathtt{rsk}_{|\vec{V}|})$ // For each V_l: if $V_l \in \overline{\mathcal{R}^H}$, s_{l+1} is V_l's secret key; else $\bot$.
      $c \leftarrow \Pi.Forge_{\mathtt{pp}}(\mathtt{spk}_i, \vec{v}' \coloneqq (\mathtt{rpk}_{\mathtt{pp}}, v_1, \ldots, v_{|\vec{v}|}), m, \vec{s})$
      if $c \in$ CtxtSet **:** Abort
      CtxtSet $\leftarrow$ CtxtSet $\cup \{c\}$
      Ctxts[id] $\leftarrow c$ // Add entry to map Ctxts.
    outputList $\leftarrow$ outputList $\cup \{(\mathtt{id}, \text{Ctxts}[\mathtt{id}])\}$ // Fetch value of entry from map Ctxts.
  for $(l \in \mathbb{N}, \mathtt{id}, l' \in \mathbb{N}) \in \mathbf{T}$-READ **:**
    if id $\notin$ Ctxts **:**
      $c \leftarrow \Pi.Forge_{\mathtt{pp}}(\mathtt{spk}_1, {\mathtt{rpk}_{\mathtt{pp}}}^{l+1}, {0^{l'}}, {\bot}^{l+1})$
      if $c \in$ CtxtSet **:** Abort
      CtxtSet $\leftarrow$ CtxtSet $\cup \{c\}$
      Ctxts[id] $\leftarrow c$
    outputList $\leftarrow$ outputList $\cup \{(\mathtt{id}, \text{Ctxts}[\mathtt{id}])\}$
  OUTPUT(outputList $\cup$ **INS**-READ)

Algorithm 17 Reduction $\mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}$ for Theorem 11.

▷ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V}\rangle, m)$
$c \leftarrow \mathcal{O}_E\big(A_i, \vec{V}' \coloneqq (B_{\mathsf{pp}}, V_1, \ldots, V_{|\vec{V}|}), m\big)$
$\mathtt{id} \leftarrow$WRITE(c)
CtxtDec[id] $\leftarrow$ DECRYPTION(B_{pp}, c)
OUTPUT(id)

▷ $(P \in \overline{\mathcal{P}^H})$-WRITE$(c)$
$\mathtt{id} \leftarrow$WRITE(c)
CtxtDec[id] $\leftarrow$ DECRYPTION(B_{pp}, c)
OUTPUT(id)

▷ $(P \in \mathcal{F})$-WRITE$(\langle[\text{Forge}]A_i \to \vec{V}\rangle, m)$
$c \leftarrow$ FORGE$\big(A_i, \vec{V}, m, \text{Set}(\vec{V}) \cap \overline{\mathcal{P}^H}\big)$
$\mathtt{id} \leftarrow$WRITE(c)
CtxtDec[id] $\leftarrow$ DECRYPTION(B_{pp}, c)
OUTPUT(id)

▷ $(P \in \overline{\mathcal{P}^H})$-READ
OUTPUT(READ)

▷ $(B_j \in \mathcal{R}^H)$-READ
outputList $\leftarrow \emptyset$
ciphertextSet $\leftarrow \emptyset$
for $(\mathtt{id}, c) \in$ READ **with** $c \notin$ ciphertextSet **:**
  ciphertextSet $\leftarrow$ ciphertextSet $\cup \{c\}$
  $(\langle A_i \to \vec{V}\rangle, m) \leftarrow$ DECRYPTION(B_j, c)
  if $(\langle A_i \to \vec{V}\rangle, m) \neq \perp$ **:**
    outputList $\leftarrow$ outputList $\cup \{(\mathtt{id}, (\langle A_i \to \vec{V}\rangle, m))\}$
OUTPUT(GETDELIVERED$(B_j$, outputList$))$

Algorithm 18 Reduction $\mathbf{C}^{\mathsf{Cons\text{-}H}}$ for Theorem 11. Below we only show the differences (highlighted) relative to $\mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}$.

▷ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V}\rangle, m)$
$c \leftarrow \mathcal{O}_E\big(A_i, \vec{V}' \coloneqq (B_{\mathsf{pp}}, V_1, \ldots, V_{|\vec{V}|}), m\big)$
if $c \in$ CtxtHon **:** Abort // Event ξ_1 : Ciphertext Already Exists
CtxtHon $\leftarrow$ CtxtHon $\cup \{c\}$
$\mathtt{id} \leftarrow$WRITE(c)
CtxtDec[id] $\leftarrow$ DECRYPTION(B_{pp}, c)
OUTPUT(id)

▷ $(P \in \mathcal{F})$-WRITE$(\langle[\text{Forge}]A_i \to \vec{V}\rangle, m)$
$c \leftarrow$ FORGE$\big(A_i, \vec{V}, m, \text{Set}(\vec{V}) \cap \overline{\mathcal{P}^H}\big)$
CtxtForge $\leftarrow$ CtxtForge $\cup \{c\}$
$\mathtt{id} \leftarrow$WRITE(c)
CtxtDec[id] $\leftarrow$ DECRYPTION(B_{pp}, c)
OUTPUT(id)

▷ $(P \in \overline{\mathcal{P}^H})$-WRITE$(c)$
$\mathtt{id} \leftarrow$WRITE(c)
if $c \notin$ CtxtHon $\cup$ CtxtForge **:**
  if $c \notin$ CtxtDis **:**
    CtxtDec[id] $\leftarrow$ DECRYPTION(B_{pp}, c)
  CtxtDis $\leftarrow$ CtxtDis $\cup \{c\}$
OUTPUT(id)

Proof. Let $\mathbf{R}$ be the real world system

$$\mathbf{R} \coloneqq \mathsf{Snd}^{\mathcal{S}^H}\,\mathsf{Rcv}^{\mathcal{R}^H}\,\mathsf{Forge}^{\mathcal{F}}\perp^J[\mathbf{KGA}, \mathsf{Net}\cdot\mathbf{INS}],$$

$\mathbf{T}$ be the ideal repository defined in Equation 6.2, i.e.

$$\mathbf{T} \coloneqq \begin{pmatrix} \mathsf{ConfAnon}^{\overline{\mathcal{P}^H}} \\ \cdot\,\mathsf{Otr}^{\overline{\mathcal{P}^H}} \end{pmatrix} \cdot \begin{bmatrix} \mathsf{Net}\cdot\perp^{\mathsf{Auth\text{-}Intf}}\cdot\Big[\langle A_i \to \vec{V}\rangle^{\{A_i\}\cup\overline{\mathcal{P}^H}}_{\text{Set}(\vec{V})\cup\overline{\mathcal{P}^H}}\Big]_{\substack{A_i\in\mathcal{S}\\ \vec{V}\in\mathcal{R}^+}} \\ \Big[\langle[\text{Forge}]A_i \to \vec{V}\rangle^{\mathcal{F}}_{\overline{\mathcal{P}^H}}\Big]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+} \end{bmatrix}, \quad \text{(D.2)}$$

Algorithm 19 Reduction $\mathbf{C}^{\mathsf{Cons}}$ for Theorem 11. We only show the differences (highlighted) relative to $\mathbf{C}^{\mathsf{Cons\text{-}H}}$.

▷ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V}\rangle, m)$
  $c \leftarrow \mathcal{O}_E\big(A_i, \vec{V}' := (B_{\mathsf{pp}}, V_1, \ldots, V_{|\vec{V}|}), m\big)$
  if $c \in$ CtxtHon $\cup$ CtxtDis **:** Abort // Event $\xi_1 \vee \xi_2$: Ciphertext Already Exists
  CtxtHon $\leftarrow$ CtxtHon $\cup \{c\}$
  id $\leftarrow$WRITE(c)
  CtxtDec[id] $\leftarrow$ DECRYPTION(B_{pp}, c)
  OUTPUT(id)

Algorithm 20 Reduction $\mathbf{C}^{\mathsf{Corr}}$ for Theorem 11. We only show the differences (highlighted) relative to $\mathbf{C}^{\mathsf{Cons}}$.

▷ $(B_j \in \mathcal{R}^H)$-READ
  outputList $\leftarrow \emptyset$
  ciphertextSet $\leftarrow \emptyset$
  for $(\mathtt{id}, c) \in$ READ **with** $c \notin$ ciphertextSet **:**
    ciphertextSet $\leftarrow$ ciphertextSet $\cup \{c\}$
    $(\langle A_i \to \vec{V}\rangle, m) \leftarrow$ CtxtDec[id]
    if $(\langle A_i \to \vec{V}\rangle, m) \neq \perp \wedge B_j \in \vec{V}$ **:**
      outputList $\leftarrow$ outputList $\cup \{(\mathtt{id}, (\langle A_i \to \vec{V}\rangle, m))\}$
  OUTPUT(GETDELIVERED$(B_j,$ outputList))

Algorithm 21 Reduction $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}$ for Theorem 11. We only show the differences (highlighted) relative to $\mathbf{C}^{\mathsf{Corr}}$.

▷ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V}\rangle, m)$
  $c \leftarrow \mathcal{O}_E\big(A_i, \vec{V}' := (B_{\mathsf{pp}}, V_1, \ldots, V_{|\vec{V}|}), m\big)$
  if $c \in$ CtxtHon $\cup$ CtxtDis **:** Abort // Event $\xi_1 \vee \xi_2$: Ciphertext Already Exists
  CtxtHon $\leftarrow$ CtxtHon $\cup \{c\}$
  id $\leftarrow$WRITE(c)
  CtxtDec[id] $\leftarrow (\langle A_i \to \vec{V}\rangle, m)$
  OUTPUT(id)

▷ $(P \in \mathcal{F})$-WRITE$(\langle[\text{Forge}]A_i \to \vec{V}\rangle, m)$
  $c \leftarrow$ FORGERED$\big(A_i, \vec{V}, m, \text{Set}(\vec{V}) \cap \overline{\mathcal{P}^H}\big)$
  CtxtForge $\leftarrow$ CtxtForge $\cup \{c\}$
  id $\leftarrow$WRITE(c)
  CtxtDec[id] $\leftarrow$ DECRYPTION(B_{pp}, c)
  OUTPUT(id)

// Local procedure.
◇ FORGERED$(A_i, \vec{V}, m, \mathcal{C})$
  if $\vec{V} \in (\mathcal{R}^H)^+$ **:**
    return $\mathcal{O}_{Forge}(A_1, B_{\mathsf{pp}}{}^{|\vec{V}|+1}, 0^{|m|}, \emptyset)$
  return $\mathcal{O}_{Forge}\big(A_i, \vec{V}' := (B_{\mathsf{pp}}, V_1, \ldots, V_{|\vec{V}|}), m, \text{Set}(\vec{V}) \cap \overline{\mathcal{P}^H}\big)$

Algorithm 22 Reduction $\mathbf{C}^{\mathsf{R\text{-}Unforg}}$ for Theorem 11. We only show the differences (highlighted) relative to $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}$.

▷ $(P \in \mathcal{F})$-WRITE$(\langle[\text{Forge}]A_i \to \vec{V}\rangle, m)$
  $c \leftarrow$ FORGE$\big(A_i, \vec{V}, m, \text{Set}(\vec{V}) \cap \overline{\mathcal{P}^H}\big)$
  CtxtForge $\leftarrow$ CtxtForge $\cup \{c\}$
  id $\leftarrow$WRITE(c)
  CtxtDec[id] $\leftarrow \perp$
  OUTPUT(id)

Algorithm 23 Reduction $\mathbf{C}^{\mathsf{CCA}}$ for Theorem 11. We only show the differences (highlighted) relative to $\mathbf{C}^{\mathsf{R\text{-}Unforg}}$.

▷ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V}\rangle, m)$
  $\alpha_{\mathbf{0}} \coloneqq (A_i, \vec{V}' \coloneqq (B_{\mathsf{pp}}, V_1, \ldots, V_{|\vec{V}|}), m)$, $\alpha_{\mathbf{1}} \coloneqq (A_1, \overbrace{(B_{\mathsf{pp}}, \ldots, B_{\mathsf{pp}})}^{|\vec{V}|+1 \text{ times}}, 0^{|m|})$
  if $\vec{V} \notin (\mathcal{R}^H)^+$ **:** // This allows for a reduction to $\{\mathsf{IND}, \mathsf{IK}\}\text{-}\mathsf{CCA}_S$.
    $\alpha_{\mathbf{1}} \coloneqq \alpha_{\mathbf{0}}$
  $c \leftarrow \mathcal{O}_E(\alpha_{\mathbf{0}}, \alpha_{\mathbf{1}})$
  if $c \in \text{CtxtHon} \cup \text{CtxtDis}$ **:** Abort // Event $\xi_1 \vee \xi_2$: Ciphertext Already Exists
  $\text{CtxtHon} \leftarrow \text{CtxtHon} \cup \{c\}$
  $\mathtt{id} \leftarrow$WRITE(c)
  $\text{CtxtDec}[\mathtt{id}] \leftarrow (\langle A_i \to \vec{V}\rangle, m)$
  OUTPUT$(\mathtt{id})$

▷ $(P \in \overline{\mathcal{P}^H})$-WRITE$(c)$
  $\mathtt{id} \leftarrow$WRITE(c)
  if $c \notin \text{CtxtHon} \cup \text{CtxtForge}$ **:**
    if $c \notin \text{CtxtDis}$ **:**
      $(\langle A_i \to \vec{V}\rangle, m) \leftarrow$DECRYPTION$(B_{\mathsf{pp}}, c)$
      if $A_i \in \mathcal{S}^H$ **:** Abort // Valid Ciphertext Forgery
      $\text{CtxtDec}[\mathtt{id}] \leftarrow (\langle A_i \to \vec{V}\rangle, m)$
    $\text{CtxtDis} \leftarrow \text{CtxtDis} \cup \{c\}$
  OUTPUT$(\mathtt{id})$

Algorithm 24 Reduction $\mathbf{C}^{\mathsf{OTR}}$ for Theorem 11. We only show the differences (highlighted) relative to $\mathbf{C}^{\mathsf{CCA}}$.

▷ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V}\rangle, m)$
  if $\vec{V} \in (\mathcal{R}^H)^+$ **:**
    $c \leftarrow \mathcal{O}_E(\mathtt{sig}, A_1, \overbrace{(B_{\mathsf{pp}}, \ldots, B_{\mathsf{pp}})}^{|\vec{V}|+1 \text{ times}}, 0^{|m|}, \emptyset)$
  else
    $c \leftarrow \mathcal{O}_E(\mathtt{sig}, A_i, \vec{V}' \coloneqq (B_{\mathsf{pp}}, V_1, \ldots, V_{|\vec{V}|}), m, \mathrm{Set}(\vec{V}) \cap \overline{\mathcal{P}^H})$
  if $c \in \text{CtxtHon} \cup \text{CtxtDis}$ **:** Abort // Event $\xi_1 \vee \xi_2$: Ciphertext Already Exists
  $\text{CtxtHon} \leftarrow \text{CtxtHon} \cup \{c\}$
  $\mathtt{id} \leftarrow$WRITE(c)
  $\text{CtxtDec}[\mathtt{id}] \leftarrow (\langle A_i \to \vec{V}\rangle, m)$
  OUTPUT$(\mathtt{id})$

and let sim be the simulator specified in Algorithms 13 and 16. The remainder of the proof bounds $\Delta^{\mathbf{D}}(\mathbf{R}, \mathsf{sim}^{\overline{\mathcal{P}^H}}\mathbf{T})$ by proceeding in a sequence of hybrids. In the following, we consider the reduction systems defined in the lemma's statement.

$\mathbf{R} \rightsquigarrow \mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}\mathbf{G}^{\mathsf{Cons}}$: It is easy to see that $\mathbf{R}$ and $\mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}\mathbf{G}^{\mathsf{Cons}}$ are the same sequence of conditional probability distributions—conditioned on the event that all parties public keys are distinct—by considering, on one hand, the definition of $\mathbf{R}$—i.e. the definitions of converters Snd, Rcv and Forge (Algorithm 9), the definition of the **KGA** resource (Algorithm 8), and the definitions of **INS** (Algorithm 1) and of Net (Algorithm 3)—and, on the other hand, the definition of $\mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}\mathbf{G}^{\mathsf{Cons}}$—i.e. the definition of $\mathbf{G}^{\mathsf{Cons}}$ and its oracles (Definition 8 and Section 4.1) and the definition of $\mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}$ (Algorithms 14, 15 and 17). Since by assumption the MDRS-PKE scheme is (m, n)-Party ε-Public Key Collision Resistant and there are m senders and n receivers, it follows

$$\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}\mathbf{G}^{\mathsf{Cons}}) \leq \varepsilon.$$

$\mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}\mathbf{G}^{\mathsf{Cons}} \rightsquigarrow \mathbf{C}^{\mathsf{Cons\text{-}H}}\mathbf{G}^{\mathsf{Cons}}$: It is easy to see that $\mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}\mathbf{G}^{\mathsf{Cons}}$ and $\mathbf{C}^{\mathsf{Cons\text{-}H}}\mathbf{G}^{\mathsf{Cons}}$ are the same sequence of conditional probability distributions conditioned on event ξ_1 not occurring (see Section D.1.1). By Lemma 2, it follows

$$\begin{aligned}\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}\mathbf{G}^{\mathsf{Cons}}, \mathbf{C}^{\mathsf{Cons\text{-}H}}\mathbf{G}^{\mathsf{Cons}}) \leq 4 \cdot \Big(& Adv^{\mathsf{OTR}}(\mathbf{D}\perp^J\mathbf{C}^{\mathsf{OTR}\text{-}\xi_1}) \\ & + Adv^{\mathsf{Corr}}(\mathbf{D}\perp^J\mathbf{C}^{\mathsf{Corr}\text{-}\xi_1}) \\ & + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{D}\perp^J\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1})\Big).\end{aligned}$$

$\mathbf{C}^{\mathsf{Cons\text{-}H}}\mathbf{G}^{\mathsf{Cons}} \rightsquigarrow \mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}$: As for the previous step $\mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}\mathbf{G}^{\mathsf{Cons}} \rightsquigarrow \mathbf{C}^{\mathsf{Cons\text{-}H}}\mathbf{G}^{\mathsf{Cons}}$, it is easy to see that the two systems are the exact same sequence of conditional probability distributions conditioned on event ξ_2 not occurring (see Section D.1.1). It then follows by Lemma 3

$$\begin{aligned}\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Cons\text{-}H}}\mathbf{G}^{\mathsf{Cons}}, \mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}) \leq\ & Adv^{\mathsf{OTR}}(\mathbf{D}\perp^J\mathbf{C}^{\mathsf{OTR}\text{-}\xi_2}) \\ & + Adv^{\mathsf{Cons}}(\mathbf{D}\perp^J\mathbf{C}^{\mathsf{Cons\text{-}0}\text{-}\xi_2}) + Adv^{\mathsf{Cons}}(\mathbf{D}\perp^J\mathbf{C}^{\mathsf{Cons\text{-}1}\text{-}\xi_2}) \\ & + Adv^{\mathsf{Corr}}(\mathbf{D}\perp^J\mathbf{C}^{\mathsf{Corr}\text{-}\xi_2}) + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{D}\perp^J\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2}).\end{aligned}$$

$\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}} \rightsquigarrow \mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$: The only difference between $\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$ and $\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}$ is that in $\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$ each ciphertext that is either generated by a WRITE operation at the interface of an honest sender $A_i \in \mathcal{S}^H$ or input to a WRITE operation at the interface of a dishonest party $P \in \overline{\mathcal{P}^H}$ is decrypted only once, and decryption uses the secret key $\mathtt{rsk}_{\mathtt{pp}}$ corresponding to the public parameters public key $\mathtt{rpk}_{\mathtt{pp}}$. (For more details see Algorithms 19 and 20). Given $\mathbf{C}^{\mathsf{Cons}}$ does not query for the secret key of any receiver $B_j \in \mathcal{R}^H$ nor for the secret key of $B_{\mathtt{pp}}$, the advantage of a distinguisher $\mathbf{D}$ in distinguishing $\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}$ and $\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$ is bounded by the advantage of adversary $\mathbf{D}\mathbf{C}^{\mathsf{Cons}}$ in winning the consistency

game $\mathbf{G}^{\mathsf{Cons}}$ (note that $\mathbf{C}^{\mathsf{Cons}}$ makes a query to $\mathcal{O}_D$ on party $B_{\mathtt{pp}}$ when queried for any WRITE operation, and when queried for a READ operation at the interface of a receiver $B_j \in \mathcal{R}^H$ makes a query to $\mathcal{O}_D$ for each $(\mathtt{id}, c)$ in the reduction's internal repository **INS**) implying

$$\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}, \mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}) \leq Adv^{\mathsf{Cons}}(\mathbf{D}\mathbf{C}^{\mathsf{Cons}}).$$

$\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}} \rightsquigarrow \mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$**:** System $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$ differs from $\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$ in that ciphertexts generated by WRITE operations issued at the interface of honest senders are no longer decrypted by a query to $\mathcal{O}_D$ on party $B_{\mathtt{pp}}$, and instead the result of their decryption is simply assumed to be the correct label-message pair. (Note that using procedure FORGE or using REDFORGE is perfectly indistinguishable, in an information-theoretic sense.) Since we are already assuming no two parties have the same public key, **D**'s advantage in distinguishing $\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$ and $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$ is upper bounded by the advantage of $\mathbf{D}\mathbf{C}^{\mathsf{Corr}}$ in winning the correctness game $\mathbf{G}^{\mathsf{Corr}}$, implying

$$\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}, \mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}) \leq Adv^{\mathsf{Corr}}(\mathbf{D}\mathbf{C}^{\mathsf{Corr}}).$$

$\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}} \rightsquigarrow \mathbf{C}^{\mathsf{R\text{-}Unforg}}\mathbf{G}^{\mathsf{R\text{-}Unforg}}$**:** In $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$, ciphertexts generated on queries $\text{WRITE}(\langle[\text{Forge}]A_i \to \vec{V}\rangle, m)$ are no longer decrypted by a query to $\mathcal{O}_D$ on party $B_{\mathtt{pp}}$, and instead the result of their decryption is assumed to fail (i.e. resulting in $\perp$). **D**'s distinguishing advantage between $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$ and $\mathbf{C}^{\mathsf{R\text{-}Unforg}}\mathbf{G}^{\mathsf{R\text{-}Unforg}}$ is upper bounded by the advantage of $\mathbf{D}\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}$ in winning the forgery invalidity game $\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$, implying

$$\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}, \mathbf{C}^{\mathsf{R\text{-}Unforg}}\mathbf{G}^{\mathsf{R\text{-}Unforg}}) \leq Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{D}\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}).$$

$\mathbf{C}^{\mathsf{R\text{-}Unforg}}\mathbf{G}^{\mathsf{R\text{-}Unforg}} \rightsquigarrow \mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{0}}^{\{\mathsf{IND,IK}\}\text{-}\mathsf{CCA}}$**:** The main things to note for this step are that 1. **D** has no access to the secret key corresponding to $\mathtt{rpk}_{\mathtt{pp}}$ (i.e. the public parameters public key); 2. since J has a converter $\perp$ attached to her interface, **D** also has no access to the secret key of any honest sender $A_i \in \mathcal{S}^H$; and 3. the only case in which $\mathbf{C}^{\mathsf{R\text{-}Unforg}}\mathbf{G}^{\mathsf{R\text{-}Unforg}}$ may differ from $\mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{0}}^{\{\mathsf{IND,IK}\}\text{-}\mathsf{CCA}}$ is if **D** makes a query for a WRITE operation at the interface of a dishonest party $P \in \overline{\mathcal{S}^H} \cup \overline{\mathcal{R}^H}$ with input ciphertext c whose decryption results in a label $\langle A_i \to \vec{V}\rangle$ where $A_i \in \mathcal{S}^H$ and yet there was no WRITE operation at the interface of A_i that resulted in ciphertext c. This allows us to bound the advantage of **D** in distinguishing $\mathbf{C}^{\mathsf{R\text{-}Unforg}}\mathbf{G}^{\mathsf{R\text{-}Unforg}}$ and $\mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{0}}^{\{\mathsf{IND,IK}\}\text{-}\mathsf{CCA}}$ by the advantage of $\mathbf{D}\mathbf{C}^{\mathsf{R\text{-}Unforg}}$ in winning $\mathbf{G}^{\mathsf{R\text{-}Unforg}}$, implying

$$\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{R\text{-}Unforg}}\mathbf{G}^{\mathsf{R\text{-}Unforg}}, \mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{0}}^{\{\mathsf{IND,IK}\}\text{-}\mathsf{CCA}}) \leq Adv^{\mathsf{R\text{-}Unforg}}(\mathbf{D}\mathbf{C}^{\mathsf{R\text{-}Unforg}}).$$

$\mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{1}}^{\{\mathsf{IND,IK}\}\text{-}\mathsf{CCA}} \rightsquigarrow \mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}$**:** Systems $\mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{1}}^{\{\mathsf{IND,IK}\}\text{-}\mathsf{CCA}}$ and $\mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}$ are perfectly indistinguishable. It follows

$$\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{1}}^{\{\mathsf{IND,IK}\}\text{-}\mathsf{CCA}}, \mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}) = 0.$$

$\mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}} \rightsquigarrow \overline{\mathsf{sim}^{\mathcal{P}^H}}\mathbf{T}$**:** It is easy to see, by considering the definitions of $\mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}$—i.e. of $\mathbf{C}^{\mathsf{OTR}}$, Algorithms 14, 15 and 24, and of $\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}$ and its oracles, Definition 10 and Section 2.4—and $\overline{\mathsf{sim}^{\mathcal{P}^H}}\mathbf{T}$—i.e. of simulator sim, Algorithms 13 and 16, and of $\mathbf{T}$, Equation D.2—that these are perfectly indistinguishable; it follows

$$\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}, \overline{\mathsf{sim}^{\mathcal{P}^H}}\mathbf{T}) = 0.$$

To conclude the proof we use triangle inequality:

$$\begin{aligned}
\Delta^{\mathbf{D}}(\mathbf{R}, \overline{\mathsf{sim}^{\mathcal{P}^H}}\mathbf{T}) &\leq \Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}\mathbf{G}^{\mathsf{Cons}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}\mathbf{G}^{\mathsf{Cons}}, \mathbf{C}^{\mathsf{Cons\text{-}H}}\mathbf{G}^{\mathsf{Cons}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Cons\text{-}H}}\mathbf{G}^{\mathsf{Cons}}, \mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}, \mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}, \mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}, \mathbf{C}^{\mathsf{R\text{-}Unforg}}\mathbf{G}^{\mathsf{R\text{-}Unforg}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{R\text{-}Unforg}}\mathbf{G}^{\mathsf{R\text{-}Unforg}}, \mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{0}}^{\{\mathsf{IND,IK}\}\text{-}\mathsf{CCA}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{0}}^{\{\mathsf{IND,IK}\}\text{-}\mathsf{CCA}}, \mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{1}}^{\{\mathsf{IND,IK}\}\text{-}\mathsf{CCA}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{1}}^{\{\mathsf{IND,IK}\}\text{-}\mathsf{CCA}}, \mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}, \mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}, \overline{\mathsf{sim}^{\mathcal{P}^H}}\mathbf{T}) \\
&\leq \varepsilon + 4 \cdot \Big(Adv^{\mathsf{OTR}}(\mathbf{D}\perp^J \mathbf{C}^{\mathsf{OTR}\text{-}\xi_1}) + Adv^{\mathsf{Corr}}(\mathbf{D}\perp^J \mathbf{C}^{\mathsf{Corr}\text{-}\xi_1}) \\
&\qquad + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{D}\perp^J \mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1}) \Big) \\
&\quad + Adv^{\mathsf{OTR}}(\mathbf{D}\perp^J \mathbf{C}^{\mathsf{OTR}\text{-}\xi_2}) + Adv^{\mathsf{Cons}}(\mathbf{D}\perp^J \mathbf{C}^{\mathsf{Cons}\text{-}0\text{-}\xi_2}) \\
&\quad + Adv^{\mathsf{Cons}}(\mathbf{D}\perp^J \mathbf{C}^{\mathsf{Cons}\text{-}1\text{-}\xi_2}) + Adv^{\mathsf{Corr}}(\mathbf{D}\perp^J \mathbf{C}^{\mathsf{Corr}\text{-}\xi_2}) \\
&\quad + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{D}\perp^J \mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2}) + Adv^{\mathsf{Cons}}(\mathbf{D}\mathbf{C}^{\mathsf{Cons}}) \\
&\quad + Adv^{\mathsf{Corr}}(\mathbf{D}\mathbf{C}^{\mathsf{Corr}}) + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{D}\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}) + Adv^{\mathsf{R\text{-}Unforg}}(\mathbf{D}\mathbf{C}^{\mathsf{R\text{-}Unforg}}) \\
&\quad + Adv^{\{\mathsf{IND,\,IK}\}\text{-}\mathsf{CCA}}(\mathbf{D}\mathbf{C}^{\mathsf{CCA}}) + Adv^{\mathsf{OTR}}(\mathbf{D}\mathbf{C}^{\mathsf{OTR}}).
\end{aligned}$$

□

D.1.3 Proof of Theorem 12

Proof. This proof is similar to the one from Theorem 11. We present it for completeness.

Let $\mathbf{R}$ be the real world system

$$\mathbf{R} \coloneqq \mathsf{Snd}^{\mathcal{S}^H}\,\mathsf{Rcv}^{\mathcal{R}^H}\,\mathsf{Forge}^{\mathcal{F}}[\mathbf{KGA}, \mathsf{Net} \cdot \mathbf{INS}],$$

Algorithm 25 Description of the behavior of the simulator considered in the proof of Theorem 12 for the (sub-)interfaces of dishonest parties in $\overline{\mathcal{P}^H}$ that correspond to an interface of $\mathsf{Net} \cdot \mathbf{INS}$ in the real world system. In the following $\mathbf{S}$ is as defined in Equation 6.1 and Equation D.3.

▷ (J)-SenderKeyPair($A_i \in \mathcal{S}^H$)
  Output($\mathtt{spk}_i, \mathtt{ssk}_i$)

▷ ($P \in \overline{\mathcal{P}^H}$)-Write($c$)
  if $c \notin$ CtxtSet **:**
    CtxtSet $\leftarrow$ CtxtSet $\cup \{c\}$
    $(\mathtt{spk}, \vec{v}', m) \leftarrow \Pi.D_{\mathtt{pp}}(\mathtt{rsk}_{\mathtt{pp}}, c)$
    if $(\mathtt{spk}, \vec{v}', m) \neq \bot$ **:**
      $\langle A_i \to \vec{V} \rangle \leftarrow$ GetLabel($\mathtt{spk}, \vec{v}'$)
      if $\langle A_i \to \vec{V} \rangle \neq \bot$ **:**
        $\mathtt{id} \leftarrow$ **S**-Write($\langle A_i \to \vec{V} \rangle, m$)
        Ctxts[id] $\leftarrow c$ // Add entry to map Ctxts.
        Output(id)
  Output(**INS**-Write(c))

▷ ($P \in \overline{\mathcal{P}^H}$)-Read
  outputList $\leftarrow \emptyset$
  for $(\langle A_i \to \vec{V} \rangle, \mathtt{id}, m) \in$ **S**-Read **:**
    if id $\notin$ Ctxts **:** // $A_i \in \mathcal{S}^H$
      $\vec{s} := (\bot, \mathtt{rsk}_1, \ldots, \mathtt{rsk}_{|\vec{V}|})$ // For each V_l: if $V_l \in \overline{\mathcal{R}^H}$, s_{l+1} is V_l's secret key; else $\bot$.
      $c \leftarrow \Pi.Forge_{\mathtt{pp}}(\mathtt{spk}_i, \vec{v}' := (\mathtt{rpk}_{\mathtt{pp}}, v_1, \ldots, v_{|\vec{v}|}), m, \vec{s})$
      if $c \in$ CtxtSet **:** Abort // Event $\xi_1 \vee \xi_2$.
      CtxtSet $\leftarrow$ CtxtSet $\cup \{c\}$
      Ctxts[id] $\leftarrow c$
    outputList $\leftarrow$ outputList $\cup \{(\mathtt{id}, \text{Ctxts}[\mathtt{id}])\}$
  for $(l \in \mathbb{N}, \mathtt{id}, l' \in \mathbb{N}) \in$ **S**-Read **:**
    if id $\notin$ Ctxts **:** // $A_i \in \mathcal{S}^H$
      $c \leftarrow \Pi.Forge_{\mathtt{pp}}(\mathtt{spk}_1, {\mathtt{rpk}_{\mathtt{pp}}}^{l+1}, {0^{l'}}, {\bot^{l+1}})$
      if $c \in$ CtxtSet **:** Abort // Event $\xi_1 \vee \xi_2$.
      CtxtSet $\leftarrow$ CtxtSet $\cup \{c\}$
      Ctxts[id] $\leftarrow c$
    outputList $\leftarrow$ outputList $\cup \{(\mathtt{id}, \text{Ctxts}[\mathtt{id}])\}$
  Output(outputList $\cup$ **INS**-Read)

$\mathbf{S}$ be the ideal world's repository from Equation 6.1

$$\mathbf{S} \coloneqq \mathsf{ConfAnon}^{\overline{\mathcal{P}^H}} \cdot \mathsf{Otr}^{\overline{\mathcal{P}^H}} \cdot \begin{bmatrix} \mathsf{Net} \cdot \left[\langle A_i \to \vec{V} \rangle^{\overline{\mathcal{P}^H}}_{\mathrm{Set}(\vec{V}) \cup \overline{\mathcal{P}^H}}\right]_{A_i \in \mathcal{S}, \vec{V} \in \mathcal{R}^+} \\ \left[\langle [\mathrm{Forge}] A_i \to \vec{V} \rangle^{\mathcal{F}}_{\overline{\mathcal{P}^H}}\right]_{A_i \in \mathcal{S}, \vec{V} \in \mathcal{R}^+} \end{bmatrix}, \tag{D.3}$$

and let sim be the simulator specified in Algorithms 13 and 25. One can bound $\Delta^{\mathbf{D}}(\mathbf{R}, \mathsf{sim}^{\overline{\mathcal{P}^H}}\mathbf{S})$ by proceeding in a sequence of hybrids that is essentially the same as the one given in the proof of Theorem 11; the only differences are:

- each reduction provides an additional interface to allow for queries for honest senders' secret keys;
- there is no reduction to unforgeability.

As before, the main thing to note in the reductions is that the distinguisher is not given access to the secret keys of any honest receivers nor to the secret key of B_{pp}, which is necessary to ensure we can use the adversary to win the underlying security games. The following sequence of hybrids is essentially the same as in Theorem 11 (with the change explained above, that the reduction can query for honest senders' secret keys).

$$\begin{aligned} \mathbf{R} &\rightsquigarrow \mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}\mathbf{G}^{\mathsf{Cons}} \\ &\rightsquigarrow \mathbf{C}^{\mathsf{Cons\text{-}H}}\mathbf{G}^{\mathsf{Cons}} \\ &\rightsquigarrow \mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}} \\ &\rightsquigarrow \mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}} \\ &\rightsquigarrow \mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}} \end{aligned}$$

$\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}} \rightsquigarrow \mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{0}}^{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}}$**:** Analogous to proof of Theorem 11's step $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}} \rightsquigarrow \mathbf{C}^{\mathsf{R\text{-}Unforg}}\mathbf{G}^{\mathsf{R\text{-}Unforg}}$.

Finally, note the following hops, with the changes explained above, are also analogous:

$$\begin{aligned} \mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{1}}^{\{\mathsf{IND},\mathsf{IK}\}\text{-}\mathsf{CCA}} &\rightsquigarrow \mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} \\ \mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}} &\rightsquigarrow \mathsf{sim}^{\overline{\mathcal{P}^H}}\mathbf{S}. \end{aligned}$$

It then follows by triangle inequality:

$$
\begin{aligned}
\Delta^{\mathbf{D}}(\mathbf{R}, \mathsf{sim}^{\overline{\mathcal{P}^H}}\mathbf{S}) \leq\ & \Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}\mathbf{G}^{\mathsf{Cons}}) \\
& + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{PK\text{-}Coll\text{-}Res}}\mathbf{G}^{\mathsf{Cons}}, \mathbf{C}^{\mathsf{Cons\text{-}H}}\mathbf{G}^{\mathsf{Cons}}) \\
& + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Cons\text{-}H}}\mathbf{G}^{\mathsf{Cons}}, \mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}) \\
& + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}, \mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}) \\
& + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}, \mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}) \\
& + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}, \mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{0}}^{\{\mathsf{IND,IK}\}\text{-}\mathsf{CCA}}) \\
& + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{0}}^{\{\mathsf{IND,IK}\}\text{-}\mathsf{CCA}}, \mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{1}}^{\{\mathsf{IND,IK}\}\text{-}\mathsf{CCA}}) \\
& + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{CCA}}\mathbf{G}_{\mathbf{1}}^{\{\mathsf{IND,IK}\}\text{-}\mathsf{CCA}}, \mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}) \\
& + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}, \mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}) \\
& + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}, \mathsf{sim}^{\overline{\mathcal{P}^H}}\mathbf{S}) \\
\leq\ & \varepsilon + 4 \cdot \Big(Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR\text{-}}\xi_1}) + Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr\text{-}}\xi_1}) \\
& + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid\text{-}}\xi_1}) \Big) \\
& + Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR\text{-}}\xi_2}) + Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons\text{-}0\text{-}}\xi_2}) + Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons\text{-}1\text{-}}\xi_2}) \\
& + Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr\text{-}}\xi_2}) + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid\text{-}}\xi_2}) + Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons}}) \\
& + Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}}) + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}}) \\
& + Adv^{\{\mathsf{IND,\,IK}\}\text{-}\mathsf{CCA}}(\mathbf{DC}^{\mathsf{CCA}}) + Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}}).
\end{aligned}
$$

□

D.1.4 Proof of Helper Claim: Lemma 2 Consider adversary $\mathbf{DC}^{\mathsf{OTR\text{-}}\xi_1}$ interacting with $\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}$: if event ${\xi_1}'$ occurs[12] $\mathbf{DC}^{\mathsf{OTR\text{-}}\xi_1}$ wins the game; if ${\xi_1}'$ does not occur, $\mathbf{DC}^{\mathsf{OTR\text{-}}\xi_1}$ wins the game with probability $1/2$. Now, suppose $\mathbf{DC}^{\mathsf{OTR\text{-}}\xi_1}$ interacts with $\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}$: if ${\xi_1}'$ does not occur $\mathbf{DC}^{\mathsf{OTR\text{-}}\xi_1}$ wins the game with probability $1/2$. If event ${\xi_1}'$ occurs then $\mathbf{DC}^{\mathsf{OTR\text{-}}\xi_1}$ does not win $\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}$; however, one can bound the probability of event ${\xi_1}'$ occurring (when $\mathbf{DC}^{\mathsf{OTR\text{-}}\xi_1}$ is interacting with $\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}$) by the probability that $\mathbf{DC}^{\mathsf{Corr\text{-}}\xi_1}$ wins the correctness game plus the probability that $\mathbf{DC}^{\mathsf{Forge\text{-}Invalid\text{-}}\xi_1}$ wins the Forgery Invalidity game. It follows

$$
\begin{aligned}
\Pr[\mathbf{DC}^{\mathsf{OTR\text{-}}\xi_1}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}} \neq \mathtt{win}] \leq\ & \frac{1}{2} + \Pr[\mathbf{DC}^{\mathsf{Corr\text{-}}\xi_1}\mathbf{G}^{\mathsf{Corr}} = \mathtt{win}] \\
& + \Pr[\mathbf{DC}^{\mathsf{Forge\text{-}Invalid\text{-}}\xi_1}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}} = \mathtt{win}].
\end{aligned}
$$

[12] See Algorithm 26 for a definition of event ${\xi_1}'$.

By Definition 10,

$$\begin{aligned}
Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}) &:= \Big|\Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} = \mathtt{win}] + \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}} = \mathtt{win}] - 1\Big| \\
&= \Big|\Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} = \mathtt{win} \mid {\xi_1}'] \cdot \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} \Rightarrow {\xi_1}'] \\
&\quad + \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} = \mathtt{win} \mid \neg{\xi_1}'] \cdot \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} \Rightarrow \neg{\xi_1}'] \\
&\quad + \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}} = \mathtt{win}] - 1\Big| \\
&= \Big|\Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} \Rightarrow {\xi_1}'] + \frac{1}{2} \cdot \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} \Rightarrow \neg{\xi_1}'] \\
&\quad + \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}} = \mathtt{win}] - 1\Big| \\
&= \Big|\Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} \Rightarrow {\xi_1}'] \cdot \frac{1}{2} - \frac{1}{2} + \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}} = \mathtt{win}]\Big|.
\end{aligned}$$

We consider the two possible cases:

$$Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}) = \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} \Rightarrow {\xi_1}'] \cdot \frac{1}{2} - \frac{1}{2} + \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}} = \mathtt{win}],$$

and

$$Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}) = \frac{1}{2} - \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} \Rightarrow {\xi_1}'] \cdot \frac{1}{2} - \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}} = \mathtt{win}].$$

For the first case, we have

$$\begin{aligned}
&Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}) = \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} \Rightarrow {\xi_1}'] \cdot \frac{1}{2} - \frac{1}{2} + \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}} = \mathtt{win}] \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Leftrightarrow \\
&Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}) + \frac{1}{2} - \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}} = \mathtt{win}] = \frac{1}{2} \cdot \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} \Rightarrow {\xi_1}'] \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Rightarrow \\
&\qquad\qquad Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}) + \frac{1}{2} - \left(\frac{1}{2} - \Big(\Pr[\mathbf{DC}^{\mathsf{Corr}\text{-}\xi_1}\mathbf{G}^{\mathsf{Corr}} = \mathtt{win}] + \right. \\
&\left.\Pr[\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}} = \mathtt{win}]\Big)\right) \geq \frac{1}{2} \cdot \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} \Rightarrow {\xi_1}'] \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \Leftrightarrow \\
&\qquad\qquad Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}) + \Pr[\mathbf{DC}^{\mathsf{Corr}\text{-}\xi_1}\mathbf{G}^{\mathsf{Corr}} = \mathtt{win}] \\
&+ \Pr[\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}} = \mathtt{win}] \geq \frac{1}{2} \cdot \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} \Rightarrow {\xi_1}'].
\end{aligned}$$

For the second, we have

$$Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}) = \frac{1}{2} - \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} \Rightarrow \xi_1{}'] \cdot \frac{1}{2} - \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}} = \mathtt{win}]$$

$$\Leftrightarrow$$

$$\frac{1}{2} \cdot \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} \Rightarrow \xi_1{}'] = \frac{1}{2} - Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}) - \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}} = \mathtt{win}]$$

$$\Rightarrow$$

$$\frac{1}{2} \cdot \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} \Rightarrow \xi_1{}'] \leq \frac{1}{2} + Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}) - \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}} = \mathtt{win}]$$

$$\Rightarrow$$

$$\frac{1}{2} \cdot \Pr[\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}} \Rightarrow \xi_1{}'] \leq Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}) + \Pr[\mathbf{DC}^{\mathsf{Corr}\text{-}\xi_1}\mathbf{G}^{\mathsf{Corr}} = \mathtt{win}] + \Pr[\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}} = \mathtt{win}].$$

Putting things together one can then upper bound the probability that $\xi_1{}'$ occurs by

$$\begin{aligned} & 2 \cdot \Big(Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}) + \Pr[\mathbf{DC}^{\mathsf{Corr}\text{-}\xi_1}\mathbf{G}^{\mathsf{Corr}} = \mathtt{win}] \\ & \qquad + \Pr[\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}} = \mathtt{win}] \Big) \\ = \; & 2 \cdot \Big(Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}) + Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}\text{-}\xi_1}) \\ & \qquad + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1}) \Big). \end{aligned}$$

To conclude, note that the probability for event $\xi_1{}'$ to occur is half of the probability that event ξ_1 occurs. □

D.1.5 Proof of Helper Claim: Lemma 3 The proof of this result follows similar lines to the proof of Lemma 2; in the following, events $\xi_{2,0}$ and $\xi_{2,1}$ are as defined in Algorithm 29.

Interacting with $\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}$*:* First, consider adversary $\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_2}$ interacting with $\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}$: if $\xi_{2,0}$ occurs $\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_2}$ wins the game; if $\xi_{2,1}$ occurs, it does not win the game; and otherwise it wins the game with probability $1/2$. We can bound the probability that $\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_2}$ does not win $\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}$ due to event $\xi_{2,1}$ occurring by reducing to winning either the consistency or the correctness games. Concretely, we bound the probability of $\xi_{2,1}$ occurring when $\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_2}$ is interacting with $\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}$ by

$$Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons\text{-}0}\text{-}\xi_2}) + Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}\text{-}\xi_2}).$$

Interacting with $\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}$*:* Conversely, consider $\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_2}$ is now interacting with $\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}$: if $\xi_{2,0}$ occurs $\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_2}$ does not win; if $\xi_{2,1}$ occurs, it wins the game, and otherwise it wins the game with probability $1/2$. As before we bound the

Algorithm 26 Reduction $\mathbf{C}^{\mathsf{OTR}\text{-}\xi_1}$ for Lemma 2.

◇ Initialization
CtxtChall ← ∅ // Additional Initialization.
CtxtNonChall ← ∅

▷ $(A_i \in \mathcal{S}^H)$-Write$(\langle A_i \to \vec{V}\rangle, m)$
$b \leftarrow \$\{0,1\}$ // Sample bit b uniformly at random.
if $b = 0$ **:**
  $c \leftarrow \mathcal{O}_E\big(\mathtt{sig}, A_i, \vec{V}' \coloneqq (B_{\mathtt{pp}}, V_1, \ldots, V_{|\vec{V}|}), m, \mathrm{Set}(\vec{V}) \cap \overline{\mathcal{P}^H}\big)$
  CtxtChall ← CtxtChall ∪ $\{c\}$
else
  $c \leftarrow \Pi.E_{\mathtt{pp}}(\mathtt{ssk}_i, \vec{v}' \coloneqq (\mathtt{rpk}_{\mathtt{pp}}, v_1, \ldots, v_{|\vec{v}|}), m)$
  CtxtNonChall ← CtxtNonChall ∪ $\{c\}$
Define event $\xi_1{}'$ as: $\xi_1{}' \coloneqq \mathrm{CtxtChall} \cap \mathrm{CtxtNonChall} \neq \emptyset$
if $\xi_1{}'$ **:**
  Guess(0) // Makes reduction output 0 as its guess.
Output(Write(c))

▷ $(P \in \mathcal{F})$-Write$(\langle[\mathrm{Forge}]A_i \to \vec{V}\rangle, m)$
Output(Write(Forge$\big(A_i, \vec{V}, m, \mathrm{Set}(\vec{V}) \cap \overline{\mathcal{P}^H}\big)$))

▷ $(B_j \in \mathcal{R}^H)$-Read
outputList ← ∅
ciphertextSet ← ∅
for $(\mathtt{id}, c) \in$ Read **with** $c \notin$ ciphertextSet **:**
  ciphertextSet ← ciphertextSet ∪ $\{c\}$
  $(\langle A_i \to \vec{V}\rangle, m) \leftarrow$ Decryption(B_j, c)
  if $(\langle A_i \to \vec{V}\rangle, m) \neq \perp$ **:**
    outputList ← outputList ∪ $\{(\mathtt{id}, (\langle A_i \to \vec{V}\rangle, m))\}$
Output(GetDelivered$(B_j, \mathrm{outputList})$)

▷ (J)-SenderKeyPair$(A_i \in \mathcal{S})$
Output$(\mathcal{O}_{SK}(A_i))$

▷ $(P \in \overline{\mathcal{P}^H})$-Write$(c)$
Output(Write(c))

▷ $(P \in \overline{\mathcal{P}^H})$-Read
Output(Read)

◇ Termination
$b \leftarrow \$\{0,1\}$
Guess(b) // Makes reduction output b as its guess.

Algorithm 27 Reduction $\mathbf{C}^{\mathsf{Corr}\text{-}\xi_1}$ for Lemma 2. Below, we only specify the reduction for operations for which it differs from $\mathbf{C}^{\mathsf{OTR}\text{-}\xi_1}$.

▷ $(A_i \in \mathcal{S}^H)$-Write$(\langle A_i \to \vec{V}\rangle, m)$
$b \leftarrow \$\{0,1\}$ // Sample bit b uniformly at random.
if $b = 0$ **:**
  $\vec{v}' \coloneqq (\mathtt{rpk}_{\mathtt{pp}}, v_1, \ldots, v_{|\vec{V}|})$
  $\vec{s} \coloneqq (\perp, \mathtt{rsk}_1, \ldots, \mathtt{rsk}_{|\vec{V}|})$ // For each V_l: if $V_l \in \overline{\mathcal{P}^H}$, s_{l+1} is V_l's secret key; else is $\perp$.
  $c \leftarrow \Pi.Forge_{\mathtt{pp}}(\mathtt{spk}_i, \vec{v}', m, \vec{s})$
else
  $c \leftarrow \mathcal{O}_E\big(A_i, \vec{V}' \coloneqq (B_{\mathtt{pp}}, V_1, \ldots, V_{|\vec{V}|}), m\big)$
$\mathcal{O}_D(B_{\mathtt{pp}}, c)$ // Allows winning the correctness and forgery invalidity games.
Output(Write(c))

▷ $(P \in \overline{\mathcal{P}^H})$-Write$(c)$
Output(Write(c))

Algorithm 28 Reduction $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1}$ for Lemma 2. As for Algorithm 27, we only specify the reduction for operations for which it differs from $\mathbf{C}^{\mathsf{OTR}\text{-}\xi_1}$.

$\triangleright$ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V}\rangle, m)$
$b \leftarrow \$\{0,1\}$ // Sample bit b uniformly at random.
if $b = 0$ **:**
$c \leftarrow \mathcal{O}_{Forge}(A_i, \vec{V}' \coloneqq (B_{\mathtt{pp}}, V_1, \ldots, V_{|\vec{V}|}), m, \mathrm{Set}(\vec{V}) \cap \overline{\mathcal{P}^H})$
else
$c \leftarrow \Pi.E_{\mathtt{pp}}(\mathtt{ssk}_i, \vec{v}' \coloneqq (\mathtt{rpk}_{\mathtt{pp}}, v_1, \ldots, v_{|\vec{v}|}), m)$
$\mathcal{O}_D(B_{\mathtt{pp}}, c)$ // Allows winning the correctness and forgery invalidity games.
OUTPUT(WRITE(c))

$\triangleright$ $(P \in \overline{\mathcal{P}^H})$-WRITE$(c)$
OUTPUT(WRITE(c))

probability that $\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_2}$ does not win $\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}$ due to event $\xi_{2,0}$ occurring by reducing to winning either the consistency or the forgery invalidity games. This means the probability of $\xi_{2,0}$ occurring when $\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_2}$ is interacting with $\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}$ is bounded by

$$Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons\text{-}1}\text{-}\xi_2}) + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2}).$$

Obtaining the final bound: Putting these facts together then allows to upper bound the probability of event ξ_2 occurring:

$$\begin{aligned} &Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_2}) + Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons\text{-}0}\text{-}\xi_2}) + Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons\text{-}1}\text{-}\xi_2}) \\ &+ Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}\text{-}\xi_2}) + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2}). \end{aligned}$$

□

E Application Semantics of MDVS Game Notions

Recall $\mathcal{F} \coloneqq \mathcal{S} \cup \mathcal{R}$. We now establish composable semantics for the MDVS game-based notions.

Theorem 13. *Consider simulator* sim *defined in Algorithms 34 and 37, reductions* $\mathbf{C}^0$, $\mathbf{C}^{\mathsf{Cons}\text{-}H}$, $\mathbf{C}^{\mathsf{Cons}}$, $\mathbf{C}^{\mathsf{Corr}}$, $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}$, $\mathbf{C}^{\mathsf{R\text{-}Unforg}}$ *and* $\mathbf{C}^{\mathsf{OTR}}$ *(defined, respectively, in Algorithms 35, 36 and 38, Algorithms 35, 36 and 39, Algorithms 35, 36 and 40, Algorithms 35, 36 and 41, Algorithms 35, 36 and 42, Algorithms 35, 36 and 43 and Algorithms 35, 36 and 44), and reductions* $\perp^J \cdot \mathbf{C}^{\mathsf{OTR}\text{-}\xi_1}$, $\perp^J \cdot \mathbf{C}^{\mathsf{Corr}\text{-}\xi_1}$, $\perp^J \cdot \mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1}$, $\perp^J \cdot \mathbf{C}^{\mathsf{OTR}\text{-}\xi_2}$, $\perp^J \cdot \mathbf{C}^{\mathsf{Cons}\text{-}0\text{-}\xi_2}$, $\perp^J \cdot \mathbf{C}^{\mathsf{Cons}\text{-}1\text{-}\xi_2}$, $\perp^J \cdot \mathbf{C}^{\mathsf{Corr}\text{-}\xi_2}$ *and* $\perp^J \cdot \mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2}$ *(where* $\mathbf{C}^{\mathsf{OTR}\text{-}\xi_1}$, $\mathbf{C}^{\mathsf{Corr}\text{-}\xi_1}$, $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1}$, $\mathbf{C}^{\mathsf{OTR}\text{-}\xi_2}$, $\mathbf{C}^{\mathsf{Cons}\text{-}0\text{-}\xi_2}$, $\mathbf{C}^{\mathsf{Cons}\text{-}1\text{-}\xi_2}$, $\mathbf{C}^{\mathsf{Corr}\text{-}\xi_2}$ *and* $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2}$ *are the reductions from Lemmata 4 and 5).*

Algorithm 29 Reduction $\mathbf{C}^{\mathsf{OTR}\text{-}\xi_2}$ for Lemma 3.

▷ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V}\rangle, m)$
  $c \leftarrow \mathcal{O}_E\big(\mathtt{sig}, A_i, \vec{V}' \coloneqq (B_{\mathsf{pp}}, V_1, \ldots, V_{|\vec{V}|}), m, \mathrm{Set}(\vec{V}) \cap \overline{\mathcal{P}^H}\big)$
  if $c \in$ Dec **:** // Event ξ_2
    Define event $\xi_{2,0} \coloneqq (\mathrm{Dec}[c] \neq \bot)$
    Define event $\xi_{2,1} \coloneqq (\mathrm{Dec}[c] = \bot)$
    if $\xi_{2,0}$ **:**
      Guess(0) // Makes reduction output guess 0.
    else // Event $\xi_{2,1}$ occurred
      Guess(1) // Reduction outputs guess 1.
  OUTPUT(WRITE(c))

▷ $(P \in \mathcal{F})$-WRITE$(\langle[\mathrm{Forge}]A_i \to \vec{V}\rangle, m)$
  OUTPUT(WRITE(FORGE$\big(A_i, \vec{V}, m, \mathrm{Set}(\vec{V}) \cap \overline{\mathcal{P}^H}\big)$))

▷ $(B_j \in \mathcal{R}^H)$-READ
  outputList $\leftarrow \emptyset$
  ciphertextSet $\leftarrow \emptyset$
  for $(\mathtt{id}, c) \in$ READ **with** $c \notin$ ciphertextSet **:**
    ciphertextSet $\leftarrow$ ciphertextSet $\cup \{c\}$
    $(\langle A_i \to \vec{V}\rangle, m) \leftarrow$ DECRYPTION(B_j, c)
    if $(\langle A_i \to \vec{V}\rangle, m) \neq \bot$ **:**
      outputList $\leftarrow$ outputList $\cup \{(\mathtt{id}, (\langle A_i \to \vec{V}\rangle, m))\}$
  OUTPUT(GETDELIVERED$(B_j,$ outputList$)$)

▷ (J)-SENDERKEYPAIR$(A_i \in \mathcal{S})$
  OUTPUT$(\mathcal{O}_{SK}(A_i))$

▷ $(P \in \overline{\mathcal{P}^H})$-WRITE$(c)$
  $\alpha \leftarrow \mathcal{O}_D(B_{\mathsf{pp}}, c)$
  if $\alpha \neq \mathtt{test}$ **:** // c not written before by WRITE operation at some interface $A_i \in \mathcal{S}^H$.
    Dec$[c] \leftarrow \alpha$
  OUTPUT(WRITE(c))

▷ $(P \in \overline{\mathcal{P}^H})$-READ
  OUTPUT(READ)

◇ TERMINATION
  $b \leftarrow \$\{0, 1\}$
  Guess(b) // Makes reduction output b as its guess.

Algorithm 30 Reduction $\mathbf{C}^{\mathsf{Cons}\text{-}0\text{-}\xi_2}$ for Lemma 3. We only specify the reduction for operations for which it differs from $\mathbf{C}^{\mathsf{OTR}\text{-}\xi_2}$.

◇ INITIALIZATION
  CtxtChall $\leftarrow \emptyset$ // Additional Initialization.

▷ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V}\rangle, m)$
  $c \leftarrow \mathcal{O}_E\big(A_i, \vec{V}' \coloneqq (B_{\mathsf{pp}}, V_1, \ldots, V_{|\vec{V}|}), m\big)$
  CtxtChall $\leftarrow$ CtxtChall $\cup \{c\}$
  if $c \in$ Dec **:**
    $\mathcal{O}_D(B_{\mathsf{pp}}, c)$
  OUTPUT(WRITE(c))

▷ $(P \in \overline{\mathcal{P}^H})$-WRITE$(c)$
  if $c \notin$ CtxtChall **:**
    Dec$[c] \leftarrow \mathcal{O}_D(B_{\mathsf{pp}}, c)$
  OUTPUT(WRITE(c))

Algorithm 31 Reduction $\mathbf{C}^{\mathsf{Cons\text{-}1\text{-}}\xi_2}$ for Lemma 3. As for Algorithm 30, we only specify the reduction for operations for which it differs from $\mathbf{C}^{\mathsf{OTR\text{-}}\xi_2}$.

$\diamond$ Initialization
CtxtChall $\leftarrow \emptyset$ // Additional Initialization.

$\triangleright$ $(A_i \in \mathcal{S}^H)$-Write$(\langle A_i \to \vec{V}\rangle, m)$
$\vec{v}' \coloneqq (\mathtt{rpk}_{\mathtt{pp}}, v_1, \ldots, v_{|\vec{V}|})$
$\vec{s} \coloneqq (\bot, \mathtt{rsk}_1, \ldots, \mathtt{rsk}_{|\vec{V}|})$ // For each V_l: if $V_l \in \overline{\mathcal{P}^H}$, s_{l+1} is V_l's secret key; else is $\bot$.
$c \leftarrow \Pi.Forge_{\mathtt{pp}}(\mathtt{spk}_i, \vec{v}', m, \vec{s})$
CtxtChall $\leftarrow$ CtxtChall $\cup \{c\}$
if $c \in$ Dec **:**
  $\mathcal{O}_D(B_{\mathtt{pp}}, c)$
Output(Write(c))

$\triangleright$ $(P \in \overline{\mathcal{P}^H})$-Write$(c)$
if $c \notin$ CtxtChall **:**
  Dec$[c] \leftarrow \mathcal{O}_D(B_{\mathtt{pp}}, c)$
Output(Write(c))

Algorithm 32 Reduction $\mathbf{C}^{\mathsf{Corr\text{-}}\xi_2}$ for Lemma 3. We only present this reduction for completeness (note that it is the same reduction as $\mathbf{C}^{\mathsf{Cons\text{-}0\text{-}}\xi_2}$).

$\diamond$ Initialization
CtxtChall $\leftarrow \emptyset$ // Additional Initialization.

$\triangleright$ $(A_i \in \mathcal{S}^H)$-Write$(\langle A_i \to \vec{V}\rangle, m)$
$c \leftarrow \mathcal{O}_E\big(A_i, \vec{V}' \coloneqq (B_{\mathtt{pp}}, V_1, \ldots, V_{|\vec{V}|}), m\big)$
CtxtChall $\leftarrow$ CtxtChall $\cup \{c\}$
if $c \in$ Dec **:**
  $\mathcal{O}_D(B_{\mathtt{pp}}, c)$
Output(Write(c))

$\triangleright$ $(P \in \overline{\mathcal{P}^H})$-Write$(c)$
if $c \notin$ CtxtChall **:**
  Dec$[c] \leftarrow \mathcal{O}_D(B_{\mathtt{pp}}, c)$
Output(Write(c))

Algorithm 33 Reduction $\mathbf{C}^{\mathsf{Forge\text{-}Invalid\text{-}}\xi_2}$ for Lemma 3. As before, we only specify the reduction for operations for which it differs from $\mathbf{C}^{\mathsf{OTR\text{-}}\xi_2}$.

$\diamond$ Initialization
CtxtChall $\leftarrow \emptyset$ // Additional Initialization.

$\triangleright$ $(A_i \in \mathcal{S}^H)$-Write$(\langle A_i \to \vec{V}\rangle, m \in \mathcal{M})$
$c \leftarrow \mathcal{O}_{Forge}\big(A_i, \vec{V}' \coloneqq (B_{\mathtt{pp}}, V_1, \ldots, V_{|\vec{V}|}), m, \mathrm{Set}(\vec{V}) \cap \overline{\mathcal{P}^H}\big)$
CtxtChall $\leftarrow$ CtxtChall $\cup \{c\}$
if $c \in$ Dec **:**
  $\mathcal{O}_D(B_{\mathtt{pp}}, c)$
Output(Write(c))

$\triangleright$ $(P \in \overline{\mathcal{P}^H})$-Write$(c)$
if $c \notin$ CtxtChall **:**
  Dec$[c] \leftarrow \mathcal{O}_D(B_{\mathtt{pp}}, c)$
Output(Write(c))

For any distinguisher $\mathbf{D}$*,*

$$
\begin{aligned}
&\Delta^{\mathbf{D}}\Bigg(\mathsf{Snd}^{\mathcal{S}^H}\,\mathsf{Rcv}^{\mathcal{R}^H}\,\mathsf{Forge}^{\mathcal{F}}\perp^{J}[\mathbf{KGA},\mathsf{Net}\cdot\mathbf{INS}],\\
&\quad \mathsf{sim}^{\overline{\mathcal{P}^H}}\cdot\mathsf{Otr}^{\overline{\mathcal{P}^H}}\cdot\begin{bmatrix}\mathsf{Net}\cdot\perp^{\mathsf{Auth\text{-}Intf}}\cdot\left[\langle A_i\to\vec{V}\rangle^{\{A_i\}\cup\overline{\mathcal{P}^H}}_{Set(\vec{V})\cup\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+}\\ \left[\langle[Forge]A_i\to\vec{V}\rangle^{\mathcal{F}}_{\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+}\end{bmatrix}\Bigg)\\
&\leq 4\cdot\Big(Adv^{\mathsf{OTR}}(\mathbf{D}\perp^{J}\mathbf{C}^{\mathsf{OTR}\text{-}\xi_1})+Adv^{\mathsf{Corr}}(\mathbf{D}\perp^{J}\mathbf{C}^{\mathsf{Corr}\text{-}\xi_1})\\
&\qquad+Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{D}\perp^{J}\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1})\Big)\\
&\quad+Adv^{\mathsf{OTR}}(\mathbf{D}\perp^{J}\mathbf{C}^{\mathsf{OTR}\text{-}\xi_2})+Adv^{\mathsf{Cons}}(\mathbf{D}\perp^{J}\mathbf{C}^{\mathsf{Cons}\text{-}0\text{-}\xi_2})\\
&\quad+Adv^{\mathsf{Cons}}(\mathbf{D}\perp^{J}\mathbf{C}^{\mathsf{Cons}\text{-}1\text{-}\xi_2})+Adv^{\mathsf{Corr}}(\mathbf{D}\perp^{J}\mathbf{C}^{\mathsf{Corr}\text{-}\xi_2})\\
&\quad+Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{D}\perp^{J}\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2})+Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons}})\\
&\quad+Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}})+Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}})+Adv^{\mathsf{R\text{-}Unforg}}(\mathbf{DC}^{\mathsf{R\text{-}Unforg}})\\
&\quad+Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}}).
\end{aligned}
$$

Theorem 14. *Consider simulator* sim *(analogous to the one from Theorem 13),*[13] *reductions* $\mathbf{C}^0$*,* $\mathbf{C}^{\mathsf{Cons}\text{-}H}$*,* $\mathbf{C}^{\mathsf{Cons}}$*,* $\mathbf{C}^{\mathsf{Corr}}$*,* $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}$ *and* $\mathbf{C}^{\mathsf{OTR}}$ *(analogous to the ones given for Theorem 13), and reductions* $\mathbf{C}^{\mathsf{OTR}\text{-}\xi_1}$*,* $\mathbf{C}^{\mathsf{Corr}\text{-}\xi_1}$*,* $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1}$*,* $\mathbf{C}^{\mathsf{OTR}\text{-}\xi_2}$*,* $\mathbf{C}^{\mathsf{Cons}\text{-}0\text{-}\xi_2}$*,* $\mathbf{C}^{\mathsf{Cons}\text{-}1\text{-}\xi_2}$*,* $\mathbf{C}^{\mathsf{Corr}\text{-}\xi_2}$ *and* $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2}$ *(i.e. the reductions from Lemmata 4 and 5). For any distinguisher* $\mathbf{D}$*,*

$$
\begin{aligned}
&\Delta^{\mathbf{D}}(\mathsf{Snd}^{\mathcal{S}^H}\,\mathsf{Rcv}^{\mathcal{R}^H}\,\mathsf{Forge}^{\mathcal{F}}[\mathbf{KGA},\mathsf{Net}\cdot\mathbf{INS}],\\
&\quad \mathsf{sim}^{\overline{\mathcal{P}^H}}\cdot\ \mathsf{ConfAnon}^{\overline{\mathcal{P}^H}}\cdot\ \mathsf{Otr}^{\overline{\mathcal{P}^H}}\cdot\quad\begin{bmatrix}\mathsf{Net}\ \cdot\ \left[\langle A_i\to\vec{V}\rangle^{\overline{\mathcal{P}^H}}_{Set(\vec{V})\cup\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+}\\ \left[\langle[Forge]A_i\to\vec{V}\rangle^{\mathcal{F}}_{\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+}\end{bmatrix})\\
&\leq 4\cdot\Big(Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1})+Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}\text{-}\xi_1})\\
&\qquad+Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1})\Big)\\
&\quad+Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_2})+Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons}\text{-}0\text{-}\xi_2})+Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons}\text{-}1\text{-}\xi_2})\\
&\quad+Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}\text{-}\xi_2})+Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2})\\
&\quad+Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons}})+Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}})+Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}})\\
&\quad+Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}}).
\end{aligned}
$$

[13] The only difference between the simulators is that the one for this theorem does not abort when input a valid signature by an honest sender: in this case the simulator proceeds as if the sender were dishonest.

E.1 Proofs

For simplicity, in Algorithm 34 we describe the behavior of the simulators we will consider in the proofs of Theorems 13 and 14 for the (sub-)interfaces of dishonest parties that correspond to an interface of the **KGA** resource in the real world system. Similarly, in Algorithm 35 we describe the behavior of the reductions we will consider in these proofs for the same (sub-)interfaces.

Algorithm 34 Description of (part of) the simulators considered in the proofs of Theorems 13 and 14 for the (sub-)interfaces of (dishonest) parties that correspond to an interface of the **KGA** resource in the real world system. In the following, $k \in \mathbb{N}$ is the (implicitly defined) security parameter.

◇ Initialization
INS-Initialization
Sigs $\leftarrow \emptyset$
$\mathtt{pp} \leftarrow \Pi.S(1^k)$
for $A_i \in \mathcal{S}$ **:** $(\mathtt{spk}_i, \mathtt{ssk}_i) \leftarrow \Pi.G_S(\mathtt{pp})$
for $B_j \in \mathcal{R}$ **:** $(\mathtt{vpk}_j, \mathtt{vsk}_j) \leftarrow \Pi.G_V(\mathtt{pp})$

▷ $(P \in \overline{\mathcal{P}^H})$-PublicParameters
Output($\mathtt{pp}$)

▷ $(P \in \overline{\mathcal{P}^H})$-SenderKeyPair($A_i \in \overline{\mathcal{S}^H}$)
Output($\mathtt{spk}_i, \mathtt{ssk}_i$)

▷ $(P \in \overline{\mathcal{P}^H})$-SenderPublicKey($A_i \in \mathcal{S}$)
Output($\mathtt{spk}_i$)

▷ $(P \in \overline{\mathcal{P}^H})$-ReceiverKeyPair($B_j \in \overline{\mathcal{R}^H}$)
Output($\mathtt{vpk}_j, \mathtt{vsk}_j$)

▷ $(P \in \overline{\mathcal{P}^H})$-ReceiverPublicKey($B_j \in \mathcal{R}$)
Output($\mathtt{vpk}_j$)

Algorithm 35 Description of the reductions considered in the proofs of Lemmata 4 and 5 and Theorems 13 and 14 for the (sub-)interfaces of (dishonest) parties that correspond to **KGA** interface in the real world system, plus the Deliver interface.

◇ Initialization
INS-Initialization
SigVal $\leftarrow \emptyset$
(SigHon,SigForge,SigDis) $\leftarrow (\emptyset, \emptyset, \emptyset)$
for $A_i \in \mathcal{S}$ **:** $\mathcal{O}_{SPK}(A_i)$
for $B_j \in \mathcal{R}$ **:** $\mathcal{O}_{VPK}(B_j)$
for $B_j \in \mathcal{R}$ **:** Received$[B_j] \leftarrow \emptyset$

▷ $(P \in \overline{\mathcal{P}^H})$-PublicParameters
Output($\mathcal{O}_{PP}$)

▷ Deliver(P, $\mathtt{id}$)
Received$[P] \leftarrow$ Received$[P] \cup \{\mathtt{id}\}$

▷ $(P \in \overline{\mathcal{P}^H})$-SenderKeyPair($A_i \in \overline{\mathcal{S}^H}$)
Output($\mathcal{O}_{SK}(A_i)$)

▷ $(P \in \overline{\mathcal{P}^H})$-SenderPublicKey($A_i \in \mathcal{S}$)
Output($\mathcal{O}_{SPK}(A_i)$)

▷ $(P \in \overline{\mathcal{P}^H})$-ReceiverKeyPair($B_j \in \overline{\mathcal{R}^H}$)
Output($\mathcal{O}_{VK}(B_j)$)

▷ $(P \in \overline{\mathcal{P}^H})$-ReceiverPublicKey($B_j \in \mathcal{R}$)
Output($\mathcal{O}_{VPK}(B_j)$)

E.1.1 Helper Claims We now state two results analogous to Lemmata 2 and 3 that help in simplifying the proofs of Theorems 13 and 14. Consider the following events:

Algorithm 36 Helper functions used in the reductions considered in the proofs of Theorems 13 and 14.

◇ GetDelivered(P, list) // Local procedure.
filteredList $\leftarrow \emptyset$
for $(\mathtt{id}, x) \in$ list **with** $\mathtt{id} \in$ Received$[P]$ **:**
  filteredList $\leftarrow$ filteredList $\cup \{(\mathtt{id}, x)\}$
return filteredList

◇ Forge($A_i, \vec{V}, m, \mathcal{C}$) // Local procedure.
$\vec{v} \coloneqq (v_1, \ldots, v_{|\vec{V}|})$
$\vec{s} \coloneqq (\mathtt{vsk}_1, \ldots, \mathtt{vsk}_{|\vec{V}|})$ // For each V_l: if $V_l \in \mathcal{C}$, s_{l+1} is V_l's secret key; else it is $\perp$.
return $\Pi.Forge_{\mathsf{pp}}(\mathtt{spk}_i, \vec{v}, m, \vec{s})$

Event ξ_1 There are two Write queries at the interface of an honest party $A_i \in \mathcal{S}^H$ that output $\mathtt{id}$ and $\mathtt{id}'$ with $\mathtt{id} \neq \mathtt{id}'$, such that the contents of the registers with these identifiers (i.e. $\mathtt{id}$ and $\mathtt{id}'$) are the same.

Event ξ_2 There is a Write query at a dishonest party's interface with input quadruple $(m, \sigma, (A_i, \vec{V}))$ that outputs a register identifier $\mathtt{id}$ and there is a later Write query at the interface of an honest party $A_i \in \mathcal{S}^H$ that outputs a register identifier $\mathtt{id}'$ such that the contents of the registers with identifiers $\mathtt{id}$ and $\mathtt{id}'$ are the same.

Lemma 4. *For any distinguisher* $\mathbf{D}$*, the probability that event* ξ_1 *occurs when it interacts with the real world system* $\mathbf{R}$ *(Equation 5.1) is upper bounded by*

$$4 \cdot \Big(Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_1}) + Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}\text{-}\xi_1}) + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1}) \Big).$$

where $\mathbf{C}^{\mathsf{OTR}\text{-}\xi_1}$, $\mathbf{C}^{\mathsf{Corr}\text{-}\xi_1}$ *and* $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1}$ *are the* MDVS *analogous of reductions Algorithms 14 and 26, Algorithms 14 and 27, and Algorithms 14 and 28.*

Lemma 5. *For any distinguisher* $\mathbf{D}$*, the probability that event* ξ_2 *occurs when it interacts with the real world system* $\mathbf{R}$ *(Equation 5.1) is upper bounded by*

$$Adv^{\mathsf{OTR}}(\mathbf{DC}^{\mathsf{OTR}\text{-}\xi_2}) + Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons}\text{-}0\text{-}\xi_2}) + Adv^{\mathsf{Cons}}(\mathbf{DC}^{\mathsf{Cons}\text{-}1\text{-}\xi_2}) + Adv^{\mathsf{Corr}}(\mathbf{DC}^{\mathsf{Corr}\text{-}\xi_2}) + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2}).$$

where $\mathbf{C}^{\mathsf{OTR}\text{-}\xi_2}$, $\mathbf{C}^{\mathsf{Cons}\text{-}0\text{-}\xi_2}$, $\mathbf{C}^{\mathsf{Cons}\text{-}1\text{-}\xi_2}$, $\mathbf{C}^{\mathsf{Corr}\text{-}\xi_2}$ *and* $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2}$ *are the* MDVS *analogous of reductions Algorithms 14 and 29, Algorithms 14 and 30, Algorithms 14 and 31, Algorithms 14 and 32, and Algorithms 14 and 33.*

The proofs follow from arguments that are analogous to their MDRS-PKE counterparts (see Lemmata 2 and 3).

E.1.2 Proof of Theorem 13

Algorithm 37 Description of the behavior of the simulator considered in the proof of Theorem 13 for the (sub-)interfaces of dishonest parties that correspond to an interface of $\mathsf{Net} \cdot \mathbf{INS}$ in the real world system. In the following, $\mathbf{T}$ is as defined in Equations 5.6 and E.1.

▷ $(P \in \overline{\mathcal{P}^H})$-WRITE$(m, \sigma, (A_i, \vec{V}))$
  if $\vec{V} \in \overline{\mathcal{R}^H}^+$ **:** OUTPUT(**INS**-WRITE$(m, \sigma, (A_i, \vec{V}))$)
  Consider least $l \in \{1, \ldots, |\vec{V}|\}$ with $V_l \in \mathcal{P}^H$
  if $\neg \Pi.Vfy_{\mathtt{pp}}(\mathtt{spk}_i, \mathtt{vsk}_l, \vec{v}, m, \sigma)$ **:** OUTPUT(**INS**-WRITE$(m, \sigma, (A_i, \vec{V}))$)
  if $A_i \in \mathcal{S}^H$ **:** Abort // Signature forged.
  $\mathtt{id}' \leftarrow \mathbf{S_{rep}}$-WRITE$(\langle A_i \to \vec{V} \rangle, m)$
  Sigs[$\mathtt{id}'$] $\leftarrow (m, \sigma, (A_i, \vec{V}))$
  OUTPUT($\mathtt{id}'$)

▷ $(P \in \overline{\mathcal{P}^H})$-READ
  for $(\langle A_i \to \vec{V} \rangle, \mathtt{id}, m) \in P$-$\mathbf{S_{rep}}$-READ **with** $\mathtt{id} \notin$ Sigs **:**
    $\sigma \leftarrow \Pi.Sig_{\mathtt{pp}}(\mathtt{ssk}_i, \vec{v}, m)$
    Sigs[id] $\leftarrow (m, \sigma, (A_i, \vec{V}))$
  OUTPUT(Sigs $\cup$ **INS**-READ)

Algorithm 38 Reduction $\mathbf{C}^0$ for Theorem 13. This is a "dummy" reduction: when connected to the MDVS oracles defined in Section 2.3, it has exactly the same behavior as the real world resource.

▷ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V} \rangle, m)$
  $\sigma \leftarrow \mathcal{O}_S(A_i, \vec{V}, m)$
  id $\leftarrow$WRITE$(m, \sigma, (A_i, \vec{V}))$
  OUTPUT(id)

▷ $(P \in \mathcal{F})$-WRITE$(\langle [\text{Forge}] A_i \to \vec{V} \rangle, m)$
  $\sigma \leftarrow$ FORGE$(A_i, \vec{V}, m, \text{Set}(\vec{V}) \cap \overline{\mathcal{P}^H})$
  id $\leftarrow$WRITE$(m, \sigma, (A_i, \vec{V}))$
  OUTPUT(id)

▷ $(B_j \in \mathcal{R}^H)$-READ
  (list, sigSet) $\leftarrow (\emptyset, \emptyset)$
  for $(\mathtt{id}, (m, \sigma, (A_i, \vec{V}))) \in$ READ **with** $(B_j \in \text{Set}(\vec{V})) \wedge ((m, \sigma, (A_i, \vec{V})) \notin \text{sigSet})$ **:**
    sigSet $\leftarrow$ sigSet $\cup \{(m, \sigma, (A_i, \vec{V}))\}$
    if $\mathcal{O}_V(A_i, B_j, \vec{V}, m, \sigma)$ **:** list $\leftarrow$ list $\cup \{(\mathtt{id}, (\langle A_i \to \vec{V} \rangle, m))\}$
  OUTPUT(GETDELIVERED$(B_j$, list))

▷ $(P \in \overline{\mathcal{P}^H})$-WRITE$(m, \sigma, (A_i, \vec{V}))$
  id $\leftarrow$WRITE$(m, \sigma, (A_i, \vec{V}))$
  OUTPUT(id)

▷ $(P \in \overline{\mathcal{P}^H})$-READ
  OUTPUT(READ)

Algorithm 39 Reduction $\mathbf{C}^{\mathsf{Cons\text{-}H}}$ for Theorem 13. Below we only show the differences (highlighted) relative to $\mathbf{C}^0$.

▷ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V}\rangle, m)$
  $\sigma \leftarrow \mathcal{O}_S(A_i, \vec{V}, m)$
  if $(m, \sigma, (A_i, \vec{V})) \in$ SigHon **:** Abort // Event ξ_1: Signature Already Exists
  SigHon $\leftarrow$ SigHon $\cup \{(m, \sigma, (A_i, \vec{V}))\}$
  id $\leftarrow$WRITE$(m, \sigma, (A_i, \vec{V}))$
  OUTPUT(id)

▷ $(P \in \mathcal{F})$-WRITE$(\langle [\text{Forge}]A_i \to \vec{V}\rangle, m)$
  $\sigma \leftarrow$ FORGE$(A_i, \vec{V}, m, \text{Set}(\vec{V}) \cap \overline{\mathcal{P}^H})$
  SigForge $\leftarrow$ SigForge $\cup \{(m, \sigma, (A_i, \vec{V}))\}$
  id $\leftarrow$WRITE$(m, \sigma, (A_i, \vec{V}))$
  OUTPUT(id)

▷ $(P \in \overline{\mathcal{P}^H})$-WRITE$(m, \sigma, (A_i, \vec{V}))$
  id $\leftarrow$WRITE$(m, \sigma, (A_i, \vec{V}))$
  if $(m, \sigma, (A_i, \vec{V})) \notin$ SigHon $\cup$ SigForge **:**
    SigDis $\leftarrow$ SigDis $\cup \{(m, \sigma, (A_i, \vec{V}))\}$
  OUTPUT(id)

Algorithm 40 Reduction $\mathbf{C}^{\mathsf{Cons}}$ for Theorem 13. Below we only show the differences (highlighted) relative to $\mathbf{C}^{\mathsf{Cons\text{-}H}}$.

▷ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V}\rangle, m)$
  $\sigma \leftarrow \mathcal{O}_S(A_i, \vec{V}, m)$
  if $(m, \sigma, (A_i, \vec{V})) \in$ SigHon $\cup$ SigDis **:** Abort // Event $\xi_1 \vee \xi_2$: Signature Already Exists
  SigHon $\leftarrow$ SigHon $\cup \{(m, \sigma, (A_i, \vec{V}))\}$
  id $\leftarrow$WRITE$(m, \sigma, (A_i, \vec{V}))$
  if $\vec{V} \notin \overline{\mathcal{R}^H}^+$ **:** SigVal[id] $\leftarrow \mathcal{O}_V(A_i, V_l, \vec{V}, m, \sigma)$, for least $l \in [|\vec{V}|]$ with $V_l \in \mathcal{P}^H$
  OUTPUT(id)

▷ $(P \in \mathcal{F})$-WRITE$(\langle [\text{Forge}]A_i \to \vec{V}\rangle, m)$
  $\sigma \leftarrow$ FORGE$(A_i, \vec{V}, m, \text{Set}(\vec{V}) \cap \overline{\mathcal{P}^H})$
  SigForge $\leftarrow$ SigForge $\cup \{(m, \sigma, (A_i, \vec{V}))\}$
  id $\leftarrow$WRITE$((m, \sigma, (A_i, \vec{V})))$
  if $\vec{V} \notin \overline{\mathcal{R}^H}^+$ **:** SigVal[id] $\leftarrow \mathcal{O}_V(A_i, V_l, \vec{V}, m, \sigma)$, for least $l \in [|\vec{V}|]$ with $V_l \in \mathcal{P}^H$
  OUTPUT(id)

▷ $(P \in \overline{\mathcal{P}^H})$-WRITE$(m, \sigma, (A_i, \vec{V}))$
  id $\leftarrow$WRITE$(m, \sigma, (A_i, \vec{V}))$
  if $(m, \sigma, (A_i, \vec{V})) \notin$ SigHon $\cup$ SigForge **:**
    if $(m, \sigma, (A_i, \vec{V})) \notin$ SigDis **:**
      if $\vec{V} \notin \overline{\mathcal{R}^H}^+$ **:** SigVal[id] $\leftarrow \mathcal{O}_V(A_i, V_l, \vec{V}, m, \sigma)$, for least $l \in [|\vec{V}|]$ with $V_l \in \mathcal{P}^H$
    SigDis $\leftarrow$ SigDis $\cup \{(m, \sigma, (A_i, \vec{V}))\}$
  OUTPUT(id)

Algorithm 41 Reduction $\mathbf{C}^{\mathsf{Corr}}$ for Theorem 13. Below we only show the differences (highlighted) relative to $\mathbf{C}^{\mathsf{Cons}}$.

▷ $(B_j \in \mathcal{R}^H)$-READ
$(\text{list}, \text{sigSet}) \leftarrow (\emptyset, \emptyset)$
for $(\texttt{id}, (m, \sigma, (A_i, \vec{V}))) \in$ READ **with** $(B_j \in \text{Set}(\vec{V})) \wedge ((m, \sigma, (A_i, \vec{V})) \notin \text{sigSet})$ **:**
  $\text{sigSet} \leftarrow \text{sigSet} \cup \{(m, \sigma, (A_i, \vec{V}))\}$
  if $\texttt{SigVal[id]}$ **:** $\text{list} \leftarrow \text{list} \cup \{(\texttt{id}, (\langle A_i \rightarrow \vec{V}\rangle, m))\}$
OUTPUT(GETDELIVERED(B_j, list))

Algorithm 42 Reduction $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}$ for Theorem 13. Below we only show the differences (highlighted) relative to $\mathbf{C}^{\mathsf{Corr}}$.

▷ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \rightarrow \vec{V}\rangle, m)$
$\sigma \leftarrow \mathcal{O}_S(A_i, \vec{V}, m)$
if $(m, \sigma, (A_i, \vec{V})) \in \text{SigHon} \cup \text{SigDis}$ **:** Abort // Event $\xi_1 \vee \xi_2$: Signature Already Exists
$\text{SigHon} \leftarrow \text{SigHon} \cup \{(m, \sigma, (A_i, \vec{V}))\}$
$\texttt{id} \leftarrow$WRITE$(m, \sigma, (A_i, \vec{V}))$
$\texttt{SigVal[id]} \leftarrow 1$
OUTPUT(id)

▷ $(P \in \mathcal{F})$-WRITE$(\langle[\text{Forge}]A_i \rightarrow \vec{V}\rangle, m)$
$\sigma \leftarrow$ FORGERED$(A_i, \vec{V}, m, \text{Set}(\vec{V}) \cap \overline{\mathcal{P}^H})$
$\text{SigForge} \leftarrow \text{SigForge} \cup \{(m, \sigma, (A_i, \vec{V}))\}$
$\texttt{id} \leftarrow$WRITE$(m, \sigma, (A_i, \vec{V}))$
if $\vec{V} \notin \overline{\mathcal{R}^H}^+$ **:** $\text{SigVal[id]} \leftarrow \mathcal{O}_V(A_i, V_l, \vec{V}, m, \sigma)$, for least $l \in [|\vec{V}|]$ with $V_l \in \mathcal{P}^H$
OUTPUT(id)

// Local procedure.
◇ FORGERED$(A_i, \vec{V}, m, \mathcal{C})$
return $\mathcal{O}_{Forge}(A_i, \vec{V}, m, \text{Set}(\vec{V}) \cap \overline{\mathcal{P}^H})$

Algorithm 43 Reduction $\mathbf{C}^{\mathsf{R\text{-}Unforg}}$ for Theorem 13. Below we only show the differences (highlighted) relative to $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}$.

▷ $(P \in \mathcal{F})$-WRITE$(\langle[\text{Forge}]A_i \rightarrow \vec{V}\rangle, m)$
$\sigma \leftarrow$ FORGE$(A_i, \vec{V}, m, \text{Set}(\vec{V}) \cap \overline{\mathcal{P}^H})$
$\text{SigForge} \leftarrow \text{SigForge} \cup \{(m, \sigma, (A_i, \vec{V}))\}$
$\texttt{id} \leftarrow$WRITE$((m, \sigma, (A_i, \vec{V})))$
$\texttt{SigVal[id]} \leftarrow 0$
OUTPUT(id)

Algorithm 44 Reduction $\mathbf{C}^{\mathsf{OTR}}$ for Theorem 13. Below we only show the differences (highlighted) relative to $\mathbf{C}^{\mathsf{R\text{-}Unforg}}$.

▷ $(A_i \in \mathcal{S}^H)$-WRITE$(\langle A_i \to \vec{V}\rangle, m)$
  $\sigma \leftarrow \mathcal{O}_S(\mathtt{sig}, A_i, \vec{V}, m, \mathrm{Set}(\vec{V}) \cap \overline{\mathcal{P}^H})$
  if $(m, \sigma, (A_i, \vec{V})) \in \mathrm{SigHon} \cup \mathrm{SigDis}$ **:** Abort // Event $\xi_1 \vee \xi_2$: Signature Already Exists
  $\mathrm{SigHon} \leftarrow \mathrm{SigHon} \cup \{(m, \sigma, (A_i, \vec{V}))\}$
  $\mathtt{id} \leftarrow$WRITE$(m, \sigma, (A_i, \vec{V}))$
  $\mathrm{SigVal}[\mathtt{id}] \leftarrow 1$
  OUTPUT(id)

▷ $(P \in \overline{\mathcal{P}^H})$-WRITE$(m, \sigma, (A_i, \vec{V}))$
  $\mathtt{id} \leftarrow$WRITE$(m, \sigma, (A_i, \vec{V}))$
  if $(m, \sigma, (A_i, \vec{V})) \notin \mathrm{SigHon} \cup \mathrm{SigForge}$ **:**
    if $(m, \sigma, (A_i, \vec{V})) \notin \mathrm{SigDis}$ **:**
      if $(A_i \in \mathcal{S}^H)$ **:** $\mathrm{SigVal}[\mathtt{id}] \leftarrow 0$
      else if $\vec{V} \notin \overline{\mathcal{R}^H}^+$ **:** $\mathrm{SigVal}[\mathtt{id}] \leftarrow \mathcal{O}_V(A_i, V_l, \vec{V}, m, \sigma)$, for least $l \in [|\vec{V}|]$ with $V_l \in \mathcal{P}^H$
    $\mathrm{SigDis} \leftarrow \mathrm{SigDis} \cup \{(m, \sigma, (A_i, \vec{V}))\}$
  OUTPUT(id)

Proof. Let $\mathbf{R}$ be the real world system

$$\mathbf{R} := \mathsf{Snd}^{\mathcal{S}^H}\,\mathsf{Rcv}^{\mathcal{R}^H}\,\mathsf{Forge}^{\mathcal{F}}\,\bot^J[\mathbf{KGA}, \mathsf{Net} \cdot \mathbf{INS}],$$

$\mathbf{T}$ be the ideal repository defined in Equation 5.6, i.e.

$$\mathbf{T} := \mathsf{Otr}^{\overline{\mathcal{P}^H}} \cdot \left[\begin{array}{c} \mathsf{Net} \cdot \bot^{\mathsf{Auth\text{-}Intf}} \cdot \left[\langle A_i \to \vec{V}\rangle^{\{A_i\}\cup\overline{\mathcal{P}^H}}_{\mathrm{Set}(\vec{V})\cup\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+} \\ \left[\langle [\mathrm{Forge}]A_i \to \vec{V}\rangle^{\mathcal{F}}_{\overline{\mathcal{P}^H}}\right]_{A_i\in\mathcal{S},\vec{V}\in\mathcal{R}^+} \end{array}\right], \tag{E.1}$$

and let sim be the simulator specified in Algorithms 34 and 37. The remainder of the proof bounds $\Delta^{\mathbf{D}}(\mathbf{R}, \mathsf{sim}^{\overline{\mathcal{P}^H}}\mathbf{T})$ by proceeding in a sequence of hybrids. In the following, we consider the reduction systems defined in the lemma's statement.

$\mathbf{R} \rightsquigarrow \mathbf{C}^0\mathbf{G}^{\mathsf{Cons}}$: It is easy to see that $\mathbf{R}$ and $\mathbf{C}^0\mathbf{G}^{\mathsf{Cons}}$ are the same sequence of conditional probability distributions by considering, on one hand, the definition of $\mathbf{R}$—i.e. the definitions of converters Snd, Rcv and Forge (Algorithm 5), the definition of the $\mathbf{KGA}$ resource (Algorithms 4 and 6), and the definitions of $\mathbf{INS}$ (Algorithm 1) and of Net (Algorithm 3)—and, on the other hand, the definition of $\mathbf{C}^0\mathbf{G}^{\mathsf{Cons}}$—i.e. the definition of $\mathbf{G}^{\mathsf{Cons}}$ and its oracles (Definition 3 and Section 3.1) and the definition of $\mathbf{C}^0$ (Algorithms 35, 36 and 38). It follows

$$\Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{C}^0\mathbf{G}^{\mathsf{Cons}}) = 0.$$

$\mathbf{C}^0\mathbf{G}^{\mathsf{Cons}} \rightsquigarrow \mathbf{C}^{\mathsf{Cons\text{-}H}}\mathbf{G}^{\mathsf{Cons}}$: It is easy to see that $\mathbf{C}^0\mathbf{G}^{\mathsf{Cons}}$ and $\mathbf{C}^{\mathsf{Cons\text{-}H}}\mathbf{G}^{\mathsf{Cons}}$ are the same sequence of conditional probability distributions conditioned on event

ξ_1 not occurring (see Section E.1.1). By Lemma 4, it follows

$$\begin{aligned}\Delta^{\mathbf{D}}(\mathbf{C}^{0}\mathbf{G}^{\mathsf{Cons}}, \mathbf{C}^{\mathsf{Cons\text{-}H}}\mathbf{G}^{\mathsf{Cons}}) \leq 4 \cdot \Big(& Adv^{\mathsf{OTR}}(\mathbf{D}\perp^{J}\mathbf{C}^{\mathsf{OTR}\text{-}\xi_1}) \\ & + Adv^{\mathsf{Corr}}(\mathbf{D}\perp^{J}\mathbf{C}^{\mathsf{Corr}\text{-}\xi_1}) \\ & + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{D}\perp^{J}\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_1})\Big).\end{aligned}$$

$\mathbf{C}^{\mathsf{Cons\text{-}H}}\mathbf{G}^{\mathsf{Cons}} \rightsquigarrow \mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}$**:** As for the previous step $\mathbf{C}^{0}\mathbf{G}^{\mathsf{Cons}} \rightsquigarrow \mathbf{C}^{\mathsf{Cons\text{-}H}}\mathbf{G}^{\mathsf{Cons}}$, it is easy to see that the two systems are the exact same sequence of conditional probability distributions conditioned on event ξ_2 not occurring (see Section E.1.1). It then follows by Lemma 5

$$\begin{aligned}\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Cons\text{-}H}}\mathbf{G}^{\mathsf{Cons}}, \mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}) \leq\ & Adv^{\mathsf{OTR}}(\mathbf{D}\perp^{J}\mathbf{C}^{\mathsf{OTR}\text{-}\xi_2}) \\ & + Adv^{\mathsf{Cons}}(\mathbf{D}\perp^{J}\mathbf{C}^{\mathsf{Cons\text{-}0}\text{-}\xi_2}) + Adv^{\mathsf{Cons}}(\mathbf{D}\perp^{J}\mathbf{C}^{\mathsf{Cons\text{-}1}\text{-}\xi_2}) \\ & + Adv^{\mathsf{Corr}}(\mathbf{D}\perp^{J}\mathbf{C}^{\mathsf{Corr}\text{-}\xi_2}) + Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{D}\perp^{J}\mathbf{C}^{\mathsf{Forge\text{-}Invalid}\text{-}\xi_2}).\end{aligned}$$

$\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}} \rightsquigarrow \mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$**:** The only difference between $\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$ and $\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}$ is that in $\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$ the validity of each quadruple $q \coloneqq (m, \sigma, (A_i, \vec{V}))$ where $\vec{V}$ contains at least one honest receiver, such that q was either generated by a WRITE operation at the interface of an honest sender $A_i \in \mathcal{S}^H$ or input to a WRITE operation at the interface of a dishonest party $P \in \overline{\mathcal{P}^H}$ is only verified once, and this verification is made for the first party in vector $\vec{V}$ who is honest (as described in the reduction; for more details see Algorithms 40 and 41). Given $\mathbf{C}^{\mathsf{Cons}}$ does not query for the secret key of any receiver $B_j \in \mathcal{R}^H$, the advantage of a distinguisher $\mathbf{D}$ in distinguishing $\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}$ and $\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$ is bounded by the advantage of adversary $\mathbf{D}\mathbf{C}^{\mathsf{Cons}}$ in winning the consistency game $\mathbf{G}^{\mathsf{Cons}}$: note that $\mathbf{C}^{\mathsf{Cons}}$ makes a query to $\mathcal{O}_V$ with the first honest verifier appearing in vector $\vec{V}$ when queried for any WRITE operation, and when queried for a READ operation at the interface of a receiver $B_j \in \mathcal{R}^H$ makes verification queries to $\mathcal{O}_V$ for each $(\mathtt{id}, (m, \sigma, (A_i, \vec{V})))$ in the reduction's internal repository $\mathbf{INS}$. This implies

$$\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}, \mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}) \leq Adv^{\mathsf{Cons}}(\mathbf{D}\mathbf{C}^{\mathsf{Cons}}).$$

$\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}} \rightsquigarrow \mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$**:** System $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$ differs from $\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$ in that signatures generated by WRITE operations—whose vector $\vec{V}$ contains at least one honest receiver—issued at the interface of honest senders are no longer verified by a query to $\mathcal{O}_V$ and instead the result of their verification is simply assumed to be 1. $\mathbf{D}$'s advantage in distinguishing $\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}$ and $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$ is upper bounded by the advantage of $\mathbf{D}\mathbf{C}^{\mathsf{Corr}}$ in winning the correctness game $\mathbf{G}^{\mathsf{Corr}}$, implying

$$\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}, \mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}) \leq Adv^{\mathsf{Corr}}(\mathbf{D}\mathbf{C}^{\mathsf{Corr}}).$$

$\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}} \rightsquigarrow \mathbf{C}^{\mathsf{R\text{-}Unforg}}\mathbf{G}^{\mathsf{R\text{-}Unforg}}$**:** In $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$, signatures generated on queries $\textsc{Write}(\langle[\text{Forge}]A_i \to \vec{V}\rangle, m)$ are assumed to be invalid. Distinguisher $\mathbf{D}$'s distinguishing advantage between $\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$ and $\mathbf{C}^{\mathsf{R\text{-}Unforg}}\mathbf{G}^{\mathsf{R\text{-}Unforg}}$ is upper bounded by the advantage of $\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}}$ in winning the forgery invalidity game $\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}$, implying

$$\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Forge\text{-}Invalid}}\mathbf{G}^{\mathsf{Forge\text{-}Invalid}}, \mathbf{C}^{\mathsf{R\text{-}Unforg}}\mathbf{G}^{\mathsf{R\text{-}Unforg}}) \leq Adv^{\mathsf{Forge\text{-}Invalid}}(\mathbf{DC}^{\mathsf{Forge\text{-}Invalid}}).$$

$\mathbf{C}^{\mathsf{R\text{-}Unforg}}\mathbf{G}^{\mathsf{R\text{-}Unforg}} \rightsquigarrow \mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}$**:** The main things to note for this step are that 1. $\mathbf{D}$ has no access to the secret key of any honest receiver $B_j \in \mathcal{R}^H$; 2. since J has a converter $\perp$ attached to her interface, $\mathbf{D}$ has no access to the secret key of any honest sender $A_i \in \mathcal{S}^H$; and 3. the only case in which $\mathbf{C}^{\mathsf{R\text{-}Unforg}}\mathbf{G}^{\mathsf{R\text{-}Unforg}}$ may differ from $\mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}$ is if $\mathbf{D}$ makes a query for a $\textsc{Write}$ operation at the interface of a dishonest party $P \in \overline{\mathcal{S}^H} \cup \overline{\mathcal{R}^H}$ with input quadruple $(m, \sigma, (A_i, \vec{V}))$ where $A_i \in \mathcal{S}^H$ and $\vec{V}$ contains at least one honest receiver, and the verification of σ by the first honest receiver in $\vec{V}$ with respect to sender A_i, vector $\vec{V}$ and message m is 1, and yet there was no matching $\textsc{Write}$ operation at the interface of A_i that resulted in the same quadruple (in particular with the same signature). This allows us to bound the advantage of $\mathbf{D}$ in distinguishing $\mathbf{C}^{\mathsf{R\text{-}Unforg}}\mathbf{G}^{\mathsf{R\text{-}Unforg}}$ and $\mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}$ by the advantage of $\mathbf{DC}^{\mathsf{R\text{-}Unforg}}$ in winning $\mathbf{G}^{\mathsf{R\text{-}Unforg}}$, implying

$$\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{R\text{-}Unforg}}\mathbf{G}^{\mathsf{R\text{-}Unforg}}, \mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}) \leq Adv^{\mathsf{R\text{-}Unforg}}(\mathbf{DC}^{\mathsf{R\text{-}Unforg}}).$$

$\mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}} \rightsquigarrow \mathsf{sim}^{\overline{\mathcal{P}^H}}\mathbf{T}$**:** It is easy to see, by considering the definitions of $\mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}$—i.e. of $\mathbf{C}^{\mathsf{OTR}}$, Algorithms 35, 36 and 44, and of $\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}$ and its oracles, Definition 5 and Section 3.1—and $\mathsf{sim}^{\overline{\mathcal{P}^H}}\mathbf{T}$—i.e. of simulator sim, Algorithms 34 and 37, and of $\mathbf{T}$, Equation E.1—that these are perfectly indistinguishable; it follows

$$\Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}, \mathsf{sim}^{\overline{\mathcal{P}^H}}\mathbf{T}) = 0.$$

To conclude the proof we use triangle inequality:

$$
\begin{aligned}
\Delta^{\mathbf{D}}(\mathbf{R}, \mathsf{sim}^{\overline{\mathcal{P}^H}}\mathbf{T}) &\leq \Delta^{\mathbf{D}}(\mathbf{R}, \mathbf{C}^0\mathbf{G}^{\mathsf{Cons}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^0\mathbf{G}^{\mathsf{Cons}}, \mathbf{C}^{\text{Cons-H}}\mathbf{G}^{\mathsf{Cons}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\text{Cons-H}}\mathbf{G}^{\mathsf{Cons}}, \mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Cons}}\mathbf{G}^{\mathsf{Cons}}, \mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}, \mathbf{C}^{\text{R-Unforg}}\mathbf{G}^{\text{R-Unforg}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{Corr}}\mathbf{G}^{\mathsf{Corr}}, \mathbf{C}^{\text{Forge-Invalid}}\mathbf{G}^{\text{Forge-Invalid}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\text{Forge-Invalid}}\mathbf{G}^{\text{Forge-Invalid}}, \mathbf{C}^{\text{R-Unforg}}\mathbf{G}^{\text{R-Unforg}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\text{R-Unforg}}\mathbf{G}^{\text{R-Unforg}}, \mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{0}}^{\mathsf{OTR}}, \mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}) \\
&\quad + \Delta^{\mathbf{D}}(\mathbf{C}^{\mathsf{OTR}}\mathbf{G}_{\mathbf{1}}^{\mathsf{OTR}}, \mathsf{sim}^{\overline{\mathcal{P}^H}}\mathbf{T}) \\
&\leq 4 \cdot \Big(Adv^{\mathsf{OTR}}(\mathbf{D}\perp^J\mathbf{C}^{\text{OTR-}\xi_1}) + Adv^{\mathsf{Corr}}(\mathbf{D}\perp^J\mathbf{C}^{\text{Corr-}\xi_1}) \\
&\qquad\quad + Adv^{\text{Forge-Invalid}}(\mathbf{D}\perp^J\mathbf{C}^{\text{Forge-Invalid-}\xi_1}) \Big) \\
&\quad + Adv^{\mathsf{OTR}}(\mathbf{D}\perp^J\mathbf{C}^{\text{OTR-}\xi_2}) + Adv^{\mathsf{Cons}}(\mathbf{D}\perp^J\mathbf{C}^{\text{Cons-0-}\xi_2}) \\
&\quad + Adv^{\mathsf{Cons}}(\mathbf{D}\perp^J\mathbf{C}^{\text{Cons-1-}\xi_2}) + Adv^{\mathsf{Corr}}(\mathbf{D}\perp^J\mathbf{C}^{\text{Corr-}\xi_2}) \\
&\quad + Adv^{\text{Forge-Invalid}}(\mathbf{D}\perp^J\mathbf{C}^{\text{Forge-Invalid-}\xi_2}) + Adv^{\mathsf{Cons}}(\mathbf{D}\mathbf{C}^{\mathsf{Cons}}) \\
&\quad + Adv^{\mathsf{Corr}}(\mathbf{D}\mathbf{C}^{\mathsf{Corr}}) + Adv^{\text{Forge-Invalid}}(\mathbf{D}\mathbf{C}^{\text{Forge-Invalid}}) \\
&\quad + Adv^{\text{R-Unforg}}(\mathbf{D}\mathbf{C}^{\text{R-Unforg}}) + Adv^{\mathsf{OTR}}(\mathbf{D}\mathbf{C}^{\mathsf{OTR}}).
\end{aligned}
$$

□

E.1.3 Proof of Theorem 14

Proof. A proof can be obtained by applying modifications to the proof of Theorem 13 that are analogous to the changes made in the proof of Theorem 12 (from the proof of Theorem 11). □