Anonymous Rerandomizable RCCA Encryption

Manoj Prabhakaran Mike Rosulek
Department of Computer Science, University of Illinois, Urbana-Champaign.
{mmp,rosulek}@cs.uiuc.edu

April 1, 2007

Abstract

We give the first rerandomizable Replayabale-CCA (RCCA [7]) secure encryption scheme, answering an open problem posed in [19]. We also extend the notion of receiver-anonymity (or key-privacy) [3] to the setting of rerandomizable RCCA-secure encryption schemes, and show that our protocol enjoys this property as well. To justify our definitions, we define a powerful "Replayabale Message Posting" functionality in the Universally Composable (UC) framework, and show that any encryption scheme that satisfies our definitions of rerandomizability, RCCA security, and receiver-anonymity is a UC-secure implementation of this functionality. We point out that this is perhaps the most sophisticated functionality that has been UC-securely realized in the standard model, without global setups, super-polynomial simulation, or an honest majority assumption.

Our encryption scheme is a non-trivial extension of the popular Cramer-Shoup encryption, which we call the *Double-strand Cramer-Shoup scheme*. Its security is based on the standard DDH assumption.

Contents

1 Introduction

Non-malleability and rerandomizability are opposing demands to place on an encryption scheme. Non-malleability requires that an adversary should not be able to use one ciphertext to produce another one which would decrypt to a related ciphertext. Rerandomizability on the other hand requires that anyone can alter a ciphertext into another ciphertext in an unlinkable way, such that both will decrypt to the same plaintext. It is precisely this requirement that is formalized in the notion of Rerandomizable Replayable-CCA security [19]. Achieving it (without restricting security to generic group adversaries) was left as an open problem there.

Another very useful property that a public-key encryption scheme can have is receiver-anonymity (or key-privacy) [3]. Indeed, Bellare, et al. [3] show that the popular CCA-secure Cramer-Shoup encryption scheme [11] already enjoys such a security guarantee.

In this work, we combine these two properties to obtain a powerful encryption tool. In a system with multiple users it is unlikely that rerandomizability by itself would be useful. For instance, while rerandomizability allows unlinkability of multiple encryptions in terms of their contents, without anonymity they could all be linked as going to the same receiver. Adding anonymity brings out the power of rerandomizability and yields a potent cryptographic primitive.

Our results. We present the first rerandomizable Replayable-CCA-secure public-key encryption scheme. We also show that it is receiver-anonymous. Our scheme is a non-trivial variant of the Cramer-Shoup scheme, which we call the *Double-strand Cramer-Shoup* encryption. As with that scheme, the security of our scheme is based on the Decisional Diffie Hellman (DDH) assumption.

Going further, we give a combined security definition in the Universally-Composable (UC) security framework by specifying a "Replayable Message Posting" functionality $\mathcal{F}_{\text{RMP}}$, and show that any scheme which satisfies our notions of rerandomizability, Replayable-CCA security, and receiver-anonymity is also a *UC-secure realization* of the functionality $\mathcal{F}_{\text{RMP}}$ (without any setup assumptions, and without superpolynomial simulation). The functionality $\mathcal{F}_{\text{RMP}}$ allows users to post messages and obtain "handles" for them. These handles can then be used by any party to repost a message and obtain a new handle. The handles do not reveal the content or the receiver of the messages. Nor is it revealed if a handle is freshly created or obtained by reposting an old handle. The adversary is allowed to post and repost handles, and arbitrarily interact with the environment (which instructs honest players to post, repost or retrieve messages).

Once we achieve this UC-secure functionality, simple modifications can be made to add extra functionality like replay-testability (by the designated receiver) and authentication.

We point out that this is perhaps the most sophisticated functionality that has been UC-securely realized in the standard model, without super-polynomial simulation and without using an honest majority assumption. However, in this work, we do restrict ourselves to static adversaries (as opposed to adversaries who corrupt the parties adaptively).

Related work. Replayable-CCA (RCCA) security was defined by Canetti et. al. [7] as a *relaxation* of CCA security (also often called CCA2 security, to distinguish it from the "lunch-time attack" version). Gröth [19] raised the problem of rerandomizable RCCA security, but achieved only a weaker notion. For the full-fledged RCCA security, [19] presents a scheme with a security analysis in the idealized so-called generic group model. Our work improves on [19], answering the open problem posed there. Our encryption scheme is also more efficient compared to that of [19].

Bellare et. al. [3] formally introduced notions of key-privacy and showed that the El Gamal encryption scheme (which is only secure against chosen plaintext attacks) and the Cramer-Shoup scheme already achieve their notions of key-privacy. Our work is an extension of theirs in the sense that we consider anonymity for rerandomizable encryption schemes.

As mentioned before, our encryption scheme is based on the Cramer-Shoup scheme [11, 12], which in turn is modeled after the El Gamal encryption [16]. Security of these schemes are based on the DDH assumption (see, e.g. [4]). In [12], Cramer and Shoup show a wide range of encryption schemes based on various assumptions, which provide CCA security, under a framework subsuming the original Cramer-Shoup scheme [11]. We believe that much of their generalization can be adapted to our current work as well, though we do not investigate this in detail here. (See the remark in the concluding section.)

[27] and [1] introduced a variant of RCCA security, called *benignly malleable* security or gCCA2 security. It is similar to RCCA security, but uses an arbitrary equivalence relation over ciphertexts to define the notion of replaying. However, these definitions preclude rerandomizability by requiring that the equivalence relation be efficiently computable *publicly*. A simple extension of our scheme achieves a similar definition of RCCA security and receiver-anonymity, where the replay-equivalence relation is computable only by the ciphertext's designated recipient. Such a functionality precludes perfect rerandomization, though our modification does achieve a computational relaxation of the rerandomization requirement.

A motivating example. We sketch the problem which motivated our investigation, and for which the functionality $\mathcal{F}_{\text{RMP}}$ provides a fitting solution. Consider the following network routing problem (adapted from a private communication [20]) in a futuristic secure network, with the following requirements: (1) each packet should carry its entire path to the destination, (2) each node, including the node from which a packet originates, should not get any information about the path encoded in the packet, other than the length of the path and where that node must send the packet to next (next-hop), and (3) there should be a mechanism to broadcast link-failure information so that any node holding a packet can check if the failed link occurs in (the remainder of) the packet's path (without gaining any other information). The nodes obtain objects encoding the paths by listening to advertisements broadcast by one's immediate neighbors who have already discovered a path (i.e., have an object encoding a path) to the particular destination. It is beyond the scope of this paper to consider specific networking scenarios where such a scheme could be deployed; however, we mention peer-to-peer networks as a likely situation where the nodes in a path may wish to remain anonymous.

In fact a lot of work on anonymity has been carried out both in theoretical cryptography and in government funded implementations [8, 9, 10, 26, 18, 13, 5, 15, 21] (also see references collected at [25]). Of these, the problem most similar to the one above is "Onion Routing" [18, 13, 5, 21]. However, it turns out that adding requirement (3) above makes the above problem fundamentally different from previous works.

We point out some of the implicit anonymity requirements in our example. If a node receives multiple path objects (during path discovery or during packet routing), it should not be able to tell if the paths share common links (any more than it can tell by just knowing what is considered legitimate for it to know, like the next/previous hops in the paths). In general, given many paths, one should not be able to tell anything more than is revealed by their lengths and one's own immediate neighbors in each path, and when given link-failure information, it should only let each

node know which paths use that link, and nothing more about the identity of the link.

An approach to solving this routing problem is by employing the functionality $\mathcal{F}_{\text{RMP}}$. The path objects are collections of handles, one for each node in the path, inserted into the collection by the node itself. These handles correspond to self-addressed messages; the message includes the next-hop information, which would let a node route a packet correctly later when it reaches the node. Whenever a path object is passed on, during path discovery or during routing, each handle in it is turned into a new handle, and the collection of handles re-sorted. In routing, the data packets are also carried using handles which are renewed each time. Finally, to declare a link failure, the corresponding private key is broadcast (this feature is not modeled in $\mathcal{F}_{\text{RMP}}$, but is not hard to incorporate). The details of the problem and the solution are deferred to future work.

2 Definitions

We call a function ν *negligible* if it asymptotically approaches zero faster than any inverse polynomial; that is, $\nu(n) = n^{-\omega(1)}$. We call a function *noticeable* if it is non-negligible. A probability is *overwhelming* if it is negligibly close to 1 (negligible in an implicit security parameter). In all the encryption schemes we consider, the security parameter is the number of bits needed to represent an element from the underlying cyclic group.

2.1 Encryption and security definitions

In this section we formally define the strong notions mentioned above. First we give the syntax of a perfectly rerandomizable encryption scheme, and then state two security requirements for it. These security definitions are of the indistinguishability flavour: specific games are described to capture the security notion and the adversary is required to have only a negligible advantage in the game. Then we give a strong and convincing definition of security which combines all these properties. This definition is in the simulation-based setting with an arbitrary network environment (also variously known as UC (Universally Composable) security [6, 22], environmental security [17, 24] and network-aware security [23]). As a justification of our indistinguishability-based definitions, we show in Theorem 2 that a rerandomizable encryption scheme satisfying both indistinguishability definitions also satisfies the simulation-based definition.

2.1.1 Perfectly rerandomizable encryption (syntax and correctness)

A perfectly rerandomizable encryption scheme consists of four efficient algorithms (i.e., polynomial time in the security parameter): KeyGen, Enc, Rerand, and Dec.

- KeyGen is a randomized algorithm which generates a *public key* PK and a *private key* SK.
- Enc is a randomized encryption algorithm which takes a *plaintext* msg (from a plaintext space) and a public key produced by KeyGen, and outputs a *ciphertext* $\zeta \leftarrow \mathsf{Enc}_{PK}(\mathsf{msg})$.
- Rerand is a randomized algorithm which takes a ciphertext and outputs another ciphertext.
- Dec is a deterministic decryption algorithm which takes a private key and a ciphertext, and outputs either a plaintext or an error indicator $\perp$.

The correctness requirements on Dec are as follows:

- For any key pair (PK, SK) generated by KeyGen, all plaintexts msg and all honestly-generated ciphertexts $\zeta \leftarrow \mathsf{Enc}_{PK}(\mathsf{msg})$, we must have $\mathsf{Dec}_{SK}(\zeta) = \mathsf{msg}$.

The correctness requirement on Rerand is as follows:

- For any message msg, any key pair (PK, SK), and any honestly generated ciphertext $\zeta \leftarrow \mathsf{Enc}_{PK}(\mathsf{msg})$, the distribution of $\mathsf{Rerand}(\zeta)$ is identical to that of $\mathsf{Enc}_{PK}(\mathsf{msg})$.
- For key pair (PK, SK), any (purported) ciphertext ζ, and any $\zeta' \leftarrow \mathsf{Rerand}(\zeta)$, we must have $\mathsf{Dec}_{SK}(\zeta') = \mathsf{Dec}_{SK}(\zeta)$.

In other words, rerandomizing an honestly generated ciphertext induces the same distribution as an independent encryption of the same message, while the only guarantee for an adversarially generated ciphertext is that rerandomization preserves the value of its decryption (under every key).[1]

2.1.2 Replayable CCA (RCCA) security

We use the definition of Replayable CCA security from Canetti, et al. [7]. The encryption scheme is said to be *RCCA secure* if the advantage of any PPT adversary in the following experiment is negligible.

1. **Setup:** Pick $(PK, SK) \leftarrow \mathsf{KeyGen}$. $\mathcal{A}^{\text{rcca}}$ is given PK.
2. **Phase I:** $\mathcal{A}^{\text{rcca}}$ gets access to decryption oracle $\mathsf{Dec}_{SK}(\cdot)$.
3. **Challenge:** Pick $b \leftarrow \{0, 1\}$. $\mathcal{A}^{\text{rcca}}$ outputs a pair of plaintexts $(\mathsf{msg}_0, \mathsf{msg}_1)$. Let $\zeta^* \leftarrow \mathsf{Enc}_{PK}(\mathsf{msg}_b)$. $\mathcal{A}^{\text{rcca}}$ is given ζ^*.
4. **Phase II:** $\mathcal{A}^{\text{rcca}}$ gets access to a *guarded decryption oracle* $\mathsf{GDec}_{SK}^{(\mathsf{msg}_1, \mathsf{msg}_2)}$ which on input ζ, first checks if $\mathsf{Dec}_{SK}(\zeta) \in \{\mathsf{msg}_1, \mathsf{msg}_2\}$. If so, it returns replay; otherwise it returns $\mathsf{Dec}_{SK}(\zeta)$.
5. **Guess:** $\mathcal{A}^{\text{rcca}}$ outputs a bit b'.

The advantage of the adversary is $\Pr[b' = b] - \frac{1}{2}$.

2.1.3 Receiver-anonymity

Our receiver-anonymity (or key-privacy) definition is similar to that of [3], but modified to allow replayability. An encryption scheme is said to be *RCCA receiver-anonymous* (or simply *receiver-anonymous*) if the advantage of any PPT adversary in the following experiment is negligible.

1. **Setup:** Pick $(PK_0, SK_0) \leftarrow \mathsf{KeyGen}$ and $(PK_1, SK_1) \leftarrow \mathsf{KeyGen}$. $\mathcal{A}^{\text{anon}}$ is given (PK_0, PK_1)
2. **Phase I:** $\mathcal{A}^{\text{anon}}$ gets access to decryption oracles $\mathsf{Dec}_{SK_0}(\cdot)$ and $\mathsf{Dec}_{SK_1}(\cdot)$
3. **Challenge:** Pick $b \leftarrow \{0, 1\}$. $\mathcal{A}^{\text{anon}}$ outputs a plaintext msg. Let $\zeta^* \leftarrow \mathsf{Enc}_{PK_b}(\mathsf{msg})$. $\mathcal{A}^{\text{anon}}$ is given ζ^*.

[1] A stronger requirement for rerandomization would be that for all (purported) ciphertexts ζ such that $\mathsf{Dec}_{SK}(\zeta) \neq \perp$, the distribution of $\mathsf{Rerand}(\zeta)$ is identical to that of $\mathsf{Enc}_{PK}(\mathsf{Dec}_{SK}(\zeta))$. This is stronger than we need in our target application. As adversarially generated ciphertexts may be addressed to the adversary anyway, we provide no guarantee on the distribution of rerandomizations of such ciphertext.

4. **Phase II:** $\mathcal{A}^{\text{anon}}$ gets access a *guarded decryption oracle* $\mathsf{GDec}^{\mathsf{msg}}_{SK_0,SK_1}(\cdot)$, on input ζ, first checks if $\mathsf{Dec}_{SK_0}(\zeta) = \mathsf{msg}$ or $\mathsf{Dec}_{SK_1}(\zeta) = \mathsf{msg}$. If so, it returns replay; otherwise it returns $(\mathsf{Dec}_{SK_0}(\zeta), \mathsf{Dec}_{SK_1}(\zeta))$.
5. **Guess:** $\mathcal{A}^{\text{anon}}$ outputs a bit b'.

The advantage of the adversary is $\Pr[b' = b] - \frac{1}{2}$.

In both experiments, we call ζ^* the *challenge ciphertext*.

2.2 Replayable message posting functionality

We define the "Replayable Message Posting" functionality $\mathcal{F}_{\text{RMP}}$ in the Universally Composable (UC) framework which concisely presents the security achieved by an encryption scheme which simultaneously enjoys the RCCA security, anonymity and rerandomizability. The functionality allows parties to publicly post messages which are represented by *handles*. The handles are arbitrary strings provided by the adversary. The adversary is only told who posts a message but not the contents of the message, nor to whom it is addressed. Only the party to whom it is addressed is allowed to obtain a message from the functionality.

Further, $\mathcal{F}_{\text{RMP}}$ provides a reposting functionality: any party can request $\mathcal{F}_{\text{RMP}}$ to "repost" (i.e, make a copy of) a handle handle that has already been posted. Requesting a repost does not reveal the message or its designated recipient. To everyone except the party requesting the repost, the repost behaves exactly like a normal message posting (unless the original handle handle was posted by the adversary or derived as a repost of a handle posted by the adversary).[2] In particular, neither the adversary nor the recipient of the message is told whether a handle is the result of a repost of a previous handle.

In Section 7 we define the functionality $\mathcal{F}_{\text{RMP}}$ in full detail. There we also incorporate a registration feature as part of $\mathcal{F}_{\text{RMP}}$ which allows parties to (dynamically) obtain identities.

2.3 Decisional Diffie-Hellman (DDH) assumption

Our constructions are proven secure under the standard Decisional Diffie-Hellman (DDH) assumption. Consider a (multiplicative) cyclic group $\mathbb{G} = \langle g \rangle$ of prime order p.[3] The *Decisional Diffie-Hellman (DDH) assumption in* $\mathbb{G}$ is that the following two distributions are computationally indistinguishable:

- DDH **distribution.** Pick random elements $g_1, g_2 \leftarrow \mathbb{G}$, and pick a random $v \leftarrow \mathbb{Z}_p$. Output (g_1, g_2, g_1^v, g_2^v).
- Rand **distribution.** Pick random elements $g_1, g_2 \leftarrow \mathbb{G}$, and pick random $v_1, v_2 \leftarrow \mathbb{Z}_p$. Output $(g_1, g_2, g_1^{v_1}, g_2^{v_2})$.

Our Double-strand Cramer-Shoup construction requires two (multiplicative) cyclic groups with a specific relationship: $\mathbb{G}$ of prime order p, and $\widehat{\mathbb{G}}$ of prime order q, where $\widehat{\mathbb{G}}$ is a subgroup of $\mathbb{Z}_p^*$. We require the DDH assumption to hold in both groups (with respect to the same security parameter).

[2] Note that we cannot expect to give any significant security guarantee when a handle that was originally posted by the adversary is reposted, because it could have been addressed to the adversary itself.

[3] It is likely that our security analysis can be extended to groups of orders with large prime factors, as is done in [12]. For simplicity, we do not consider this here.

As a concrete example, the DDH assumption is believed to hold in $\mathbb{QR}_p^*$, the group of quadratic residues modulo p, when p and $(p-1)/2$ are both prime (i.e, p is a *safe prime*). Supposing we have a sequence of prime numbers of the form $q, 2q+1, 4q+3$, the two groups $\widehat{\mathbb{G}} = \mathbb{QR}_{2q+1}^*$ and $\mathbb{G} = \mathbb{QR}_{4q+3}^*$ satisfy the needs of our construction. A sequence of primes of this form is called a *Cunningham chain* (of the first kind) of length 3 (See [2]). Such Cunningham chains are known to exist having q as large as 20,000 bits. It is conjectured that there are infinitely many such chains.

3 Motivating the double-strand construction

Conceptually, the crucial enabling idea in our construction is that of using two "strands" of ciphertexts (from an extended version of the Cramer-Shoup scheme) which can then be recombined with each other for rerandomization, without changing the encrypted value. To motivate this idea, we consider the much simpler case of security against chosen plaintext atacks. In this case a simple variant of the El Gamal encryption scheme works:

Recall that in an El Gamal encryption scheme over a group $\mathbb{G}$ of order p, a message $\mu \in \mathbb{G}$ is encrypted into $(g^v, \mu(g^a)^v)$, where $a \in \mathbb{Z}_p$ is the private key and g^a is the public key. To encrypt a message $\mu \in \mathbb{G}$, in the "Double-strand El Gamal" scheme, we pick a random exponents $v, w \in \mathbb{Z}_p$ and output the ciphertext $\zeta = (g^v, \mu g^{av}, g^w, g^{aw})$. To rerandomize a given ciphertext $\zeta = (A, B, C, D)$, let $\mathsf{Rerand}(\zeta) = (AC^r, BD^r, C^s, D^s)$ for random $r, s \in \mathbb{Z}_p$. Finally $\zeta = (A, B, C, D)$ can be decrypted as $\mu = BA^{-a}$. The security of this scheme can be proven under the DDH assumption in $\mathbb{G}$.

Our Double-strand Cramer-Shoup scheme tries to adapt this paradigm of rerandomization using two pieces of ciphertexts, to the case when adaptive chosen ciphertext attacks are considered. The main technical difficulty is in ensuring that the scheme remains secure even though two correlated "strands" of encryption are made available, and even though the additional functionality of rerandomization is provided, which the adversary may be able to exploit.

Cramer-Shoup encryption. We briefly recall the Cramer-Shoup scheme. We work over a group $\mathbb{G}$ of prime order p in which the DDH assumption is believed to hold. The private key is $b_1, b_2, c_1, c_2, d_1, d_2 \in \mathbb{Z}_p$ and the public key is $g_1, g_2 \in \mathbb{G}$, $B = \prod_{i=1}^2 g_i^{b_i}$, $C = \prod_{i=1}^2 g_i^{c_i}$, and $D = \prod_{i=1}^2 g_i^{d_i}$.

To encrypt a message msg, first pick $x \in \mathbb{Z}_p$. and for $i = 1, 2$ let and $X_i = g_i^x$. Encode msg into an element μ in $\mathbb{G}$. The ciphertext is $(X_1, X_2, \mu B^x, (CD^m)^x)$ where $m = \mathsf{H}(X_1, X_2, \mu B^x)$ (H being a collision resistant hash function from $\mathbb{G}^3$ to $\mathbb{Z}_p$).

In our scheme the ciphertext will contain two "strands," each one similar to a Cramer-Shoup ciphertext, and allowing rerandomization as in the El Gamal example above. However, instead of pairs we require 5-tuples of g_i, b_i, c_i, d_i (i.e., for $i = 1, \ldots, 5$). To allow for rerandomization, we use a direct encoding of the message for the exponent m (instead of a hash of part of the ciphertext). Finally, we thwart attacks which splice together multiple encryptions by correlating the two strands with shared random mask values.

Our security analysis is fairly more complicated than the ones in [11, 3, 19]. However all these analyses as well as the current one follow the basic idea that if the encryptions were to be carried out using the secret key in a "bad" way, it will remain indistinguishable from the actual encryptions (by the DDH assumption), but will also become independent of the message and the public key.

4 Results

We present two main results below. The first says that the DSCS encryption scheme presented in Section 5 meets the security guarantees defined earlier using standalone games. The second result says that any construction which meets these guarantees yields a UC-secure realization of the $\mathcal{F}_{\text{RMP}}$ functionality.

Theorem 1 *The DSCS encryption scheme (Section 5) is a perfectly rerandomizable encryption scheme which achieves the definitions of RCCA security and receiver-anonymity, under the DDH assumption in* $\mathbb{G}$ *and* $\widehat{\mathbb{G}}$.

Proof overview: The construction itself is given in Section 5. Here we give an outline of the structure of the proof, without getting into the details of the construction. The full proof appears in Section 6.

The correctness and perfect rerandomization properties of our construction are straight-forward to verify. We focus on proving RCCA security and receiver-anonymity. It is convenient to formulate our proof in terms of alternate encryption and decryption procedures – AltEnc which uses the private key for encryption and AltDec which uses the public key for decryption – as described below. We remark that this outline is similar to that used in previous proofs related to the Cramer-Shoup construction [11, 3, 12, 14]. However the implementation is significantly more involved in our case.

- First, we would like to argue that the ciphertexts hide the message and the public key used in the encryption. For this we describe an alternate encryption procedure AltEnc. When using this procedure to generate the challenge ciphertext in either security experiment, the difference is indistinguishable to any adversary (including when the adversary queries the decryption oracle which has the private keys). AltEnc will actually use the private key (not the public key) to generate the challenge ciphertext. That the adversary's behavior in the experiment will not change significantly when using such an alternate ciphertext follows from the DDH assumption in $\mathbb{G}$ and $\widehat{\mathbb{G}}$.

  Secondly, the ciphertexts produced by AltEnc are information-theoretically independent of the message (even given the public key) and of the public key (even given the message). This property will be useful in proving RCCA security and receiver-anonymity, respectively.

- However, an adversary may be able to get information about the message or the public key used in the encryption not only from the challenge ciphertext, but also from the answers to the decryption queries that it makes. Indeed, since the decryption oracle uses the private key there is a danger that information about the private key is leaked, especially when answering maliciously crafted ciphertexts. To show that this is not the case with our encryption scheme, we describe an alternate decryption procedure AltDec to be used in the security experiments, which can be implemented using only the public key(s) and challenge ciphertext (things which are already known to the adversary). AltDec will be computationally unbounded, but as it is only accessed as an oracle, this is not problematic. More importantly, its functionality will be indistinguishable from the honest decryption procedure (even when the adversary is computationally unbounded and is given a ciphertext generated by AltEnc).

We conclude that with these two modifications – alternate challenge ciphertext and the alternate decryption procedure – the adversary's view in either experiment is independent of the secret bit

b chosen in the experiment, and so the adversary has no advantage in guessing the bit. Further, the outcome of this modified experiment is negligibly different from the outcome of the original experiment, so the security claims follow. $\triangleleft$

Realizing $\mathcal{F}_{\text{RMP}}$. Given an encryption scheme which meets our RCCA security, rerandomizability and anonymity definitions, $\mathcal{F}_{\text{RMP}}$ can be realized in the standard UC model (and without super-polynomial simulation) using a simple "protocol." For simplicity we consider all communications to use a broadcast channel. The identities are the public keys, and the handles are the ciphertexts. Registering oneself involves generating a public-private key pair and publishing (broadcasting) the public key as one's identity. To post a message addressed to a party, one simply encrypts the message under the intended party's public key and publishes the encryption. Reposting a handle (i.e., a ciphertext) involves rerandomizing the ciphertext. To retrieve a message in a handle one simply decrypts it using one's private key.

Theorem 2 *Any rerandomizable encryption scheme which satisfies the definitions of Replayable-CCA2 security, rerandomizability and receiver-anonymity, is a secure realization of the replayable message posting functionality $\mathcal{F}_{\text{RMP}}$.*

To prove this theorem we need to demonstrate a simulator $\mathcal{S}$ for each adversary $\mathcal{A}$. The simulator internally runs $\mathcal{A}$ and behaves as follows.

- First it generates a key pair (PK^*, SK^*). Every time an honest party registers, it generates a key pair and sends the public key as that party's identity to $\mathcal{F}_{\text{RMP}}$ and to $\mathcal{A}$. Public keys published by $\mathcal{A}$ are registered by $\mathcal{S}$ as identities (noted by $\mathcal{F}_{\text{RMP}}$ as corrupted identities).
- When $\mathcal{F}_{\text{RMP}}$ receives a post or repost request – say the i^{th} one – from an honest party addressed to an honest party, it requests $\mathcal{S}$ for a handle handle_i (without telling it what the message is, who the receiver is or whether it was a post or a repost). At this point $\mathcal{S}$ picks a random message $\widetilde{\mathsf{msg}}_i$, encrypts it under PK^* and sends this ciphertext to $\mathcal{F}_{\text{RMP}}$ as handle_i. $\mathcal{F}_{\text{RMP}}$ will keep this handle marked as HONEST. In its simulation $\mathcal{S}$ uses this ciphertext as the one broadcast by the sender.
- When $\mathcal{A}$ sends out a ciphertext ζ, $\mathcal{S}$ behaves as follows:
  - First it checks if $\mathsf{Dec}_{SK^*}(\zeta) = \widetilde{\mathsf{msg}}_i$, the random message chosen for some previous i. If so, $\mathcal{S}$ requests $\mathcal{F}_{\text{RMP}}$ to *repost* handle_i.
  - Otherwise it checks if $\mathsf{Dec}_{SK}(\zeta) = \mathsf{msg} \neq \perp$ for any of the private keys SK that it picked while registering honest parties. If so, $\mathcal{S}$ asks $\mathcal{F}_{\text{RMP}}$ to post msg, addressed to the party for which SK was picked as the private key.
  - If ζ is not decrypted by any of these private keys then $\mathcal{S}$ requests $\mathcal{F}_{\text{RMP}}$ to post a dummy message to a dummy user.

  In all the above cases, $\mathcal{S}$ sends ζ as the handle to $\mathcal{F}_{\text{RMP}}$. $\mathcal{F}_{\text{RMP}}$ keeps this handle marked as ADVERSARIAL, as it was posted by the adversary (possibly as a repost of a handle marked HONEST).

- When $\mathcal{F}_{\text{RMP}}$ receives a post request from an honest party addressed to a corrupt identity, it sends the message being posted to $\mathcal{S}$, along with the identity to which it is addressed. $\mathcal{S}$ generates a handle by encrypting the message under the corrupt public key and sends it to $\mathcal{F}_{\text{RMP}}$ and $\mathcal{A}$ as before.
- When $\mathcal{F}_{\text{RMP}}$ receives a repost request for a handle marked ADVERSARIAL it sends $\mathcal{S}$ this handle. $\mathcal{S}$ applies the rerandomization procedure to this handle and returns the resulting handle to $\mathcal{F}_{\text{RMP}}$, which $\mathcal{F}_{\text{RMP}}$ publishes. In its simulation $\mathcal{S}$ uses the rerandomized ciphertext as the one broadcast by the sender.

In Section 8 a detailed analysis of this simulator is given.

5 Our construction

In this section we describe the Double-strand Cramer-Shoup (DSCS) encryption scheme referred to in Theorem 1. First, we introduce a simpler encryption scheme that is used as a component of the main scheme.

5.1 Double-strand malleable encryption scheme

We first define a rerandomizable encryption scheme which we call the "Double-strand malleable encryption" (DSME). As its name suggests, it is malleable, so it does not achieve our notions of RCCA security. However, it introduces the double-strand paradigm for rerandomization which we will use in our main construction. We will also use our DSME scheme as a component in our main construction, where its malleability will be a vital feature.

System parameters. A cyclic multiplicative group $\widehat{\mathbb{G}}$ of prime order q. $\widehat{\mathbb{G}}$ also acts as the message space for this scheme.

Key generation. Pick random generators $\widehat{g}_1, \widehat{g}_2, \widehat{g}_3$ from $\widehat{\mathbb{G}}$, and random $\mathbf{a} = (a_1, a_2, a_3)$ from $\mathbb{Z}_q$. The private key is $\mathbf{a}$. The public key consists of $\widehat{g}_1$, $\widehat{g}_2$, $\widehat{g}_3$, and $A = \prod_{j=1}^{3} \widehat{g}_j^{a_j}$

Encryption: $\mathsf{MEnc}_{MPK}(u \in \widehat{\mathbb{G}})$**:**

- Pick random $v, w \in \mathbb{Z}_q$. For $j = 1, 2, 3$: let $V_j = \widehat{g}_j^v$ and $W_j = \widehat{g}_j^w$.
- Output $(\mathbf{V}, uA^v, \mathbf{W}, A^w)$, where $\mathbf{V} = (V_1, V_2, V_3)$ and $\mathbf{W} = (W_1, W_2, W_3)$.

Decryption: $\mathsf{MDec}_{MSK}(U = (\mathbf{V}, A_V, \mathbf{W}, A_W))$**:**

- **Check ciphertext integrity:** Check if $A_W \stackrel{?}{=} \prod_{j=1}^{3} W_j^{a_j}$. If not, output $\perp$.
- **Derive message:** Output $A_V / \prod_{j=1}^{3} V_j^{a_j}$.

Rerandomization: $\mathsf{MRerand}(U = (\mathbf{V}, A_V, \mathbf{W}, A_W))$**.** The only randomness used in generating a ciphertext is the choice of v and w in $\widehat{\mathbb{G}}$. We can perfectly rerandomize both of these quantities by choosing random $s, t \in \mathbb{Z}_q$ and constructing a new ciphertext U' with corresponding randomness $v' = v + sw$, $w' = tw$, as follows:

- For $j = 1, 2, 3$, set $V'_j = V_j \cdot W_j^s$.

- For $j = 1, 2, 3$, set $W'_j = W_j^t$.
- $A'_V = A_V \cdot A_W^s$.
- $A'_W = A_W^t$.

The rerandomized ciphertext is $(\mathbf{V}', A'_V, \mathbf{W}', A'_W)$.

Homomorphism (multiplication by known value): Let $u' \in \widehat{\mathbb{G}}$ and let $U = (\mathbf{V}, A_V, \mathbf{W}, A_W)$ be a DSME ciphertext. We define the following operation:

$$u' \otimes U \stackrel{\text{def}}{=} (\mathbf{V}, u' \cdot A_V, \mathbf{W}, A_W)$$

It is not hard to see that for all private keys MSK, if $\mathsf{MDec}_{MSK}(U) \neq \perp$ then $\mathsf{MDec}_{MSK}(u' \otimes U) = u' \cdot \mathsf{MDec}_{MSK}(U)$, and if $\mathsf{MDec}_{MSK}(U) = \perp$ then $\mathsf{MDec}_{MSK}(U') = \perp$ as well.

Observe that this scheme is malleable under more than just multiplication by a known quantity. For instance, given an encryption of u, we can produce an encryption of u^r for a known $r \in \mathbb{Z}_q$. As it turns out, the way we use DSME in the main construction ensures that we achieve our final security despite such additional malleabilities.

Correctness properties. We require that the DSME scheme satisfy the correctness properties of a perfectly rerandomizable encryption scheme, as defined in Section 2.1.1. These are proved in Section 6.

5.2 Double-strand Cramer-Shoup encryption scheme

Now we give our main construction: an anonymous, rerandomizable, RCCA-secure encryption scheme called the "Double-strand Cramer-Shoup" (DSCS) encryption scheme. At the high level it has two Cramer-Shoup encryption strands, one carrying the message, and the other to help rerandomize it. But unlike in Cramer-Shoup we need to allow rerandomization, and so we do not use a prefix of the ciphertext itself in ensuring consistency; instead we use the message. Further, we need to restrict the ways in which recombinations of ciphertexts can yield related valid ciphertexts. This leads us to longer strands (more components). Finally, in order to prevent the possibility of mixing together two separate encryptions (say, in the manner in which rerandomizability allows two strands to be mixed together) – which will let the adversary check if they have the same message or public key, via a chosen ciphertext attack – we correlate the two strands of the ciphertext with shared random masks. These masks are random exponents which are separately encrypted using the malleable DSME scheme described above (so that they may be hidden from everyone but the designated recipient, but also be rerandomized via the DSME scheme's homomorphic operation). A formal description follows.

System parameters. A cyclic multiplicative group $\mathbb{G}$ of prime order p. A space of messages. An injective encoding $\mathsf{encode}_{\mathbb{G}}$ of messages into $\mathbb{G}$. An injective mapping $\mathsf{encode}_{\mathbb{Z}_p}$ of messages into $\mathbb{Z}_p$ (or into $\mathbb{Z}_p^*$, without any significant difference). These functions should be efficiently computable in both directions.

We also require a secure DSME scheme over a group $\widehat{\mathbb{G}}$ of prime order q, where $\widehat{\mathbb{G}}$ is also a subgroup of $\mathbb{Z}_p^*$. This is crucial, as the homomorphic operation $\otimes$ of the DSME scheme must coincide with multiplication in the exponent in $\mathbb{G}$.

Finally, we require a fixed vector $\mathbf{z} = (z_1, \ldots, z_5) \in (\mathbb{Z}_p)^5$ with a certain degenerate property. For our purposes, $\mathbf{z} = (0, 0, 0, 1, 1)$ is sufficient.

Key generation. Generate 5 keypairs for the DSME scheme in $\widehat{\mathbb{G}}$, as described above. Call them $A_i, \mathbf{a}_i$ for $i = 1, \ldots, 5$.

Pick random generators $g_1, \ldots, g_5 \in \mathbb{G}$, and random $\mathbf{b} = (b_1, \ldots, b_5), \mathbf{c} = (c_1, \ldots, c_5), \mathbf{d} = (d_1, \ldots, d_5)$ from $\mathbb{Z}_p$. The private key consists of $\mathbf{b}, \mathbf{c}, \mathbf{d}$ and the 5 private keys for the DSME scheme. The public key consists of $(g_1, \ldots g_5)$, the 5 public keys for the DSME scheme, and the values:

$$B = \prod_{i=1}^{5} g_i^{b_i}, \qquad C = \prod_{i=1}^{5} g_i^{c_i}, \qquad D = \prod_{i=1}^{5} g_i^{d_i}$$

Encryption: $\mathsf{Enc}_{PK}(\mathsf{msg})$:

- Pick random $x, y \in \mathbb{Z}_p^*$ and random $u_1, \ldots, u_5 \in \widehat{\mathbb{G}}$.
- For $i = 1, \ldots, 5$: let $X_i = g_i^{(x+z_i)u_i}$; $Y_i = g_i^{yu_i}$; and $U_i = \mathsf{MEnc}_{A_i}(u_i)$.
- Let $\mu = \mathsf{encode}_{\mathbb{G}}(\mathsf{msg})$, and $m = \mathsf{encode}_{\mathbb{Z}_p}(\mathsf{msg})$.
- Output:
$$(\mathbf{X}, \mu B^x, (CD^m)^x, \mathbf{Y}, B^y, (CD^m)^y, \mathbf{U})$$
where $\mathbf{U} = (U_1, \ldots, U_5), \mathbf{X} = (X_1, \ldots, X_5), \mathbf{Y} = (Y_1, \ldots, Y_5)$.

Decryption: $\mathsf{Dec}_{SK}(\zeta = (\mathbf{X}, B_X, P_X, \mathbf{Y}, B_Y, P_Y, \mathbf{U}))$:

- **Decrypt U_i's:** For $i = 1, \ldots, 5$: set $u_i = \mathsf{MDec}_{\mathbf{a}_i}(U_i)$. If any $u_i = \perp$, immediately output $\perp$.
- **Strip u_i and z_i:** For $i = 1, \ldots, 5$: set $\overline{X}_i = X_i^{1/u_i} g_i^{-z_i}$ and $\overline{Y}_i = Y_i^{1/u_i}$.
- **Derive purported message:** Set $\mu = B_X / \prod_{i=1}^{5} \overline{X}_i^{b_i}$, $\mathsf{msg} = \mathsf{encode}_{\mathbb{G}}^{-1}(\mu)$, and $m = \mathsf{encode}_{\mathbb{Z}_p}(\mathsf{msg})$.
- **Check ciphertext integrity:** Check the following conditions:
$$B_Y \stackrel{?}{=} \prod_{i=1}^{5} \overline{Y}_i^{b_i}; \qquad P_X \stackrel{?}{=} \prod_{i=1}^{5} \overline{X}_i^{c_i + d_i m}; \qquad P_Y \stackrel{?}{=} \prod_{i=1}^{5} \overline{Y}_i^{c_i + d_i m}$$
If any checks fail, output $\perp$. Otherwise output msg.

Rerandomization: $\mathsf{Rerand}(\zeta = (\mathbf{X}, B_X, P_X, \mathbf{Y}, B_Y, P_Y, \mathbf{U}))$**.** The only randomness used in generating ζ is the choice of x, y, $\mathbf{u} = (u_1, \ldots, u_5)$, and the randomness used in each instance of MEnc. We can rerandomize each of these quantities by choosing random $r_1, \ldots, r_5 \in \widehat{\mathbb{G}}$, random $s, t \in \mathbb{Z}_p^*$, and constructing a new ciphertext with corresponding randomness $u_i' = u_i r_i$, $x' = x + ys$, and $y' = yt$:

- For $i = 1, \ldots, 5$: $U_i' = \mathsf{MRerand}(r_i \otimes U_i)$.
- For $i = 1, \ldots, 5$: $X_i' = (X_i Y_i^s)^{r_i}$.
- For $i = 1, \ldots, 5$: $Y_i' = Y_i^{r_i t}$.
- $B_X' = B_X B_Y^s$ and $P_X' = P_X P_Y^s$.
- $B_Y' = B_Y^t$ and $P_Y' = P_Y^t$.

The rerandomized ciphertext is $\zeta' = (\mathbf{X}', B_X', P_X', \mathbf{Y}', B_Y', P_Y', \mathbf{U}')$

5.3 Complexity

A DSCS ciphertext consists of 40 elements from $\widehat{\mathbb{G}}$ and 14 elements from $\mathbb{G}$. See Figure 5.3 for the complexity of the DSCS algorithms.[4]

	exponentiations		multiplications		inversions	
	$\widehat{\mathbb{G}}$	$\mathbb{G}$	$\mathbb{Z}_p^*$	$\mathbb{G}$	$\mathbb{Z}_p^*$	$\mathbb{G}$
Enc	40	16	15	3	0	0
Dec (worst case)	30	35	40	22	10	1
Rerand	36	19	40	7	0	0

Figure 1: Group operations performed in DSCS algorithms.

Clearly our scheme is much less efficient than the Cramer-Shoup encryption scheme. On the other hand, it is much more efficient than the only previously proposed rerandomizable (weak) RCCA secure scheme [19], which used $O(k)$ group elements to encode a k-bit message (or in other words, to be able to encode group elements, it uses $O(\log p)$ group elements). In fact, if we restrict ourselves to weak RCCA security (and a computational version of rerandomizability), our construction can be simplified to have only 10 group elements. (We omit the details of that construction in this paper.)

Rerandomizable RCCA security (anonymous or not) is a significantly harder problem by our current state of knowledge. Despite the inefficiency, we believe that by providing the first complete solution (not in generic group model) we have not only solved the problem from a theoretical perspective, but also have opened up the possibility of efficient and practical constructions.

6 Proof of Theorem 1

We first show that the DSCS scheme satisfies the correctness requirements of a perfectly rerandomizable encryption scheme.

As mentioned in Section 4, to demonstrate the two security properties (RCCA security and receiver-anonymity), we will demonstrate alternate encryption and decryption procedures. The alternate encryption procedure AltEnc is described in Section 6.5 and the alternate decryption procedure AltDec is described in Section 6.6. The lemmas in the rest of this section carry out the proof outlined in Section 4.

6.1 Correctness properties

The correctness property of decryption (i.e, that it is the inverse of encryption) is straight-forward to verify. The first correctness property of Rerand (i.e, that a rerandomization of an honestly generated ciphertext is distributed as a fresh re-encryption) is also straight-forward to verify. We now prove the final correctness property (i.e, that a rerandomization of an adversarially generated ciphertext decrypts to the same value, under any secret key) for both the DSME and DSCS schemes.

Lemma 1 *For all key pairs (MPK, MSK), all (purported) ciphertexts U, and all $U' \leftarrow \mathsf{MRerand}(U)$, we must have $\mathsf{MDec}_{MSK}(U') = \mathsf{MDec}_{MSK}(U)$.*

[4] Multiplication and inversion operations in $\mathbb{Z}_p^*$ include operations in the subgroup $\widehat{\mathbb{G}}$. We assume that for $\widehat{g}_i$ elements of the public key, $\widehat{g}_i^{-1}$ can be precomputed.

PROOF: Let $MSK = (a_1, a_2, a_3)$ be a private key, and $U = (\mathbf{V}, A_V, \mathbf{W}, A_W)$ be a ciphertext such that $\mathsf{MDec}_{MSK}(U) = u \neq \perp$. It is easy to see that this happens if and only if the following two conditions hold:

$$A_V/u = \prod_{j=1}^{3} V_i^{a_i}; \qquad A_W = \prod_{j=1}^{3} W_i^{a_i}$$

Now let $U' = (\mathbf{V}', A'_V, \mathbf{W}', A'_W) = \mathsf{MRerand}(U)$. Suppose the randomness used in $\mathsf{MRerand}$ is $s, t \in \mathbb{Z}_q$. Then, substituting according to the two above constraints and the computations in $\mathsf{MRerand}$, we have:

$$A'_V/u = (A_V A_W^s)/u = \left[\prod_{j=1}^{3} V_i^{a_i}\right] \left[\prod_{j=1}^{3} W_i^{a_i}\right]^s = \prod_{j=1}^{3} (V_i W_i^s)^{a_i} = \prod_{j=1}^{3} (V'_i)^{a_i}$$
$$A'_W = A_W^t = \left[\prod_{j=1}^{3} W_i^{a_i}\right]^s = \prod_{j=1}^{3} (W_i^s)^{a_i} = \prod_{j=1}^{3} (W'_i)^{a_i}$$

Thus $\mathsf{MDec}_{MSK}(U') = u$ as well.

Likewise, $\mathsf{MDec}_{MSK}(U) = \perp$ if and only if $A_W \neq \prod_{j=1}^{3} W_i^{a_i}$. When this is the case, the corresponding constraint on A'_W does not hold, as can be seen by a similar argument. Thus $\mathsf{MDec}_{MSK}(U') = \perp$ as well. □

Lemma 2 *For all key pairs (PK, SK), all (purported) ciphertexts ζ, and all $\zeta' \leftarrow \mathsf{Rerand}(\zeta)$, we must have $\mathsf{Dec}_{SK}(\zeta') = \mathsf{Dec}_{SK}(\zeta)$.*

PROOF: Fix a private key SK and let $\zeta = (\mathbf{X}, B_X, P_X, \mathbf{Y}, B_Y, P_Y, \mathbf{U})$ be a (purported) ciphertext such that $\mathsf{Dec}_{SK}(\zeta) \neq \perp$. Suppose each U_i decrypts to u_i under the ith DSME private key contained in SK. Let $\zeta' = (\mathbf{X}', B'_X, P'_X, \mathbf{Y}', B'_Y, P'_Y, \mathbf{U}')$, where:

$$X'_i = (X_i Y_i^s)^{r_i}; \qquad Y'_i = Y_i^{r_i t}; \qquad B'_X = B_X B_Y^s; \qquad P'_X = P_X P_Y^s; \qquad B'_Y = B_Y^t; \qquad P'_Y = P_Y^t$$

By the correctness properties of the DSME scheme, each U'_i of the rerandomized ciphertext decrypts to $r_i u_i$ under the ith DSME private key contained in SK.

When decrypting ζ, the decryption procedure will strip the u_i's (and z_i's) to obtain: $\overline{X}_i = X_i^{1/u_i} g_i^{-z_i}$ and $\overline{Y}_i = Y_i^{1/u_i}$. However, when decrypting the rerandomization ζ', the computed values will be:

$$\overline{X}'_i = (X'_i)^{1/u'_i} g_i^{-z_i} = (X_i Y_i^s)^{r_i/(r_i u_i)} g_i^{-z_i} = \overline{X}_i \overline{Y}_i^s$$
$$\overline{Y}'_i = (Y'_i)^{1/u'_i} = Y_i^{r_i t/r_i u_i} = \overline{Y}_i^t$$

Next, the decryption procedure computes the purported message of ζ' by the following computation (substituting according to the identities above and the fact that the check on B_Y succeeds while decrypting ζ):

$$\mu = \frac{B'_X}{\prod_{i=1}^{5} (\overline{X}'_i)^{b_i}} = \frac{B_X}{\prod_{i=1}^{5} \overline{X}_i^{b_i}} \frac{B_Y^s}{\prod_{i=1}^{5} (\overline{Y}_i^s)^{b_i}} = \frac{B_X}{\prod_{i=1}^{5} \overline{X}_i^{b_i}} \cdot 1$$

In other words, the same purported message is computed as in the decryption of ζ. In the same way, the integrity checks also succeed while decrypting ζ', and the decryption procedure indeed returns the same message.

On the other hand, if $\mathsf{Dec}_{SK}(\zeta) = \perp$, then $\mathsf{Dec}_{SK}(\zeta')$ will output $\perp$ at the same point in the algorithm; either while decrypting a certain U_i or while performing an integrity check on the ciphertext (though the purported ciphertexts may be different). □

6.2 Replay interactions

Instead of arguing separately about the two security experiments (RCCA security and receiver-anonymity), we define a generic "experiment" called a *replay interaction*. Both security experiments can be implemented in terms of replay interactions.

Replay interaction. Consider the following interaction against an adversary $\mathcal{A}$.

1. Honestly generate a keypair $(PK, SK) \leftarrow \mathsf{KeyGen}$, and give PK to $\mathcal{A}$.
2. (Phase I) Let $\mathcal{A}$ access the decryption oracle $\mathsf{Dec}_{SK}(\cdot)$.
3. $\mathcal{A}$ gives a message msg^*. Give $\zeta^* = \mathsf{Enc}_{PK}(\mathsf{msg}^*)$ to $\mathcal{A}$.
4. (Phase II) Let $\mathcal{A}$ access a special guarded decryption oracle with the following restrictions:
   - On input ζ, if $\mathsf{Dec}_{SK}(\zeta) = \mathsf{msg}^*$, the oracle's response must be either msg^* or replay (possibly arbitrarily).
   - Otherwise, the oracle's response must match $\mathsf{Dec}_{SK}(\zeta)$.
5. $\mathcal{A}$ outputs a bit.

An alternate decryption procedure is said to *faithfully implement* the oracle in Phases I & II if for all adversaries $\mathcal{A}$, the procedure's answers to the adversary's queries queries satisfy the given restrictions, except with negligible probability (over the randomness in KeyGen and ciphertext generation).

Claim 1 *There is an alternate encryption procedure* AltEnc *which uses the private key SK instead of the public key PK, and whose output is indistinguishable from the honest encryption procedure. However, even though it uses the private key, its output is independent of the message and corresponding public key, except with negligible probability over its randomness.*

Claim 2 *There is an alternate decryption procedure* AltDec *that can be implemented using only ζ^* and the public key PK, and which faithfully implements the decryption oracles of the replay interaction, even when ζ^* was generated by* AltEnc.

Note that AltDec does not have access to SK or msg^*, yet must (in part) emulate $\mathsf{GDec}_{SK}^{\mathsf{msg}^*}(\cdot)$.

Later, we describe AltEnc and AltDec, and prove the above claims.

Proving RCCA security and receiver-anonymity given Claim 1 and Claim 2. First, we show that both of the security experiments can be implemented by playing as the adversary in replay interactions.

- To implement the RCCA security experiment against $\mathcal{A}$, we participate as an adversary in a replay interaction, and receive a public-key PK, which we pass on to $\mathcal{A}$. The decryption oracle of Phase I is implemented using access to the replay interaction's decryption oracle. In the challenge phase, when $\mathcal{A}$ gives a pair of messages $\mathsf{msg}_0, \mathsf{msg}_1$, we flip a coin b and pass $\mathsf{msg}^* = \mathsf{msg}_b$ to the replay interaction. On obtaining ζ^* from the replay interaction, we return it to the adversary. In Phase II, when $\mathcal{A}$ wants to query $\mathsf{GDec}_{SK}^{(\mathsf{msg}_0,\mathsf{msg}_1)}(\zeta)$, we use the special guarded decryption oracle from the replay interaction. If it returns replay or

one of $\mathsf{msg}_0, \mathsf{msg}_1$, we return replay; otherwise we return its result. From the condition on the special guarded decryption oracle, it follows that this is a perfect implementation of the RCCA security experiment. Finally, we output 1 if $\mathcal{A}$ correctly guesses b.

- To implement the receiver-anonymity experiment against $\mathcal{A}$, we participate in two replay interaction sessions, and receive two public-keys PK_0 and PK_1 which we pass to $\mathcal{A}$. The decryption oracles of Phase I of the receiver-anonymity experiment are implemented using access to Dec_{SK_0} and Dec_{SK_1} of the replay interaction. In the challenge phase, when the adversary gives a message msg^*, we flip a coin b, give msg^* to both interactions, and return $\zeta^* = \mathsf{Enc}_{PK_b}(\mathsf{msg}^*)$ to the adversary (discarding $\mathsf{Enc}_{PK_{1-b}}(\mathsf{msg}^*)$). In Phase II, when $\mathcal{A}$ wants to query $\mathsf{GDec}^{\mathsf{msg}^*}_{SK_0,SK_1}(\zeta)$, we pass the query to both decryption oracles from the replay interactions. If either oracle responds with msg^* or replay, we return replay. Otherwise, we return the results of both queries. Finally, we output 1 if $\mathcal{A}$ correctly guesses b.

By Claim 1, generating the encryptions using AltEnc instead of Enc does not noticeably affect the outcome of the replay interaction. Then, by Claim 2, further replacing the decryption oracle with AltDec also does not noticeably affect the outcome of the replay interaction. Call such an interaction which uses both AltEnc and AltDec an *alternate replay interaction*.

It suffices to show that when implementing either security experiment in terms of alternate replay interactions, the adversary has no advantage in guessing b. In such an implementation of either experiment, the adversary's view includes only the following quantities:

- Public key(s). These are obviously distributed independently of b.
- Answers to decryption queries in Phase I. By Claim 2, these can be computed using only the public key(s), which the adversary already knows.
- The challenge ciphertext ζ^*. Here, the bit b was used to choose either the message or the public key with which to generate the ciphertext. However, by Claim 1, the ciphertext is distributed independently of this choice (i.e, independent of b).
- Answers to decryption queries in Phase II. By Claim 2, these can be computed using only the public key(s) and ζ^*, both of which the adversary knows.

Thus in either security experiment (when implemented in terms of alternate replay interactions), the adversary's view is independent of b, and hence the adversary has zero advantage.

6.3 Decisional Diffie-Hellman assumption

We now describe a more intricate indistinguishability assumption, which is implied by the standard DDH assumption in $\mathbb{G}$ and $\widehat{\mathbb{G}}$.

First, consider the following two distributions:

- $\mathsf{DDH}(\mathbb{G}, n)$ **distribution.** Pick random elements $g_1, \ldots, g_n \in \mathbb{G}$, and pick a random $v \in \mathbb{Z}_p$, where $|\mathbb{G}| = p$. Output $(g_1, \ldots, g_n, g_1^v, \ldots, g_n^v)$.
- $\mathsf{Rand}(\mathbb{G}, n)$ **distribution.** Pick random elements $g_1, \ldots, g_n \in \mathbb{G}$, and pick random $v_1, \ldots, v_n \in \mathbb{Z}_p$, where $|\mathbb{G}| = p$. Output $(g_1, \ldots, g_n, g_1^{v_1}, \ldots, g_n^{v_n})$.

We will require distributions of this form with $n = 3$ and $n = 5$, in different groups. Note that for fixed n, the standard DDH assumption in $\mathbb{G}$ (which is the special case of $n = 2$) implies that the above distributions are indistinguishable. To see this, consider a hybrid distribution in which the first k exponents are randomly chosen, and the remaining $n - k$ are all equal. The standard DDH assumption is easily seen to imply that the kth hybrid distribution is indistinguishable from the $(k + 1)$st.

Now consider the following two "double-strand" distributions:

- $\mathsf{DS\text{-}DDH}(\mathbb{G}, n)$ **distribution.** Pick random elements $g_1, \ldots, g_n \in \mathbb{G}$, and pick random $v, w \in \mathbb{Z}_p$, where $|\mathbb{G}| = p$. Output $(g_1, \ldots, g_n, g_1^v, \ldots, g_n^v, g_1^w, \ldots, g_n^w)$.
- $\mathsf{DS\text{-}Rand}(\mathbb{G}, n)$ **distribution.** Pick random elements $g_1, \ldots, g_n \in \mathbb{G}$, and pick random $v_1, \ldots, v_n, w_1, \ldots, w_n \in \mathbb{Z}_p$, where $|\mathbb{G}| = p$. Output $(g_1, \ldots, g_n, g_1^{v_1}, \ldots, g_n^{v_n}, g_1^{w_1}, \ldots, g_n^{w_n})$.

Again, a simple hybrid argument shows that if the $\mathsf{DDH}(\mathbb{G}, n)$ and $\mathsf{Rand}(\mathbb{G}, n)$ distributions are indistinguishable, then so are $\mathsf{DS\text{-}DDH}(\mathbb{G}, n)$ and $\mathsf{DS\text{-}Rand}(\mathbb{G}, n)$. We call elements in the support of these distributions *double-strand tuples of length n.*

Finally, our security proofs rely on the indistinguishability of the following two distributions:

- Pick K_0 from $\mathsf{DS\text{-}DDH}(\mathbb{G}, 5)$, and pick $K_1, \ldots, K_5$ from $\mathsf{DS\text{-}DDH}(\widehat{\mathbb{G}}, 3)$. Output $(K_0, \ldots, K_5)$.
- Pick K_0 from $\mathsf{DS\text{-}Rand}(\mathbb{G}, 5)$, and pick $K_1, \ldots, K_5$ from $\mathsf{DS\text{-}Rand}(\widehat{\mathbb{G}}, 3)$. Output $(K_0, \ldots, K_5)$.

A final hybrid argument shows that if $\mathsf{DS\text{-}DDH}(\mathbb{G}, 5)$ and $\mathsf{DS\text{-}Rand}(\mathbb{G}, 5)$ are indistinguishable, and $\mathsf{DS\text{-}DDH}(\widehat{\mathbb{G}}, 3)$ and $\mathsf{DS\text{-}Rand}(\widehat{\mathbb{G}}, 3)$ are also indistinguishable, then the above two distributions are indistinguishable.

6.4 Encryption and decryption as linear algebra

Before describing the alternate encryption and decryption procedures, we give a characterization of our construction using linear algebra.

Definition 1 *Let $U = (\mathbf{V}, A_V, \mathbf{W}, A_W)$ be a DSME ciphertext. The two* DSME strands *of U with respect to a public key $(\widehat{g}_1, \widehat{g}_2, \widehat{g}_3, A)$ are:*

$$\mathbf{v} = (v_1, v_2, v_3), \text{ where } v_j = \log_{\widehat{g}_j} V_j$$
$$\mathbf{w} = (w_1, w_2, w_3), \text{ where } w_j = \log_{\widehat{g}_j} W_j$$

Observe that rerandomizing U gives a ciphertext whose two strands are of the form $\mathbf{v} + r\mathbf{w}$ and $s\mathbf{w}$, for random $r, s \in \mathbb{Z}_q$. In ciphertexts generated by MEnc, both strands are scalar multiples of the all-ones vector.

For simplicity later on, we call the all-ones vector the public key's strand.

For DSCS ciphertexts, we define a similar notion of strands. However, in a DSCS ciphertext, the first strand is "masked" by u_i's and z_i's, and the second strand is masked by u_i's. We separately consider "masked" and "unmasked" definitions of strands.

Definition 2 *Let* $\zeta = (\mathbf{X}, B_X, P_X, \mathbf{Y}, B_Y, P_Y, \mathbf{U})$ *be a DSCS ciphertext. The* masked DSCS strands *of* ζ *with respect to a public key* $(g_1, \ldots, g_5, B, C, D)$ *are:*

$$\mathbf{x} = (x_1, \ldots, x_5), \text{ where } x_i = \log_{g_i} X_i$$
$$\mathbf{y} = (y_1, \ldots, y_5), \text{ where } y_i = \log_{g_i} Y_i$$

The unmasked DSCS strands *of* ζ *with respect to a public key* $(g_1, \ldots, g_5, B, C, D)$ *and* $\mathbf{u} = (u_1, \ldots, u_5)$ *are:*

$$\overline{\mathbf{x}} = \left(\frac{x_1}{u_1} - z_1, \ldots, \frac{x_5}{u_5} - z_5\right), \qquad \overline{\mathbf{y}} = \left(\frac{y_1}{u_1}, \ldots, \frac{y_5}{u_5}\right)$$

As with DSME ciphertexts, rerandomizing ζ gives a ciphertext whose two (unmasked) strands are of the form $\overline{\mathbf{x}} + r\overline{\mathbf{y}}$ and $s\overline{\mathbf{y}}$, for $r, s \in \mathbb{Z}_p$. In ciphertexts generated by Enc, both unmasked strands are scalar multiples of the all-ones vector.

For simplicity later on, we call the all-ones vector the public key's (unmasked) strand.

Adversary's view in a replay interaction. In a replay interaction, the view of the adversary is a linear function of the private key in the following way:

For each DSME private key $\mathbf{a} = (a_1, a_2, a_3)$, the adversary sees only the corresponding public key A and, as part of the challenge ciphertext, a DSME ciphertext $U^* = (\mathbf{V}^*, A_V^*, \mathbf{W}^*, A_W^*)$ which decrypts to some u^*. Of these quantities, only A, A_V^*, and A_W^* are dependent on the private key. Let $\mathbf{w}^*$ and $\mathbf{v}^*$ be the strands of U^* with respect to $\widehat{g}_1, \widehat{g}_2, \widehat{g}_3$ of the public key. The following constraints must hold:

$$\begin{bmatrix} 1 & 1 & 1 \\ w_1^* & w_2^* & w_3^* \\ v_1^* & v_2^* & v_3^* \end{bmatrix} \begin{bmatrix} \log \widehat{g}_1 & 0 & 0 \\ 0 & \log \widehat{g}_2 & 0 \\ 0 & 0 & \log \widehat{g}_3 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} \log A \\ \log A_W^* \\ \log(A_V^*/u^*) \end{bmatrix} \tag{1}$$

The logarithm is with respect to any fixed generator of $\widehat{\mathbb{G}}$. In Phase I of the replay interaction, only the first row constraint is relevant.

Similarly, let B, C, D be the corresponding parts of the DSCS public key, and let $\zeta^* = (\mathbf{X}^*, B_X^*, P_X^*, \mathbf{Y}^*, P_Y^*, \mathbf{U}^*)$ denote the challenge ciphertext. Let $\mathbf{x}^*$ and $\mathbf{y}^*$ denote the unmasked DSCS strands of ζ^*, with respect to the decryptions of each U_i^*, and the generators $g_1, \ldots, g_5$ in the DSCS public key. The following constraints must hold:

$$\begin{bmatrix} \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} \\ \overline{\mathbf{x}}^* & 0 & 0 \\ \overline{\mathbf{y}}^* & 0 & 0 \\ 0 & \overline{\mathbf{x}}^* & m\overline{\mathbf{x}}^* \\ 0 & \overline{\mathbf{y}}^* & m\overline{\mathbf{y}}^* \end{bmatrix} \begin{bmatrix} G & 0 & 0 \\ 0 & G & 0 \\ 0 & 0 & G \end{bmatrix} \begin{bmatrix} \mathbf{b}^T \\ \mathbf{c}^T \\ \mathbf{d}^T \end{bmatrix} = \begin{bmatrix} \log B \\ \log C \\ \log D \\ \log(B_X^*/\mu) \\ \log B_Y^* \\ \log P_X^* \\ \log P_Y^* \end{bmatrix}, \text{ where } G = \begin{bmatrix} \log g_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \log g_5 \end{bmatrix} \tag{2}$$

Again, the logarithm is with respect to any fixed generator of $\mathbb{G}$. In Phase I of the replay interaction, only the first three row constraints are relevant.

In the analysis that follows, we consider the adversary's behavior over the remaining (independent) randomness of the key generation.

6.5 The alternate encryption functionality

We now describe the alternate encryption functionality AltEnc needed for Claim 1. As a component, it uses $\mathsf{AltMEnc}$, an alternate encryption functionality for the DSME scheme.

DSME alternate encryption: $\mathsf{AltMEnc}_{\mathbf{a}}(u)$**.**

- Pick random $v_1, v_2, v_3, w_1, w_2, w_3 \in \mathbb{Z}_q$. For $j = 1, 2, 3$ let $V_j = \widehat{g}_j^{v_j}$ and $W_j = \widehat{g}_j^{w_j}$ (alternatively, in the analysis below we also consider V_i, W_i as inputs instead).
- Output $(\mathbf{V}, A_V, \mathbf{W}, A_W)$, where

$$A_V = u \cdot \prod_{j=1}^{3} V_j^{a_j} \qquad \mathbf{V} = (V_1, V_2, V_3)$$
$$A_W = \prod_{j=1}^{3} W_j^{a_j} \qquad \mathbf{W} = (W_1, W_2, W_3)$$

DSCS alternate encryption: $\mathsf{AltEnc}_{SK}(\mathsf{msg})$**.**

- Pick random $x_1, \ldots, x_5, y_1, \ldots, y_5 \in \mathbb{Z}_p^*$. For $i = 1, \ldots, 5$, set $\overline{X}_i = g_i^{x_i}$ and $\overline{Y}_i = g_i^{y_i}$, (alternatively, in the analysis below we also consider $\overline{X}_i, \overline{Y}_i$ as inputs instead).
- Pick random $u_1, \ldots, u_5 \in \widehat{\mathbb{G}}$ and for $i = 1, \ldots, 5$, set $X_i = (\overline{X}_i g_i^{z_i})^{u_i}$, $Y_i = \overline{Y}_i^{u_i}$, and $U_i = \mathsf{AltMEnc}_{\mathbf{a}_i}(u_i)$.
- Let $\mu = \mathsf{encode}_{\mathbb{G}}(\mathsf{msg})$, and $m = \mathsf{encode}_{\mathbb{Z}_p}(\mathsf{msg})$.
- Output $(\mathbf{X}, B_X, P_X, \mathbf{Y}, B_Y, P_Y, \mathbf{U})$, where

$$B_X = \mu \cdot \prod_{i=1}^{5} \overline{X}_i^{b_i} \qquad \mathbf{U} = (U_1, \ldots, U_5)$$
$$P_X = \prod_{i=1}^{5} \overline{X}_i^{c_i + d_i m} \qquad \mathbf{X} = (X_1, \ldots, X_5)$$
$$B_Y = \prod_{i=1}^{5} \overline{Y}_i^{b_i} \qquad \mathbf{Y} = (Y_1, \ldots, Y_5)$$
$$P_Y = \prod_{i=1}^{5} \overline{Y}_i^{c_i + d_i m}$$

Observe that both of these alternate encryption procedures generate ciphertexts whose two (unmasked) strands are random vectors. The remainder of the ciphertext is constructed using the private key to ensure that it (and any of its malleations or rerandomizations) will decrypt properly.

The following two lemmas complete the proof of Claim 1:

Lemma 3 *In a replay interaction where ciphertexts are generated with* AltEnc*, the challenge ciphertext is distributed independently of the choice of message and public key. except with negligible probability over the randomness in* AltEnc*.*

PROOF: The parts of the adversary's view that depend on the key are given in the linear constraints of Equation 1 and Equation 2.

Consider the ith DSME component U_i^* of the challenge ciphertext. When $\mathsf{AltMEnc}$ is used to generate U_i^*, its two strands are random vectors. Therefore, the matrix in Equation 1 is nonsingular with overwhelming probability. Conditioned on this event, there are an equal number of private keys consistent with each choice of right-hand side values in the equation. The right-hand side includes the choice of public key and message u_i^*, thus the view of the adversary is independent of these choices.

Similarly, when AltEnc is used to generate the rest of the challenge ciphertext, its unmasked strands are also random vectors (with respect to any choice of $\mathbf{u}^*$). Thus the first matrix in equation Equation 2 is nonsingular with overwhelming probability, for all values of m (this happens when $\{\mathbf{1}, \mathbf{x}^*, \mathbf{y}^*\}$ are linearly independent). By the same reasoning as above, the adversary's view is independent of the choice of ciphertext and message. □

Lemma 4 *For any adversary in a replay interaction, its advantage when the challenge ciphertext is generated using* AltEnc *is negligibly close to its advantage in the original interaction (when the ciphertext is generated using* Enc*), if the DDH assumption holds in* $\mathbb{G}$ *and* $\widehat{\mathbb{G}}$*.*

Proof: If the DDH assumption holds for $\widehat{\mathbb{G}}$ and $\mathbb{G}$, then two the distributions described in Section 6.3 are computationally indistinguishable. Elements in the support of these distributions consist of 1 double-strand tuple of length 5 from $\mathbb{G}$, and 5 double-strand tuples of length 3 from $\widehat{\mathbb{G}}$.

Now consider a simulation of a replay interaction, where the input is from one of the above distributions. For each $i = 1, \ldots, 5$, let $(\widehat{g}_1, \widehat{g}_2, \widehat{g}_3, V_1, V_2, V_3, W_1, W_2, W_3)$ be the ith double-strand tuple from $\widehat{\mathbb{G}}$. Set $(\widehat{g}_1, \widehat{g}_2, \widehat{g}_3)$ as the corresponding part of the ith DSME public key, and generate the remainder of the ith keypair honestly. To simulate the encryption of u_i^* from the challenge ciphertext with this keypair, use AltMEnc with the input values $V_1, V_2, V_3, W_1, W_2, W_3$.

Similarly, let $(g_1, \ldots, g_5, \overline{X}_1, \ldots, \overline{X}_5, \overline{Y}_1, \ldots, \overline{Y}_5)$ be the double-strand tuple from $\mathbb{G}$. Set $(g_1, \ldots, g_5)$ as the corresponding part of the DSCS public key and generate the remainder of the DSCS keypair honestly. To simulate the encryption of the challenge ciphertext, use AltEnc with the input values $\overline{X}_1, \ldots, \overline{X}_5, \overline{Y}_1, \ldots, \overline{Y}_5$.

It is easy to see that when the input is sampled from the first distribution (i.e, each tuple comes from the appropriate DS-DDH distribution), the ciphertext is distributed as an honest encryption with Enc (and MEnc). If the input is sampled from the second distribution (i.e, each tuple comes from the appropriate DS-Rand distribution), then the ciphertext is distributed as an encryption with AltEnc (and AltMEnc).

The rest of this simulation of the replay interaction can be implemented in polynomial time. Thus, the outcomes of the two simulations must not differ by more than a negligible amount. □

6.6 The alternate decryption procedure

We now describe alternate decryption procedures for the DSME and DSCS schemes. They are computationally unbounded, as they computes the strands of ciphertexts (i.e, they compute discrete logarithms in $\widehat{\mathbb{G}}$ and $\mathbb{G}$). Depending on whether they are being called in Phase I or Phase II of a replay interaction, they have access to the challenge ciphertext ζ^*.

DSME alternate decryption ($\mathsf{AltMDec}^{U^*}_{MPK}(U)$)**.** Let $U^* = (\mathbf{V}^*, A_V^*, \mathbf{W}^*, A_W^*)$ denote the challenge ciphertext that was created with AltMEnc using the corresponding private key (if called in Phase II of the interaction). Let $U = (\mathbf{V}, A_V, \mathbf{W}, A_W)$ denote the ciphertext query given as input.

First, compute the strands $\mathbf{v}$ and $\mathbf{w}$ of U, and the strands $\mathbf{v}^*$ and $\mathbf{w}^*$ of U^* (if given), both with respect to the given public key. If in Phase I, consider the *known strands* to be $\mathbf{1}$ (the public key strand). If in Phase II, consider the *known strands* to be $\{\mathbf{1}, \mathbf{v}^*, \mathbf{w}^*\}$. Hereafter, we somewhat abuse notation and talk about linear combinations of $\{\mathbf{1}, \mathbf{v}^*, \mathbf{w}^*\}$ and components

of U_i^*, both of which are misleading in Phase I. However, it should be understood that when U^* is not given, the coefficients of $\mathbf{v}^*$ and $\mathbf{w}^*$ in a linear combination are zero, and in those cases, the components of U^* in fact cancel out from the expressions we use.

We now check that $\mathbf{v}, \mathbf{w}$ are both linear combinations of the known strands. If not, then output $\perp$. Otherwise, let $\mathbf{w} = \alpha\mathbf{v}^* + \beta\mathbf{w}^* + \gamma\mathbf{1}$. If $\alpha \neq 0$, then output $\perp$. Otherwise check that

$$A_W \stackrel{?}{=} (A_W^*)^\beta A^\gamma$$

where A comes from the public key. If the check fails, output $\perp$.

Now let $\mathbf{v} = \pi\mathbf{v}^* + \beta\mathbf{w}^* + \gamma\mathbf{1}$. Output the pair:

$$\left(\delta = \frac{A_V}{(A_V^*)^\pi (A_W^*)^\beta A^\gamma}, \pi\right)$$

Below, we prove that these values (δ, π) are such that $\mathsf{MDec}_{MSK}(U) = \delta\mathsf{MDec}_{MSK}(U^*)^\pi$, where MSK is the private key used to generate U^*. If U^* has not been given yet, observe that π and β must be zero, and in fact δ is the correct decryption of U.

The following lemma establishes the correctness of the output of $\mathsf{AltMDec}$ when it is used in the context of a replay interaction.

Lemma 5 *Fix a DSME key pair (MPK, MSK). Let U^* be a DSME ciphertext generated by $\mathsf{AltMEnc}_{MSK}$. If $\mathsf{AltMDec}_{MPK}^{U^*}(U)$ outputs (δ, π), then $\mathsf{MDec}_{MSK}(U) = \delta\mathsf{MDec}_{MSK}(U^*)^\pi$.*

PROOF: Let $\mathbf{v}, \mathbf{w}$ be the strands of U, and let $\mathbf{v}^*, \mathbf{w}^*$ be the strands of U^*. If we can write $\mathbf{v} = \pi\mathbf{v}^* + \beta\mathbf{w}^* + \gamma\mathbf{1}$ for some π, β, γ, then each $V_j = (V_j^*)^\pi (W_j^*)^\beta \widehat{g}_j^\gamma$.

If $MSK = (a_1, a_2, a_3)$ was the private key used to generate U^* and A, we have:

$$\begin{aligned}\mathsf{MDec}_{MSK}(U) &= \frac{A_V}{\prod_{j=1}^3 V_j^{a_j}} = \frac{A_V}{\left(\prod_j (V_j^*)^{a_j}\right)^\pi \left(\prod_j (W_j^*)^{a_j}\right)^\beta \left(\prod_j \widehat{g}_j^{a_j}\right)^\gamma} = \frac{A_V}{\left(\frac{A_V^*}{\mathsf{MDec}_{MSK}(U^*)}\right)^\pi (A_W^*)^\beta A^\gamma} \\ &= \left[\frac{A_V}{(A_V^*)^\pi (A_W^*)^\beta A^\gamma}\right] \mathsf{MDec}_{MSK}(U^*)^\pi\end{aligned}$$

□

We now describe the DSCS alternate decryption procedure AltDec.

DSCS alternate decryption ($\mathsf{AltDec}_{PK}^{\zeta^*}(\zeta)$). Let $\zeta^* = (\mathbf{X}^*, B_X^*, P_X^*, \mathbf{Y}^*, B_Y^*, P_Y^*, \mathbf{U}^*)$ denote the challenge ciphertext that was created with AltEnc (if called in Phase II of the interaction). Let $\zeta = (\mathbf{X}, B_X, P_X, \mathbf{Y}, B_Y, P_Y, \mathbf{U})$ denote the ciphertext query given as input.

The first step in the honest decryption procedure is to decrypt each U_i, a DSME ciphertext. We try to simulate the behavior of this step in the first part of the alternate decryption procedure.

- For $i = 1, \ldots, 5$: Call $\mathsf{AltMDec}_{MPK_i}^{U_i^*}(U_i)$, where MPK_i is the ith DSME public key contained in PK (omitting U_i^* if called in Phase I). If it returns $\perp$, immediately return $\perp$. Otherwise, store the pair (δ_i, π_i) that it returned.

Next, we simulate how the honest decryption procedure strips the u_i and z_i terms from the exponents of each X_i and Y_i. To do this, we compute the masked strands $\mathbf{x}^*$ and $\mathbf{y}^*$ of ζ^* (if given), and compute the masked strands $\mathbf{x}$ and $\mathbf{y}$ of ζ (both with respect to the public key).

- **Case 1:** ($\forall i : \pi_i = 0$ or $y_i = x_i = 0$). In this case, we can unmask each component of $\mathbf{x}$ and $\mathbf{y}$ as follows:
  - If $\pi_i = 0$, then the honest DSME decryption procedure would have decrypted U_i to $u_i = \delta_i$. We can unmask this component with respect to this u_i.
  - If $y_i = x_i = 0$, then regardless of how the honest decryption procedure would have decrypted U_i, x_i unmasks to $-z_i$ and y_i unmasks to 0.

  After computing these unmasked strands of ζ (call them $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$), we check that both are scalar multiples of $\mathbf{1}$. If not, we return $\perp$. Otherwise, let $\overline{\mathbf{x}} = x\mathbf{1}$ and $\overline{\mathbf{y}} = y\mathbf{1}$.
  Set $\mu = B_X/B^x$, $\mathsf{msg} = \mathsf{encode}_{\mathbb{G}}^{-1}(\mu)$, and $m = \mathsf{encode}_{\mathbb{Z}_p}(\mathsf{msg})$. We perform the following checks:
  $$P_X \stackrel{?}{=} (CD^m)^x; \qquad B_Y \stackrel{?}{=} B^y; \qquad P_Y \stackrel{?}{=} (CD^m)^y$$
  If any check fails, return $\perp$. Otherwise return msg.

If we are called in Phase I, output $\perp$ at this point, as the following case requires the challenge ciphertext ζ^*.

- **Case 2:** ($\forall i : \pi_i = 1$ or $y_i = x_i = 0$). In this case, we try to determine if the corresponding *unmasked strands* $\overline{\mathbf{x}}^*, \overline{\mathbf{y}}^*, \overline{\mathbf{x}}, \overline{\mathbf{y}}$ are linear combinations of the following form:
  $$\overline{\mathbf{x}} = \overline{\mathbf{x}}^* + \beta\overline{\mathbf{y}}^*, \qquad \overline{\mathbf{y}} = \gamma\overline{\mathbf{y}}^*$$
  We cannot determine this directly, as we cannot decrypt the U_i's and U_i^*'s to unmask these strands. However, suppose we did have the correct DSME private keys and proceeded to decrypt $u_i = \mathsf{MDec}_{\mathbf{a}_i}(U_i)$ and $u_i^* = \mathsf{MDec}_{\mathbf{a}_i}(U_i^*)$, for each i. Then we have:
  $$\begin{aligned}
  &\left(\frac{x_1}{u_1} - z_1, \ldots, \frac{x_5}{u_5} - z_5\right) = \left(\frac{x_1^*}{u_1^*} - z_1, \ldots, \frac{x_5^*}{u_5^*} - z_5\right) + \beta\left(\frac{y_1^*}{u_1^*}, \ldots, \frac{y_5^*}{u_5^*}\right) \\
  \iff &\left(\frac{x_1}{u_1}, \ldots, \frac{x_5}{u_5}\right) = \left(\frac{x_1^*}{u_1^*}, \ldots, \frac{x_5^*}{u_5^*}\right) + \beta\left(\frac{y_1^*}{u_1^*}, \ldots, \frac{y_5^*}{u_5^*}\right) \\
  \iff &\left(x_1\frac{u_1^*}{u_1}, \ldots, x_5\frac{u_5^*}{u_5}\right) = (x_1^*, \ldots, x_5^*) + \beta(y_1^*, \ldots, y_5^*)
  \end{aligned}$$
  and likewise
  $$\left(\frac{y_1}{u_1}, \ldots, \frac{y_5}{u_5}\right) = \gamma\left(\frac{y_1^*}{u_1^*}, \ldots, \frac{y_5^*}{u_5^*}\right) \iff \left(y_1\frac{u_1^*}{u_1}, \ldots, y_5\frac{u_5^*}{u_5}\right) = \gamma(y_1^*, \ldots, y_5^*)$$
  If $\pi_i = 1$, we know that the ratio $u_i/u_i^* = \delta_i$. Otherwise, if $x_i = 0$, then the product $x_i\frac{u_i^*}{u_i} = 0$, independent of u_i^* and u_i. In either case, we know every component of the vectors in these final two equalities, without having to explicitly decrypt the U_i's and U_i^*'s. Thus we can decide whether the *unmasked* strands are linear combinations of the

appropriate form, without actually unmasking them. If they are not of this form, we return $\perp$. Otherwise, we perform the following checks:

$$B_X \stackrel{?}{=} B_X^*(B_Y^*)^\beta; \qquad P_X \stackrel{?}{=} P_X^*(P_Y^*)^\beta; \qquad B_Y \stackrel{?}{=} (B_Y^*)^\gamma; \qquad P_Y \stackrel{?}{=} (P_Y^*)^\gamma$$

If any check fails, return $\perp$, otherwise return replay.

- **Case 3:** If the previous two cases do not hold, return $\perp$.

The following lemma establishes the correctness of some of the cases where AltDec outputs $\perp$.

Lemma 6 *Consider a replay interaction in which the challenge ciphertext ζ^* is generated with* AltEnc *and the honest decryption procedure is used. Call a DSCS ciphertext "bad" if its component DSME ciphertexts decrypt successfully, and one of its unmasked strands is linearly independent of the known strands (the all-ones vector in Phase I, or the all-ones vector along with the unmasked strands of the challenge ciphertext in Phase II).*

With overwhelming probability over the key generation and randomness in AltEnc*, all bad queries to the* Dec *are rejected (i.e, it returns $\perp$).*

PROOF: Consider the first bad ciphertext (with unmasked strands $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$) submitted by the adversary. Assume without loss of generality that the independent strand is $\overline{\mathbf{x}}$. After decrypting the DSME components, the honest decryption algorithm computes a purported message msg (and corresponding μ and m) and checks the following constraint:

$$\begin{bmatrix}0 & \overline{\mathbf{x}} & m\overline{\mathbf{x}}\end{bmatrix}\begin{bmatrix}G & 0 & 0\\ 0 & G & 0\\ 0 & 0 & G\end{bmatrix}\begin{bmatrix}\mathbf{b}^T\\ \mathbf{c}^T\\ \mathbf{d}^T\end{bmatrix} = \begin{bmatrix}\log P_X\end{bmatrix}, \text{ where } G = \begin{bmatrix}\log g_1 & \cdots & 0\\ \vdots & \ddots & \vdots\\ 0 & \cdots & \log g_5\end{bmatrix}$$

The row $[0 \quad \overline{\mathbf{x}} \quad m\overline{\mathbf{x}}]$ is independent of the existing rows in the first matrix in Equation 2. Among the private keys consistent with Equation 2, exactly a $1/p$ fraction of them are also consistent with any fixed value of $\log P_X$. Thus, conditioned on the adversary's view (Equation 2), the "correct" value of P_X is uniformly distributed in $\mathbb{G}$. The adversary has a $1/p$ chance of submitting a query with that correct value.

Each time a decryption query is rejected, the adversary learns that at most a $1/p$ fraction of private keys are no longer possible. The actual private key remains uniformly distributed among the remaining values, from the adversary's point of view. By a union bound, if the adversary makes Q bad queries, one of them will be accepted with probability at most $Q/(p-Q)$. Since the adversary can make only polynomially many queries, this probability is negligible. □

We now prove a similar claim about some of the cases where AltMDec returns $\perp$.

Lemma 7 *During a replay interaction in which the challenge ciphertext ζ^* is generated with* AltEnc *and the honest decryption procedure is used, let U_i^* denote the ith DSME ciphertext component of ζ^*. Call a purported DSME ciphertext "bad" if:*

- *In Phase I, its second strand is linearly independent of the public key strand; or,*
- *In Phase II, its second strand is linearly dependent on the first strand of the corresponding U_i^*.*

With overwhelming probability (over the key generation) every bad query submitted to MDec *is rejected.*

Note that with overwhelming probability over the randomness in AltMEnc, the known strands in Phase II span the space of all strands, so we ignore the case where a Phase II query is linearly independent of the known strands.

PROOF: As in the previous lemma, consider the first bad ciphertext submitted to MDec. We consider the two cases:

- Phase I: The second strand is linearly independent of the known strands. The adversary's view in the replay interaction contains only the first row of the constraints in Equation 1. Since the second strand is independent of this row, the "correct" value for MDec's integrity check on A_W is distributed independently of this view. Over the remaining randomness in the key generation, the value A_W in the ciphertext is correct with only $1/q$ probability.
- Phase II: Suppose the second strand is linearly dependent on the first strand of the alternate ciphertext $U^* = (\mathbf{V}^*, A_V^*, \mathbf{W}^*, A_W^*)$. Over the remaining randomness in the key generation, the value of $\mathsf{MDec}(U^*)$ is distributed independently of the adversary's view in the replay interaction, by Lemma 3. Thus the value $A_V^*/\mathsf{MDec}(U^*) = \prod_i (V_i^*)^{a_i}$ is distributed independently as well. However, the integrity check on the given ciphertext must contain the correct combination of this value, or else the check fails. This can happen only with $1/q$ probability.

By a similar union bound as in the previous lemma, all bad queries are rejected with overwhelming probability. □

Lemma 8 *Consider a DSME ciphertext given in Phase I, whose first strand is linearly independent of the public key strand. Then either* MDec *rejects the ciphertext with probability 1, or the decryption of this ciphertext with* MDec *is uniformly distributed over $\widehat{\mathbb{G}}$, over the remaining randomness in the key generation.*

PROOF: By a similar argument as above, if the first strand is linearly independent of the known strands, then the value computed by MDec to derive the purported message is distributed uniformly over the group. Thus the purported message that MDec outputs is uniformly distributed. □

Lemma 9 *Suppose the adversary gives a decryption query to the honest decryption oracle whose DSME components decrypt successfully, and whose unmasked strands $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$ are linearly dependent on $\{\overline{\mathbf{x}}^*, \overline{\mathbf{y}}^*, \mathbf{1}\}$. Then, except with negligible probability (over the randomness in* AltEnc*), one of the following cases must hold:*

- $(\forall i : \pi_i = 0 \text{ or } x_i = y_i = 0)$*;* $\overline{\mathbf{x}} = x\mathbf{1}$*; and* $\overline{\mathbf{y}} = y\mathbf{1}$ *(for some x, y).*
- $(\forall i : \pi_1 = 1 \text{ or } x_i = y_i = 0)$*;* $\overline{\mathbf{x}} = \overline{\mathbf{x}}^* + \beta\overline{\mathbf{y}}^*$*; and* $\mathbf{y} = \gamma\overline{\mathbf{y}}^*$ *(for some β, γ).*

PROOF: We view this linear dependence condition as a game:

- We give an encryption from AltEnc, and its masked strands $\mathbf{x}^*$ and $\mathbf{y}^*$ are fixed. However, from Lemma 3, the $\mathbf{u}^*$ values in this ciphertext are distributed independently, and we may choose them later.
- The adversary submits a ciphertext to the decryption oracle. This fixes its masked strands $\mathbf{x}$ and $\mathbf{y}$. From Lemma 5, this also fixes a relationship between $\mathbf{u}^*$ and $\mathbf{u}$ (i.e, δ_i, π_i such that $u_i = \delta_i (u_i^*)^{\pi_i}$).
- The decryption procedure will proceed to decrypt the U_i values to unmask the strands. At this point, we can now randomly choose $\mathbf{u}^*$, which determines $\mathbf{u}$ according to the relationship in the previous step. Given these values, the corresponding unmasked strands $\overline{\mathbf{x}}^*, \overline{\mathbf{y}}^*, \overline{\mathbf{x}}, \overline{\mathbf{y}}$ are fixed. The adversary succeeds if $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$ are linearly dependent on $\{\overline{\mathbf{x}}^*, \overline{\mathbf{y}}^*, \mathbf{1}\}$.

We assume that the adversary has a strategy whereby he succeeds with noticeable probability over the choices of $\mathbf{x}^*$, $\mathbf{y}^*$, and $\mathbf{u}^*$. We will show that the only cases that do not lead to a contradiction are the two cases given in the statement of the lemma.

The strand $\overline{\mathbf{x}}$ is linearly dependent on $\{\overline{\mathbf{x}}^*, \overline{\mathbf{y}}^*, \mathbf{1}\}$ if there exist α, β, γ such that:

$$\begin{aligned}
&\overline{\mathbf{x}} = \alpha \overline{\mathbf{x}}^* + \beta \overline{\mathbf{y}}^* + \gamma \mathbf{1} \\
\iff &\left(\frac{x_1}{u_1} - z_1, \ldots, \frac{x_5}{u_5} - z_5 \right) = \alpha \left(\frac{x_1^*}{u_1^*} - z_1, \ldots, \frac{x_5^*}{u_5^*} - z_5 \right) + \beta \left(\frac{y_1^*}{u_1^*}, \ldots, \frac{y_5^*}{u_5^*} \right) + \gamma \mathbf{1} \\
\iff &\left(\frac{x_1}{\delta_1 (u_1^*)^{\pi_1}}, \ldots, \frac{x_5}{\delta_5 (u_5^*)^{\pi_5}} \right) = \alpha \left[\left(\frac{x_1^*}{u_1^*}, \ldots, \frac{x_5^*}{u_5^*} \right) - \mathbf{z} \right] + \beta \left(\frac{y_1^*}{u_1^*}, \ldots, \frac{y_5^*}{u_5^*} \right) + \gamma \mathbf{1} + \mathbf{z} \qquad (3)
\end{aligned}$$

Multiplying the ith row on both sides by $(u_i^*)^{\pi_i}$ yields the following equation:

$$\begin{bmatrix} x_1/\delta_1 \\ \vdots \\ x_5/\delta_5 \end{bmatrix} = \begin{bmatrix} (u_1^*)^{\pi_1 - 1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & (u_5^*)^{\pi_5 - 1} \end{bmatrix} \begin{bmatrix} x_1^* - z_1 u_1^* & y_1^* & u_1^* \\ \vdots & \vdots & \vdots \\ x_5^* - z_5 u_5^* & y_5^* & u_5^* \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} + \begin{bmatrix} z_1 (u_1^*)^{\pi_1} \\ \vdots \\ z_1 (u_5^*)^{\pi_5} \end{bmatrix}$$

In each case, we use the following similar argument. We will show that with overwhelming probability, the first 3 choices of $\mathbf{u}^*$ determine the coefficients of the linear combination. The adversary succeeds only if the constraint holds in the other components with noticeable probability over the choices of the remaining 2 values of $\mathbf{u}^*$. Recall that u_i^* is distributed randomly in $\widehat{\mathbb{G}}$, which is an order-q subgroup of $\mathbb{Z}_p^*$. In particular, $(u_i^*)^\pi$ is also distributed randomly unless $\pi = 0 \bmod q$.

Substituting our choice of $\mathbf{z} = (0, 0, 0, 1, 1)$ makes the following analysis simpler. We write the first 3 rows as above:

$$\begin{aligned}
\begin{bmatrix} x_1/\delta_1 \\ x_2/\delta_2 \\ x_3/\delta_3 \end{bmatrix} &= \overbrace{\begin{bmatrix} (u_1^*)^{\pi_1 - 1} & 0 & 0 \\ 0 & (u_2^*)^{\pi_2 - 1} & 0 \\ 0 & 0 & (u_3^*)^{\pi_3 - 1} \end{bmatrix}}^{1} \overbrace{\begin{bmatrix} x_1^* & y_1^* & u_1^* \\ x_2^* & y_2^* & u_2^* \\ x_3^* & y_3^* & u_3^* \end{bmatrix}}^{2} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} \\
&= \begin{bmatrix} x_1^* (u_1^*)^{\pi_1 - 1} & y_1^* (u_1^*)^{\pi_1 - 1} & (u_1^*)^{\pi_1} \\ x_2^* (u_2^*)^{\pi_2 - 1} & y_2^* (u_2^*)^{\pi_2 - 1} & (u_2^*)^{\pi_2} \\ x_3^* (u_3^*)^{\pi_3 - 1} & y_3^* (u_3^*)^{\pi_3 - 1} & (u_3^*)^{\pi_3} \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} \qquad (4)
\end{aligned}$$

With overwhelming probability, (x_1^*, x_2^*, y_3^*) and (y_1^*, y_2^*, y_3^*) are independent, as are (x_4^*, x_5^*) and (y_4^*, y_5^*). Matrix 1 is always nonsingular, as $u_i^* \neq 0$. With overwhelming probability, matrix 2 is nonsingular, and thus α, β, γ are fixed by the first 3 choices of $\mathbf{u}^*$, as desired.

We now write the remaining two constraints in the following form:

$$\begin{bmatrix} x_4/\delta_4 \\ x_5/\delta_5 \end{bmatrix} = \underbrace{\begin{bmatrix} (u_4^*)^{\pi_4 - 1} & 0 \\ 0 & (u_5^*)^{\pi_5 - 1} \end{bmatrix}}_{3} \underbrace{\begin{bmatrix} x_4^* & y_4^* \\ x_5^* & y_5^* \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix}}_{4} + \underbrace{\begin{bmatrix} (u_4^*)^{\pi_4} & 0 \\ 0 & (u_5^*)^{\pi_5} \end{bmatrix}}_{5} \underbrace{\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \gamma \\ 1 - \alpha \end{bmatrix}}_{6} \tag{5}$$

We fix the left hand side of this equation and matrices 4 and 6, and assume the equation holds with noticeable probability over the choice of u_4^*, u_5^*. We consider two cases:

- For some $i \in \{1, 2, 3\}$, we have $\pi_i = 1$. Without loss of generality, let $i = 1$. Then applying Cramer's rule to solve for γ in matrices 1 and 2, we see that $\gamma = \Delta/(\theta u_1^* + \rho)$, where Δ is the determinant of the following matrix:

$$\begin{bmatrix} x_1^* & y_1^* & x_1/\delta_1 \\ x_2^*(u_2^*)^{\pi_2 - 1} & y_2^*(u_2^*)^{\pi_2 - 1} & x_2/\delta_2 \\ x_3^*(u_3^*)^{\pi_3 - 1} & y_3^*(u_3^*)^{\pi_3 - 1} & x_3/\delta_3 \end{bmatrix}$$

  and θ, ρ are independent of u_1^*. Also, $\theta \neq 0$ except with negligible probability.

  Suppose $\Delta \neq 0$. Then γ varies in a one-to-one fashion as u_1^* varies. the fourth constraint has the following form:

$$x_4/\delta_4 = (u_4^*)^{\pi_4 - 1}\Big[\alpha x_4^* + \beta y_4^*\Big] + (u_4^*)^{\pi_4}\Big[\gamma + 1 - \alpha\Big]$$

  and for any fixed u_4^*, the right-hand side varies as u_1^* varies. This constraint only holds with negligible probability. We conclude that $\gamma = \Delta = 0$.

  We can only have $\Delta = 0$ with noticeable probability when either:

  - $\pi_2 = \pi_3 = 1$ and $x_i/\delta_i = \alpha x_i^* + \beta y_i^*$, or
  - $x_1 = x_2 = x_3 = 0$

  We now consider two subcases:

  - If $\alpha = 1$, the second term in Equation 5 vanishes. We must have either $\pi_i = 1$ or $x_i = 0$, for $i = 4, 5$ (because if $\pi_i \neq 1$ and $x_i \neq 0$, then the ith constraint varies with the choice of u_i^*).

    In this cases we get the required property that $(\forall i : x_i = 0 \text{ or } \pi_i = 1)$, and $\alpha = 1$ and $\gamma = 0$.
  - If $\alpha \neq 1$, we must have $\pi_4 = \pi_5 = 0$, since otherwise the second term of Equation 5 varies with u_4^*, u_5^*. However if $\pi_4 = \pi_5 = 0$, matrix 4 must contain all zeroes, which implies $\alpha = \beta = 0$.

    Now, $\alpha = \beta = \gamma = 0$. Substituting into Equation 4, we see that $x_1 = x_2 = x_3 = 0$. In this case we get the property that $(\forall i : x_i = 0 \text{ or } \pi_i = 0)$ and $\alpha = \beta = 0$.

- For all $i \in \{1,2,3\}$, we have $\pi_i = 0$. Note that the first two columns in the matrix of Equation 4 are randomized by $(u_i^*)^{-1}$ in each row, but the last column is all ones. Again we consider 2 subcases:
  - If $x_1/\delta_1 = x_2/\delta_2 = x_3/\delta_3$, then in the Equation 4, we see that we must have $\alpha = \beta = 0$. If $\gamma = -1$, then we get $x_4 = x_5 = 0$. Otherwise we require $\pi_4 = \pi_5 = 0$ for the conditions in Equation 5 to hold as u_i^* varies.
  - If $x_1/\delta_1, x_2/\delta_2, x_3/\delta_3$ are not all equal, then consider solving the Equation 4 for α, β, γ. Over the choice of u_1^*, u_2^*, u_3^*, it is only with negligible probability that we can obtain $\gamma = 0$ or $\gamma = \alpha - 1$. To see this, note that if γ were to obey either of these two equations, then u_3^* could be uniquely solved in terms of the other variables (by first substituting γ and then solving α and β from the first two rows of the system of equations, and then solving $(u_3^*)^{-1}$). So for Equation 5 to hold, it must be the case that $\pi_4 = \pi_5 = 0$. But then, as before, we require that matrix 4 contain all zeroes. This implies $\alpha = \beta = 0$.

  In either case, we get the property that $(\forall i : x_i = 0 \text{ or } \pi_i = 0)$ and $\alpha = \beta = 0$.

The arguments concerning linear combinations of the $\overline{\mathbf{y}}$ strand are very similar, and omitted here for brevity. When substituting $\overline{\mathbf{y}}$ for $\overline{\mathbf{x}}$, we lose the final $\mathbf{z}$ term in Equation 3. This accounts for the difference in the possible linear combinations for $\overline{\mathbf{y}}$ compared to $\overline{\mathbf{x}}$. □

Lemma 10 AltDec *faithfully implements the decryption oracle in the replay interaction described in Section 6.2, even when the encryptions are generated using* AltEnc.

PROOF: Consider a query $\zeta = (\mathbf{X}, B_X, P_X, \mathbf{Y}, B_Y, P_Y, \mathbf{U})$ made to AltDec. First, it calls AltMDec on each U_i.

- According to Lemma 7, if some U_i is "bad", then with overwhelming probability, the honest MDec procedure would have rejected and so would have Dec. This is also what AltDec does.
- If in Phase I, some U_i's first strand is linearly independent of the public key strand, then by Lemma 8, the output of MDec would be uniformly distributed in the group. When the honest Dec procedure continued to (implicitly) unmask the strands of ζ, the strands would be linearly independent of the known strands, with overwhelming probability. By Lemma 6, Dec would reject with overwhelming probability. This is also what AltDec does in this case.
- Whenever AltMDec performs an integrity check, it is easy to see that it corresponds to the same integrity check that the MDec procedure would perform. So if AltMDec outputs $\perp$ due to a failed integrity check, so would have MDec.
- Otherwise, if $\mathsf{AltMDec}(U_i)$ returns (δ_i, π_i), then by Lemma 5, these values are such that $\mathsf{MDec}(U_i) = \delta_i \mathsf{MDec}(U_i^*)^{\pi_i}$ for any consistent private key.

Given the correctness of the (δ_i, π_i) values that are computed, the AltDec procedure does check for the correct linear combinations of the ciphertext's strands (correct with respect to the $\mathbf{u}$ that the honest decryption procedure would compute).

If the conditions of Lemma 9 do not hold, then the honest decryption procedure would reject with overwhelming probability. In these cases, AltDec rejects as well (Case 3 in its description).

If Case 1 of the alternate decryption algorithm holds, the honest decryption would compute values of $\overline{X}_i = g_i^x$ and $\overline{Y}_i = g_i^y$. It then computes:

$$\mu = \frac{B_X}{\prod_{i=1}^{5} \overline{X}_i^{b_i}} = \frac{B_X}{\left(\prod_{i=1}^{5} g_i^{b_i}\right)^x} = B_X / B^x$$

which is exactly what the alternate decryption procedure computes. Similarly, the alternate decryption procedure performs the following checks:

$$P_X \stackrel{?}{=} (CD^m)^x; \qquad B_Y \stackrel{?}{=} B^y; \qquad P_Y \stackrel{?}{=} (CD^m)^y$$

which are easily verified to coincide with the honest decryption procedure's checks in this case.

If Case 2 of the alternate decryption algorithm holds, the honest decryption procedure would compute values of $\overline{X}_i = \overline{X}_i^* (\overline{Y}_i^*)^\beta$ and $\overline{Y}_i = (\overline{Y}_i^*)^\gamma$. Recall that msg^* is the message used to generate the challenge ciphertext ζ^*. The alternate decryption procedure's first check is $B_X \stackrel{?}{=} B_X^* (B_Y^*)^\beta$. If this check does not succeed, then the honest decryption procedure would compute a purported message msg' different than msg^*. Then the additional constraints

$$\begin{bmatrix} 0 & \overline{\mathbf{x}} & m'\overline{\mathbf{x}} \\ 0 & \overline{\mathbf{y}} & m'\overline{\mathbf{y}} \end{bmatrix} \begin{bmatrix} G & 0 & 0 \\ 0 & G & 0 \\ 0 & 0 & G \end{bmatrix} \begin{bmatrix} \mathbf{b}^T \\ \mathbf{c}^T \\ \mathbf{d}^T \end{bmatrix} = \left[\log P_X\right], \text{ where } G = \begin{bmatrix} \log g_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \log g_5 \end{bmatrix}$$

are linearly independent of the constraints in Equation 2, because $m' \neq m^*$. By the same logic as in the proof of Lemma 6, the honest decryption procedure would reject with overwhelming probability, which is what AltDec does as well.

Otherwise, the purported message that is computed while decrypting ζ is msg^*. Again, it can easily be checked that the alternate decryption procedure's checks:

$$P_X \stackrel{?}{=} P_X^* (P_Y^*)^\beta; \qquad B_Y \stackrel{?}{=} (B_Y^*)^\gamma; \qquad P_Y \stackrel{?}{=} (P_Y^*)^\gamma$$

coincide with the checks performed by the honest decryption procedure. If these checks succeed, the honest decryption procedure would return msg^*, whereas AltDec returns replay. This response is still considered acceptable for a replay interaction, thus we faithfully implement the guarded decryption for the replay interaction. □

7 Replayable message posting

Below is the detailed description of $\mathcal{F}_{\text{RMP}}$, the Replayable Message Posting functionality.

The functionality accepts four kinds of requests from parties.

- $\mathsf{register}$ request: on receiving a message $\mathsf{register}$ from a party sender (corrupt or honest), the functionality sends (ID-REQ, sender) to the adversary, and expects in response an identity string id.[5] If the string received in response has been already used, ignore the request. Otherwise respond to sender with the string id, and also send a message (ID-ANNOUNCE, id) to all other parties registered so far.

[5] This can be modified to have the functionality itself pick an id from a predetermined distribution specified as part of the functionality. In this case the functionality will also provide the adversary with some auxiliary information about id (e.g., the randomness used in sampling id). For simplicity we do not use such a stronger formulation.

- post request: on receiving a request $(\mathsf{post}, \mathsf{id}, \mathsf{msg})$ from a party sender, the functionality behaves as follows:
  - If id is not a registered ID, ignore the request.
  - If sender is uncorrupted and the party with ID id is uncorrupted, send $(\text{HANDLE-REQ}, \mathsf{sender})$ to the adversary, and expect in return a string handle from the adversary. Record $(\mathsf{handle}, \mathsf{id}, \mathsf{msg}, \text{HONEST})$ internally and publish $(\text{HANDLE-ANNOUNCE}, \mathsf{handle})$ to all registered parties. Here HONEST indicates that the message originated from an honest party and is addressed to an honest party.
  - If sender is corrupted, or the party with ID id is corrupted, then proceed as above, but instead of $(\text{HANDLE-REQ}, \mathsf{sender})$, send $(\text{HANDLE-REQ}, \mathsf{sender}, \mathsf{msg}, \mathsf{id})$ to the adversary while requesting the handle, and record $(\mathsf{handle}, \mathsf{id}, \mathsf{msg}, \text{ADVERSARIAL})$. Note that in this case the functionality reveals the input it received to the adversary (but as the sender or the receiver is corrupted, it is indeed legitimate for the adversary to learn the contents of the message).
- repost request: on receiving a message $(\mathsf{repost}, \mathsf{handle})$ from a party sender, the functionality behaves as follows:
  - If handle is not recorded internally, ignore the request.
  - If sender is uncorrupted and $(\mathsf{handle}, \mathsf{id}, \mathsf{msg}, \text{HONEST})$ is recorded internally, send $(\text{HANDLE-REQ}, \mathsf{sender})$ to the adversary. On receiving a new handle handle' as response from the adversary, record $(\mathsf{handle}', \mathsf{id}, \mathsf{msg}, \text{HONEST})$ internally and publish $(\text{HANDLE-ANNOUNCE}, \mathsf{handle}')$ to all registered parties.
  - If sender is corrupt or $(\mathsf{handle}, \mathsf{id}, \mathsf{msg}, \text{ADVERSARIAL})$ is recorded internally (i.e., handle is marked ADVERSARIAL), send $(\text{HANDLE-REQ}, \mathsf{sender}, \mathsf{handle})$ to the adversay (i.e., send handle too). On receiving a new handle handle' record $(\mathsf{handle}', \mathsf{id}, \mathsf{msg}, \text{ADVERSARIAL})$ internally and publish $(\text{HANDLE-ANNOUNCE}, \mathsf{handle}')$ to all registered parties. Note that if handle is ADVERSARIAL, then whenever it is reposted, the adversary learns that fact. However, the msg or id associated with handle is not revealed.
- get request: on receiving a message $(\mathsf{get}, \mathsf{handle})$ from a party, if a record $(\mathsf{handle}, \mathsf{id}, \mathsf{msg}, *)$ (i.e., marked HONEST or ADVERSARIAL) is internally recorded and the ID id was assigned to this party in response to a $\mathsf{register}$ request it made, then return $(\mathsf{id}, \mathsf{msg})$ to it. Otherwise ignore this request.

8 Proof of Theorem 2

To prove Theorem 2, for any given real-world adversary $\mathcal{A}$, we need to show an ideal-world adversary (simulator) $\mathcal{S}$, so that for all PPT environment $\mathcal{Z}$, $\text{REAL}^{\text{DSCS}}_{\mathcal{A},\mathcal{Z}} \approx \text{IDEAL}^{\mathcal{F}_{\text{RMP}}}_{\mathcal{S},\mathcal{Z}}$.

We build $\mathcal{S}$ in stages, starting from the real-world scenario and altering it step by step to get an ideal-world adversary, at every stage ensuring that the behaviours within any environment remain indistinguishable. We describe these stages below, and highlight what property of the encryption scheme is used to establish indistinguishability in that stage. All the simulators below exist in the ideal world, but are also given (progressively less) information about the inputs to the honest parties. We conveniently model this access to extra information using modified functionalities.

$\mathcal{S}_0$ and $\mathcal{F}_0$ (Correctness): $\mathcal{F}_0$ behaves exactly like $\mathcal{F}_{\text{RMP}}$ except that it also relays all the input messages it receives from honest parties to the adversary ($\mathcal{S}_0$). Thus, $\mathcal{S}_0$ has access to all the inputs to the honest parties (i.e., their communication with the functionality, or equivalently that from the environment). $\mathcal{S}_0$ internally simulates the encryption scheme algorithms for all honest parties, and lets the adversary $\mathcal{A}$ interact with these simulated parties and directly with the environment, as follows:

1. When the environment instructs an honest party to register, $\mathcal{F}_0$ relays this instruction to $\mathcal{S}_0$. Then the honest party internally simulated by $\mathcal{S}_0$ runs the key-generation algorithm and $\mathcal{S}_0$ sends the resulting public key to the functionality as the identity string for the party.

2. When an honest party sender (as instructed by the environment) sends a post request $(\mathsf{post}, \mathsf{id}, \mathsf{msg})$ to $\mathcal{F}_0$, addressed to an honest party id, $\mathcal{F}_0$ relays the message $(\mathsf{sender}, \mathsf{post}, \mathsf{id}, \mathsf{msg})$ to $\mathcal{S}_0$. Then $\mathcal{S}_0$ generates a ciphertext on behalf of sender as $\zeta = \mathsf{Enc}_{PK_{\mathsf{id}}}(\mathsf{msg})$ and sends it to $\mathcal{A}$, for delivery to all (simulated) parties. $\mathcal{S}_0$ also sends $\mathsf{handle} = \zeta$ to $\mathcal{F}_0$. $\mathcal{F}_0$ internally records $(\mathsf{handle}, \mathsf{id}, \mathsf{msg}, \text{HONEST})$. When $\mathcal{A}$ delivers an encryption to an internally simulated party, $\mathcal{S}_0$ lets the functionality deliver $(\text{HANDLE-ANNOUNCE}, \mathsf{handle})$ to the corresponding ideal honest party, who outputs it to the environment.

3. When instructed by an honest party sender to repost handle, $\mathcal{F}_0$ relays the message $(\mathsf{sender}, \mathsf{repost}, \mathsf{handle})$ to $\mathcal{S}_0$. Then $\mathcal{S}_0$ (more specifically, its internal simulation of sender) sets $\mathsf{handle}' = \mathsf{Rerand}(\mathsf{handle})$, and sends it to $\mathcal{A}$, for delivery to all parties. $\mathcal{S}_0$ sends handle' to $\mathcal{F}_0$ as the new handle. Again, when $\mathcal{A}$ delivers this ciphertext that it received to an internally simulated party, $\mathcal{S}_0$ lets $\mathcal{F}_0$ deliver $(\text{HANDLE-ANNOUNCE}, \mathsf{handle}')$ to the corresponding ideal honest party, who outputs it to the environment. Note that if $\mathcal{F}_0$ had an internal record $(\mathsf{handle}, \mathsf{id}, \mathsf{msg}, \text{HONEST})$ it will record $(\mathsf{handle}', \mathsf{id}, \mathsf{msg}, \text{HONEST})$, and if it had a record $(\mathsf{handle}, \mathsf{id}, \mathsf{msg}, \text{ADVERSARIAL})$ it will record $(\mathsf{handle}', \mathsf{id}, \mathsf{msg}, \text{ADVERSARIAL})$.

4. When the adversary creates and sends a ciphertext ζ, $\mathcal{S}_0$ does the following:

   - For each honest party's private key SK, $\mathcal{S}_0$ checks if $\mathsf{Dec}_{SK}(\zeta)$ is not $\perp$. If any of them succeeds, say for private key SK_{id} for a party with ID (public-key) id, then $\mathcal{S}_0$ sends $(\mathsf{post}, \mathsf{id}, \mathsf{msg})$ to the functionality where $\mathsf{Dec}_{SK_{\mathsf{id}}}(\zeta) = \mathsf{msg}$. When the functionality responds with a request for a handle, send $\mathsf{handle} = \zeta$.[6] Note that $\mathcal{F}_0$ will record $(\mathsf{handle}, \mathsf{id}, \mathsf{msg}, \text{ADVERSARIAL})$.
   - If none of the decryptions succeed either, then $\mathcal{S}_0$ will send a message $(\mathsf{post}, \mathsf{id}_\perp, \mathsf{msg})$ to the functionality with an arbitrary message. When the functionality responds with a request for a handle, send $\mathsf{handle} = \zeta$. $\mathcal{F}_0$ will record $(\mathsf{handle}, \mathsf{id}_\perp, \mathsf{msg}, \text{ADVERSARIAL})$.

5. When instructed to get the message from a handle, the internally simulated party runs Dec and generates corresponding output for the environment. (This is only for convenient description. The internally simulated party need not do anything here.) We call these outputs the simulated outputs. Note that the environment gets outputs from the ideal world honest parties, and the simulated outputs are simply discarded.

[6] If the decryption succeeds for more than one id, then $\mathcal{S}_0$ repeats the same for each such id, and sends the same handle to the functionality for each of the posts. This can be avoided by augmenting the correctness requirement of the encryption scheme; see the remarks in Section 9.

We denote the output of an environment $\mathcal{Z}$ when interacting with $\mathcal{S}_0$ and honest parties who interact with $\mathcal{F}_0$ by $\text{IDEAL}^{\mathcal{F}_0}_{\mathcal{S}_0,\mathcal{Z}}$.

Claim 3 *For any given PPT adversary, let $\mathcal{F}_0$ and $\mathcal{S}_0$ be as described above. Then $\forall \mathcal{Z}$ $\text{REAL}^{\text{DSCS}}_{\mathcal{A},\mathcal{Z}} \approx \text{IDEAL}^{\mathcal{F}_0}_{\mathcal{S}_0,\mathcal{Z}}$.*

PROOF: This follows from the correctness requirements of the replayable encryption scheme alone. Note that $\mathcal{S}_0$ exactly emulates the real world actions of the honest parties and $\mathcal{A}$. In addition it also interacts with the functionality to ensure that the outputs that the environment receives from the ideal honest parties are always equal to the simulated outputs. Obviously the handles output by the ideal honest parties are the same as those in simulation (because $\mathcal{S}_0$ sends these handles to the functionality). It is easy to see that the correctness of Dec and Rerand ensures that the simulated outputs of get instructions on all handles are equal to the ideal outputs. □

$\mathcal{S}_1$ and $\mathcal{F}_1$ (Rerandomizability): $\mathcal{S}_1$ is in fact identical to $\mathcal{S}_0$, but $\mathcal{F}_1$ differs from $\mathcal{F}_0$ in that it does not tell $\mathcal{S}_1$ whether a request received is a post or repost request. More specifically, $\mathcal{F}_1$ differs from $\mathcal{F}_0$ as follows:

- When a (repost, handle) command is given by an honest party sender, $\mathcal{F}_1$ checks if handle is marked ADVERSARIAL, i.e., if there is a record (handle, id, msg, ADVERSARIAL).[7] If so, $\mathcal{F}_1$ sends (sender, repost, handle) to $\mathcal{S}_1$, just as $\mathcal{F}_0$ does. On getting back a new handle handle′, it records (handle′, id, msg, ADVERSARIAL).
- If handle is marked HONEST, i.e., there is a record (handle, id, msg, HONEST) then it sends (sender, post, id, msg) to the adversary ($\mathcal{S}_1$) and requests a new handle in return (as if the command received is (post, id, msg)). On getting back a new handle handle′, it records (handle′, id, msg, HONEST).

Claim 4 *For any given PPT adversary, let $\mathcal{S}_0$, $\mathcal{F}_0$, $\mathcal{S}_1$ and $\mathcal{F}_1$ be as described above. Then $\forall \mathcal{Z}$ $\text{IDEAL}^{\mathcal{F}_0}_{\mathcal{S}_0,\mathcal{Z}} = \text{IDEAL}^{\mathcal{F}_1}_{\mathcal{S}_1,\mathcal{Z}}$.*

PROOF: This follows from the perfect rerandomizability of the replayable encryption scheme. The only way the two executions differ is in whether a ciphertext is rerandomized (as $\mathcal{S}_0$ does on receiving a repost command relayed to it by $\mathcal{F}_0$) or whether a fresh encryption is generated for the same message (as $\mathcal{S}_1$ ends up doing when $\mathcal{F}_1$ sends a post command). We point out that $\mathcal{F}_1$ replaces repost by post only for handles marked HONEST, which are guaranteed to be correctly generated ciphertexts. Then by the perfect rerandomization property of the encryption, $\mathcal{S}_1$ generates ciphertexts distributed identically to what $\mathcal{S}_0$ generates by rerandomization. Thus the two executions do not differ, and indeed $\text{IDEAL}^{\mathcal{F}_0}_{\mathcal{S}_0,\mathcal{Z}} = \text{IDEAL}^{\mathcal{F}_1}_{\mathcal{S}_1,\mathcal{Z}}$. □

[7] Recall that handle can be ADVERSARIAL if it is a ciphertext generated by or addressed to the adversary, or was generated by a simulated honest party as a rerandomization of a handle marked ADVERSARIAL.

$\mathcal{S}_2$ and $\mathcal{F}_2$ (RCCA security): $\mathcal{F}_2$ differs from $\mathcal{F}_1$ in that the adversary is not given the contents of the posted messages. When the environment instructs a party sender to post a message, $\mathcal{F}_2$ relays (sender, post, id) to $\mathcal{S}_2$ (instead of (sender, post, id, msg) as in $\mathcal{F}_1$). $\mathcal{S}_2$ behaves like $\mathcal{S}_1$, except as follows:

1. When it receives a the j^{th} message of the from (sender, post, id) from $\mathcal{F}_2$, it picks a random message $\widetilde{\mathsf{msg}}_j$, sets $\mathsf{handle}_j = \mathsf{Enc}_{PK_{\mathsf{id}}}(\widetilde{\mathsf{msg}}_j)$ and continues to behave like $\mathcal{S}_1$. In addition it internally records $(\mathsf{handle}_j, \widetilde{\mathsf{msg}}_j)$ for later reference.

2. When the adversary creates and sends a ciphertext ζ, $\mathcal{S}_2$ does the following:

   - $\mathcal{S}_2$ checks if $\mathsf{Dec}_{SK}(\zeta) = \widetilde{\mathsf{msg}}_j$ for each private key SK that it has generated so far (when honest parties send register commands to $\mathcal{F}_2$) and each randomly chosen message $\widetilde{\mathsf{msg}}_j$ (which it picked on receiving the j^{th} post command relayed by $\mathcal{F}_2$). If so, $\mathcal{S}_2$ sends the message $(\mathcal{A}, \mathsf{repost}, \mathsf{handle}_j)$ to $\mathcal{F}_2$, where handle_j is the handle that was sent in response to that post command; when $\mathcal{F}_2$ responds with a request for a new handle $\mathcal{S}_2$ sends $\mathsf{handle}' = \zeta$.[8]
   - For convenient description later, we shall have $\mathcal{S}_2$ do the same for a "dummy private key" SK^*, generated as $(PK^*, SK^*) \leftarrow \mathsf{KeyGen}$ in the beginning of the simulation.
   - If none of the decryptions match a previously stored random message, then $\mathcal{S}_2$ continues just like $\mathcal{S}_1$.

Claim 5 *For any given PPT adversary, let $\mathcal{S}_1$, $\mathcal{F}_1$, $\mathcal{S}_2$ and $\mathcal{F}_2$ be as described above. Then $\forall \mathcal{Z}$* $\text{IDEAL}^{\mathcal{F}_1}_{\mathcal{S}_1,\mathcal{Z}} \approx \text{IDEAL}^{\mathcal{F}_2}_{\mathcal{S}_2,\mathcal{Z}}$.

Proof: This follows from the RCCA security of the replayable encryption scheme. Intuitively, the only way the two executions differ is in whether the simulator provides encryptions of the actual message (as $\mathcal{S}_1$ does) or of a random message (as $\mathcal{S}_2$ does). Note that in the execution $\text{IDEAL}^{\mathcal{F}_2}_{\mathcal{S}_2,\mathcal{Z}}$, if the adversary sends an encryption to the random message chosen by $\mathcal{S}_2$ or to the actual message that was sent by the honest party to $\mathcal{F}_2$, $\mathcal{F}_2$ ends up recording the actual message again; this is not the case with $\text{IDEAL}^{\mathcal{F}_2}_{\mathcal{S}_2,\mathcal{Z}}$, but we can consider modifying $\mathcal{S}_1$ to pick a random message, and if the adversary sends an encryption to that message (which can happen with only negligible probability), replace it with the actual message (as $\mathcal{S}_2$ does). Then (the modified) $\mathcal{S}_1$ and $\mathcal{S}_2$ essentially implement the two indistinguishable experiments in RCCA game.

But to apply the RCCA security guarantee first we will have to reduce to the case of a single encryption. This is done using a series of hybrid experiments.

We define $\hat{\mathcal{S}}_i$ which interacts with $\mathcal{F}_1$ (and hence receives the message in the post commands) to be exactly like $\mathcal{S}_1$, but with the following differences:

- Let the j^{th} post command reported by $\mathcal{F}_1$ be $(\mathsf{post}, \mathsf{msg}_j, \mathsf{id}_j)$ $(1 \le j \le n)$. For each j, $\hat{\mathcal{S}}_i$ picks random messages $\widetilde{\mathsf{msg}}_j$. For $j \le i$ it uses $\mathsf{Enc}_{PK_{\mathsf{id}_j}}(\widetilde{\mathsf{msg}}_j)$ as the simulated encryption, but for $j > i$, it uses $\mathsf{Enc}_{PK_{\mathsf{id}_j}}(\mathsf{msg}_j)$ just like $\mathcal{S}_1$.

[8] As before, if the decryption succeeds for more than one id, then $\mathcal{S}_2$ repeats the same for each such id, and sends the same handle to the functionality for each of the posts.

- When the adversary produces a ciphertext ζ, $\hat{\mathcal{S}}_i$ checks if $\mathsf{Dec}_{SK}(\zeta)$ equals msg_j *or* $\widetilde{\mathsf{msg}}_j$ for any j, for any private key SK it generated so far (including SK^*). If so it sends $(\mathcal{A}, \mathsf{post}, \mathsf{msg}_j, \mathsf{id})$ to $\mathcal{F}_1$ (where id corresponds to the private key that yielded the decryption), and follows up by sending ζ as the handle.[9] Recall that in contrast, $\hat{\mathcal{S}}_i$ would send $(\mathcal{A}, \mathsf{post}, \widetilde{\mathsf{msg}}_j, \mathsf{id})$ if the decryption is to $\widetilde{\mathsf{msg}}_j$.

Clearly $\hat{\mathcal{S}}_n$ is identical to $\mathcal{S}_2$ (and it does not use the messages included in the post commands relayed by $\mathcal{F}_1$), and so $\text{IDEAL}^{\mathcal{F}_1}_{\hat{\mathcal{S}}_n,\mathcal{Z}} = \text{IDEAL}^{\mathcal{F}_2}_{\mathcal{S}_2,\mathcal{Z}}$. However $\hat{\mathcal{S}}_0$ and $\mathcal{S}_1$ differ slightly, in that if $\mathsf{Dec}_{SK}(\zeta) = \widetilde{\mathsf{msg}}_j$ (for some private key SK) then $\hat{\mathcal{S}}_0$ replaces $\widetilde{\mathsf{msg}}_j$ by msg_j. However this can happen only with negligible probability as adversary's view is independent of $\widetilde{\mathsf{msg}}_j$ in the interaction with $\hat{\mathcal{S}}_0$ up to that point. So we do have $\text{IDEAL}^{\mathcal{F}_1}_{\mathcal{S}_1,\mathcal{Z}} \approx \text{IDEAL}^{\mathcal{F}_1}_{\hat{\mathcal{S}}_0,\mathcal{Z}}$.

That $\text{IDEAL}^{\mathcal{F}_1}_{\hat{\mathcal{S}}_i,\mathcal{Z}} \approx \text{IDEAL}^{\mathcal{F}_1}_{\hat{\mathcal{S}}_{i+1},\mathcal{Z}}$ follows from the RCCA-2 security of the encryption, because $\hat{\mathcal{S}}_i$ and $\hat{\mathcal{S}}_{i+1}$ can be seen as carrying out the the RCCA experiment with the bit $b = 0$ and $b = 1$ respectively. □

$\mathcal{S}_3$ and $\mathcal{F}_3$ (Receiver-anonymity): $\mathcal{S}_3$ and $\mathcal{F}_3$ are similar to $\mathcal{S}_2$ and $\mathcal{F}_2$, except for the following:

- $\mathcal{F}_3$ does not send the ID id (i.e., the receiver's public-key) when relaying the post instructions to $\mathcal{S}_3$.
- So $\mathcal{S}_3$ uses the dummy key-pair $(PK^*, SK^*) \leftarrow \mathsf{KeyGen}$ for producing encryptions (note that it still encrypts the randomly chosen messages). That is, when the j^{th} post instruction is relayed, $\mathcal{S}_3$ picks a random message $\widetilde{\mathsf{msg}}_j$ and sets $\mathsf{handle}_j = \mathsf{Enc}_{PK^*}(\widetilde{\mathsf{msg}}_j)$.

Note that on receiving a ciphertext ζ from $\mathcal{A}$, $\mathcal{S}_3$ (just like $\mathcal{S}_2$) tries to decrypt it with SK^* as well as with the private keys for the simulated honest parties.

Claim 6 *For any given PPT adversary, let $\mathcal{S}_2$ and $\mathcal{S}_3$ be as described above. Then $\forall \mathcal{Z}$ $\text{IDEAL}^{\mathcal{F}_2}_{\mathcal{S}_2,\mathcal{Z}} \approx \text{IDEAL}^{\mathcal{F}_3}_{\mathcal{S}_3,\mathcal{Z}}$.*

PROOF: The proof uses the receiver-anonymity of the encryption scheme. A series of hybrids are used to reduce a non-negligible advantage of distinguishing the two situations to a non-negligible advantage in a receiver-anonymity game (which involves only two key-pairs and one message). The details follow. Let n be an upperbound on the number of IDs and ℓ be an upperbound on the number of messages that will be posted by honest players (ℓ and n are not predetermined; they can depend on $\mathcal{A}$ and $\mathcal{Z}$). Let the IDs be denoted by $\mathsf{id}^1, \ldots, \mathsf{id}^n$ (sorted say in the order in which they are generated); let the key-pair generated for id^i be denoted by (PK^i, SK^i) (where $PK^i = \mathsf{id}^i$). Also let the random message chosen by $\mathcal{S}_2$ for the j-th post command with id^i as the receiver be denoted by msg^i_j (for $1 \le j \le \ell$). We consider a hybrid simulator $\hat{\mathcal{S}}^i_j$ which behaves like $\mathcal{S}_2$, except as follows:

- On receiving $(\mathsf{sender}, \mathsf{post}, \mathsf{id}^{i'})$ for the $j'th$ time, with $(i', j') \le (i, j)$ (i.e., $i' < i$, or $i' = i$ and $j' \le j$) from $\mathcal{F}_2$, $\hat{\mathcal{S}}^i_j$ picks a random message $\mathsf{msg}^{i'}_{j'}$ and encrypts it using PK^* to get $\zeta^{i'}_{j'} = \mathsf{Enc}_{PK^*}(\mathsf{msg}^{i'}_{j'})$.

[9] As before, if the decryption succeeds for more than one id, the simulator repeats the same for each such id, and sends the same handle to the functionality for each of the posts.

Note that $\hat{\mathcal{S}}^n_\ell$ is the same as $\mathcal{S}_3$, and $\hat{\mathcal{S}}^1_0$ is the same as $\mathcal{S}_2$. Further $\text{IDEAL}^{\mathcal{F}_2}_{\mathcal{S}_3,\mathcal{Z}} = \text{IDEAL}^{\mathcal{F}_3}_{\mathcal{S}_3,\mathcal{Z}}$, because $\mathcal{S}_3$ does not use the extra inputs from $\mathcal{F}_2$. So by the standard hybrid argument it is enough to show that $\text{IDEAL}^{\mathcal{F}_2}_{\hat{\mathcal{S}}^i_j,\mathcal{Z}} \approx \text{IDEAL}^{\mathcal{F}_2}_{\hat{\mathcal{S}}^i_{j-1},\mathcal{Z}}$ (where $\hat{\mathcal{S}}^i_0$ is the same as $\hat{\mathcal{S}}^{i-1}_\ell$, or in the case of $i = 1$, same as $\mathcal{S}_2$). The difference between $\hat{\mathcal{S}}^i_j$ and $\hat{\mathcal{S}}^i_{j-1}$ is whether msg^i_j was encrypted using PK^* or PK^i (and whether ζ^i_j is decrypted using SK^* or SK^i). We now argue that an environment $\mathcal{Z}$ and adversary $\mathcal{A}$ who can distinguish between these two cases for some (i, j) can be converted to an adversary $\mathcal{A}^{\text{anon}}$ with a non-negligible advantage in the receiver-anonymity game. Without loss of generality, assume that $\Pr[\text{IDEAL}^{\mathcal{F}_2}_{\hat{\mathcal{S}}^i_j,\mathcal{Z}} = 0] > \Pr[\text{IDEAL}^{\mathcal{F}_2}_{\hat{\mathcal{S}}^i_{j-1},\mathcal{Z}} = 0]$ (and $\Pr[\text{IDEAL}^{\mathcal{F}_2}_{\hat{\mathcal{S}}^i_1,\mathcal{Z}} = 0] > \Pr[\text{IDEAL}^{\mathcal{F}_2}_{\hat{\mathcal{S}}^{i-1}_\ell,\mathcal{Z}} = 0]$).

First consider $\mathcal{A}^{\text{anon}}$ running in the game with $b = 0$, as follows. $\mathcal{A}^{\text{anon}}$ internally runs the system with $\mathcal{Z}$, $\hat{\mathcal{S}}^i_j$ and the ideal functionality $\mathcal{F}_3$, but with the following modifications.

- Let $(PK^*, PK^i) = (PK_0, PK_1)$ where the latter keys are provided by the game. The game also provides access to the decryption oracles (guarded against the challenge message in Phase II).
- Note that now $\hat{\mathcal{S}}^i_j$ does not have the secret keys SK^* and SK^i, and hence it needs to get access to oracles to carry out decryptions.
- When $\hat{\mathcal{S}}^i_j$ wants to encrypt msg^i_j $\mathcal{A}^{\text{anon}}$ outputs this message in the challenge phase. It receives $\zeta^* \leftarrow \mathsf{Enc}_{PK_b}(\mathsf{msg}^i_j)$, which it uses as the encryption for $\hat{\mathcal{S}}^i_j$. When $b = 0$ this is an encryption using PK^*, just as $\hat{\mathcal{S}}^i_j$ requires.
- After this note that the guarded decryption oracles are enough to run $\hat{\mathcal{S}}^i_j$. This is because when $\mathcal{A}$ produces a ciphertext ζ, $\hat{\mathcal{S}}^i_j$ needs to only check if $\mathsf{Dec}_{SK}(\zeta) = \widetilde{\mathsf{msg}}_j$ for *any private key SK it has produced* (and in particular if either $\mathsf{Dec}_{SK^i}(\zeta)$ or $\mathsf{Dec}_{SK^*}(\zeta)$ equals msg^i_j; in either case $\hat{\mathcal{S}}^i_j$ will request a repost). So $\mathcal{A}^{\text{anon}}$ can simply use the guarded decryption oracle for this.

Finally, $\mathcal{A}^{\text{anon}}$ outputs what $\mathcal{Z}$ outputs in this internal simulation. As sketched above, if $b = 0$, this is a perfect simulation of $\hat{\mathcal{S}}^i_j$ running with $\mathcal{Z}$ and the ideal parties and the functionality $\mathcal{F}_3$. On the other hand if $b = 1$, then this is a perfect simulation of $\hat{\mathcal{S}}^i_{j-1}$ (or $\hat{\mathcal{S}}^{i-1}_\ell$, if j=1) running likewise. Then a non-negligible advantage for $\mathcal{Z}$ to distinguish between the two cases translates to a non-negligible advantage for $\mathcal{A}^{\text{anon}}$ in the receiver-anonymity game. □

Concluding the proof. Combining the above claims we get that $\text{REAL}^{\text{DSCS}}_{\mathcal{A},\mathcal{Z}} \approx \text{IDEAL}^{\mathcal{F}_3}_{\mathcal{S}_3,\mathcal{Z}}$. Note that $\mathcal{F}_3$ is in fact identical to $\mathcal{F}_{\text{RMP}}$. So letting $\mathcal{S} = \mathcal{S}_3$ completes the proof.

9 Extensions

Once our constructions are made available as a UC-secure realization of $\mathcal{F}_{\text{RMP}}$, it is easier to extend in a modular fashion. We describe a few extensions which are easily achieved, yet can be very useful.

In some applications, it is convenient for the recipient of a ciphertext to be able to check whether it is a rerandomization of another ciphertext, or a "fresh" re-encryption of the same message. We

call such a functionality a *replay-test* functionality. A replay-test precludes having perfect or even statistical rerandomization functionality, and so we must settle for a computational definition of rerandomization.

We point out that redefining RCCA security, receiver-anonymity and rerandomizability in this case is a non-trivial extension of our current definitions. In particular, note that in a chosen ciphertext attack, an adversary can now access the replay-test oracle as well as the decryption oracle, and queries to the decryption responses are now guarded based on the replay-test check.

However instead of modifying our security definitions based on standalone experiments, we can directly formulate a UC functionality. The new functionality is identical to $\mathcal{F}_{\text{RMP}}$, but in addition provides the designated user with a Test command: a party can give two handles, and if it is the designated receiver of both the handles, then the functionality tells it whether the two handles were derived as reposting of the same original post. For this the functionality maintains some extra book-keeping internally. This functionality can be easily achieved starting from $\mathcal{F}_{\text{RMP}}$: to each message, a random nonce is appended, before posting it. Test is implemented by the receiver retrieving the messages of the two handles and comparing their random nonces.

The second extension we point out is of adding authentication. As should be intuitive, this can be achieved by signing the messages using a public-key signature scheme, before posting them. In terms of the functionality, a separate register feature is provided which allows *senders* to register themselves (this corresponds to publishing the signature verification key). Then the get command is augmented to provide not only the message in the handle, but also who originally posted the handle. The identities for receiving messages and sending messages are separate, but they can be tied together (by signing the encryption key and publishing it), so that only the signature verification keys can be used as identities in the system.

The way we have presented our encryption scheme, there is a hard limit on the message length, as the message must be encoded as an element in a group of fixed size. However $\mathcal{F}_{\text{RMP}}$ can easily be extended to handle messages of variable lengths: for this the longer message is split into smaller pieces; to each piece is appended a serial number and a common random nonce; further the first piece also carries the total number of pieces. Then each piece is posted using the fixed-length $\mathcal{F}_{\text{RMP}}$ functionality. Decryption performs the obvious simple integrity checks on receiving a set of ciphertexts and discards them if they are not all consistent and complete. Note that the new variable-length $\mathcal{F}_{\text{RMP}}$ functionality achieved thus informs the adversary of the length of the message (i.e., number of pieces) while posting or reposting a handle. It is easy to construct a simulator (on receiving a handle and a length, the simulator creates the appropriate number of handles and reports to the adversary; when the adversary reposts handles, the simulator will not make a repost unless all handles it generated for one variable-length handle are reposted together).

Finally, the $\mathcal{F}_{\text{RMP}}$ functionality can be strengthened to disallow duplicate handles. This corresponds to ensuring that a ciphertext in the rerandomizable RCCA encryption scheme does not successfully decrypt under multiple private keys. This requires us to introduce an additional security condition for the encryption scheme, namely, an adversary, when given two public keys and access to their decryption oracles, cannot generate a ciphertext which decrypts to valid plaintexts under both private keys. It can be shown that our RCCA encryption scheme does meet this requirement, with minor modifications.

Note that these extensions can be applied one after the other.

10 Conclusion

There are several interesting directions that this work leads to. One question relates to improving the efficiency of the constructions. Public-key encryption schemes like Cramer-Shoup are much less efficient than the private-key schemes. To exploit the best of both worlds, one can use a hybrid encryption scheme which uses a public-key encryption scheme to share a private-key over the public-channel, and then carry out the private-key encryption for the actual voluminous data. It is interesting to ask if this kind of a hybrid scheme can be devised for a rerandomizable encryption schemes. Consider using a stream cipher (or pseudorandom generator, PRG) as the private-key encryption scheme: here the output of the PRG would be used like a one-time pad on the data, and the input to it would be encrypted using the public-key encryption scheme. One approach for making such a scheme rerandomizable would be to build some sort of a homomorphic PRG and a corresponding homomorphic public-key encryption scheme which would allow anyone to rerandomize an encryption of the input for the PRG in such a way that the output of the PRG is rerandomized in a predictable way.

This leads us to the second question of extending Replayable CCA security definition to allow some homomorphic properties of the encryption. Note that CCA security requires non-malleability and homomorphism seeks to demand (specific kinds of) malleability. A definition combining the two should allow just the specific kinds of malleability as required by the homomorphic property and be non-malleable otherwise. It is an interesting direction to explore potential applications of such schemes and if such schemes can indeed be created.

Finally, we based our schemes on the DDH assumption. However as we mentioned before it is likely that the extensions of [12] can be adapted for our problem too. But we point out that our requirements on the "Universal Hash Proofs" would be a little more demanding than what [12] required. In particular, when using the double-strand approach, we seem to require 5-universality and not 2-universality. (This corresponds to our use of five bases $g_1, \ldots, g_5$ instead of just two in [11].) We leave it for future work to investigate this.

Acknowledgment

We would like to acknowledge useful discussions with Rui Xue about the work in [12].

References

[1] J. H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In L. R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 83–107. Springer, 2002.

[2] J. K. Andersen and E. W. Weisstein. Cunningham chain. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/CunninghamChain.html, 2005.

[3] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In C. Boyd, editor, *ASIACRYPT*, volume 2248 of *Lecture Notes in Computer Science*, pages 566–582. Springer, 2001.

[4] D. Boneh. The decision diffie-hellman problem. In J. Buhler, editor, *ANTS*, volume 1423 of *Lecture Notes in Computer Science*, pages 48–63. Springer, 1998.

[5] J. Camenisch and A. Lysyanskaya. A formal treatment of onion routing. In V. Shoup, editor, *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 169–187. Springer, 2005.

[6] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Electronic Colloquium on Computational Complexity (ECCC) TR01-016, 2001. Previous version "A unified framework for analyzing security of protocols" availabe at the ECCC archive TR01-016. Extended abstract in FOCS 2001.

[7] R. Canetti, H. Krawczyk, and J. B. Nielsen. Relaxing chosen-ciphertext security. In D. Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 565–582. Springer, 2003.

[8] D. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 4(2), February 1981.

[9] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[10] D. Chaum. The dining cryptographers problem: Unconditional sender and recipient untraceability. *Journal of Cryptology*, 1:65–75, 1988.

[11] R. Cramer and V. Shoup. A practical public key cryptosystem provably secure against adaptive chosen ciphertext attack. In H. Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*. Springer, 1998.

[12] R. Cramer and V. Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In L. R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2002.

[13] R. Dingledine, N. Mathewson, and P. F. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, pages 303–320. USENIX, 2004.

[14] E. Elkind and A. Sahai. A unified methodology for constructing public-key encryption schemes secure against adaptive chosen-ciphertext attack. Cryptology ePrint Archive, Report 2002/042, 2002. http://eprint.iacr.org/.

[15] Free haven project. http://freehaven.net/.

[16] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, pages 10–18, 1984.

[17] O. Goldreich. *Foundations of Cryptography: Basic Applications*. Cambridge University Press, 2004.

[18] D. M. Goldschlag, M. G. Reed, and P. F. Syverson. Onion routing. *Commun. ACM*, 42(2):39–41, 1999.

[19] J. Gröth. Rerandomizable and replayable adaptive chosen ciphertext attack secure cryptosystems. In M. Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 152–170. Springer, 2004.

[20] M. Lad. Personal communication, 2005.

[21] The onion routing program. http://www.onion-router.net/. A program sponsored by the Office of Naval Research, DARPA and the Naval Research Laboratory.

[22] B. Pfitzmann and M. Waidner. Composition and integrity preservation of secure reactive systems. In *ACM Conference on Computer and Communications Security*, pages 245–254, 2000.

[23] M. Prabhakaran. *New Notions of Security*. PhD thesis, Department of Computer Science, Princeton University, 2005.

[24] M. Prabhakaran and A. Sahai. New notions of security: achieving universal composability without trusted setup. In *STOC*, pages 242–251. ACM, 2004.

[25] F. H. Project. Anonymity bibliography. http://freehaven.net/anonbib/, 2006.

[26] M. Reiter and A. Rubin. Crowds: Anonymity for web transactions. *ACM Transactions on Information and System Security*, 1(1), June 1998.

[27] V. Shoup. A proposal for an iso standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001. http://eprint.iacr.org/.