Rerandomizable RCCA Encryption[*]

MANOJ PRABHAKARAN
University of Illinois
Urbana-Champaign, IL
mmp@uiuc.edu

MIKE ROSULEK
University of Illinois
Urbana-Champaign, IL
rosulek@uiuc.edu

June 27, 2007

Abstract

We give the first perfectly rerandomizable, Replayable-CCA (RCCA) secure encryption scheme, positively answering an open problem of Canetti et al. (*CRYPTO* 2003). Our encryption scheme, which we call the *Double-strand Cramer-Shoup scheme*, is a non-trivial extension of the popular Cramer-Shoup encryption. Its security is based on the standard DDH assumption. To justify our definitions, we define a powerful "Replayable Message Posting" functionality in the Universally Composable (UC) framework, and show that any encryption scheme that satisfies our definitions of rerandomizability and RCCA security is a UC-secure implementation of this functionality. Finally, we enhance the notion of rerandomizable RCCA security by adding a receiver-anonymity (or key-privacy) requirement, and show that it results in a correspondingly enhanced UC functionality. We leave open the problem of constructing a scheme achieving this enhancement.

[*]An extended abstract of this work appears in the proceedings of *CRYPTO 2007*, published by Springer. This full version is available from `http://eprint.iacr.org/2007/119`, and includes improvements over the extended abstract, and over the previous (April 2007) full version.

Contents

1 Introduction

Non-malleability and rerandomizability are opposing requirements to place on an encryption scheme. Non-malleability insists that an adversary should not be able to use one ciphertext to produce another one which decrypts to a related value. 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 value. Achieving this delicate tradeoff was proposed as an open problem by Canetti et al. [7].

We present the first (perfectly) rerandomizable, RCCA-secure public-key encryption scheme. Because our scheme is a non-trivial variant of the Cramer-Shoup scheme, we call it the *Double-strand Cramer-Shoup* encryption. Like the original Cramer-Shoup scheme, the security of our scheme is based on the Decisional Diffie Hellman (DDH) assumption. Additionally, our method of using ciphertext components from two related groups may be of independent interest.

Going further, we give a combined security definition in the Universally-Composable (UC) security framework by defining a "Replayable Message Posting" functionality $\mathcal{F}_{\text{RMP}}$. As a justification of the original definitions of rerandomizability and RCCA security, we show that any scheme which satisfies these definitions is also a *UC-secure realization* of the functionality $\mathcal{F}_{\text{RMP}}$. (Here we restrict ourselves to static adversaries, as opposed to adversaries who corrupt the parties adaptively.) As an additional contribution on the definitional front, in Section 9.1, we introduce a notion of receiver anonymity for RCCA encryptions, and a corresponding UC functionality.

$\mathcal{F}_{\text{RMP}}$ is perhaps the most sophisticated functionality that has been UC-securely realized in the standard model, i.e., without super-polynomial simulation, global setups, or an honest majority assumption.

Once we achieve this UC-secure functionality, simple modifications can be made to add extra functionality to our scheme, such as authentication and replay-testability (the ability for a ciphertext's recipient to check whether it was obtained via rerandomization of another ciphertext, or was encrypted independently).

Related work. Replayable-CCA security was proposed by Canetti et al. [7] as a relaxation of standard CCA security. They also raised the question of whether a scheme could be simultaneously *rerandomizable* and RCCA secure. Gröth [18] presented a rerandomizable scheme that achieved a weaker form of RCCA security, and another with full RCCA security in the *generic groups* model. Our work improves on [18], in that our scheme is more efficient, and we achieve full RCCA security in a standard model.

Rerandomizable encryption schemes also appear using the term *universal re-encryption* schemes (*universal* refers to the fact that the rerandomization/re-encryption routine does not require the public key), introduced by Golle et al. [17]. Their CPA-secure construction is based on El Gamal, and our construction can be viewed as a non-trivial extension of their approach, applied to the Cramer-Shoup construction.

The notion of receiver-anonymity (key-privacy) that we consider in Section 9.1 is an extension to the RCCA setting, of a notion due to Bellare et al. [3] (who introduced it for the simpler CPA and CCA settings).

As mentioned before, our encryption scheme is based on the Cramer-Shoup scheme [9, 10], which in turn is modeled after El Gamal encryption [14]. The security of these schemes and our own is based on the DDH assumption (see, e.g. [4]). Cramer and Shoup [10] later showed a wide range of encryption schemes based on various assumptions which provide CCA security, under a framework subsuming their original scheme [9]. 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).

Shoup [25] and An et al. [1] introduced a variant of RCCA security, called *benignly malleable*, 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 modified definition of RCCA security, where the replay-equivalence relation is computable only by the ciphertext's designated recipient. Such a functionality also precludes perfect rerandomization, though our modification does achieve a computational relaxation of the rerandomization requirement.

Motivating applications. Golle et al. [17] propose a CPA-secure rerandomizable encryption scheme for use in mixnets [8] with applications to RFID tag anonymization. Implementing a re-encryption mixnet using a rerandomizable encryption scheme provides a significant simplification over previous implementations, which require distributed key management. Golle et al. call such networks *universal mixnets*. Some attempts have been made to strengthen their scheme against a naïve chosen-ciphertext attack, including by Klonowski et al. [19], who augment the scheme with a rerandomizable RSA signature. However, these modifications still do not prevent all practical chosen-ciphertext attacks, as demonstrated by Danezis [11].

We anticipate that by achieving full RCCA security, our construction will be an important step towards universal mixnets that do not suffer from active chosen-ciphertext attacks. However, mixnet applications tend to also require a "receiver-anonymity" property (see Section 9.1) from the underlying encryption scheme. In fact, the utility of rerandomizable RCCA encryption is greatly enhanced by this anonymity property. We do not have a scheme which achieves this. However, our current result is motivated in part by the power of such a scheme. We illustrate its potential with another example application (adapted from a private communication [20]). Consider a (peer-to-peer) network routing scenario, with the following requirements: (1) each packet should carry a *path object* which encodes its entire path to the destination; (2) each node in the network should not get any information from a path object other than the length of the path and the next hop in the path; and (3) there should be a mechanism to broadcast link-failure information so that any node holding a path object can check if the failed link occurs in that path, without gaining any additional information. This problem is somewhat similar to "Onion Routing" [5, 12, 16, 21]. However, adding requirement (3) makes the above problem fundamentally different. Using an *anonymous*, rerandomizable, RCCA-secure encryption scheme one can achieve this selective revealing property as well as anonymity. We defer a more formal treatment of this scenario to future work.

2 Definitions

We call a function ν *negligible in* n if it asymptotically approaches zero faster than any inverse polynomial in n; 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 give the syntax of a perfectly rerandomizable encryption scheme, and then state our security requirements, which are formulated as indistinguishability experiments. Later, we justify these indistinguishability-based definitions by showing that any scheme which satisfies them is a secure realization of a powerful functionality in the UC security model, which we define in Section 5.

Syntax and correctness of a perfectly rerandomizable encryption scheme. A perfectly rerandomizable encryption scheme consists of four polynomial-time algorithms (polynomial in the implicit security parameter):

1. KeyGen: a randomized algorithm which outputs a *public key* PK and a corresponding *private key* SK.
2. Enc: a randomized encryption algorithm which takes a *plaintext* (from a plaintext space) and a public key, and outputs a *ciphertext.*
3. Rerand: a randomized algorithm which takes a ciphertext and outputs another ciphertext.
4. Dec: a deterministic decryption algorithm which takes a private key and a ciphertext, and outputs either a plaintext or an error indicator $\perp$.

We emphasize that the Rerand procedure takes only a ciphertext as input, and in particular, no public key.

We require the scheme to satisfy the following correctness properties for all key pairs $(PK, SK) \leftarrow \mathsf{KeyGen}$:

- For every plaintext msg and every (honestly generated) ciphertext $\zeta \leftarrow \mathsf{Enc}_{PK}(\mathsf{msg})$, we must have $\mathsf{Dec}_{SK}(\zeta) = \mathsf{msg}$.
- For every independently chosen $(PK', SK') \leftarrow \mathsf{KeyGen}$, the sets of honestly generated ciphertexts under PK and PK' are disjoint, with overwhelming probability over the randomness of KeyGen.
- For every plaintext msg and every (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 every (purported) ciphertext ζ and every $\zeta' \leftarrow \mathsf{Rerand}(\zeta)$, we must have $\mathsf{Dec}_{SK}(\zeta') = \mathsf{Dec}_{SK}(\zeta)$.

In other words, decryption is the inverse of encryption, and ciphertexts can be labeled “honestly generated” for at most one honestly generated key pair. We require that 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 all private keys).

Perfect vs. computational rerandomization. For simplicity, we only consider statistically perfect rerandomization. However, for most purposes (including our UC functionality), a computational relaxation suffices. Computational rerandomization can be formulated as an indistinguishability experiment against an adversary; given two ciphertexts (of a chosen plaintext), no adversary can have a significant advantage in determining whether they are independent encryptions or if one is a rerandomization of the other. As in our other security experiments, the adversary is given access to a decryption oracle.

Replayable-CCA (RCCA) security. We use the definition from Canetti et al. [7]. An encryption scheme is said to be *RCCA secure* if the advantage of any PPT adversary $\mathcal{A}$ in the following experiment is negligible:

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

Tightness of decryption. An encryption scheme is said to have *tight decryption* if the success probability of any PPT adversary $\mathcal{A}$ in the following experiment is negligible:

1. Pick $(PK, SK) \leftarrow \mathsf{KeyGen}$ and give PK to $\mathcal{A}$.
2. $\mathcal{A}$ gets access to the decryption oracle $\mathsf{Dec}_{SK}(\cdot)$.
3. $\mathcal{A}$ outputs a ciphertext ζ. $\mathcal{A}$ is said to *succeed* if $\mathsf{Dec}_{SK}(\zeta) = \mathsf{msg} \neq \perp$ for some msg, yet ζ is *not* in the range of $\mathsf{Enc}_{PK}(\mathsf{msg})$.

Observe that when combined with correctness property (2), this implies that an adversary cannot generate a ciphertext which successfully decrypts under more than one honestly generated key. Such a property is useful in achieving a more robust definition of our UC functionality $\mathcal{F}_{\text{RMP}}$ in Section 5 (without it, a slightly weaker yet still meaningful definition is achievable).

2.2 Decisional Diffie-Hellman (DDH) Assumption

Let $\mathbb{G}$ be a (multiplicative) cyclic group of prime order p. The *Decisional Diffie-Hellman (DDH) assumption in* $\mathbb{G}$ is that the following two distributions are computationally indistinguishable:

$$\{(g, g^a, g^b, g^{ab})\}_{g \leftarrow \mathbb{G}; a,b \leftarrow \mathbb{Z}_p} \quad \text{and} \quad \{(g, g^a, g^b, g^c)\}_{g \leftarrow \mathbb{G}; a,b,c \leftarrow \mathbb{Z}_p}.$$

Here, $x \leftarrow X$ denotes that x is drawn uniformly at random from a set X.

Cunningham chains. Our construction requires two (multiplicative) cyclic groups with a specific relationship: $\mathbb{G}$ of prime[1] 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).

As a concrete example, the DDH assumption is believed to hold in $\mathbb{QR}_p^*$, the group of quadratic residues modulo p, where p and $\frac{p-1}{2}$ are prime (i.e, p is a *safe prime*). Given a sequence of primes $(q, 2q+1, 4q+3)$, 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 which can be recombined with each other for rerandomization without changing the encrypted value. To motivate this idea, we sketch the rerandomizable scheme of Golle et al. [17], which is based on the El Gamal scheme and secure against chosen plaintext attacks.

Recall that in an El Gamal encryption scheme over a group $\mathbb{G}$ of order p, the private key is $a \in \mathbb{Z}_p$ and the corresponding public key is $A = g^a$. A message $\mu \in \mathbb{G}$ is encrypted into the pair $(g^v, \mu A^v)$ for a random $v \in \mathbb{Z}_p$.

To encrypt a message $\mu \in \mathbb{G}$ in a "Double-strand El Gamal" scheme, we generate two (independent) El Gamal ciphertexts: one of μ (say, C_0) and one of the identity element in $\mathbb{G}$ (say, C_1). Such a double-strand ciphertext (C_0, C_1) can be rerandomized by computing $(C_0', C_1') = (C_0 C_1^r, C_1^s)$ for random $r, s \leftarrow \mathbb{Z}_p$ (where the operations on C_0 and C_1 are component-wise).

Our construction adapts this paradigm of rerandomization for Cramer-Shoup ciphertexts, and when chosen ciphertext attacks are considered. The main technical difficulty is in ensuring that the prescribed rerandomization procedure is the *only* way in which "strands" can be used to generate a valid ciphertexts.

Cramer-Shoup encryption. The Cramer-Shoup scheme [9] uses 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 $m = \mathsf{H}(X_1, X_2, \mu B^x)$ and H is a collision-resistant hash function.

In our scheme the ciphertext will contain two "strands," each one similar to a Cramer-Shoup ciphertext, allowing rerandomization as in the example above. However, instead of pairs we require 5-tuples of g_i, b_i, c_i, d_i (i.e., for $i = 1, \dots, 5$). 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 strands from different encryptions by correlating the two strands with shared random masks.

Our security analysis is more complicated than the ones in [3, 9, 18]. However all these analyses as well as the current one follow the basic idea that if an encryption were to be carried out using

[1]It is likely that our security analysis can be extended to groups of orders with large prime factors, as is done in [10]. For simplicity, we do not consider this here.

the secret key in a "bad" way, the result will remain indistinguishable from an actual encryption (by the DDH assumption), but will also become statistically independent of the message and the public key.

4 Our Construction

In this section we describe our main construction, the *Double-strand Cramer-Shoup* (DSCS) encryption scheme. First, we introduce a simpler encryption scheme that is used as a component of the main scheme.

4.1 Double-strand Malleable Encryption Scheme

We now 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 actually 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 $(\mathbb{Z}_q)^3$. 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 plaintext:** 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 MEnc is the choice of v and w in $\widehat{\mathbb{G}}$. We can rerandomize both of these quantities by choosing random $s, t \in \mathbb{Z}_q$ and outputting the following ciphertext:

$$U' = (\mathbf{V}\mathbf{W}^s, A_V \cdot A_W^s, \mathbf{W}^t, A_W^t).$$

Here $\mathbf{V}\mathbf{W}^s$ and $\mathbf{W}^t$ denote component-wise operations. It is not hard to see that if U is in the range of $\mathsf{MEnc}_{MPK}(u)$ (with random choices v and w), then U' is in the range of $\mathsf{MEnc}_{MPK}(u)$ with corresponding random choices $v' = v + sw$ and $w' = tw$.

Homomorphic operation (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 $r \in \mathbb{Z}_q$ and an encryption of u, one can derive an encryption of u^r. As it turns out, the way we use DSME in the main construction ensures that we achieve our final security despite such additional malleabilities.

4.2 Double-strand Cramer-Shoup Encryption Scheme

Now we give our main construction: a rerandomizable, RCCA-secure encryption scheme called the "Double-strand Cramer-Shoup" (DSCS) 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 the Cramer-Shoup scheme, we need to allow rerandomization, and so we do not use a prefix of the ciphertext itself in ensuring consistency; instead we use a direct encoding of the plaintext.

Further, we must prevent the possibility of mixing together strands from two *different* encryptions of the same message (say, in the manner in which rerandomizability allows two strands to be mixed together) to obtain a ciphertext which successfully decrypts, which would yield a successful adversarial strategy in our security experiments. For this, we correlate the two strands of a 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).

Finally, we must restrict the ways in which a ciphertext's two strands can be recombined, so that essentially the only way in which the two strands can be used to generate a ciphertext that decrypts successfully is to combine the two strands in the manner prescribed in the Rerand algorithm. To accomplish this, we perturb the exponents of the message-carrying strand by an additional (fixed) vector. Intuitively, this additive perturbance must remain present in the message-carrying strand of a ciphertext, which restricts the ways in which that strand can be combined with things. As a side-effect, our construction requires longer strands (i.e., more components) than in the original Cramer-Shoup scheme.

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 relationship 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 $\widehat{\mathbb{G}}$. 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 $(\mathbb{Z}_p)^5$. 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 following 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's and z_i's:** 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 plaintext:** 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 Enc 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 ciphertext which corresponds to an encryption of the same message, with corresponding random choices $u_i' = u_i r_i$, $x' = x + ys$, and $y' = yt$:

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

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

4.3 Complexity

The complexities of the DSCS scheme are summarized in Table 1 and Table 2.[2] Clearly our scheme

Table 1: Number of elements

	$\widehat{\mathbb{G}}$	$\mathbb{Z}_q$	$\mathbb{G}$	$\mathbb{Z}_p$
Public key	20	-	8	-
Private key	-	15	-	15
Ciphertext	40	-	14	-

Table 2: Group operations performed

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

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 [18], which used $O(k)$ group elements to encode a k-bit message (or in other words, to be able to use the group itself as the message space, it used $O(\log p)$ group elements). In fact, if we restrict ourselves to weak RCCA security (as defined in [18]) 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 is a significantly harder problem by our current state of knowledge. Despite the inefficiency, we believe that by providing the first complete solution (i.e., not in the generic group model) we have not only solved the problem from a theoretical perspective, but also opened up the possibility of efficient and practical constructions.

5 Replayable Message Posting

We define the "Replayable Message Posting" functionality $\mathcal{F}_{\text{RMP}}$ in the Universally Composable (UC) security framework [6, 22], also variously known as environmental security [15, 24] and network-aware security [23] framework.

This functionality concisely presents the security achieved by a rerandomizable, RCCA-secure encryption scheme. The functionality allows parties to publicly post messages which are represented by abstract *handles*, arbitrary strings provided by the adversary. The adversary is not told the actual message (unless, of course, the recipient is corrupted by the adversary). Only the designated receiver is allowed to obtain the corresponding message from the functionality.

Additionally, $\mathcal{F}_{\text{RMP}}$ provides a reposting functionality: any party can "repost" (i.e., make a copy of) any existing handle. Requesting a repost does not reveal the message. To the other parties (including the adversary and the original message's recipient), the repost appears exactly like a normal message posting; i.e, the functionality's external behavior is no different for a repost versus a normal post.

A similar functionality $\mathcal{F}_{\text{RPKE}}$ was defined by Canetti et. al [7] to capture (not necessarily rerandomizable) RCCA security. $\mathcal{F}_{\text{RPKE}}$ is itself a modification of the $\mathcal{F}_{\text{PKE}}$ functionality of [6], which modeled CCA security. Both of these functionalities similarly represent messages via abstract handles. However, the most important distinction between these two functionalities is that

[2]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.

$\mathcal{F}_{\text{RMP}}$ provides the ability to repost handles as a *feature*; thus, it does not include the notion of "decrypting" handles which are not previously known to the functionality.

We now formally define the behavior of $\mathcal{F}_{\text{RMP}}$. It accepts the following four kinds of requests from parties:

Registration: On receiving a message $\mathsf{register}$ from a party sender, the functionality sends $(\textsc{id-req}, \mathsf{sender})$ to the adversary, and expects in response an identifier string id.[3] 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 $(\textsc{id-announce}, \mathsf{id})$ to all other parties.

Additionally, we reserve a special identifier $\mathsf{id}_\perp$ for the adversary. The adversary need not explicitly register to use this identifier, nor is it announced to the other parties. We also insist that only corrupted parties are allowed to post messages for $\mathsf{id}_\perp$ (though honest parties may repost the resulting handles).[4]

Message posting: On receiving a request $(\mathsf{post}, \mathsf{id}, \mathsf{msg})$ from a party sender, the functionality behaves as follows:[5] If id is not registered, ignore the request.

If id is registered to an uncorrupted party, send $(\textsc{handle-req}, \mathsf{sender}, \mathsf{id})$ to the adversary; otherwise send $(\textsc{handle-req}, \mathsf{sender}, \mathsf{id}, \mathsf{msg})$ to the adversary. In both cases, expect a string handle in return. If handle has been previously used, ignore this request. Otherwise, record $(\mathsf{handle}, \mathsf{sender}, \mathsf{id}, \mathsf{msg})$ internally and publish $(\textsc{handle-announce}, \mathsf{handle}, \mathsf{id})$ to all registered parties.

Note that if the recipient of a message is corrupted, it is reasonable for the functionality to reveal msg to the adversary when requesting the handle.

Message reposting: 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.

Otherwise, suppose $(\mathsf{handle}, \mathsf{sender}', \mathsf{id}, \mathsf{msg})$ is recorded internally. If id is registered to an uncorrupted party, send $(\textsc{handle-req}, \mathsf{sender}, \mathsf{id})$ to the adversary; otherwise send $(\textsc{handle-req}, \mathsf{sender}, \mathsf{id}, \mathsf{msg})$ to the adversary. In both cases, expect a string handle' in return. If handle' has been previously used, ignore this request. Otherwise, record $(\mathsf{handle}', \mathsf{sender}, \mathsf{id}, \mathsf{msg})$ internally and publish $(\textsc{handle-announce}, \mathsf{handle}', \mathsf{id})$ to all registered parties.

As above, if the message's recipient is corrupted, the functionality can legitimately reveal msg to the adversary when requesting the handle.

Message reading: On receiving a message $(\mathsf{get}, \mathsf{handle})$ from a party, if a record $(\mathsf{handle}, \mathsf{sender}, \mathsf{id}, \mathsf{msg})$ is recorded internally, and id is registered to this party, then return $(\mathsf{id}, \mathsf{msg})$ to it. Otherwise ignore this request.

[3] 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.

[4] $\mathsf{id}_\perp$ models the fact that an adversary may generate key pairs without announcing them, and broadcast encryptions under those keys.

[5] We assume that msg is from a predetermined message space, with size superpolynomial in the security parameter; otherwise the request is ignored.

6 Results

We present two main results below. The first is that the DSCS encryption scheme presented in Section 4 achieves the security definitions defined in Section 2.1. The second result is that any construction which meets these guarantees is a secure realization of the $\mathcal{F}_{\text{RMP}}$ functionality defined in Section 5. The complete proofs appear in Section 7 and Section 8.

Theorem 1 *The DSCS scheme (Section 4) is a perfectly rerandomizable encryption scheme which satisfies the definitions of RCCA security and tight decryption under the DDH assumption in $\mathbb{G}$ and $\widehat{\mathbb{G}}$.*

Proof overview: Here we sketch an outline of the proof of RCCA security.

It is convenient to formulate our proof in terms of alternate encryption and decryption procedures. We remark that this outline is similar to that used in previous proofs related to the Cramer-Shoup construction [3, 9, 10, 13]. However, the implementation is significantly more involved in our case.

Alternate encryption. 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. AltEnc actually uses the *private key* to generate ciphertexts. In short, instead of using $\{X_i = g_i^x\}$ and $\{Y_i = g_i^y\}$, AltEnc picks random group elements for these ciphertext components, then uses the private key to generate the other components according to the quantities which are computed by Dec.

When AltEnc is used to generate the challenge ciphertext in the RCCA security experiment, it follows from the DDH assumption in $\mathbb{G}$ and $\widehat{\mathbb{G}}$ that for any adversary the experiment's outcome does not change significantly. Additionally, the ciphertexts produced by AltEnc are information-theoretically independent of the message.

Alternate decryption. An adversary may be able to get information about the message 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 the oracle answers maliciously crafted ciphertexts. To show that our scheme does leak information in this way, 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 (quantities which are already known to the adversary). AltDec will be computationally unbounded, but since it is accessed as an oracle, this does not affect the analysis. More importantly, its functionality is statistically indistinguishable from the honest decryption procedure (even when the adversary is given a ciphertext generated by AltEnc).

By computing discrete logarithms of some components of its input and comparing with the public key and challenge ciphertext, the alternate decryption procedure can check whether its input is "looks like" an honest encryption or a rerandomization of the challenge ciphertext, and give the correct response in these cases. To establish the correctness of this approach, we show that ciphertexts which are rejected by AltDec would be rejected by the normal decryption algorithm with overwhelming probability as well. The $\mathbf{u}$ and $\mathbf{z}$ components of our construction are vital in preventing all other ways of combining the challenge strands and the public key. This is the most delicate part of our proof.

We conclude that with these two modifications – alternate challenge ciphertext and the alternate decryption procedure – the adversary's view in the RCCA security experiment is independent of the secret bit b, and so the adversary's advantage is zero. Furthermore, the outcome of this modified experiment is only negligibly different from the outcome of the original experiment, so the security claim follow. $\lhd$

Theorem 2 *Every rerandomizable encryption scheme which is RCCA-secure, and has tight decryption*[6] *is a secure realization of $\mathcal{F}_{\text{RMP}}$ in the standard UC model.*

PROOF OVERVIEW: For simplicity we consider all communications to use a broadcast channel. The scheme yields a protocol for $\mathcal{F}_{\text{RMP}}$ in the following natural way: public keys correspond to identifiers and ciphertexts correspond to handles. To register oneself, one generates a key pair and broadcasts the public key. To post a message to a party, one simply encrypts the message under his public key. To repost a handle, one simply applies the rerandomization procedure. To retrieve a message in a handle, one simply decrypts it using one's private key.

To prove the security of this protocol, we demonstrate a simulator $\mathcal{S}$ for each adversary $\mathcal{A}$. The simulator $\mathcal{S}$ internally runs $\mathcal{A}$ and behaves as follows:

When $\mathcal{F}_{\text{RMP}}$ receives a registration request from an honest party, it requests an identifier from $\mathcal{S}$. $\mathcal{S}$ generates a key pair, sends the public key as the identifier string, and simulates to $\mathcal{A}$ that an honest party broadcasted this public key.

When $\mathcal{F}_{\text{RMP}}$ receives a post request addressed to an honest party (or a repost request for such a handle), it requests a new handle from $\mathcal{S}$, without revealing the message. The ith time this happens, $\mathcal{S}$ generates the handle handle_i^H by picking a random message msg_i^H and encrypting it under the given identity (say, public key PK_i). In its simulation, $\mathcal{S}$ uses this ciphertext as the one broadcast by the sender. The correctness of this simulated ciphertext follows from the scheme's RCCA security property.

When $\mathcal{A}$ broadcasts a public key, $\mathcal{S}$ registers it as an identifier in $\mathcal{F}_{\text{RMP}}$. When $\mathcal{A}$ broadcasts a ciphertext ζ, $\mathcal{S}$ behaves as follows:

1. If for some i, $\mathsf{Dec}_{SK_i}(\zeta) = \mathsf{msg}_i^H$ (the ith random message chosen to simulate a ciphertext between honest parties), then $\mathcal{S}$ instructs $\mathcal{F}_{\text{RMP}}$ to repost handle_i^H.

2. If $\mathsf{Dec}_{SK}(\zeta) = \mathsf{msg} \neq \perp$ for any of the private keys SK that it picked while registering honest parties, then $\mathcal{S}$ instructs $\mathcal{F}_{\text{RMP}}$ to post msg, addressed to the corresponding honest party.

3. Otherwise, ζ does not successfully decrypt under any of these private keys. The jth time this happens, $\mathcal{S}$ picks a random message msg_j^A and instructs $\mathcal{F}_{\text{RMP}}$ to post msg_j^A to $\mathsf{id}_\perp$. It also remembers $\mathsf{handle}_j^A = \zeta$.

In all the above cases, $\mathcal{S}$ sends ζ to $\mathcal{F}_{\text{RMP}}$ as the handle for this new message. Tight decryption ensures that at most one of the above decryptions succeeds. Further, if one does succeed, the

[6]By relaxing the requirement that the scheme have tight decryption (and also the correctness requirement that ciphertexts are not honest ciphertexts for more than one honest key pair), we can still realize a weaker variant of $\mathcal{F}_{\text{RMP}}$. In this variant, handles may be re-used, and the adversary is notified any time an honest party reposts any handle which the adversary posted/reposted. We omit the details here.

ciphertext must be in the support of honest encryptions, so the perfect rerandomization condition holds for it.

When $\mathcal{F}_{\text{RMP}}$ receives a post request addressed to a corrupted party (or a repost request for such a handle), it sends the corresponding message msg and identifier id to $\mathcal{S}$, and requests a new handle.

1. If $\mathsf{id} = \mathsf{id}_\perp$ and $\mathsf{msg} = \mathsf{msg}_j^A$ (the jth random message chosen to simulate an adversarial ciphertext to $\mathcal{F}_{\text{RMP}}$), then $\mathcal{S}$ generates a handle by rerandomizing the corresponding handle_j^A.
2. Otherwise, $\mathcal{S}$ generates a handle by encrypting the message under the appropriate public key.

In its simulation, $\mathcal{S}$ uses this ciphertext as the one broadcast by the sender. $\lhd$

7 Proof of Theorem 1

We first show that the DSCS scheme satisfies the correctness requirements of a perfectly rerandomizable encryption scheme. Then, as mentioned in Section 6, we focus on the proofs of RCCA security. To do this, we will demonstrate alternate encryption and decryption procedures. The alternate encryption procedure AltEnc is described in Section 7.5 and the alternate decryption procedure AltDec is described in Section 7.6. The lemmas in the rest of this section carry out the proof outlined in Section 6. Finally, we observe that the tight decryption property of our construction is a direct consequence of Lemma 6.

7.1 Correctness Properties

The correctness properties of decryption are 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 $\mathsf{MDec}_{MSK}(U') = \mathsf{MDec}_{MSK}(U)$.*

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 MRerand is $s, t \in \mathbb{Z}_q$. Then, substituting according to the two above constraints and the computations in 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. $\square$

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). $\square$

7.2 Replay Interactions

Instead of arguing about the RCCA security experiment, we define a generic "experiment" called a *replay interaction.*

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 plaintext message given the public key (conditioned on a negligible-probability event not happening).*

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 given Claim 1 and Claim 2. First, we show that the RCCA security experiment can be implemented by playing as the adversary in a replay interaction.

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.

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 the RCCA security experiment in terms of alternate replay interactions, the adversary has no advantage in guessing b. In such an implementation, the adversary's view includes only the following quantities:

- The public key, clearly distributed independently of b.

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

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

7.3 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.

7.4 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.

7.5 The Alternate Encryption Procedure

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

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

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

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

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, 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 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 message. $\square$

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 7.3 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 $\mathsf{AltMEnc}$ 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 $\mathsf{DS\text{-}DDH}$ 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 $\mathsf{DS\text{-}Rand}$ distribution), then the ciphertext is distributed as an encryption with AltEnc (and $\mathsf{AltMEnc}$).

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. $\square$

7.6 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 $\mathsf{AltMEnc}$ 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 $\mathsf{AltMDec}^{U^*}_{MPK}(U)$ 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:

$$\mathsf{MDec}_{MSK}(U) = \frac{A_V}{\prod_{j=1}^3 V_j^{a_j}} = \frac{A_V}{\left(\prod_j (V^*_j)^{a_j}\right)^\pi \left(\prod_j (W^*_j)^{a_j}\right)^\beta \left(\prod_j \widehat{g}_j^{a_j}\right)^\gamma} = \frac{A_V}{\left(\frac{A^*_V}{\mathsf{MDec}_{MSK}(U^*)}\right)^\pi (A^*_W)^\beta A^\gamma}$$
$$= \left[\frac{A_V}{(A^*_V)^\pi (A^*_W)^\beta A^\gamma}\right] \mathsf{MDec}_{MSK}(U^*)^\pi$$

□

We now describe the DSCS alternate decryption procedure AltDec.

DSCS alternate decryption ($\mathsf{AltDec}^{\zeta^*}_{PK}(\zeta)$). 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 $\mathsf{AltMDec}^{U^*_i}_{MPK_i}(U_i)$, 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 $\mu = B_X / B^x$, $\mathsf{msg} = \mathsf{encode}^{-1}_{\mathbb{G}}(\mu)$, and $m = \mathsf{encode}_{\mathbb{Z}_p}(\mathsf{msg})$. We perform the following checks:
  $$P_X \overset{?}{=} (CD^m)^x; \qquad B_Y \overset{?}{=} B^y; \qquad P_Y \overset{?}{=} (CD^m)^y$$
  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 lemmas establish the correctness of some of the cases where AltDec outputs $\perp$:

Lemma 6 *In Phase I of a replay interaction, call a query ζ to the decryption oracle "bad" if one of the following holds:*

1. *One of its component DSME ciphertexts has a strand that is linearly independent of* $\mathbf{1}$*; or,*
2. *All of its component DSME ciphertexts decrypt successfully, but one of its unmasked DSCS strands (with respect to these values) is linearly independent of* $\mathbf{1}$*.*

Then with overwhelming probability over the remaining randomness in KeyGen*,* Dec *would return* $\perp$ *for all such "bad" queries.*

Proof: Consider the first such "bad" ciphertext submitted by the adversary. We seperately consider the two cases above:

1. Suppose one of the component DSME ciphertexts U_i has a strand that is linearly independent of $\mathbf{1}$. Let A and $\mathbf{a} = (a_1, a_2, a_3)$ be the corresponding DSME public and private key, respectively. In Phase I so far, the part of the adversary's view that depends on this keypair is:

$$\begin{bmatrix}1 & 1 & 1\end{bmatrix}\begin{bmatrix}\log\widehat{g}_1 & 0 & 0\\ 0 & \log\widehat{g}_2 & 0\\ 0 & 0 & \log\widehat{g}_3\end{bmatrix}\begin{bmatrix}a_1\\ a_2\\ a_3\end{bmatrix} = \begin{bmatrix}\log A\end{bmatrix}$$

Let $\mathbf{v}, \mathbf{w}$ be the two strands of the DSME ciphertext U_i. It successfully decrypts to u_i if and only if:

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

If the second strand is independent of $\mathbf{1}$, then the "correct" value of A_W is distributed uniformly in $\widehat{\mathbb{G}}$, and independently of the adversary's view. Thus, this ciphertext can decrypt successfully with probability at most $1/q$.

Likewise, if the first strand is independent of **1**, then if the ciphertext decrypts at all, it decrypts to a value u_i that is distributed uniformly and independently of the adversary's view. As a result, with respect to this u_i, the unmasked DSCS strands will be independent of **1** with overwhelming probability. Thus, according to the following argument, Dec will return $\perp$ with overwhelming probability.

2. Suppose an unmasked DSCS strand of ζ is linearly independent of **1**. By a linear algebra argument analagous to the one above, the "correct" value of the P_X or P_Y component of ζ is distributed uniformly in $\mathbb{G}$, and independently of the adversary's view. Thus, the ciphertext can decrypt successfully with probabilitiy at most $1/p$.

When a ciphertext is rejected by Dec, the adversary learns that some consistency check failed. This (information-theoretically) leaks some information about the space of possible private keys. However, at most a $1/q$ or $1/p$ fraction of consistent private keys are excluded by learning that some constraint did not hold, depending on which group the constraint was in. By a union bound, if the adversary makes N such bad queries, one of them will be accepted with probability at most $N/(q-N)$. Since N is polynomial in the security parameter, this quantity remains negligible. □

Tightness of decryption. Observe that the experiment in the definition of tight decryption (Section 2.1) is essentially Phase I of a replay interaction. A ciphertext ζ is "honestly generated" if its strands are all scalar multiples of **1** (and its consistency-check components are all correct). Clearly a ciphertext whose strands are all multiples of **1**, but whose consistency-check components are incorrect will be unconditionally rejected by Dec. Additionally, by Lemma 6, all other non-honestly-generated ciphertexts will be rejected with overwhelming probability. This establishes the decryption tightness property for our construction.

Lemma 7 *In Phase II of a replay interaction, where the challenge ciphertext ζ^* was generated by* AltEnc*, call a query ζ to the decryption oracle "bad" if one of the following holds:*

1. *One of its component DSME ciphertexts U_i has a second strand that is linearly dependent on the first strand of the corresponding U_i^*;*[7] *or,*

2. *All of its component DSME ciphertexts decrypt successfully, but one of its unmasked DSCS strands (with respect to these values) is linearly independent of* **1** *and the unmasked strands of ζ^*.*

Then with overwhelming probability over the remaining randomness in KeyGen*,* Dec *would return $\perp$ for all such "bad" queries.*

PROOF: As above, consider the first such "bad" ciphertext submitted by the adversary. In Phase II, the view of the adversary now includes the strands contained in ζ^*.

- Suppose a component DSME ciphertext U_i has a second strand that is linearly dependent on the first strand of the corresponding U_i^*. Recall that by Lemma 3, the decrypted value of U_i^* is distributed independently of the adversary's view. Thus the value $A_V^*/\mathsf{MDec}(U_i^*) =$

[7]Note that the strands of U_i^* along with **1** span the space of possible strands with overwhelming probability, so we do not consider the case where the strands of U_i are independent of these strands.

$\prod_{i=1}^{3}(V_i^*)^{a_i}$ is also distributed independently. As the first strand of U_i depends linearly on the first strand of U_i^*, the decryption procedure will computed the purported plaintext u_i using a nontrivial factor of $\prod_{i=1}^{3}(V_i^*)^{a_i}$. Thus, as above, the decryption of U_i is distributed independently of the adversary's view, and the unmasked DSCS strand is independent of the adversary's view with overwhelming probability.

- Suppose an unmasked DSCS strand of ζ is linearly independent of the known strands ($\mathbf{1}$, and the two unmasked strands of ζ^*). Then as above, the "correct" value of the P_X or P_Y component of ζ is distributed independently of the adversary's view. Thus, the ciphertext can decrypt successfully with probabilitiy at most $1/p$.

Similar to above, a union bound shows that all such bad ciphertexts are rejected with overwhelming probability. □

The previous two lemmas establish the validity of AltDec rejecting its input when its DSCS strands are linearly independent of the adversary's view. However, it also rejects its input if the DSCS strands are linearly dependent but not of the form given by the Rerand procedure. The following lemma shows that this case happens only negligibly often.

Lemma 8 *Let $\overline{\mathbf{x}}^*, \overline{\mathbf{y}}^*$ be the unmasked strands of the challenge ciphertext ζ^*, which was generated by* AltEnc. *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:*

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

$$\overline{\mathbf{x}} = \alpha\overline{\mathbf{x}}^* + \beta\overline{\mathbf{y}}^* + \gamma\mathbf{1}$$

$$\iff \left(\frac{x_1}{u_1} - z_1, \dots, \frac{x_5}{u_5} - z_5\right) = \alpha\left(\frac{x_1^*}{u_1^*} - z_1, \dots, \frac{x_5^*}{u_5^*} - z_5\right) + \beta\left(\frac{y_1^*}{u_1^*}, \dots, \frac{y_5^*}{u_5^*}\right) + \gamma\mathbf{1}$$

$$\iff \left(\frac{x_1}{\delta_1(u_1^*)^{\pi_1}}, \dots, \frac{x_5}{\delta_5(u_5^*)^{\pi_5}}\right) = \alpha\left[\left(\frac{x_1^*}{u_1^*}, \dots, \frac{x_5^*}{u_5^*}\right) - \mathbf{z}\right] + \beta\left(\frac{y_1^*}{u_1^*}, \dots, \frac{y_5^*}{u_5^*}\right) + \gamma\mathbf{1} + \mathbf{z} \tag{3}$$

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

$$\begin{bmatrix} x_1/\delta_1 \\ \vdots \\ x_5/\delta_5 \end{bmatrix} = \begin{bmatrix} (u_1^*)^{\pi_1-1} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & (u_5^*)^{\pi_5-1} \end{bmatrix} \begin{bmatrix} x_1^* - z_1u_1^* & y_1^* & u_1^* \\ \vdots & \vdots & \vdots \\ x_5^* - z_5u_5^* & y_5^* & u_5^* \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} + \begin{bmatrix} z_1(u_1^*)^{\pi_1} \\ \vdots \\ z_1(u_5^*)^{\pi_5} \end{bmatrix}$$

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:

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

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 ($\forall i : x_i = 0$ 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 ($\forall i : x_i = 0$ 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 ($\forall i : x_i = 0$ 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 9 AltDec *faithfully implements the decryption oracle in the replay interaction described in Section 7.2, 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.

If any of the cases given in Lemma 6 and Lemma 7 hold, AltDec rejects its input (either directly, or by AltMDec rejecting). As proven in these lemmas, this is what Dec would do as well, with overwhelming probability.

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 AltMDec(U_i) returns (δ_i, π_i), then by Lemma 5, these values are such that MDec$(U_i) = \delta_i$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).

AltDec also rejects its input if its unmasked DSCS strands are linearly dependent on the adversary's view, but not of the form of an honest ciphertext or a rerandomization of the challenge ciphertext. By Lemma 8, this event happens only with negligible probability.

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

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

are linearly independent of the constraints in Equation 2, because $m' \neq m^*$. By the same logic as in the proofs of Lemma 6 and Lemma 7, 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. □

8 Proof of Theorem 2

Let $\Pi = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rerand})$ be a rerandomizable RCCA-secure encryption scheme with tight decryption. To prove Theorem 2, for any real-world adversary $\mathcal{A}$, we must demonstrate an ideal-world adversary (simulator) $\mathcal{S}$, so that for all PPT environments $\mathcal{Z}$, $\textsc{real}^{\Pi}_{\mathcal{A},\mathcal{Z}} \approx \textsc{ideal}^{\mathcal{F}_{\text{RMP}}}_{\mathcal{S},\mathcal{Z}}$.

We build $\mathcal{S}$ in stages, starting from the real-world interactions 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 in its HANDLE-REQ interactions with the adversary, it reveals the message, recipient, and whether the handle is being requested for a repost. Thus $\mathcal{S}_0$ effectively learns all the honest parties' inputs to $\mathcal{F}_0$. $\mathcal{S}_0$ 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 an honest party sends a $\mathsf{register}$ command to $\mathcal{F}_0$, the functionality sends $(\textsc{id-req}, \mathsf{sender})$ to $\mathcal{S}_0$ and expects an identity. $\mathcal{S}_0$ generates a key pair $(PK_{\mathsf{id}}, SK_{\mathsf{id}}) \leftarrow \mathsf{KeyGen}$ and uses PK_{id} as the identity string. It also internally simulates to $\mathcal{A}$ that sender broadcast the public key.

2. When an honest party sender sends a command $(\mathsf{post}, \mathsf{id}, \mathsf{msg})$ to $\mathcal{F}_0$, the functionality sends $(\textsc{handle-req}, \mathsf{sender}, \mathsf{id}, \mathsf{msg})$ to $\mathcal{S}_0$ and expects a handle. $\mathcal{S}_0$ computes $\mathsf{handle} \leftarrow \mathsf{Enc}_{PK_{\mathsf{id}}}(\mathsf{msg})$ and uses it as the handle. It also internally simulates to $\mathcal{A}$ that sender broadcast handle.

3. When an honest party sender sends a command $(\mathsf{repost}, \mathsf{handle})$ to $\mathcal{F}_0$, and handle is internally recorded, the functionality sends $(\textsc{handle-req}, \mathsf{sender}, \mathsf{repost}, \mathsf{handle})$ to $\mathcal{S}_0$ and expects a handle. $\mathcal{S}_0$ computes $\mathsf{handle}' \leftarrow \mathsf{Rerand}(\mathsf{handle})$ and uses it as the handle. It also internally simulates to $\mathcal{A}$ that sender broadcast handle'.

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

   - For each honest party's private key SK_{id}, $\mathcal{S}_0$ checks if $\mathsf{Dec}_{SK_{\mathsf{id}}}(\zeta) = \mathsf{msg} \neq \perp$. If so, then $\mathcal{S}_0$ sends $(\mathsf{post}, \mathsf{id}, \mathsf{msg})$ to the functionality on behalf of $\mathcal{A}$. It sends ζ as the corresponding handle.
   - If none of the above decryptions succeed, then $\mathcal{S}_0$ picks a random message; say, msg^A_j the jth time this happens. It sends $(\mathsf{post}, \mathsf{id}_\perp, \mathsf{msg}^A_j)$ to the functionality and sends $\mathsf{handle}^A_j = \zeta$ as the corresponding handle.

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 $\textsc{ideal}^{\mathcal{F}_0}_{\mathcal{S}_0,\mathcal{Z}}$.

Claim 3 *For any given PPT adversary, let $\mathcal{F}_0$ and $\mathcal{S}_0$ be as described above. Then for all PPT environments $\mathcal{Z}$, $\textsc{real}^{\Pi}_{\mathcal{A},\mathcal{Z}} \approx \textsc{ideal}^{\mathcal{F}_0}_{\mathcal{S}_0,\mathcal{Z}}$.*

Proof: This follows from the correctness properties of encryption scheme Π. Note that $\mathcal{S}_0$ exactly emulates the real world actions of the honest parties and $\mathcal{A}$, the only exception being the negligible-probability event that there is a collision among the msg_j^A's. We also observe that by the tight decryption property of the scheme, at most one of the decryptions in step 4 succeeds. This ensures that the message sent to the functionality has a unique recipient. $\square$

$\mathcal{S}_1$ and $\mathcal{F}_1$ (Rerandomizability): $\mathcal{F}_1$ is identical to $\mathcal{F}_0$ except that it does not tell the adversary whether a HANDLE-REQ was the result of a post or repost command. The corresponding simulator $\mathcal{S}_1$ differs from $\mathcal{S}_0$ only in its servicing of handle reqeusts. The exact differences are as follows:

1. When an honest party sender sends a command $(\mathsf{repost}, \mathsf{handle})$ to $\mathcal{F}_1$, and $(\mathsf{handle}, \mathsf{sender}', \mathsf{id}, \mathsf{msg})$ is internally recorded, the functionality now sends $(\text{HANDLE-REQ}, \mathsf{sender}, \mathsf{id}, \mathsf{msg})$ to $\mathcal{S}_1$ and expects a handle (note that this is precisely what is sent when sender sends a $(\mathsf{post}, \mathsf{id}, \mathsf{msg})$ command).

2. When $\mathcal{S}_1$ receives a $(\text{HANDLE-REQ}, \mathsf{sender}, \mathsf{id}, \mathsf{msg})$ request corresponding to a post/repost request from an honest party, it first checks whether $\mathsf{id} = \mathsf{id}_\perp$ and $\mathsf{msg} = \mathsf{msg}_j^A$ for some j. If so, it generates the handle via $\mathsf{handle}' \leftarrow \mathsf{Rerand}(\mathsf{handle}_j^A)$. Otherwise, it generates the handle via $\mathsf{handle}' \leftarrow \mathsf{Enc}_{PK_{\mathsf{id}}}(\mathsf{msg})$ (as $\mathcal{S}_0$ does).

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

Proof: This follows from the perfect rerandomizability of the replayable encryption scheme. The only manner in which the two executions differ is in whether a ciphertext is generated via rerandomization (as $\mathcal{S}_0$ does on receiving a $(\text{HANDLE-REQ}, \mathsf{sender}, \mathsf{repost}, \mathsf{handle})$ command) or as a fresh encryption of the same message under the same key (as $\mathcal{S}_1$ does on receiving a $(\text{HANDLE-REQ}, \mathsf{sender}, \mathsf{id}, \mathsf{msg})$ command). We note that $\mathcal{S}_1$ only behaves differently when $\mathsf{id} \neq \mathsf{id}_\perp$, which is the identity used for adversarial ciphertexts that do not decrypt under any honest party's private key. Thus the handle being rerandomized was either honestly generated by the simulator, or it successfully decrypted under an honest party's private key. By the tight decryption property of the scheme, such a ciphertext is in the support of honestly generated encryptions and the perfect rerandomization property holds. Thus $\mathcal{S}_1$ generates ciphertexts according to the same distribution as $\mathcal{S}_0$, and we have $\text{IDEAL}^{\mathcal{F}_0}_{\mathcal{S}_0,\mathcal{Z}} = \text{IDEAL}^{\mathcal{F}_1}_{\mathcal{S}_1,\mathcal{Z}}$. $\square$

$\mathcal{S}_2$ and $\mathcal{F}_2$ (RCCA security): $\mathcal{F}_2$ is identical to $\mathcal{F}_1$ except that it does not tell the adversary the contents of the posted messages when the receiver is not corrupted. $\mathcal{S}_2$ differs from $\mathcal{S}_1$ accordingly. The exact differences are as follows:

1. Whenever $\mathcal{F}_1$ would send a send $(\text{HANDLE-REQ}, \mathsf{sender}, \mathsf{id}, \mathsf{msg})$ to the simulator (i.e, when a party posts or reposts a message), $\mathcal{F}_2$ first checks if id is registered to a corrupt party. If so, it continues as $\mathcal{F}_1$; otherwise, it sends $(\text{HANDLE-REQ}, \mathsf{sender}, \mathsf{id})$ instead (i.e, it does not include msg).

2. When $\mathcal{S}_2$ receives the ith request of the form $(\text{HANDLE-REQ}, \mathsf{sender}, \mathsf{id})$ from $\mathcal{F}_2$, it picks a random message msg_i^H and uses $\mathsf{handle}_i^H \leftarrow \mathsf{Enc}_{PK_{\mathsf{id}}}(\mathsf{msg}_i^H)$ as the handle. It internally keeps track of $(\mathsf{handle}_i^H, \mathsf{msg}_i^H, SK_{\mathsf{id}})$ for later use.

3. When the adversary broadcasts a ciphertext ζ, $\mathcal{S}_2$ does the following: For each $(\mathsf{handle}_i^H, \mathsf{msg}_i^H, SK_{\mathsf{id}})$ recorded above, $\mathcal{S}_2$ checks if $\mathsf{Dec}_{SK_{\mathsf{id}}}(\zeta) = \mathsf{msg}_i^H$. If so, $\mathcal{S}_2$ sends $(\mathsf{repost}, \mathsf{handle}_i^H)$ to $\mathcal{F}_2$ and uses ζ as the handle. If none of these decryptions succeed, then $\mathcal{S}_2$ proceeds just as $\mathcal{S}_1$.

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

Proof: This follows from the RCCA security of the 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).

To apply the RCCA security guarantee, we must reduce to to the case of a single encryption, using a series of hybrid experiments.

We define a sequence of hybrid simulators $\hat{\mathcal{S}}_1^{k,i}$ which interact with $\mathcal{F}_1$ (and hence receive the message in handle requests from the functionality) to be exactly like $\mathcal{S}_1$, but with the following differences:

- When $\hat{\mathcal{S}}_1^{k,i}$ receives a request $(\text{HANDLE-REQ}, \mathsf{sender}, \mathsf{id}, \mathsf{msg})$, it does the following: First, it chooses a new random message msg_i^H. If id is among the first k' identities registered to an honest party, and this request is among the first i' handle requests for this identity for some $(k', i') \leq (k, i)$, then $\hat{\mathcal{S}}_1^{k,i}$ generates the handle by encrypting msg_i^H (as $\mathcal{S}_2$ would do); otherwise, it encrypts the given msg (as $\mathcal{S}_1$ would do). It internally records $(\mathsf{handle}, \mathsf{msg}_i^H, \mathsf{msg}, SK_{\mathsf{id}})$; note that it internally records both msg and msg_i^H, as opposed to $\mathcal{S}_2$, which stores only the plaintext used to generate handle.

- When the adversary broadcasts a ciphertext ζ, $\hat{\mathcal{S}}_1^{k,i}$ checks if $\mathsf{Dec}_{SK}(\zeta) \in \{\mathsf{msg}, \mathsf{msg}'\}$ for each $(\mathsf{handle}, \mathsf{msg}, \mathsf{msg}', SK)$ recorded above. If so, it sends $(\mathsf{repost}, \mathsf{handle})$ to the functionality, using ζ as the handle for this repost.

Let M be a polynomial bound on the number of identies registered to honest parties, and let N be a polynomial bound on the number of messages sent by honest parties to honest identities. The only difference between $\hat{\mathcal{S}}_1^{M,N}$ and $\mathcal{S}_2$ is that when the adversary outputs a ciphertext that decrypts to an actual msg that was previously sent between honest parties, $\hat{\mathcal{S}}_1^{M,N}$ will repost the corresponding handle, while $\mathcal{S}_2$ will post that message (as it was never told the actual msg). However, both commands put the functionality in an identical state, so $\hat{\mathcal{S}}_1^{M,N}$ and $\mathcal{S}_2$ behave identically.

We also have that $\hat{\mathcal{S}}_1^{k,N}$ behaves identically to $\hat{\mathcal{S}}_1^{k+1,0}$. However, $\hat{\mathcal{S}}_1^{0,0}$ and $\mathcal{S}_1$ differ slightly, in that if the adversary outputs a ciphertext ζ which decrypts to some msg_i^H under the appropriate private key, then $\hat{\mathcal{S}}_1^{0,0}$ replaces msg_i^H by some other msg. However this can happen only with negligible probability as the adversary's view is independent of msg_i^H in the interaction up to that point. So we do have $\text{IDEAL}^{\mathcal{F}_1}_{\mathcal{S}_1,\mathcal{Z}} \approx \text{IDEAL}^{\mathcal{F}_1}_{\hat{\mathcal{S}}_1^{0,0},\mathcal{Z}}$.

Finally, it suffices to show that $\text{IDEAL}^{\mathcal{F}_1}_{\hat{\mathcal{S}}_1^{k,i},\mathcal{Z}} \approx \text{IDEAL}^{\mathcal{F}_1}_{\hat{\mathcal{S}}_1^{k,i+1},\mathcal{Z}}$. This follows directly from the RCCA security of the scheme. The only difference between $\hat{\mathcal{S}}_1^{k,i}$ and $\hat{\mathcal{S}}_1^{k,i+1}$ is whether the actual message or a random message is encrypted. To implement the simulator in terms of the RCCA security experiment, it suffices to be told whenever subsequent ciphertexts decrypt to *either* of these two plaintexts. Thus, $\hat{\mathcal{S}}_1^{k,i}$ and $\hat{\mathcal{S}}_1^{k,i+1}$ can be seen as carrying out the RCCA experiment with $b = 0$ and $b = 1$ respectively. $\square$

Concluding the proof. Combining the above claims we get that for all adversaries $\mathcal{A}$, there exists a simulator $\mathcal{S}_2$ such that $\text{REAL}^{\Pi}_{\mathcal{A},\mathcal{Z}} \approx \text{IDEAL}^{\mathcal{F}_2}_{\mathcal{S}_2,\mathcal{Z}}$ for all environments $\mathcal{Z}$. Note that $\mathcal{F}_2$ is in fact identical to $\mathcal{F}_{\text{RMP}}$. So letting $\mathcal{S} = \mathcal{S}_2$ completes the proof.

9 Extensions

Once our construction is 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.

Replay-test. 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 an independent encryption of the same plaintext. We call such a feature a *replay-test* feature. A replay-test precludes having perfect or even statistical rerandomization, and so we must settle for a computational definition of rerandomization.

We point out that redefining RCCA security and rerandomizability for schemes which include a replay-test feature is a non-trivial extension of our current definitions. In particular, note that in a chosen ciphertext attack, the adversary should be allowed to access a replay-test oracle as well as a decryption oracle, while responses from the decryption oracle should be guarded based on the replay-test instead of a check of the plaintext.

However, instead of modifying our security definitions based on standalone experiments, we can directly formulate a new UC functionality. The functionality is identical to $\mathcal{F}_{\text{RMP}}$, but it provides an additional 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 reposts of the same original post. To do this, the functionality maintains some extra book-keeping internally. This functionality can be easily achieved starting from $\mathcal{F}_{\text{RMP}}$: each message is posted with a random nonce appended. To implement test, the receiver retrieves the messages of the two handles and compares their nonces.

Authentication. As should be intuitive, authentication 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 functionality's get command is augmented to provide not only the message in the handle, but also who originally posted the handle. The identifiers for receiving messages and sending messages are separate, but they can be tied together (by signing the encryption scheme's public key and publishing it), so that only the signature verification keys need to be considered as identifiers in the system.

Variable-length plaintexts. In our presentation of our encryption scheme, there is a hard limit on the message length, because the message must be encoded as an element in a group of fixed size. However, $\mathcal{F}_{\text{RMP}}$ can easily be extended to allow messages of variable lengths: for this the longer message is split into smaller pieces; a serial number and a common random nonce are appended to each piece; the first piece also carries the total number of pieces. Then each piece is posted using the fixed-length $\mathcal{F}_{\text{RMP}}$ functionality. The decryption procedure performs the obvious simple integrity checks on a set of ciphertexts and discards them if they are not all consistent and complete. Note that the resulting modification to the $\mathcal{F}_{\text{RMP}}$ functionality tells the adversary the length of the message (i.e., number of pieces) while posting or reposting a handle. It is straight-forward 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 to the functionality unless all handles it generated for one variable-length handle are reposted together). We note that these extensions can be applied one after the other.

9.1 Anonymity

Bellare et al. [3] introduced the notion of anonymity (or key-privacy) for encryption schemes. In a system with multiple users (including in particular possible applications of rerandomizable encryption in mix-nets), 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. We note that our scheme does not achieve this definition of anonymity, and leave it as an interesting open problem.

RCCA receiver-anonymity. Our receiver-anonymity definition is similar to that of Bellare et al [3], but modified for the RCCA paradigm. An encryption scheme is said to be *RCCA receiver-anonymous* (or simply *receiver-anonymous*) if the advantage of any PPT adversary $\mathcal{A}$ in the following experiment is negligible:

1. **Setup:** Pick $(PK_0, SK_0) \leftarrow \mathsf{KeyGen}$ and $(PK_1, SK_1) \leftarrow \mathsf{KeyGen}$. $\mathcal{A}$ is given (PK_0, PK_1)
2. **Phase I:** $\mathcal{A}$ gets access to the decryption oracles $\mathsf{Dec}_{SK_0}(\cdot)$ and $\mathsf{Dec}_{SK_1}(\cdot)$.
3. **Challenge:** $\mathcal{A}$ outputs a plaintext msg. Pick $b \leftarrow \{0,1\}$ and let $\zeta^* \leftarrow \mathsf{Enc}_{PK_b}(\mathsf{msg})$. $\mathcal{A}$ is given ζ^*.
4. **Phase II:** $\mathcal{A}$ gets access to a *guarded decryption oracle* $\mathsf{GDec}^{\mathsf{msg}}_{SK_0,SK_1}(\cdot)$, which on input ζ, first checks if $\mathsf{msg} \in \{\mathsf{Dec}_{SK_0}(\zeta), \mathsf{Dec}_{SK_1}(\zeta)\}$. If so, it returns replay; otherwise it returns the pair $(\mathsf{Dec}_{SK_0}(\zeta), \mathsf{Dec}_{SK_1}(\zeta))$.
5. **Guess:** $\mathcal{A}$ outputs a bit $b' \in \{0,1\}$. The *advantage* of $\mathcal{A}$ in this experiment is $\Pr[b' = b] - \frac{1}{2}$.

Modifications to $\mathcal{F}_{\text{RMP}}$. If a rerandomizable, RCCA-secure encryption scheme additionally meets this definition of RCCA anonymity, the scheme can be used to implement an "anonymous" variant of $\mathcal{F}_{\text{RMP}}$. In this variant, the functionality does not broadcast the handle's recipient in a HANDLE-ANNOUNCE announcement, nor in the HANDLE-REQ messages it sends to the adversary (unless the handle's recipient is corrupted).

The only change in the simulator for this modified functionality is that it uses a privately-held "dummy key" to generate simulated ciphertexts addressed to honest recipients.

10 Conclusions and Future Directions

This work leads to several interesting questions. First, can the efficiency of our scheme be improved? Public-key encryption schemes like Cramer-Shoup are much less efficient than 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, and then encrypt the actual voluminous data with the private-key encryption. It is interesting to ask if such a hybrid scheme can be devised in a rerandomizable manner. Consider using a stream cipher (pseudorandom generator) 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 PRG's seed 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.

Second, can CCA-like security definitions be defined for encryption schemes with more sophisticated homomorphic features (viewing rerandomization as homomorphism under the identity function)? Note that a homomorphism feature necessitates malleability, while CCA security demands the opposite. A meaningful definition that combines the two should allow the specific form of malleability needed for the desired homomorphism, but prohibit all other forms.

Finally, we based our schemes on the DDH assumption. However, as mentioned before, it is likely that the extensions of Cramer and Shoup [10] can be adapted for our problem too. But we point out that our requirements on the "Universal Hash Proofs" would be more demanding than what they require. In particular, when using the double-strand approach, we seem to require 5-universality instead of 2-universality, corresponding to our use of five bases $g_1, \dots, g_5$ instead of just two.

Acknowledgment

We would like to acknowledge useful discussions with Rui Xue about the work in [10]. We also thank Ran Canetti and the anonymous referees for their helpful feedback.

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. Cryptology ePrint Archive, Report 2000/067, 2005.

[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] 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.

[10] 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.

[11] G. Danezis. Breaking four mix-related schemes based on universal re-encryption. In *Proceedings of Information Security Conference 2006*. Springer-Verlag, September 2006.

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

[13] 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/`.

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

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

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

[17] P. Golle, M. Jakobsson, A. Juels, and P. Syverson. Universal re-encryption for mixnets. In *Proceedings of the 2004 RSA Conference, Cryptographer's track*, San Francisco, USA, February 2004.

[18] 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.

[19] M. Klonowski, M. Kutylowski, A. Lauks, and F. Zagórski. Universal re-encryption of signatures and controlling anonymous information flow. In *WARTACRYPT '04 Conference on Cryptology*. Bedlewo/Poznan, 2006.

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