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

August 16, 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 that achieves 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.

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, $\nu(n) = n^{-\omega(1)}$. 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 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.

Correctness properties of a perfectly rerandomizable encryption scheme. We require the scheme to satisfy the following correctness properties.

For all key pairs (PK, SK) in the support of KeyGen:

- For every plaintext msg and every (honestly generated) ciphertext ζ in the support of $\mathsf{Enc}_{PK}(\mathsf{msg})$, we must have $\mathsf{Dec}_{SK}(\zeta) = \mathsf{msg}$.
- When $(PK', SK') \leftarrow \mathsf{KeyGen}$, the sets of honestly generated ciphertexts under PK and PK' are disjoint, with overwhelming probability.
- For every plaintext msg and every (honestly generated) ciphertext ζ in the support of $\mathsf{Enc}_{PK}(\mathsf{msg})$, the distribution of $\mathsf{Rerand}(\zeta)$ is identical to that of $\mathsf{Enc}_{PK}(\mathsf{msg})$.
- For every (purported, possibly maliciously generated) ciphertext ζ and every ζ' in the support of $\mathsf{Rerand}(\zeta)$, we must have $\mathsf{Dec}_{SK}(\zeta') = \mathsf{Dec}_{SK}(\zeta)$.

Perfect vs. computational rerandomization. For simplicity, we only consider *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. In this experiment, we generate a key pair $(PK, SK) \leftarrow \mathsf{KeyGen}$ and give PK to the adversary. The adversary then outputs a ciphertext ζ. If $\mathsf{Dec}_{SK}(\zeta) \neq \perp$, then we flip a fair coin and give back either $\mathsf{Enc}_{PK}(\mathsf{Dec}_{SK}(\zeta))$ or $\mathsf{Rerand}(\zeta)$. The adversary is to guess the coin, and must do so with only a negligible advantage. Similar to our other security experiments, the adversary is given access to a decryption oracle throughout the experiment.

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 for q as large as $2^{20,000}$. It is conjectured that infinitely many such chains exist, and with sufficient density.

[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.

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 $i = 1, \ldots, 5$). To allow for rerandomization, we use a direct encoding of the message for the exponent m (instead of a hash of part of the ciphertext). Finally, we thwart attacks which splice together 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 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 $\widehat{g}_1, \widehat{g}_2$ from $\widehat{\mathbb{G}}$, and random $\mathbf{a} = (a_1, a_2), \mathbf{b} = (b_1, b_2)$ from $(\mathbb{Z}_q)^2$. The private key is $(\mathbf{a}, \mathbf{b})$. The public key consists of $\widehat{g}_1$, $\widehat{g}_2$, and the values:

$$A = \prod_{j=1}^{2} \widehat{g}_j^{a_j}; \qquad B = \prod_{j=1}^{2} \widehat{g}_j^{b_j}.$$

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

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

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

- **Check ciphertext integrity:** Check the following constraints:

$$B_V \stackrel{?}{=} \prod_{j=1}^{2} V_j^{b_j}; \qquad A_W \stackrel{?}{=} \prod_{j=1}^{2} W_j^{a_j}; \qquad B_W \stackrel{?}{=} \prod_{j=1}^{2} W_j^{b_j}.$$

  If any fail, or if $W_1 = W_2 = 1$, output $\perp$.
- **Derive plaintext:** Output $A_V / \prod_{j=1}^{2} V_j^{a_j}$.

Rerandomization: $\mathsf{MRerand}(U = (\mathbf{V}, A_V, B_V, \mathbf{W}, A_W, B_W))$: 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 $s \leftarrow \mathbb{Z}_q$, $t \leftarrow \mathbb{Z}_q^*$ and outputting the following ciphertext:

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

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 $w' = tw$, and that these values are correctly distributed in the appropriate domains.

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

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

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 any fixed vector $\mathbf{z} = (z_1, \ldots, z_4) \in (\mathbb{Z}_p)^4$ which is not a scalar multiple of the all-ones vector.

Key generation. Generate a keypair (MPK, MSK) for the DSME scheme in $\widehat{\mathbb{G}}$. Pick random generators $g_1, \ldots, g_4 \in \mathbb{G}$, and random $\mathbf{c} = (c_1, \ldots, c_4), \mathbf{d} = (d_1, \ldots, d_4), \mathbf{e} = (e_1, \ldots, e_4)$ from $(\mathbb{Z}_p)^4$. The private key consists of $\mathbf{c}, \mathbf{d}, \mathbf{e}$ and MSK. The public key consists of $(g_1, \ldots, g_4)$, MPK, and the following values:

$$C = \prod_{i=1}^4 g_i^{c_i}, \qquad D = \prod_{i=1}^4 g_i^{d_i}, \qquad E = \prod_{i=1}^4 g_i^{e_i}.$$

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

- Pick random $x \leftarrow \mathbb{Z}_p$, $y \leftarrow \mathbb{Z}_p^*$, and $u \leftarrow \widehat{\mathbb{G}}$.
- For $i = 1, \ldots, 4$: let $X_i = g_i^{(x+z_i)u}$; $Y_i = g_i^{yu}$.

- Let $\mu = \mathsf{encode}_{\mathbb{G}}(\mathsf{msg})$, $m = \mathsf{encode}_{\mathbb{Z}_p}(\mathsf{msg})$, and $U = \mathsf{MEnc}_{MPK}(u)$.
- Output:
$$(\mathbf{X}, \mu C^x, (DE^m)^x, \mathbf{Y}, C^y, (DE^m)^y, U),$$
where $\mathbf{X} = (X_1, \dots, X_4), \mathbf{Y} = (Y_1, \dots, Y_4)$.

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

- **Decrypt U:** Set $u = \mathsf{MDec}_{MSK}(U)$. If $u = \perp$, immediately output $\perp$.
- **Strip u and $\mathbf{z}$:** For $i = 1, \dots, 4$: set $\overline{X}_i = X_i^{1/u} g_i^{-z_i}$ and $\overline{Y}_i = Y_i^{1/u}$.
- **Derive purported plaintext:** Set $\mu = C_X / \prod_{i=1}^4 \overline{X}_i^{c_i}$, $\mathsf{msg} = \mathsf{encode}_{\mathbb{G}}^{-1}(\mu)$, and $m = \mathsf{encode}_{\mathbb{Z}_p}(\mathsf{msg})$.
- **Check ciphertext integrity:** Check the following conditions:
$$C_Y \overset{?}{=} \prod_{i=1}^4 \overline{Y}_i^{c_i}; \qquad P_X \overset{?}{=} \prod_{i=1}^4 \overline{X}_i^{d_i + e_i m}; \qquad P_Y \overset{?}{=} \prod_{i=1}^4 \overline{Y}_i^{d_i + e_i m}.$$
If any checks fail, output $\perp$. Additionally, output $\perp$ if $Y_1 = \cdots = Y_4 = 1$. Otherwise output msg.

Rerandomization: $\mathsf{Rerand}(\zeta = (\mathbf{X}, C_X, P_X, \mathbf{Y}, C_Y, P_Y, \mathbf{U}))$: The only randomness used in Enc is the choice of x, y, u, and the randomness used in MEnc. We can rerandomize each of these quantities by choosing random $\sigma \leftarrow \widehat{\mathbb{G}}$, random $s \leftarrow \mathbb{Z}_p$, $t \leftarrow \mathbb{Z}_p^*$, and constructing a ciphertext which corresponds to an encryption of the same message, with corresponding random choices $u' = \sigma u$, $x' = x + ys$, and $y' = yt$:

- Set $U' = \mathsf{MRerand}(\sigma \otimes U)$.
- For $i = 1, \dots, 4$, set $X_i' = (X_i Y_i^s)^\sigma$; and $Y_i' = Y_i^{\sigma t}$.
- $C_X' = C_X C_Y^s$ and $P_X' = P_X P_Y^s$.
- $C_Y' = C_Y^t$ and $P_Y' = P_Y^t$.

The rerandomized ciphertext is $\zeta' = (\mathbf{X}', C_X', P_X', \mathbf{Y}', C_Y', P_Y', U')$. When the input ciphertext ζ is generated using Enc, the values u', x', and y' are uniformly distributed in the appropriate domains, and the rerandomization is perfect.

4.3 Complexity

The complexities of the DSCS scheme are summarized in Table 1 and Table 2.[2] Our scheme is clearly 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

[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.

Table 1: Number of elements

	$\widehat{\mathbb{G}}$	$\mathbb{Z}_q$	$\mathbb{G}$	$\mathbb{Z}_p$
Public key	4	-	7	-
Private key	-	4	-	12
Ciphertext	8	-	12	-

Table 2: Group operations performed

	exp.		mult.		inv.	
	$\widehat{\mathbb{G}}$	$\mathbb{G}$	$\mathbb{Z}_p^*$	$\mathbb{G}$	$\mathbb{Z}_p^*$	$\mathbb{G}$
Enc	8	14	3	3	0	0
Dec (worst case)	8	24	9	17	1	1
Rerand	8	16	6	4	0	0

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 $\mathcal{F}_{\text{RMP}}$ 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 register from a party sender, the functionality sends (ID-REQ, sender) to the adversary, and expects in response an identifier string id.[3] If the string received in response

[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.

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.

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 $\widehat{\mathbb{G}}$.*

[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.

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, given the public key, 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 follows. $\lhd$

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.*

[6]By relaxing the requirement that the scheme have tight decryption (and also the correctness requirement that

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 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. $\lhd$

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.

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 proof of RCCA security. To do this, we will demonstrate alternate encryption and decryption procedures. The alternate encryption procedure AltEnc is described in Section 7.4 and the alternate decryption procedure AltDec is described in Section 7.5. The lemmas in the rest of this section carry out the proof outlined in Section 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' in the support of $\mathsf{MRerand}(U)$, we have $\mathsf{MDec}_{MSK}(U') = \mathsf{MDec}_{MSK}(U)$.*

Proof: Let $MSK = (\mathbf{a}, \mathbf{b})$ be a private key, and $U = (\mathbf{V}, A_V, B_V, \mathbf{W}, A_W, B_W)$ be a ciphertext. First, suppose $\mathsf{MDec}_{MSK}(U) = u \neq \perp$. This happens if and only if some $W_j \neq 1$ and the following conditions hold:

$$A_V/u = \textstyle\prod_{j=1}^2 V_j^{a_j}; \qquad B_V = \prod_{j=1}^2 V_j^{b_j}; \quad A_W = \prod_{j=1}^2 W_j^{a_j}; \quad B_W = \prod_{j=1}^2 W_j^{b_j}.$$

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

$$\begin{aligned} A'_V/u = (A_V A_W^s)/u = \left[\textstyle\prod_{j=1}^2 V_j^{a_j}\right]\left[\prod_{j=1}^2 W_j^{a_j}\right]^s = \prod_{j=1}^2 (V_j W_j^s)^{a_j} = \prod_{j=1}^2 (V'_j)^{a_j} \\ A'_W = A_W^t = \left[\textstyle\prod_{j=1}^2 W_j^{a_j}\right]^t = \prod_{j=1}^2 (W_j^t)^{a_j} = \prod_{j=1}^2 (W'_j)^{a_j} \end{aligned}$$

Analogously, the constraints on B_V and B_W hold, and some $W'_j = W_j^t \neq 1$ (as $t \neq 0 \bmod q$ by definition). Thus $\mathsf{MDec}_{MSK}(U') = u$ as well.

Likewise, $\mathsf{MDec}_{MSK}(U) = \perp$ if and only if $W_1 = W_2 = 1$, or one of the following three inequalities holds:

$$B_V \neq \prod_{j=1}^2 V_j^{b_j}; \quad A_W \neq \prod_{j=1}^2 W_j^{a_j}; \quad B_W \neq \prod_{j=1}^2 W_j^{b_j}.$$

When this is the case, the corresponding condititon holds for U' as well, as can be seen by a similar argument to above. Thus $\mathsf{MDec}_{MSK}(U') = \perp$ as well. □

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

PROOF: Fix a private key SK and let $\zeta = (\mathbf{X}, C_X, P_X, \mathbf{Y}, C_Y, P_Y, U)$ be a (purported) ciphertext. Let $\zeta' = (\mathbf{X}', C'_X, P'_X, \mathbf{Y}', C'_Y, P'_Y, U') \leftarrow \mathsf{Rerand}(\zeta)$, where for some $\sigma \in \widehat{\mathbb{G}}, s \in \mathbb{Z}_p, t \in \mathbb{Z}_p^*$, we have:

$$X'_i = (X_i Y_i^s)^\sigma; \quad Y'_i = Y_i^{\sigma t}; \quad C'_X = C_X C_Y^s; \quad P'_X = P_X P_Y^s;$$
$$C'_Y = C_Y^t; \quad P'_Y = P_Y^t; \quad U' \leftarrow \mathsf{MRerand}(\sigma \otimes U)$$

If U decrypts to $\perp$ under the DSME private key, then by the correctness properties of the DSME scheme, U' will also decrypt to $\perp$ and we have $\mathsf{Dec}_{SK}(\zeta) = \mathsf{Dec}_{SK}(\zeta') = \perp$.

Otherwise, suppose U decrypts to u; then U' decrypts to σu. When decrypting ζ, the decryption procedure will strip u and $\mathbf{z}$ to obtain: $\overline{X}_i = X_i^{1/u} g_i^{-z_i}$ and $\overline{Y}_i = Y_i^{1/u}$. However, when decrypting the rerandomization ζ', the computed values will be:

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

We now consider several cases:

1. First, observe that $Y_1 = \cdots Y_4 = 1$ if and only if $Y'_1 = \cdots = Y'_4 = 1$, since each $Y'_i = Y_i^{\sigma t}$ and $\sigma, t \neq 0 \bmod p$ by definition. Thus this integrity check on the ciphertext fails for ζ if and only if it fails for ζ'.

2. The $C_Y \overset{?}{=} \prod_i \overline{Y}_i^{c_i}$ integrity check fails on ζ. Then the corresponding check on ζ' also fails:
$$C_Y^t = C'_Y \overset{?}{=} \textstyle\prod_{i=1}^4 (\overline{Y}'_i)^{c_i} = \prod_{i=1}^4 (\overline{Y}_i^t)^{c_i} = \left[\prod_{i=1}^4 \overline{Y}_i^{c_i}\right]^t \neq C_Y^t$$

3. The $C_Y \overset{?}{=} \prod_i \overline{Y}_i^{c_i}$ integrity check succeeds and the $P_Y \overset{?}{=} \prod_i \overline{Y}_i^{d_i + m e_i}$ integrity check fails on ζ. Suppose that μ is the purported message of ζ computed during decryption. The decryption procedure computes the purported message μ' of ζ' as follows:
$$\mu' = \frac{C'_X}{\prod_{i=1}^4 (\overline{X}'_i)^{c_i}} = \frac{C_X}{\prod_{i=1}^4 \overline{X}_i^{c_i}} \left[\frac{C_Y}{\prod_{i=1}^4 \overline{Y}_i^{c_i}}\right]^s = \mu \cdot 1$$
I.e, the same purported message is computed during the decryption of ζ'. Analagous to case (1), the corresponding P_Y check on ζ' fails.

4. The C_Y and P_Y integrity checks succeed on ζ. Then the purported messages computed for ζ and ζ' are the same. Subsequently, the check on P_X succeeds if and only if the corresponding check on P'_X succeeds:
$$1 \overset{?}{=} \frac{P'_X}{\prod_{i=1}^4 (\overline{X}'_i)^{d_i + m e_i}} = \frac{P_X}{\prod_{i=1}^4 \overline{X}_i^{d_i + m e_i}} \left[\frac{P_Y}{\prod_{i=1}^4 \overline{Y}_i^{d_i + m e_i}}\right]^s = \frac{P_X}{\prod_{i=1}^4 \overline{X}_i^{d_i + m e_i}} \cdot 1$$

Thus, if any integrity check on ζ fails, the corresponding check in ζ' fails as well (and both decrypt to $\perp$). Otherwise, all integrity checks on ζ' succeed and the same message is returned as in the decryption of ζ. □

7.2 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 $g_1, \ldots, g_n \leftarrow \mathbb{G}$, and pick a random $v \leftarrow \mathbb{Z}_p$, 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 $g_1, \ldots, g_n \leftarrow \mathbb{G}$, and pick random $v_1, \ldots, v_n \leftarrow \mathbb{Z}_p$, 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 $n = 2$ and $n = 4$, 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 $g_1, \ldots, g_n \leftarrow \mathbb{G}$, and pick random $v, w \leftarrow \mathbb{Z}_p$, 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 $g_1, \ldots, g_n \leftarrow \mathbb{G}$, and pick random $v_1, \ldots, v_n, w_1, \ldots, w_n \leftarrow \mathbb{Z}_p$, 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 $K_0 \leftarrow \mathsf{DS\text{-}DDH}(\mathbb{G}, 4)$, and pick $K_1 \leftarrow \mathsf{DDH}(\widehat{\mathbb{G}}, 2)$. Output (K_0, K_1).

- Pick $K_0 \leftarrow \mathsf{DS\text{-}Rand}(\mathbb{G}, 4)$, and pick $K_1 \leftarrow \mathsf{Rand}(\widehat{\mathbb{G}}, 2)$. Output (K_0, K_1).

A final hybrid argument shows that if $\mathsf{DS\text{-}DDH}(\mathbb{G}, 4)$ and $\mathsf{DS\text{-}Rand}(\mathbb{G}, 4)$ are indistinguishable, and if $\mathsf{DDH}(\widehat{\mathbb{G}}, 2)$ and $\mathsf{Rand}(\widehat{\mathbb{G}}, 2)$ are also indistinguishable, then the above two distributions are indistinguishable.

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

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

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 $r \in \mathbb{Z}_q, s \in \mathbb{Z}_q^*$. In ciphertexts generated by MEnc, both strands are scalar multiples of the all-ones vector.

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

Definition 2 *Let $\zeta = (\mathbf{X}, C_X, P_X, \mathbf{Y}, C_Y, P_Y, U)$ be a DSCS ciphertext. The* masked DSCS strands *of ζ with respect to a public key $(g_1, \dots, g_4, C, D, E)$ are:*

$$\mathbf{x} = (x_1, \dots, x_4), \text{ where } x_i = \log_{g_i} X_i$$
$$\mathbf{y} = (y_1, \dots, y_4), \text{ where } y_i = \log_{g_i} Y_i$$

The unmasked DSCS strands *of ζ with respect to a public key $(g_1, \dots, g_4, C, D, E)$ and $u \in \widehat{\mathbb{G}}$ are:*

$$\overline{\mathbf{x}} = \mathbf{x}/u - \mathbf{z}, \qquad \overline{\mathbf{y}} = \mathbf{y}/u$$

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 random $r \in \mathbb{Z}_p, s \in \mathbb{Z}_p^*$. In ciphertexts generated by Enc, both unmasked strands are scalar multiples of the all-ones vector.

Adversary's view in the RCCA experiment. In the RCCA security experiment, suppose $(\mathbf{a}, \mathbf{b})$ is the DSME private key and $(\widehat{g}_1, \widehat{g}_2, A, B)$ is the DSME public key. Suppose the DSME component of the challenge ciphertext is $U^* = (\mathbf{V}^*, A_V^*, B_V^*, \mathbf{W}^*, A_W^*, B_W^*)$ which decrypts to u^*. Let $\mathbf{w}^*$ and $\mathbf{v}^*$ be the strands of U^* with respect to the public key. Then the following constraints hold (in the finite field of order q):

$$\begin{bmatrix} \mathbf{1} & 0 \\ 0 & \mathbf{1} \\ \mathbf{v}^* & 0 \\ \mathbf{w}^* & 0 \\ 0 & \mathbf{v}^* \\ 0 & \mathbf{w}^* \end{bmatrix} \begin{bmatrix} \hat{G} & 0 \\ 0 & \hat{G} \end{bmatrix} \begin{bmatrix} \mathbf{a}^T \\ \mathbf{b}^T \end{bmatrix} = \begin{bmatrix} \log A \\ \log B \\ \log(A_V^*/u^*) \\ \log A_W^* \\ \log B_V^* \\ \log B_W^* \end{bmatrix}, \text{ where } \hat{G} = \begin{bmatrix} \log \widehat{g}_1 & 0 \\ 0 & \log \widehat{g}_2 \end{bmatrix} \tag{1}$$

The logarithm is with respect to any fixed generator of $\widehat{\mathbb{G}}$.

Similarly, let $\zeta^* = (\mathbf{X}^*, C_X^*, P_X^*, \mathbf{Y}^*, C_Y^*, P_Y^*, U^*)$ denote the challenge ciphertext. Let $\overline{\mathbf{x}}^*$ and $\overline{\mathbf{y}}^*$ denote the unmasked DSCS strands of ζ^*, with respect to u^*, and the DSCS public key. The following constraints hold (in the finite field of order p):

$$\begin{bmatrix} \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} \\ \overline{\mathbf{x}}^* & 0 & 0 \\ \overline{\mathbf{y}}^* & 0 & 0 \\ 0 & \overline{\mathbf{x}}^* & m^*\overline{\mathbf{x}}^* \\ 0 & \overline{\mathbf{y}}^* & m^*\overline{\mathbf{y}}^* \end{bmatrix} \begin{bmatrix} G & 0 & 0 \\ 0 & G & 0 \\ 0 & 0 & G \end{bmatrix} \begin{bmatrix} \mathbf{c}^T \\ \mathbf{d}^T \\ \mathbf{e}^T \end{bmatrix} = \begin{bmatrix} \log C \\ \log D \\ \log E \\ \log(C_X^*/\mu) \\ \log C_Y^* \\ \log P_X^* \\ \log P_Y^* \end{bmatrix}, \text{ where } G = \begin{bmatrix} \log g_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \log g_4 \end{bmatrix} \tag{2}$$

Again, the logarithm is with respect to any fixed generator of $\mathbb{G}$. $m^* = \mathsf{encode}_{\mathbb{Z}_p}(\mathsf{msg}^*)$ and $\mu = \mathsf{encode}_{\mathbb{G}}(\mathsf{msg}^*)$, where msg^* is the plaintext used to generate ζ^*.

We call the rows of the first matrix in Equation 1 and first matrix in Equation 2 the *view constraints* for the adversary. Note that in Phase I of the RCCA experiment, only the first two constraints of Equation 1 and the first three constraints of Equation 2 are relevant. In Phase II, all constraints are relevant.

Decryption constraints. Let $\zeta = (\mathbf{X}, C_X, P_X, \mathbf{Y}, C_Y, P_Y, U)$ denote a ciphertext query given to the decryption oracle by the adversary in the RCCA experiment. Let $\mathbf{v}, \mathbf{w}$ denote the DSME strands of U, and let u be the purported plaintext of U computed during the decryption procedure. Let $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$ denote the unmasked DSCS strands of ζ^*, with respect to u, and and let msg be the purported ciphertext computed during the decryption procedure, with $\mathsf{encode}_{\mathbb{Z}_p}(\mathsf{msg}) = m$. Then the decryption of ζ succeeds (i.e, the decryption oracle returns msg) if and only if $\mathbf{w}, \mathbf{y}$ are nonzero and the following constraints hold (in the appropriate fields):

$$\begin{bmatrix} \mathbf{w} & 0 \\ 0 & \mathbf{v} \\ 0 & \mathbf{w} \end{bmatrix} \begin{bmatrix} \hat{G} & 0 \\ 0 & \hat{G} \end{bmatrix} \begin{bmatrix} \mathbf{a}^T \\ \mathbf{b}^T \end{bmatrix} = \begin{bmatrix} \log A_W \\ \log B_V \\ \log B_W \end{bmatrix}$$

$$\begin{bmatrix} \overline{\mathbf{y}} & 0 & 0 \\ 0 & \overline{\mathbf{x}} & m\overline{\mathbf{x}} \\ 0 & \overline{\mathbf{y}} & m\overline{\mathbf{y}} \end{bmatrix} \begin{bmatrix} G & 0 & 0 \\ 0 & G & 0 \\ 0 & 0 & G \end{bmatrix} \begin{bmatrix} \mathbf{c}^T \\ \mathbf{d}^T \\ \mathbf{e}^T \end{bmatrix} = \begin{bmatrix} \log C_Y \\ \log P_X \\ \log P_Y \end{bmatrix}$$

Similar to above, we call the rows of the first matrix in each line the *decryption constraints.*

Lemma 3 *In the RCCA experiment, call a ciphertext* bad *if one of its decryption constraints is linearly independent of the corresponding set of view constraints which are relevant at that time. Then with overwhelming probability, all bad ciphertexts submitted to the decryption oracle are rejected (i.e, the oracle returns $\perp$).*

Proof: First, observe that responses from the decryption oracle for "good" ciphertexts do not (information-theoretically) reveal to the adversary any more information about the private key than is given by the view constraints, since no new (independent) constraints are revealed about the private key.

Consider the first bad ciphertext submitted to the decryption oracle. The adversary's view of the private key is constrained only by the relevant view constraints (Equation 1 and Equation 2). In other words, from the adversary's point of view, the private key is distributed uniformly among all keys consistent with these constraints. In particular, for each possible value for the right-hand side of the *independent* (bad) constraint (corresponding to the part of the ciphertext which the decryption procedure checks with this constraint), there are an equal number of private keys which are simultaneously consistent with the view constraints and this independent constraint. In other words, the "correct" value for this constraint is uniformly distributed in the appropriate group, from the adversary's point of view. The probability that it submits a ciphertext with this particular value is negligible.

However, the fact that a bad ciphertext is rejected also reveals some information about the private key. In particular, it reveals that the private key used by the decryption procedure is not one of those which was consistent with this value for the independent constraint. This rules out a $1/p$ or $1/q$ fraction of private keys (depending on whether the constraint was a DSME or DSCS constraint). However, from the adversary's view, the correct private key is still distributed uniformly among the remaining $(p-1)/p$ or $(q-1)/q$ fraction of keys. By a union bound, if the adversary makes N bad queries, one of them is accepted with probability at most $N/(q-N)$. Since the adversary makes a polynomial (in the security parameter) number of queries, and q is superpolynomial in the security parameter, this probability is negligible. $\square$

7.4 The Alternate Encryption Procedure

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

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

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

$$\mathbf{V} = (V_1, V_2) \qquad A_V = u \cdot \prod_{j=1}^{2} V_j^{a_j} \qquad A_W = \prod_{j=1}^{2} W_j^{a_j}$$
$$\mathbf{W} = (W_1, W_2) \qquad B_V = \prod_{j=1}^{2} V_j^{b_j} \qquad B_W = \prod_{j=1}^{2} W_j^{b_j}$$

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

- Pick random $x_1, \ldots, x_4, y_1, \ldots, y_4 \in \mathbb{Z}_p^*$. For $i = 1, \ldots, 4$, 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 $u \in \widehat{\mathbb{G}}$, set $U = \mathsf{AltMEnc}_{\mathbf{a}}(u)$, and for $i = 1, \ldots, 4$, set $X_i = (\overline{X}_i g_i^{z_i})^u$, $Y_i = \overline{Y}_i^u$.
- Let $\mu = \mathsf{encode}_{\mathbb{G}}(\mathsf{msg})$, and $m = \mathsf{encode}_{\mathbb{Z}_p}(\mathsf{msg})$.
- Output $(\mathbf{X}, C_X, P_X, \mathbf{Y}, C_Y, P_Y, U)$, where

$$C_X = \mu \cdot \prod_{i=1}^{4} \overline{X}_i^{c_i} \qquad \mathbf{X} = (X_1, \ldots, X_4)$$
$$P_X = \prod_{i=1}^{4} \overline{X}_i^{d_i + e_i m} \qquad \mathbf{Y} = (Y_1, \ldots, Y_4)$$
$$C_Y = \prod_{i=1}^{4} \overline{Y}_i^{c_i}$$
$$P_Y = \prod_{i=1}^{4} \overline{Y}_i^{d_i + e_i m}$$

These alternate encryption procedures differ from the normal encryption procedures in that they generate ciphertexts whose two strands are not necessarily scalar multiples of the all-ones vector. The DSME alternate encryption generates a ciphertext whose first strand is random, and the DSCS alternate encryption generates a ciphertext where both strands are random. The remainder of the ciphertexts are constructed using the private keys to ensure that they decrypt successfully to the desired value.

Lemma 4 *If the challenge ciphertext in the RCCA experiment is generated using* AltEnc*, then conditioned on a negligible-probability event not occuring, the challenge ciphertext is distributed independently of the plaintext message and the randomness* u*, even given the public key.*

PROOF: The relationships among keys, ciphertext, plaintext, and u are given in Equation 1 and Equation 2. Given a DSME ciphertext from $\mathsf{AltMEnc}$ with first $\mathbf{v}^*$, the set $\{\mathbf{v}^*, \mathbf{1}\}$ forms a basis for the space of all DSME strands, with overwhelming probability. Considering the matrix on the left-hand side of Equation 1, the 4th and 6th rows (which involve $\mathbf{w}^*$, a scalar multiple of the all-ones vector) are scalar multiples of rows which involve $\mathbf{1}$, so they do not contribute any additional constraints on the private keys. We remove these rows from consideration, and the remaining matrix is nonsingular.

Similarly, in a DSCS ciphertext from AltEnc, the two (masked) strands along with $\{\mathbf{1}, \mathbf{z}\}$ form a basis for the space of all DSCS strands, with overwhelming probability. When this is the case, the left-hand side of Equation 2 (which is in terms of the *unmasked* strands $\mathbf{x}/u - \mathbf{z}$ and $\mathbf{y}/u$) is nonsingular for any m and u.

Now fix any ciphertext from AltEnc that avoids the negligible-probability event that one of these matrixes are singular. Fixing a choice of u and message determines all of the quantities in Equation 1 and Equation 2 except for the private keys $\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}$. For each of these choices of u and message, the matrix equations are nonsingular, and there are an equal number of consistent private key solutions.

Since the underlying randomness in the RCCA experiment is the choice of random private keys, the choices of u and message are independent of each other and of the generation of ciphertext and public keys. $\square$

Lemma 5 *For every adversary, its advantage in the RCCA experiment when the challenge ciphertext is generated using* AltEnc *is negligibly close to its advantage in the experiment (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 7.2 are computationally indistinguishable. Elements in the support of these distributions consist of a double-strand tuple of length 4 from $\mathbb{G}$, and a double-strand tuple of length 3 from $\widehat{\mathbb{G}}$.

Now consider a simulation of the RCCA experiment, where the input is from one of the above distributions. Let $(\widehat{g}_1, \widehat{g}_2, V_1, V_2)$ be the sample from either $\mathsf{DDH}(\widehat{\mathbb{G}}, 2)$ or $\mathsf{Rand}(\widehat{\mathbb{G}}, 2)$. Set $(\widehat{g}_1, \widehat{g}_2)$ as the corresponding part of the DSME public key, and generate the remainder of the keypair $(\mathbf{a}, \mathbf{b})$ honestly. To simulate the encryption of u^* from the challenge ciphertext with this keypair, use $\mathsf{AltMEnc}$ with the input values V_1, V_2.

Similarly, let $(g_1, \ldots, g_4, \overline{X}_1, \ldots, \overline{X}_4, \overline{Y}_1, \ldots, \overline{Y}_4)$ be the sample from either $\mathsf{DS\text{-}DDH}(\mathbb{G}, 4)$ or $\mathsf{DS\text{-}Rand}(\mathbb{G}, 4)$. Set $(g_1, \ldots, g_4)$ as the corresponding part of the DSCS public key and generate the remainder of the DSCS keypair $(\mathbf{c}, \mathbf{d}, \mathbf{e})$ honestly. To simulate the encryption of the challenge ciphertext, use AltEnc with the input values $\overline{X}_1, \ldots, \overline{X}_4, \overline{Y}_1, \ldots, \overline{Y}_4$.

It is straight-forward to check that when the input is sampled from the first distribution (i.e, the 2 tuples come from the appropriate DDH distributions), the ciphertext is distributed statistically close to an honest encryption with Enc and MEnc (the distribution is identical when conditioned to avoid the negligible-probability event that $\overline{Y}_1 = \cdots = \overline{Y}_4 = 1$). If the input is sampled from the second distribution (i.e, the 2 tuples comes from the appropriate random distributions), then the ciphertext is distributed identically as an encryption with AltEnc and $\mathsf{AltMEnc}$.

The rest of this simulation of the RCCA experiment can be implemented in polynomial time. Thus, the outcomes of the two simulations must not differ by more than a negligible amount. $\square$

7.5 The Alternate Decryption Procedure

We now describe (computationally unbounded) alternate decryption procedures for the DSME and DSCS schemes. 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 ($\mathsf{AltMDec}^{U^*}_{MPK}(U)$). If in Phase II (i.e, after the challenge ciphertext has been generated), let $U^* = (\mathbf{V}^*, A_V^*, B_V^*, \mathbf{W}^*, A_W^*, B_W^*)$ denote the DSME component of the challenge ciphertext (created with $\mathsf{AltMEnc}$). Let $U = (\mathbf{V}, A_V, B_V, \mathbf{W}, A_W, B_W)$ 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 in Phase II), both with respect to the given public key (by computing discrete logs in $\widehat{\mathbb{G}}$). If $\mathbf{w}$ is the all-zeroes vector, then reject just as in MDec.

If in Phase I (before the challenge ciphertext has been generated), check that $\mathbf{v}$ and $\mathbf{w}$ are both scalar multiples of the all-ones vector, say, $\mathbf{v} = \epsilon\mathbf{1}$ and $\mathbf{w} = \gamma\mathbf{1}$, for $\gamma \neq 0$. If not, output $\perp$. Otherwise, check that the following conditions hold:

$$B_V \stackrel{?}{=} B^\epsilon; \qquad A_W \stackrel{?}{=} A^\gamma; \qquad B_W \stackrel{?}{=} B^\gamma.$$

If any check fails, output $\perp$. Otherwise, output the pair $(\sigma = A_V/A^\epsilon, \pi = 0)$.

If in Phase II, then by Lemma 4, we may write $\mathbf{v}$ and $\mathbf{w}$ each as a unique linear combination of $\{\mathbf{v}^*, \mathbf{1}\}$, say, $\mathbf{v} = \pi\mathbf{v}^* + \epsilon\mathbf{1}$ and $\mathbf{w} = \alpha\mathbf{v}^* + \gamma\mathbf{1}$. If $\alpha \neq 0$, then output $\perp$. Otherwise, if $\gamma = 0$, then $\mathbf{w}$ is the all-zeroes vector, and we reject as well. Otherwise, $\alpha = 0, \gamma \neq 0$. Check that the following conditions hold:

$$B_V \stackrel{?}{=} (B_V^*)^\pi B^\epsilon; \qquad A_W \stackrel{?}{=} A^\gamma; \qquad B_W \stackrel{?}{=} B^\gamma.$$

If any check fails, output $\perp$. Otherwise output the following pair:

$$\left(\sigma = \frac{A_V}{(A_V^*)^\pi A^\epsilon}, \pi\right).$$

Below, we prove that these values (σ, π) are such that $\mathsf{MDec}_{MSK}(U) = \sigma \mathsf{MDec}_{MSK}(U^*)^\pi$, where MSK is the private key used to generate U^*.

The following lemma establishes the correctness of the output of $\mathsf{AltMDec}$.

Lemma 6 *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 (σ, π), then* $\mathsf{MDec}_{MSK}(U) = \sigma\mathsf{MDec}_{MSK}(U^*)^\pi$.

PROOF: Let $\mathbf{v}, \mathbf{w}$ be the strands of U, let $\mathbf{v}^*, \mathbf{w}^*$ be the strands of U^*, and let $MSK = (\mathbf{a}, \mathbf{b})$ be the private key used to generate U^*. Suppose $\mathsf{AltMDec}$ does not output $\perp$. As in the description of $\mathsf{AltMDec}$, let $\mathbf{v} = \pi\mathbf{v}^* + \epsilon\mathbf{1}$ and $\mathbf{w} = \gamma\mathbf{1}$ for $\gamma \neq 0$. Upon decrypting U, the real decryption procedure MDec will return a non-$\perp$ value if and only if the three integrity constraints pass.

Considering the constraint involving B_V, we see that it passes if and only if

$$1 = \frac{B_V}{\prod_{j=1}^2 V_j^{b_j}} = \frac{B_V}{\left(\prod_{j=1}^2 (V_j^*)^{b_j}\right)^\pi \left(\prod_{j=1}^2 (\widehat{g}_j^*)^{b_j}\right)^\epsilon} = \frac{B_V}{(B_V^*)^\pi B^\epsilon}$$

which is exactly the condition checked in AltMDec. The other two constraints (involving A_W, B_W) are analogous. Thus if AltMDec returns a non-$\perp$ value on U, then so does MDec. To show the correctness of AltMDec's output, we observe that:

$$\begin{aligned}\mathsf{MDec}_{MSK}(U) &= \frac{A_V}{\prod_{j=1}^{2} V_j^{a_j}} = \frac{A_V}{\left(\prod_j (V_j^*)^{a_j}\right)^{\pi}\left(\prod_j \widehat{g}_j^{a_j}\right)^{\epsilon}} = \frac{A_V}{\left(\frac{A_V^*}{\mathsf{MDec}_{MSK}(U^*)}\right)^{\pi} A^{\epsilon}} \\ &= \left[\frac{A_V}{(A_V^*)^{\pi} A^{\epsilon}}\right] \mathsf{MDec}_{MSK}(U^*)^{\pi}\end{aligned}$$

These values correspond exactly with the (σ, π) returned by AltMDec (the behavior of AltMDec in Phase I when it does not output $\perp$ can easily be seen to be a special case of the above, restricted to $\pi = 0$). $\square$

We now describe the DSCS alternate decryption procedure AltDec.

DSCS alternate decryption ($\mathsf{AltDec}_{PK}^{\zeta^*,\mathsf{msg}_0,\mathsf{msg}_1}(\zeta)$). Let $\zeta^* = (\mathbf{X}^*, C_X^*, P_X^*, \mathbf{Y}^*, C_Y^*, P_Y^*, U^*)$ denote the challenge ciphertext (created with AltEnc) and $\zeta = (\mathbf{X}, C_X, P_X, \mathbf{Y}, C_Y, P_Y, U)$ denote the ciphertext query given as input.

1. First, let $(\sigma, \pi) \leftarrow \mathsf{AltMDec}_{MPK}^{U^*}(U)$. If it returns $\perp$, then also return $\perp$. If $\pi \notin \{0, 1\}$, return $\perp$ as well.
2. If $\pi = 1$, check whether there exist $s \in \mathbb{Z}_p$, $t \in \mathbb{Z}_p^*$ such that the following equalities hold:

$$\begin{aligned} \mathbf{X} &= (\mathbf{X}^*(\mathbf{Y}^*)^s)^{\sigma} & \mathbf{Y} &= (\mathbf{Y}^*)^{\sigma t} \quad \text{(componentwise)} \\ C_X &= C_X^*(C_Y^*)^s & C_Y &= (C_Y^*)^t \\ P_X &= P_X^*(P_Y^*)^s & P_Y &= (P_Y^*)^t \end{aligned}$$

   If so, output replay; otherwise output $\perp$.
3. If $\pi = 0$, then check whether there exist msg, $x \in \mathbb{Z}_p$, $y \in \mathbb{Z}_p^*$ such that the following equalities hold:

$$\begin{aligned} X_i &= g_i^{(x+z_i)\sigma} & Y_i &= g_i^{y\sigma} \\ C_X &= \mathsf{encode}_{\mathbb{G}}(\mathsf{msg}) \cdot C^x & C_Y &= C^y \\ P_X &= (DE^{\mathsf{encode}_{\mathbb{Z}_p}(\mathsf{msg})})^x & P_Y &= (DE^{\mathsf{encode}_{\mathbb{Z}_p}(\mathsf{msg})})^y \end{aligned}$$

   If not, output $\perp$. If $\mathsf{msg} \in \{\mathsf{msg}_0, \mathsf{msg}_1\}$, then output replay; otherwise, output msg.

In other words, check that ζ looks like a rerandomization of ζ^* or an honestly-generated encryption of msg, according to how their DSME components are related. Note that in Phase I of the RCCA experiment (before the challenge ciphertext is generated), only the $\pi = 0$ case (which does not use the challenge ciphertext) is possible.

By the correctness of AltMDec proven in Lemma 6 and the correctness properties of rerandomization and decryption, it is straight-forward to see that whenever AltDec outputs replay or some msg, its output agrees with the output of the guarded decryption oracle GDec from the RCCA experiment.

Lemma 7 *In the cases where* AltDec *outputs* $\perp$*, the normal decryption procedure* Dec *would also output* $\perp$ *with overwhelming probability (over the remaining randomness in the private key generation).*

Proof: Let $\zeta^* = (\mathbf{X}^*, C_X^*, P_X^*, \mathbf{Y}^*, C_Y^*, P_Y^*, U^*)$ denote the challenge ciphertext (created with AltEnc) and $\zeta = (\mathbf{X}, C_X, P_X, \mathbf{Y}, C_Y, P_Y, U)$ denote the ciphertext query given as input.

There are three cases where AltDec outputs $\perp$ due to AltMDec outputting $\perp$. We show that MDec would also output $\perp$ on these DSME ciphertexts with overwhelming probability:

- If in Phase I, one of the strands of $U = (\mathbf{V}, A_V, B_V, \mathbf{W}, A_W, B_W)$ is linearly independent of the all-ones vector. Such a ciphertext is *bad*, in the terminology of Lemma 3, and thus the ciphertext will be rejected with overwhelming probability.
- The second strand of $U = (\mathbf{V}, A_V, B_V, \mathbf{W}, A_W, B_W)$ is a linear combination of the form $\mathbf{w} = \alpha \mathbf{v}^* + \gamma \mathbf{1}$, where $\alpha \neq 0$, and $\mathbf{v}^*, \mathbf{w}^*$ are the strands of $U^* = (\mathbf{V}^*, A_V^*, B_V^*, \mathbf{W}^*, A_W^*, B_W^*)$.

  In this case, in order for U to pass the integrity check on A_W, we must have:

  $$A_W = \prod_{j=1}^3 W_j^{a_j} = \left(\prod_{j=1}^3 (V_j^*)^{a_j}\right)^\alpha \left(\prod_{j=1}^3 \widehat{g}_j^{a_j}\right)^\gamma = \left(\frac{A_V^*}{u^*}\right)^\alpha A^\gamma$$

  where $u^* = \mathsf{MDec}_{SK}(U^*)$. All the quantities on the right-hand side of this equation are fixed from the advesary's point of view, except u^*, which is distributed uniformly in $\widehat{\mathbb{G}}$ (according to Lemma 4). If $\alpha \neq 0$, then by a similar argument to Lemma 3, equality will hold (and U decrypts successfully) for only a negligible fraction of private keys $\mathbf{a}$.
- AltMDec outputs $\perp$ due to an integrity check failing (on B_V, A_W, B_W, or on the second strand being a nonzero vector). It is straightforward to check that the corresponding check performed by MDec will also fail (unconditionally over the choice of consistent private keys).

Otherwise, suppose that $\mathsf{AltMDec}_{MPK}^{U^*}(U) = (\sigma, \pi)$. By Lemma 6, these values are such that $\mathsf{MDec}_{MSK}(U^*) = \sigma \mathsf{MDec}_{MSK}(U)^\pi$. We now consider the constraint corresponding to the row $[0\ \overline{\mathbf{x}}\ m\overline{\mathbf{x}}]$ in Equation 2, and show that there are only two ways this constraint could hold with non-negligible probability over the remaining choice of private keys:

As parts of the challenge ciphertext, the adversary is given the values:

$$\{X_i^* = g_i^{(x_i^* + z_i)u^*}\} \qquad \{Y_i^* = g_i^{y_i^* u^*}\}$$

for some u^* corresponding to the decryption of U^* (though the "correct" value of u^* is distributed independently of the adversary's view). When submitting a ciphertext to the decryption oracle, the adversary supplies the values:

$$\{X_i = g_i^{(x_i + z_i)u}\} \qquad \{Y_i = g_i^{y_i u}\}$$

for some u, where u is related to u^* via $u = \sigma(u^*)^\pi$. We argued earlier that with overwhelming probability, $\{\overline{\mathbf{x}}^*, \overline{\mathbf{y}}^*, \mathbf{z}, \mathbf{1}\}$ span the space of strands. Then also $\{(\overline{\mathbf{x}}^* + \mathbf{z})u^*, \overline{\mathbf{y}}^* u^*, \mathbf{z}, \mathbf{1}\}$ span the space for any value of u^*. Thus we can write the following unique linear combination:

$$(\overline{\mathbf{x}} + \mathbf{z})u = \alpha\Big((\overline{\mathbf{x}}^* + \mathbf{z})u^*\Big) + \beta(\overline{\mathbf{y}}^* u^*) + \gamma \mathbf{1} + \delta \mathbf{z} \qquad (3)$$

Note that the coefficients $\alpha, \beta, \gamma, \delta$ of this linear combination are fixed (implicitly by the adversary) independently of the randomness in u^*. We will proceed by showing that if these coefficients are not fixed in a particular way, then the ciphertext would be rejected by Dec with overwhelming probability over the remaining randomness in u^* and the private key $(\mathbf{c}, \mathbf{d}, \mathbf{e})$.

Suppose the decryption procedure computes a purported message msg with $\mathsf{encode}_{\mathbb{Z}_p}(\mathsf{msg}) = m$. Let us consider whether the constraint $[0\ \overline{\mathbf{x}}\ m\overline{\mathbf{x}}]$ is further satisfied. Solving for $\overline{\mathbf{x}}$ in the above equation, and substituting, we have:

$$[0\ \overline{\mathbf{x}}\ m\overline{\mathbf{x}}] = \frac{\gamma}{u}[0\ \mathbf{1}\ m\mathbf{1}] + \frac{u^*}{u}\Big(\alpha[0\ \overline{\mathbf{x}}^*\ m\overline{\mathbf{x}}^*] + \beta[0\ \overline{\mathbf{y}}^*\ m\overline{\mathbf{y}}^*]\Big) + \left(\alpha\frac{u^*}{u} + \frac{\delta}{u} - 1\right)[0\ \mathbf{z}\ m\mathbf{z}]$$

This constraint must be a linear combination of relevant constraints on the adversary's view, with non-negligible probability over u^*, or else the ciphertext will be rejected with overwhelming probability over $(\mathbf{c}, \mathbf{d}, \mathbf{e})$, by Lemma 3. Furthermore, the coefficients must be *fixed* with non-negligible probability over the randomness (in u^*), otherwise the "correct" value of the constraint will be distributed randomly in a superpolynomial-size domain as u^* varies.

We consider several cases on the value π returned by $\mathsf{AltMDec}$ (the value such that $u = \sigma(u^*)^\pi$):

- $\pi = 0$: Then $u = \sigma$ (independent of u^*) while u^*/u is distributed uniformly over $\widehat{\mathbb{G}}$. Then $\alpha = \beta = 0$, otherwise the coefficients of $[0\ \overline{\mathbf{x}}^*\ m\overline{\mathbf{x}}^*]$ and $[0\ \overline{\mathbf{y}}^*\ m\overline{\mathbf{y}}^*]$ are distributed randomly with u^*/u. Further, observe that $[0\ \mathbf{z}\ m\mathbf{z}]$ is never a linear combination of the constraints on the adversary's view, since $\mathbf{z}$ is linearly independent of $\{\mathbf{1}, \overline{\mathbf{x}}^*, \overline{\mathbf{y}}^*\}$. Thus, the coefficient of this row must be zero with non-negligible probability. Substituting, we get that δ must equal σ.

- $\pi = 1$: Then $u^*/u = 1/\sigma$ while u (when appearing alone) is distributed uniformly over $\widehat{\mathbb{G}}$. We must have $\gamma = 0$, otherwise the coefficient of $[0\ \mathbf{1}\ m\mathbf{1}]$ is distributed randomly with u^*. Again, we must have the coefficient of $[0\ \mathbf{z}\ m\mathbf{z}]$ equal to zero with non-negligible probability. Substituting, we get that $\delta = 0$ and $\alpha = \sigma$. Now what remains is a (fixed) linear combination of $[0\ \overline{\mathbf{x}}^*\ m\overline{\mathbf{x}}^*]$ and $[0\ \overline{\mathbf{y}}^*\ m\overline{\mathbf{y}}^*]$. In order for this to be a linear combination of the constraints on the adversary's view, we must have $m = m^* = \mathsf{encode}_{\mathbb{Z}_p}(\mathsf{msg}^*)$; i.e., ζ and ζ^* must yield the same purported plaintext.

- $\pi \notin \{0, 1\}$. Below (when discussing the second strand), we show that the ciphertext would be rejected by Dec with overwhelming probability.

Similarly, we consider which linear combinations of $\mathbf{y}$ allow the ciphertext to decrypt with non-negligible probability. We write:

$$\overline{\mathbf{y}}u = \alpha'\Big((\overline{\mathbf{x}}^* + \mathbf{z})u^*\Big) + \beta'(\overline{\mathbf{y}}^* u^*) + \gamma'\mathbf{1} + \delta'\mathbf{z}$$

Then we examine the two integrity checks on the ciphertext's second strand:

$$[0\ \overline{\mathbf{y}}\ m\overline{\mathbf{y}}] = \frac{\gamma'}{u}[0\ \mathbf{1}\ m\mathbf{1}] + \frac{u^*}{u}\Big(\alpha'[0\ \overline{\mathbf{x}}^*\ m\overline{\mathbf{x}}^*] + \beta'[0\ \overline{\mathbf{y}}^*\ m\overline{\mathbf{y}}^*]\Big) + \left(\alpha'\frac{u^*}{u} + \frac{\delta'}{u}\right)[0\ \mathbf{z}\ m\mathbf{z}]$$

$$[\overline{\mathbf{y}}\ 0\ 0] = \frac{\gamma'}{u}[\mathbf{1}\ 0\ 0] + \frac{u^*}{u}\Big(\alpha'[\overline{\mathbf{x}}^*\ 0\ 0] + \beta'[\overline{\mathbf{y}}^*\ 0\ 0]\Big) + \left(\alpha'\frac{u^*}{u} + \frac{\delta'}{u}\right)[\mathbf{z}\ 0\ 0]$$

Similar to above, we see that the coefficient $\alpha' u^*/u + \delta'/u$ must be zero with non-negligible probability over the randomness in u^*. This is only possible with $\alpha' = \delta' = 0$. We further consider 3 cases of π:

- If $\pi = 0$, then $u = \sigma$ and u^*/u is uniform in $\widehat{\mathbb{G}}$. We see that the coefficient of $[0\ \bar{\mathbf{y}}^*\ m\bar{\mathbf{y}}^*]$ is $\beta' u^*/u$, so we must have $\beta' = 0$. Then $\gamma' \neq 0$, as otherwise $\bar{\mathbf{y}}$ is the all-zeroes vector, and ciphertexts with this property are unconditionally rejected by Dec.
- If $\pi = 1$, then $u^*/u = 1/\sigma$, while u is uniform in $\widehat{\mathbb{G}}$ when appearing alone. We see that the coefficient of $[0\ \mathbf{1}\ 0]$ is γ'/u, so we must have $\gamma' = 0$. Then $\beta' \neq 0$, as otherwise $\bar{\mathbf{y}}$ is the all-zeroes vector and the ciphertext is rejected by Dec.
- $\pi \notin \{0, 1\}$: First, if $m \neq m^*$ (i.e, the ciphertext yields a different purported message than ζ^*), then the row $[0\ \bar{\mathbf{y}}^*\ m\bar{\mathbf{y}}^*]$ is linearly independent of the constraints on the adversary's view. Thus its coefficient in the above expression must be zero noticeably often. This implies $\beta' = 0$. We see that $[0\ \bar{\mathbf{y}}\ m\bar{\mathbf{y}}] = \gamma'/u[0\ \mathbf{1}\ m\mathbf{1}]$. However, u is distributed uniformly in $\widehat{\mathbb{G}}$, so we must have $\gamma' = 0$. This implies $\bar{\mathbf{y}}$ is the all-zeroes vector and thus the ciphertext is rejected by Dec.

  Otherwise, suppose $m = m^*$, i.e, the ciphertext yields the same purported message as ζ^*. Rewriting the above constraints in matrix form and simplifying via $\alpha' = \delta' = 0$, we get that the following condition must hold with non-negligible probability:

$$\begin{aligned}
\begin{bmatrix} \log P_Y \\ \log C_Y \end{bmatrix} &= \begin{bmatrix} 0 & \bar{\mathbf{y}} & m\bar{\mathbf{y}} \\ \bar{\mathbf{y}} & 0 & 0 \end{bmatrix} \begin{bmatrix} G & 0 & 0 \\ 0 & G & 0 \\ 0 & 0 & G \end{bmatrix} \begin{bmatrix} \mathbf{c}^T \\ \mathbf{d}^T \\ \mathbf{e}^T \end{bmatrix} \\
&= \begin{bmatrix} 0 & \gamma'/u & m\gamma'/u & 0 & \beta' u^*/u \\ \gamma/u & 0 & 0 & \beta' u^*/u & 0 \end{bmatrix} \begin{bmatrix} \mathbf{1} & 0 & 0 \\ 0 & \mathbf{1} & 0 \\ 0 & 0 & \mathbf{1} \\ \bar{\mathbf{y}}^* & 0 & 0 \\ 0 & \bar{\mathbf{y}}^* & m^*\bar{\mathbf{y}}^* \end{bmatrix} \begin{bmatrix} G & 0 & 0 \\ 0 & G & 0 \\ 0 & 0 & G \end{bmatrix} \begin{bmatrix} \mathbf{c}^T \\ \mathbf{d}^T \\ \mathbf{e}^T \end{bmatrix} \\
&= \begin{bmatrix} 0 & \gamma'/u & m\gamma'/u & 0 & \beta' u^*/u \\ \gamma/u & 0 & 0 & \beta' u^*/u & 0 \end{bmatrix} \begin{bmatrix} \log C \\ \log D \\ \log E \\ \log C_Y^* \\ \log P_Y^* \end{bmatrix}
\end{aligned}$$

  If we multiply through both of these constraints and substitute according to $u = \sigma(u^*)^\pi$, we get the following two polynomials in u^*, which must be simultaneously zero with non-negligible probability:

$$\begin{aligned}
(\sigma \log P_Y)(u^*)^\pi \quad &- (\beta' \sigma \log P_Y^*)u^* \quad - (\gamma'(\log D + m \log E)) = 0 \\
(\sigma \log C_Y)(u^*)^\pi \quad &- (\beta' \sigma \log C_Y^*)u^* \quad - (\gamma' \log C) = 0
\end{aligned}$$

  Note that these are polynomials in u^* of degree π, and no terms collect together, as $\pi \notin \{0, 1\}$. We now argue that these two polynomials cannot be simultaneously zero with non-negligible probability, unless the ciphertext is degenerate and would be rejected on other grounds:

  – If one of the polynomials is not identically zero but has some coefficient equal to zero, then this polynomial is equivalent to (i.e, has the same roots as) an affine polynomial in a single variable; either u^* or $(u^*)^\pi$ or $(u^*)^{\pi-1}$. Each of these variables is uniform in $\widehat{\mathbb{G}}$, so the equation is satisfied with only negligible probability.

– Otherwise, if the two polynomials have all nonzero coefficients and are identical up to scalar multiplication, then the three pairs of matching coefficients have the same ratios. In particular, we have the following equality (after cancellation):

$$\frac{\log D + m \log E}{\log C} = \frac{\log P_Y^*}{\log C_Y^*}$$

The challenge ciphertext (including the components P_Y^* and C_Y^*) is generated after C, D, E, and m are fixed. It is only with negligible probability over the randomness of AltEnc that P_Y^* an C_Y^* satisfy this condition. Thus it does not affect the outcome of our analysis to condition the entire RCCA experiment on this event not happening.

– If the two polynomials have all nonzero coefficients and are not identical up to scalar multiplication, then some linear combination of them is affine either in the variable u^* or in $(u^*)^\pi$. The two original polynomial equations must have been simultaneously satisfied with non-negligible probability. When both equations hold, then so does any linear combination of the two. But we have demonstrated a linear combination of the equations that is affine on a single variable which is distributed uniformly over $\widehat{\mathbb{G}}$, and thus is satisfied with only negligible probability.

– Otherwise one polynomial is identically zero. It is only with negligible probability that AltEnc generates a ciphertext with $\log C_Y^* = 0$ or $\log P_Y^* = 0$. Thus our analysis may be conditioned on these events not happening. We must have $\beta' = 0$ to make the u^* coefficient zero (since $\sigma \neq 0$ unconditionally). This makes a coefficient in the other polynomial zero as well. This case overlaps with the first case unless both polynomials are in fact identically zero. If both are identically zero, then by similar reasoning, we must have $\gamma' = 0$ ($\log C = 0$ only with negligible probability over the key generation). When $\beta' = \gamma' = 0$, then $\mathbf{y}$ is the all-zeroes vector and the ciphertext is rejected by Dec.

Putting everything together, the only way the adversary can generate a ciphertext which succesfully decrypts with non-negligible probability is:

- $\pi = 0$, $X_i = g_i^{(x+z_i)\sigma}$, and $Y_i = g_i^{y\sigma}$ for some $x \in \mathbb{Z}_p$, $y \in \mathbb{Z}_p^*$. This is exactly the form of case (2) in the description of AltDec. It is straight-forward to see that the remaining integrity constraints succeed if and only if there exists a plaintext msg such that the remaining components of the ciphertext are as given in the description of case (2) of AltDec.

- $\pi = 1$, $\mathbf{X} = (\mathbf{X}^*(\mathbf{Y}^*)^s)^\sigma$, $\mathbf{Y} = (\mathbf{Y}^*)^{t\sigma}$ for some $s \in \mathbb{Z}_p, t \in \mathbb{Z}_p^*$, and also the purported plaintext is the same as in ζ^*. When $\mathbf{X}$ and $\mathbf{Y}$ have this form, we see that the two ciphertexts yield the same purported plaintext if and only if $C_X = C_X^*(C_Y^*)^s$. Furthermore, the integrity constraints on the remaining components succeed if and only if the equalities in case (3) of the description of AltDec hold.

The only two cases where ζ decrypts successfully with non-negligible probability over the remaining randomness in the private key are the two cases in the description of AltDec. $\square$

Lemma 8 *The DSCS scheme has tight decryption.*

PROOF: Observe that the decryption tightness experiment is no different than Phase I of the RCCA experiment. The only time that AltDec outputs $\mathsf{msg} \neq \perp$ in Phase I is actually the case that the given ciphertext is in the range of $\mathsf{Enc}_{PK}(\mathsf{msg})$. By Lemma 7, any other ciphertext given by the adversary will decrypt to $\perp$ with overwhelming probability. □

Lemma 9 *The DSCS scheme is RCCA-secure.*

PROOF: Fix an adversary $\mathcal{A}$ and consider the RCCA experiment against $\mathcal{A}$. First, modify the experiment by generating the challenge ciphertext with AltEnc instead of Enc. By Lemma 5, this does not change the outcome of the experiment more than a negligible amount. Next, replace the Dec oracle with AltDec. By Lemma 7, the two decryption procedures agree in their outputs with overwhelming probability. Furthermore, AltDec can be implemented using only the public key and challenge ciphertext. Thus the entire adversary's view (including decryption oracle responses) is a function of the public key and challenge ciphertext only. By Lemma 4, the challenge ciphertext is distributed independently of the secret bit b in the experiment. The public key is clearly distributed independently as well. Thus, in this modified RCCA experiment, the adversary has no advantage in guessing b. Since the outcome of this modified experiment is negligibly close to the outcome of the original RCCA experiment, the scheme is RCCA secure. □

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 1 *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^A_j'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. □

$\mathcal{S}_1$ and $\mathcal{F}_1$ (Rerandomizability): $\mathcal{F}_1$ is identical to $\mathcal{F}_0$ except that it does not tell the adversary whether a $\textsc{handle-req}$ 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 $(\textsc{handle-req}, \mathsf{sender}, \mathsf{id}, \mathsf{msg})$ 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 $(\textsc{handle-req}, \mathsf{sender}, \mathsf{id}, \mathsf{msg})$ request corresponding to a post/repost request from an honest party, it first checks whether $\mathsf{id} = \mathsf{id}_\perp$ and $\mathsf{msg} = \mathsf{msg}^A_j$ for some j. If so, it generates the handle via $\mathsf{handle}' \leftarrow \mathsf{Rerand}(\mathsf{handle}^A_j)$. Otherwise, it generates the handle via $\mathsf{handle}' \leftarrow \mathsf{Enc}_{PK_{\mathsf{id}}}(\mathsf{msg})$ (as $\mathcal{S}_0$ does).

Claim 2 *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 $\mathcal{Z}$, $\textsc{ideal}^{\mathcal{F}_0}_{\mathcal{S}_0,\mathcal{Z}} = \textsc{ideal}^{\mathcal{F}_1}_{\mathcal{S}_1,\mathcal{Z}}$.*

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 $(\textsc{handle-req}, \mathsf{sender}, \mathsf{repost}, \mathsf{handle})$ command) or as a fresh encryption of the same message under the same key (as $\mathcal{S}_1$ does on receiving a $(\textsc{handle-req}, \mathsf{sender}, \mathsf{id}, \mathsf{msg})$ 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 $\textsc{ideal}^{\mathcal{F}_0}_{\mathcal{S}_0,\mathcal{Z}} = \textsc{ideal}^{\mathcal{F}_1}_{\mathcal{S}_1,\mathcal{Z}}$. □

$\mathcal{S}_2$ 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 $(\textsc{handle-req}, \mathsf{sender}, \mathsf{id}, \mathsf{msg})$ 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 $(\textsc{handle-req}, \mathsf{sender}, \mathsf{id})$ instead (i.e, it does not include msg).

2. When $\mathcal{S}_2$ receives the ith request of the form $(\textsc{handle-req}, \mathsf{sender}, \mathsf{id})$ 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 3 *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 $\mathcal{Z}$, $\textsc{ideal}^{\mathcal{F}_1}_{\mathcal{S}_1,\mathcal{Z}} \approx \textsc{ideal}^{\mathcal{F}_2}_{\mathcal{S}_2,\mathcal{Z}}$.*

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 $(\textsc{handle-req}, \mathsf{sender}, \mathsf{id}, \mathsf{msg})$, 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. □

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 $g_1, \ldots, g_5$ 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, Michael Loui, and anonymous referees for their helpful feedback on earlier drafts of this manuscript.

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/`.