Rerandomizable RCCA Encryption

Manoj Prabhakaran and Mike Rosulek

Department of Computer Science, University of Illinois, Urbana-Champaign.
{mmp,rosulek}@uiuc.edu

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

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 Sect. 7.1, 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 Sect. 7.1 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 [26] 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, mix-net applications tend to also require a "receiver-anonymity" property (see Sect. 7.1) 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.

Due to lack of space, we have omitted many details in this paper. We refer the readers to the online version for a detailed presentation [24].

2 Definitions

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

2.1 Encryption and Security Definitions

In this section we give the syntax of a perfectly rerandomizable encryption scheme, and then state our security requirements, which are formulated as indistinguishability experiments. Later, we justify these indistinguishability-based definitions by showing that any scheme which satisfies them is a secure realization of a powerful functionality in the UC security model, which we define in Sect. 5.

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

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

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

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

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

In other words, decryption is the inverse of encryption, and ciphertexts can be labeled "honestly generated" for at most one honestly generated key pair. We require that rerandomizing an honestly generated ciphertext induces the same distribution as an independent encryption of the same message, while the only guarantee for an adversarially generated ciphertext is that rerandomization preserves the value of its decryption (under all private keys).

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

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

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

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

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

Observe that when combined with correctness property (2), this implies that an adversary cannot generate a ciphertext which successfully decrypts under more than one honestly generated key. Such a property is useful in achieving a more robust definition of our UC functionality $\mathcal{F}_{\text{RMP}}$ in Sect. 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*

[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.

prime). Given a sequence of primes $(q, 2q+1, 4q+3)$, the two groups $\widehat{\mathbb{G}} = \mathbb{QR}^*_{2q+1}$ and $\mathbb{G} = \mathbb{QR}^*_{4q+3}$ satisfy the needs of our construction. A sequence of primes of this form is called a *Cunningham chain* (of the first kind) of length 3 (see [2]). Such Cunningham chains are known to exist having q as large as 20,000 bits. It is conjectured that there are infinitely many such chains.

3 Motivating the Double-strand Construction

Conceptually, the crucial enabling idea in our construction is that of using two "strands" of ciphertexts which can be recombined with each other for rerandomization without changing the encrypted value. To motivate this idea, we sketch the rerandomizable scheme of Golle et al. [17], which is based on the El Gamal scheme and secure against chosen plaintext attacks.

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

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

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

Cramer-Shoup encryption. The Cramer-Shoup scheme [9] uses a group $\mathbb{G}$ of prime order p in which the DDH assumption is believed to hold. The private key is $b_1, b_2, c_1, c_2, d_1, d_2 \in \mathbb{Z}_p$ and the public key is $g_1, g_2 \in \mathbb{G}$, $B = \prod_{i=1}^2 g_i^{b_i}$, $C = \prod_{i=1}^2 g_i^{c_i}$, and $D = \prod_{i=1}^2 g_i^{d_i}$.

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

In our scheme the ciphertext will contain two "strands," each one similar to a Cramer-Shoup ciphertext, allowing rerandomization as in the example above. However, instead of pairs we require 5-tuples of g_i, b_i, c_i, d_i (i.e., for $i = 1, \ldots, 5$). 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 $\widehat{g}_1, \widehat{g}_2, \widehat{g}_3$ from $\widehat{\mathbb{G}}$, and random $\mathbf{a} = (a_1, a_2, a_3)$ from $(\mathbb{Z}_q)^3$. The private key is $\mathbf{a}$. The public key consists of $\widehat{g}_1$, $\widehat{g}_2$, $\widehat{g}_3$, and $A = \prod_{j=1}^{3} \widehat{g}_j^{a_j}$.

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

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

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

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

Rerandomization: $\mathsf{MRerand}(U = (\mathbf{V}, A_V, \mathbf{W}, A_W))$: The only randomness used in MEnc is the choice of v and w in $\widehat{\mathbb{G}}$. We can rerandomize both of these quantities by choosing random $s, t \in \mathbb{Z}_q$ and outputting the following ciphertext:

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

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

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

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

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

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

4.2 Double-strand Cramer-Shoup Encryption Scheme

Now we give our main construction: a rerandomizable, RCCA-secure encryption scheme called the "Double-strand Cramer-Shoup" (DSCS) scheme. At the high level, it has two Cramer-Shoup encryption strands, one carrying the message, and the other to help rerandomize it. But unlike in the Cramer-Shoup scheme, we need to allow rerandomization, and so we do not use a prefix of the ciphertext itself in ensuring consistency; instead we use a direct encoding of the plaintext.

Further, we must prevent the possibility of mixing together strands from two *different* encryptions of the same message (say, in the manner in which rerandomizability allows two strands to be mixed together) to obtain a ciphertext which successfully decrypts, which would yield a successful adversarial strategy in our security experiments. For this, we correlate the two strands of a ciphertext with shared random masks. These masks are random exponents which are separately encrypted using the malleable DSME scheme described above (so that they may be hidden from everyone but the designated recipient, but also be rerandomized via the DSME scheme's homomorphic operation).

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.3 Complexity

The complexities of the DSCS scheme are summarized in Table 1 and Table 2.[2] Clearly our scheme is much less efficient than the Cramer-Shoup encryption scheme. On the other hand, it is much more efficient than the only previously proposed rerandomizable (weak) RCCA-secure scheme [18], which used $O(k)$ group elements to encode a k-bit message (or in other words, to be able to use the group itself as the message space, it used $O(\log p)$ group elements). In fact, if we restrict ourselves to weak RCCA security (as defined in [18]) and a computational version of rerandomizability, our construction can be simplified to have only 10 group elements (we omit the details of that construction in this paper).

Table 1. Number of elements

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

Table 2. Group operations performed

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

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, 25] 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

[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.

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 $\mathsf{register}$ from a party sender, the functionality sends $(\textsc{id-req}, \mathsf{sender})$ to the adversary, and expects in response an identifier string id.[3] If the string received in response has been already used, ignore the request. Otherwise respond to sender with the string id, and also send a message $(\textsc{id-announce}, \mathsf{id})$ to all other parties.

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

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

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

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

Message reposting: On receiving a message $(\mathsf{repost}, \mathsf{handle})$ from a party sender, the functionality behaves as follows: If handle is not recorded internally, ignore the request.

Otherwise, suppose $(\mathsf{handle}, \mathsf{sender}', \mathsf{id}, \mathsf{msg})$ is recorded internally. If id is registered to an uncorrupted party, send $(\textsc{handle-req}, \mathsf{sender}, \mathsf{id})$ to the adversary;

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

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

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

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 Sect. 4 achieves the security definitions defined in Sect. 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 Sect. 5. For the complete proofs of these results, we refer the reader to the full version of this paper [24].

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

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

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

Alternate encryption. First, we would like to argue that the ciphertexts hide the message and the public key used in the encryption. For this we describe an alternate encryption procedure AltEnc. AltEnc actually uses the *private key* to generate ciphertexts. In short, instead of using $\{X_i = g_i^x\}$ and $\{Y_i = g_i^y\}$, AltEnc picks random group elements for these ciphertext components, then uses the private key to generate the other components according to the quantities which are computed by Dec.

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

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

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

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

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

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

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

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

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

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. $\triangleleft$

7 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 first describe a few useful extensions which are easily achieved, and then discuss extending the notion of receiver-anonymity to rerandomizable RCCA encryption schemes.

Replay-test. In some applications, it is convenient if the recipient of a ciphertext is 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.

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 bookkeeping 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.

7.1 Anonymity

Bellare et al. [3] introduced the notion of receiver-anonymity (or key-privacy) for CPA and CCA secure encryption schemes, which we extend to the RCCA setting. In a system with multiple users, rerandomizability of ciphertexts, without receiver-anonymity, may not be satisfactory. 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.

RCCA receiver-anonymity. An encryption scheme is said to be *RCCA 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 $\Pr[b' = b] - \frac{1}{2}$.

Our scheme does not achieve this definition of anonymity. We leave it as an interesting open problem.

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 reveal the handle's recipient in a HANDLE-ANNOUNCE broadcast, nor in the HANDLE-REQ messages it sends to the adversary (unless the handle's recipient is corrupted).

In proving this, compared to the proof of Theorem 2, the only change in the simulator for this modified functionality is that it uses a "dummy" public key to generate all simulated ciphertexts addressed to honest recipients, instead of using the correct key.

8 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 consider whether such a hybrid scheme can be devised in a rerandomizable manner. To achieve perfect rerandomization, the public-key scheme would have to be malleable (to rerandomize the private key), and the private-key scheme should allow reencryption which changes the key accordingly.

Second, can rerandomizable RCCA-secure schemes be constructed which are also RCCA receiver-anonymous? As mentioned earlier, many applications require not only rerandomizability of the ciphertexts, but also receiver anonymity.

Third, can CCA-like tradeoffs be defined for encryption schemes with more sophisticated homomorphic features? In this work, we give a tight tradeoff, allowing malleability via the identity function in the strongest manner, while prohibiting all other kinds of malleability. How can one define (and achieve) a similar tradeoff for, say, malleability via multiplication?

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.

Acknowledgments

We would like to acknowledge useful discussions with Rui Xue about the work in [10]. We also thank Ran Canetti, Michael Loui, and the 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 M. Rosulek. Rerandomizable rcca encryption. Cryptology ePrint Archive, Report 2007/119, 2007. `http://eprint.iacr.org/`.
25. M. Prabhakaran and A. Sahai. New notions of security: achieving universal composability without trusted setup. In *STOC*, pages 242–251. ACM, 2004.
26. V. Shoup. A proposal for an iso standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001. `http://eprint.iacr.org/`.