Rerandomizable and Replayable Adaptive Chosen Ciphertext Attack Secure Cryptosystems

Jens Groth[1,2]

[1] BRICS*, University of Aarhus, Ny Munkegade bd. 540, 8000 Århus C, Denmark
[2] Cryptomathic A/S**, Jægergårdsgade 118, 8000 Århus C, Denmark
jg@brics.dk

Abstract. Recently Canetti, Krawczyk and Nielsen defined the notion of replayable adaptive chosen ciphertext attack (RCCA) secure encryption. Essentially a cryptosystem that is RCCA secure has full CCA2 security except for the little detail that it may be possible to modify a ciphertext into another ciphertext containing the *same* plaintext.
We investigate the possibility of *perfectly* replayable RCCA secure encryption. By this, we mean that anybody can convert a ciphertext y with plaintext m into a different ciphertext y' that is distributed identically to a fresh encryption of m. We propose such a rerandomizable cryptosystem, which is secure against semi-generic adversaries.
We also define a weak form of RCCA (WRCCA) security. For this notion we provide a construction (inspired by Cramer and Shoup's CCA2 secure cryptosystems) that is both rerandomizable and provably WRCCA secure. We use it as a building block in our conjectured RCCA secure cryptosystem.

1 Introduction

Security against adaptive chosen ciphertext attacks (CCA2) has become the golden security standard for public-key cryptosystems. Dolev, Dwork and Naor gave the first construction based on standard primitives in [10] and subsequent work [13, 7, 8, 11] includes practical constructions based on a variety of assumptions. However, an unfortunate side effect of the strong security definition is the exclusion of certain cryptosystems that intuitively are secure. Consider for instance a cryptosystem that expands a CCA2 secure cryptosystem with a single bit, which is ignored in decryption. By flipping this bit it is easy to create a new encryption of the same plaintext and therefore the new cryptosystem is not CCA2 secure even though the message is protected by the same encryption. A few proposals for redefining CCA2 security to cover such cryptosystems were presented in [15, 1], but other natural examples that intuitively are "CCA2" secure but do not satisfy these definitions exist. We believe that Canetti, Krawczyk

* Basic Research in Computer Science (www.brics.dk), funded by the Danish National Research Foundation.
** www.cryptomathic.com

and Nielsen have solved this problem satisfactorily in [6] by defining replayable adaptive chosen ciphertext attack (RCCA) security.[3]

RCCA security essentially is the same as CCA2 security, except no guarantees are given against adversaries that just try to modify a ciphertext into a new ciphertext with the same plaintext. CCA2 security implies RCCA security, but not the other way around. We could hope that a weaker definition might give rise to more efficient constructions but this has so far not been the case. On the other hand, it is a proven fact that given RCCA secure encryption we can construct CCA2 secure cryptosystems. We refer the reader to [6] for several other arguments for being interested in RCCA secure encryption.

The question we seek to answer in this paper is to what extend it may be possible to maul an RCCA secure cryptosystem. We have the ambitious goal of finding a cryptosystem, which is RCCA secure and has perfect rerandomization, i.e., an efficient algorithm for converting an encryption y of plaintext m into a ciphertext y' that is perfectly indistinguishable from a fresh encryption of m.

Besides the theoretical perspective, we believe such cryptosystems may have practical applications. Consider for instance an anonymization protocol where in the end some party receives the encrypted messages and acts upon them, for instance a voting protocol based on mix-nets.[4] Here, we may want the ability to rerandomize ciphertexts in order to anonymize them. On the other hand, we may imagine an adversary that can inject ciphertexts into the anonymization protocol and therefore gets access to an adaptive chosen ciphertext attack. Rerandomizable RCCA secure encryption may be just the tool that gives us the better of two worlds.

Constructing a rerandomizable RCCA secure cryptosystem is a hard problem, and is posed as an interesting open problem in [6]. The construction has to be almost CCA2 secure and at the same time have enough mathematical structure to be rerandomizable. In particular, it seems like popular tools for building CCA2 secure encryption such as random oracles and one-time signatures cannot be used.

In this paper, we start out by defining a weaker notion of replayable security called WRCCA security. This notion is stronger than IND-CCA1 but weaker than RCCA security. It turns out that rerandomizable WRCCA secure cryptosystem can be constructed under well-known intractability assumptions.

By choosing an appropriate group to work in, we get a rerandomizable WRCCA secure cryptosystem that may be extended in a way that gives rise to a new rerandomizable cryptosystem. We believe this new cryptosystem is RCCA secure. Since it is an extension of a WRCCA secure cryptosystem, it is provably WRCCA secure. In itself, WRCCA security does not guarantee RCCA security though. We give an additional security argument by proving that a semi-generic adversary cannot break the scheme, where semi-generic means that it can only perform standard group operations on parts of the ciphertext.

[3] Independently we came up with exactly the same definition of RCCA security.

[4] Duplication of votes must be avoided, for instance by inserting a nonce in the plaintext and discarding extra pairs of the same vote and nonce.

2 Notions of Replayable Security

Notation. All algorithms and adversaries are modeled as probabilistic polynomial time (possibly interactive) Turing machines. Our proofs hold for both uniform and non-uniform adversaries.

We assume that all algorithms and adversaries get a security parameter as input. We write $p_1 \approx p_2$ if p_1 and p_2 are functions of the security parameter such that $|p_1 - p_2|$ is a negligible function in the security parameter. A function that is not negligible is said to be noticeable.

Definitions. We define a public-key cryptosystem in the usual way. The decryption function outputs `invalid` when a ciphertext does not decrypt properly to a plaintext.

Definition 1 (RCCA security). *A cryptosystem* (K, E, D) *is* RCCA *secure if for any adversary* $\mathcal{A}$ *it is the case that*

$$\begin{aligned} &P[(pk, sk) \leftarrow K(); (m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_1}(pk); y \leftarrow E_{pk}(m_0) : \mathcal{A}^{\mathcal{O}_2}(y) = 1] \\ \approx\ &P[(pk, sk) \leftarrow K(); (m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_1}(pk); y \leftarrow E_{pk}(m_1) : \mathcal{A}^{\mathcal{O}_2}(y) = 1], \end{aligned}$$

where

- $\mathcal{O}_1$ *works like* D_{sk}.
- $\mathcal{O}_2$ *works like* D_{sk} *except when the plaintext is* m_0 *or* m_1. *On* m_0 *or* m_1 *the oracle outputs* `test`.

Definition 2 (WRCCA security). *A cryptosystem* (K, E, D) *is* WRCCA *secure if for any adversary* $\mathcal{A}$ *it is the case that*

$$\begin{aligned} &P[(pk, sk) \leftarrow K(); (m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_1}(pk); y \leftarrow E_{pk}(m_0) : \mathcal{A}^{\mathcal{O}_2}(y) = 1] \\ \approx\ &P[(pk, sk) \leftarrow K(); (m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_1}(pk); y \leftarrow E_{pk}(m_1) : \mathcal{A}^{\mathcal{O}_2}(y) = 1], \end{aligned}$$

where

- $\mathcal{O}_1$ *works like* D_{sk}.
- $\mathcal{O}_2$ *works like* D_{sk} *except when the plaintext is* m_0 *or* m_1. *On* m_0 *or* m_1 *the oracle outputs* `invalid`.

Let us illustrate the two types of security with the following example. We assume that we are operating a Swiss bank, and account holders can send anonymous messages to us containing a password, the banking operation they want to perform and perhaps a counter to prevent replay attacks. We do not reply to these messages, but if the password is valid and the counter has not been used before, we perform the banking operation. Suppose a client of ours sends a ciphertext containing some banking operation he wants to perform and he is being wiretapped by somebody who wants to know which operation he carried out. Now the eavesdroppers may open an account with us, send ciphertexts to us, and see what happens with the money in their account. This means that they do have access to a chosen ciphertext attack. However, since they do not know our client's password they cannot probe the system with banking operations on his account. WRCCA security is therefore sufficient to guarantee that the eavesdroppers do not learn anything about the banking operation he performed.

Suppose we change the protocol to be user-friendlier: we send back one type of error message if a banking operation has already been executed and another type of error message if a ciphertext is invalid. Now the eavesdroppers have access to a stronger attack and we need the cryptosystem to be RCCA secure.

In general WRCCA secure cryptosystems are only appropriate in protocols where the adversary does not learn whether an injected ciphertext is valid or invalid. Often this is not the case, consider for instance Bleichenbacher's attack on the PKCS #1 protocol [4].

Other types of security. Bellare and Sahai prove in [3] that non-malleability is equivalent to indistinguishability under parallel attack. By a parallel attack we mean the adversary has access to an oracle $\mathcal{O}_2$ that decrypts any number of ciphertexts but may be invoked only once. This definition makes sense both without and with access to $\mathcal{O}_1$. They call the security notions IND-PA0 and IND-PA1. By modifying $\mathcal{O}_2$ such that it can decrypt one vector of ciphertexts and will respond with respectively `test` and `invalid` on m_0 and m_1 we get four other security notions IND-RPA0, IND-RPA1, IND-WRPA0 and IND-WRPA1.[5]

Relationship between security notions. Figure 1 in Appendix A describes completely the relationship between all the security notions. For our purposes the interesting thing to note is that CCA2 security implies RCCA security, which implies WRCCA security, which in turn implies IND-CCA1 security. On the other hand all these notions are separate; IND-CCA1 does not imply WRCCA, WRCCA does not imply RCCA, and RCCA does not imply CCA2 security.

3 Rerandomizable Weak RCCA Secure Encryption

In this section, we describe a rerandomizable WRCCA secure cryptosystem. The idea bears some resemblance to Cramer-Shoup's DDH based CCA2 secure cryptosystem [7]. In their scheme a ciphertext looks like this $(u_L = g_L^r, u_R = g_R^r, v = h^r m, \alpha = (cd^{\text{hash}(u_L,u_R,v)})^r)$.[6] If we have $h = g_L^{x_L} = g_R^{x_R}$, then α is a designated verifier zero-knowledge proof that both decryption with x_L and x_R will give the same plaintext. In the security proof, they use a hybrid argument where at one point we actually have that x_L and x_R would give different decryptions of the challenge ciphertext. At this point we simulate the designated verifier proof α. Raising d to $\text{hash}(u_1, u_2, v)$ ensures that the simulation only works when we are

[5] These forms of non-malleability should not be confused with the NM-RCCA notion in [6].

[6] L = left, R = right.

using the actual challenge ciphertext, i.e., the designated verifier proof is simulation sound. Therefore, the adversary cannot fake proofs in the oracle queries, except if it copies the challenge ciphertext directly.

In our case we wish to allow rerandomization, provided the same plaintext is used. Therefore, we wish to ensure that the adversary in the security proof cannot fake the designated verifier proof unless the same plaintext as in the challenge is used. For this reason we make a designated verifier proof that has the form $(cd^{\mathrm{hash}(m)})^r$. The cryptosystem and the proof do become more involved than standard Cramer-Shoup encryption. One of the reasons for this is that we have to take specifically into account in the hybrid argument how to shift from using $\mathrm{hash}(m_0)$ and $\mathrm{hash}(m_1)$, where in the Cramer-Shoup scheme this is always computed as $\mathrm{hash}(u_L, u_r, v)$.

Another problem with using the Cramer-Shoup cryptosystem is that even with this new type of proof we cannot rerandomize it. To solve this problem we instead encrypt the message one bit at a time as $g_{L,i}^r, g_{R,i}^r, h^{m_i r}$ where $m_i = \pm 1$. Now we can rerandomize by choosing a random exponent and then raise all parts of the ciphertext to this exponent.

Key Generation: Choose a collision-free hash-function $\mathrm{h} : \{-1,1\}^k \to \{0,1\}^t$.
Choose a cyclic group G of order n where the DDH problem is hard.[7] The order n may be a prime or a composite. We demand that the smallest prime factor of n is larger than 2^t.
Select at random elements $h_1, \ldots, h_k \in G$.
Choose $x_{L,1}, x_{R,1}, \ldots, x_{L,k}, x_{R,k}$ at random from $\mathbb{Z}_n$.
Set $g_{L,1} = h_1^{x_{L,1}^{-1}}, g_{R,1} = h_1^{x_{R,1}^{-1}}, \ldots, g_{L,k} = h_k^{x_{L,k}^{-1}}, g_{R,k} = h_k^{x_{R,k}^{-1}}$.
Select at random $k_{L,1}, k_{R,1}, \ldots, k_{L,k}, k_{R,k} \in \mathbb{Z}_n$ and $l_{L,1}, l_{R,1}, \ldots, l_{L,k}, l_{R,k} \in \mathbb{Z}_n$.
Set

$$c = \prod_{i=1}^{k} g_{L,i}^{k_{L,i}} g_{R,i}^{k_{R,i}} \qquad \text{and} \qquad d = \prod_{i=1}^{k} g_{L,i}^{l_{L,i}} g_{R,i}^{l_{R,i}} .$$

$pk = (g_{L,1}, g_{R,1}, h_1, \ldots, g_{L,k}, g_{R,k}, h_k, c, d, \mathrm{h})$.
$sk = (pk, x_{L,1}, \ldots, x_{R,k}, k_{L,1}, \ldots, k_{R,k}, l_{L,1}, \ldots, l_{R,k})$.

Encryption: Given input $m = m_1 \ldots m_k \in \{-1,1\}^k$.[8]
$E_{pk}(m; r) = (g_{L,1}^r, g_{R,1}^r, h_1^{m_1 r}, \ldots, g_{L,k}^r, g_{R,k}^r, h_k^{m_k r}, (cd^{\mathrm{h}(m)})^r)$.

Decryption: Given ciphertext $y = (u_{L,1}, u_{R,1}, v_1, \ldots, u_{L,k}, u_{R,k}, v_k, \alpha)$.
Check that all elements belong to G.
Compute for all i the $m_i \in \{-1,1\}$ that satisfies $v_i = u_{L,i}^{m_i x_{L,i}} = u_{R,i}^{m_i x_{R,i}}$.
Set $m = m_1 \ldots m_k$.
Check that

$$\alpha = \prod_{i=1}^{k} u_{L,i}^{k_{L,i} + \mathrm{h}(m) l_{L,i}} u_{R,i}^{k_{R,i} + \mathrm{h}(m) l_{R,i}} .$$

If everything works out return m, otherwise return `invalid`.

[7] Membership of G should be easy to check and it should be easy to pick a generator for the group.

[8] Using $\{-1,1\}$ instead of $\{0,1\}$ makes notation a little less cumbersome.

Rerandomization: Given ciphertext $(u_{L,1}, u_{R,1}, v_1, \ldots, u_{L,k}, u_{R,k}, v_k, \alpha)$.
Select at random $r' \in \mathbb{Z}_n^*$. Return $(u_{L,1}^{r'}, \ldots, v_k^{r'}, \alpha^{r'})$.

It is easy to see that this is a public key cryptosystem with perfect rerandomization. For security, we have the following theorem.

Theorem 1. *The cryptosystem is* WRCCA *secure provided the* DDH *assumption holds for G and the hash-function is collision-free.*

Proof. Consider the experiments in the definition of WRCCA security. The only difference is in the challenge given to the adversary. We define several probabilities $p_0, \ldots, p_6$ of $\mathcal{A}$ outputting 1 given different challenges. I.e., we set $p_i = \Pr[(pk, sk) \leftarrow K(); (m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_1}(pk); y \leftarrow \mathrm{Chal_i} : \mathcal{A}^{\mathcal{O}_2}(y) = 1]$, where $\mathrm{Chal_i}$ for the various probabilities returns the following.

p_0: $(g_{L,1}^r, g_{R,1}^r, h_1^{m_{01}r}, \ldots, g_{L,k}^r, g_{R,k}^r, h_k^{m_{0k}r}, (cd^{\mathrm{h}(m_0)})^r)$.
p_1: $(\ldots, g_{L,i}^r, g_{R,i}^{m_{0i}m_{0i}r}, h_i^{m_{0i}r}, \ldots, \prod_{i=1}^k u_{L,i}^{k_{L,i}+l_{L,i}\mathrm{h}(m_0)} u_{R,i}^{k_{R,i}+l_{R,i}\mathrm{h}(m_0)})$.
p_2: $(\ldots, g_{L,i}^r, g_{R,i}^{m_{0i}m_{1i}r}, h_i^{m_{0i}r}, \ldots, \prod_{i=1}^k u_{L,i}^{k_{L,i}+l_{L,i}\mathrm{h}(m_0)} u_{R,i}^{k_{R,i}+l_{R,i}\mathrm{h}(m_0)})$.
p_3: $(\ldots, g_{L,i}^r, g_{R,i}^{m_{0i}m_{1i}r}, h_i^{m_{0i}r}, \ldots, \prod_{i=1}^k u_{L,i}^{k_{L,i}+l_{L,i}\mathrm{h}(m_1)} u_{R,i}^{k_{R,i}+l_{R,i}\mathrm{h}(m_1)})$.
p_4: $(\ldots, g_{L,i}^{m_{0i}m_{1i}r}, g_{R,i}^r, h_i^{m_{1i}r}, \ldots, \prod_{i=1}^k u_{L,i}^{k_{L,i}+l_{L,i}\mathrm{h}(m_1)} u_{R,i}^{k_{R,i}+l_{R,i}\mathrm{h}(m_1)})$.
p_5: $(\ldots, g_{L,i}^{m_{1i}m_{1i}r}, g_{R,i}^r, h_i^{m_{1i}r}, \ldots, \prod_{i=1}^k u_{L,i}^{k_{L,i}+l_{L,i}\mathrm{h}(m_1)} u_{R,i}^{k_{R,i}+l_{R,i}\mathrm{h}(m_1)})$.
p_6: $(g_{L,1}^r, g_{R,1}^r, h_1^{m_{11}r}, \ldots, g_{L,k}^r, g_{R,k}^r, h_k^{m_{1k}r}, (cd^{\mathrm{h}(m_1)})^r)$.

p_0 and p_6 are the probabilities for the definition of WRCCA security. We must therefore prove that $p_0 \approx p_6$. To accomplish this we prove that $p_0 \approx p_1, \ldots, p_5 \approx p_6$.

The proof goes as follows. It is easy to see that $p_0 = p_1$. In p_1 we simulate the proof α, however, the simulation is perfect. $p_1 \approx p_2$ follows from Claim 11. $p_2 \approx p_3$ follows from Claim 13. $p_3 \approx p_4$ follows from Claim 14. $p_4 \approx p_5$ follows by a completely similar proof as for Claim 11. $p_5 = p_6$ is seen by inspection since again the only difference between them is a perfectly simulated proof α.

Claim 11. $p_1 \approx p_2$.

Proof. Assume for contradiction WLOG that p_1 is noticeably larger than p_2. We transform $\mathcal{A}$ into an adversary $\mathcal{B}$ that can break the following hard problem.

Hard problem. We select at random $h_1, \ldots, h_k$ and $g_{R,1}, \ldots, g_{R,k}$ from G. $\mathcal{B}$ sees these and is allowed to choose $m_0, m_1 \in \{-1, 1\}^k$. Subsequently we choose at random $r \in \mathbb{Z}_n^*$. We give either $(g_{R,1}^{m_{01}m_{01}r}, h_1^{m_{01}r}, \ldots, g_{R,k}^{m_{0k}m_{0k}r}, h_k^{m_{0k}r})$ or $(g_{R,1}^{m_{01}m_{11}r}, h_1^{m_{01}r}, \ldots, g_{R,k}^{m_{0k}m_{1k}r}, h_k^{m_{0k}r})$ to $\mathcal{B}$. $\mathcal{B}$ must now output a bit. We consider $\mathcal{B}$ successful if it can distinguish the two tuples.

The hardness of the problem relies on the DDH assumption. Suppose $\mathcal{B}$ can distinguish the two types of challenge. By a hybrid argument there is an index i and a bit b such that $\mathcal{B}$ can distinguish $(g_{R,1}^{m_{01}m_{01}r}, h_1^{m_{01}r}, \ldots, g_{R,i}^{m_{0i}m_{bi}r}, h_i^{m_{0i}r}, \ldots, g_{R,k}^{m_{0k}m_{1k}r}, h_k^{m_{0k}r})$ and

$(g_{R,1}^{m_{01}m_{01}r}, h_1^{m_{01}r}, \ldots, y, h_i^{m_{0i}r}, \ldots, g_{R,k}^{m_{0k}m_{1k}r}, h_k^{m_{0k}r})$, where y is chosen at random.

Consider now a randomly chosen DDH challenge (g, h, z, h^r) where we must determine whether $z = g^r$ or z is chosen at random from G. We set $h_i = h$ and $g_{R,i} = g$. For all $j \neq i$ we select at random $x_j, x_{R,j}$ and compute $h_j = h^{x_j}$ and $g_{R,j} = h_j^{x_{R,j}^{-1}}$. We give $g_{R,1}, h_1, \ldots, g_{R,k}, h_k$ to $\mathcal{B}$ and get the messages m_0 and m_1. Then we give $\mathcal{B}$ the challenge $(g_{R,1}^{m_{01}m_{01}r}, h_1^{m_{01}r}, \ldots, z^{m_{0i}m_{bi}}, h^{m_{0i}r}, \ldots, g_{R,k}^{m_{0k}m_{1k}r}, h_k^{m_{0k}r})$. We have now converted $\mathcal{B}$ into a DDH distinguisher.

The algorithm $\mathcal{B}$. We describe $\mathcal{B}$. In its first invocation it gets the input $g_{R,1}, \ldots, g_{R,k}, h_1, \ldots, h_k$. It selects at random $x_{L,1}, \ldots, x_{L,k} \in \mathbb{Z}_n$. It sets $g_{L,1} = h_1^{x_{L,1}^{-1}}, \ldots, g_{L,k} = h_k^{x_{L,k}^{-1}}$. After this it selects $k_{L,1}, \ldots, l_{R,k}$ and sets

$$c = \prod_{i=1}^{k} g_{L,i}^{k_{L,i}} g_{R,i}^{k_{R,i}} \qquad \text{and} \qquad d = \prod_{i=1}^{k} g_{L,i}^{l_{L,i}} g_{R,i}^{l_{R,i}}.$$

$\mathcal{B}$ now has something that looks perfectly like a public key for our cryptosystem. It does not know the full secret key since it does not know the discrete logarithms $x_{R,1}, \ldots, x_{R,k}$.

$\mathcal{B}$ runs the algorithm for $\mathcal{A}$ on the public key given above. Whenever $\mathcal{A}$ queries the oracle $\mathcal{O}_1$ then $\mathcal{B}$ answers the query by extracting a message m using its knowledge of $x_{L,1}, \ldots, x_{L,k}$. It then checks that $\alpha = \prod_{i=1}^{k} u_{L,i}^{k_{L,i}+l_{L,i}\mathrm{h}(m)} u_{R,i}^{k_{R,i}+l_{R,i}\mathrm{h}(m)}$. It returns m if everything works out OK. We can see this as $\mathcal{A}$ getting its oracle queries answered by a left-oracle $\mathcal{O}_1^L$. From Claim 12 we see that with overwhelming probability these answers correspond to the answers the real oracle $\mathcal{O}_1$ would make. $\mathcal{A}$ returns two messages m_0 and m_1. This is the output of $\mathcal{B}$ after its first invocation.

A challenge $(u_{R,1}, v_1, \ldots, u_{R,k}, v_k)$ for the hard problem is now selected and given to $\mathcal{B}$. $\mathcal{B}$ converts this challenge into what looks like a ciphertext by setting $u_{L,1} = v_1^{m_{01}x_{L,1}^{-1}}, \ldots, u_{L,k} = v_k^{m_{0k}x_{L,k}^{-1}}$ and $\alpha = \prod_{i=1}^{k} u_{L,i}^{k_{L,i}+l_{L,i}\mathrm{h}(m_0)} u_{R,i}^{k_{R,i}+l_{R,i}\mathrm{h}(m_0)}$. In case we have $u_{R,1} = g_{R,1}^{m_{01}m_{01}r}, v_1 = h_1^{m_{01}r}, \ldots, u_{R,k} = g_{R,k}^{m_{0k}m_{0k}r}, v_k = h_k^{m_{0k}r}$ then the ciphertext will be as in the challenge in p_1. In case we have $u_{R,1} = g_{R,1}^{m_{01}m_{11}r}, v_1 = h_1^{m_{01}r}, \ldots, u_{R,k} = g_{R,k}^{m_{0k}m_{1k}r}, v_k = h_k^{m_{0k}r}$ then we have a ciphertext on the form of the challenge in p_2.

$\mathcal{B}$ now runs $\mathcal{A}$ on this ciphertext. It answers queries in the same way as before, i.e., using $\mathcal{O}_2^L$ that decrypts using $x_{L,1}, \ldots, x_{L,k}$ and then checks the proof. Again using Claim 12 we get that the oracle queries are answered as the real oracle $\mathcal{O}_2$ with access to the discrete logarithms $x_{R,1}, \ldots, x_{R,k}$ would do. In the end, $\mathcal{A}$ answers with a bit. $\mathcal{B}$ uses this bit as its output.

Depending on the challenge, we have either probability p_1 for $\mathcal{B}$ outputting 1 or probability p_2 for $\mathcal{B}$ outputting 1. If the two probabilities are noticeably different, this means that we have created a distinguisher for the hard problem and thereby broken the DDH assumption.

Claim 12. It is infeasible for $\mathcal{A}$ to find a ciphertext y' with proof α' that gets answered differently by the real oracles $\mathcal{O}_1, \mathcal{O}_2$ and modified oracles $\mathcal{O}_1^L, \mathcal{O}_2^L$ that only left-decrypt, even if $\mathcal{A}$ sees a fake ciphertext y as the challenge in p_2 with simulated proof α.

Proof. Consider the difficult case, namely finding a query that $\mathcal{O}_2$ and $\mathcal{O}_2^L$ answer differently. The information available to $\mathcal{A}$ about $k_{L,1}, \dots, l_{R,k}$ comes from c, d and the fake ciphertext y. If we compute the discrete logarithms with respect to some base g for these elements, we get the following system of linear equations in $\mathbb{Z}_n$ to be satisfied, where α' is the "proof" in the newly created ciphertext.

$$\begin{pmatrix} 1 & 1 & \cdots & 0 & 0 & \cdots \\ 0 & 0 & \cdots & 1 & 1 & \cdots \\ r & r\delta_1 & \cdots & r\mathrm{h}(m_0) & r\delta_1\mathrm{h}(m_0) & \cdots \\ r_{L,1} & r_{R,1} & \cdots & r_{L,1}\mathrm{h}(m) & r_{R,1}\mathrm{h}(m) & \cdots \end{pmatrix} \begin{pmatrix} \log(g_{L,1})k_{L,1} \\ \log(g_{R,1})k_{R,1} \\ \vdots \\ \log(g_{L,1})l_{L,1} \\ \log(g_{R,1})l_{R,1} \\ \vdots \end{pmatrix} = \begin{pmatrix} \log(c) \\ \log(d) \\ \log(\alpha) \\ \log(\alpha') \end{pmatrix},$$

where we define $\delta_i = m_{0i}m_{1i}$.

Since $k_{L,1}, \dots, l_{R,k}$ are unknown and randomly chosen the only chance for the proof α' to be correct is if the last row is a linear combination of the first three rows. Already at this point we can therefore see that we must have some r_L such that for all i we have $r_L = r_{L,i}$. Reducing the matrix we get

$$\begin{pmatrix} 1 & 1 & \cdots 0 & 0 & \cdots \\ 0 & 0 & \cdots 1 & 1 & \cdots \\ 0 & \delta_i - 1 & \cdots 0 & (\delta_i - 1)\mathrm{h}(m_0) & \cdots \\ 0 & r_{R,1} - r_L & \cdots 0 & (r_{R,1} - r_L)\mathrm{h}(m) & \cdots \end{pmatrix}.$$

We see that there must be some μ such that the fourth row is μ times the third row. This means that for all i with $\delta_i = 1$ we have $r_{R,i} = r_L$. Consider from now on the remaining i's where $\delta_i = -1$. We see that for all these i's we have

$$r_{R,i} - r_L = -2\mu \qquad \text{and} \qquad (r_{R,i} - r_L)\mathrm{h}(m) = -2\mu\mathrm{h}(m_0).$$

If $\mu = 0$ then $r_{R,i} = r_L$ for all i and therefore both left-decryption and right-decryption give the same result. In that case, the left-oracle answers correctly.

If $\mu \neq 0$ then we have for these i's that $(r_{R,i} - r_L)(\mathrm{h}(m) - \mathrm{h}(m_0)) = 0$ and $r_{R,i} - r_L \neq 0$. This implies that $\mathrm{h}(m) = \mathrm{h}(m_0)$, since the hashes are smaller than the smallest prime factor of n. Collision-freeness of the hash-function now implies that $m = m_0$. But in that case, both $\mathcal{O}_2^L$ and $\mathcal{O}_2$ answer `invalid`. We therefore see that the left-oracle answers the same as the real oracle.

Claim 13. $p_2 \approx p_3$.

Proof. Let i be an index such that $m_{0i}m_{1i} = -1$. We will argue that even if $\mathcal{A}$ is computationally unbounded and given $k_{L,j}, k_{R,j}, l_{L,j}, l_{R,j}$ for all $j \neq i$ it still cannot distinguish the two challenges.

From the available information $\mathcal{A}$ can use c to compute

$$K_i = \log(g_{L,i})k_{L,i} + \log(g_{R,i})k_{R,i} \bmod n$$

and d to compute

$$L_i = \log(g_{L,i})l_{L,i} + \log(g_{R,i})l_{R,i} \bmod n$$

as well as α to compute

$$\begin{aligned} A_i &= \log(g_{L,i})(k_{L,i} + \mathrm{h}(m_b)l_{L,i}) - \log(g_{R,i})(k_{R,i} + \mathrm{h}(m_b)l_{R,i}) + \mathrm{h}(m_b)\varDelta \\ &= K_i - 2\log(g_{R,i})k_{R,i} + \mathrm{h}(m_b)(L_i - 2\log(g_{R,i})l_{R,i} + \varDelta) \bmod n, \end{aligned}$$

where $\varDelta$ depends on the other k's and l's, but not $k_{L,i}, k_{R,i}, l_{L,i}, l_{R,i}$. However, since $k_{L,i}, l_{R,i}, l_{L,i}, l_{R,i}$ are chosen at random this does not reveal whether $b = 0$ or $b = 1$.

$\mathcal{A}$ cannot use the decryption queries to learn anything. If $\mathcal{A}$ wants to make a decryption query that has noticeable chance of being valid it must be on the form $(g_{L,1}^{r_1}, g_{R,1}^{r_1}, h_1^{m_1 r_1}, \ldots, g_{L,i}^{r_i}, g_{R,i}^{r_i}, h_i^{m_i r_i}, \ldots, g_{L,k}^{r_k}, g_{R,k}^{r_k}, h_k^{m_k r_k}, \prod_{j=1}^{k} g^{K_j + \mathrm{h}(m)L_j})$. This does not reveal any new information on $k_{L,i}, k_{R,i}, l_{L,i}, l_{R,i}$ and therefore b remains hidden.

Claim 14. $p_3 \approx p_4$.

Proof. By a hybrid argument if $\mathcal{A}$ can distinguish the two challenges then there is an index i such that $\mathcal{A}$ can be used to distinguish challenges on the form $(g_{L,1}^{r}, g_{R,1}^{m_{01}m_{11}r}, h_1^{m_{01}r}, \ldots, g_{L,i}^{r}, g_{R,i}^{m_{0i}m_{1i}r}, h_i^{m_{0i}r}, \ldots, g_{L,k}^{m_{0k}m_{1k}r}, g_{R,k}^{r}, h_k^{m_{1k}r}, \prod_{i=1}^{k} u_{L,i}^{k_{L,i}+l_{L,i}\mathrm{h}(m_1)} u_{R,i}^{k_{R,i}+l_{R,i}\mathrm{h}(m_1)})$ and $(g_{L,1}^{r}, g_{R,1}^{m_{01}m_{11}r}, h_1^{m_{01}r}, \ldots, g_{L,i}^{m_{0i}m_{1i}r}, g_{R,i}^{r}, h_i^{m_{1i}r}, \ldots, g_{L,k}^{m_{0k}m_{1k}r}, g_{R,k}^{r}, h_k^{m_{1k}r}, \prod_{i=1}^{k} u_{L,i}^{k_{L,i}+l_{L,i}\mathrm{h}(m_1)} u_{R,i}^{k_{R,i}+l_{R,i}\mathrm{h}(m_1)})$.

According to the DDH assumption it is impossible to tell whether a challenge (g, h, g^r, z) has $z = h^r$ or z chosen at random from G. This implies that it is hard to distinguish (g, h, g^r, h^r) and (g, h, g^r, h^{-r}).

So given a challenge (g, h, g^r, z), where $z = h^r$ or $z = h^{-r}$, we set $h_i = h$ and for all $j \neq i$ we compute $h_j = g^{x_j}$, where we choose x_j at random. We have now selected $h_1, \ldots, h_k$ and carry out the rest of the key generation procedure. This gives us a public key and a secret key. Now we run the first invocation of $\mathcal{A}$ on this challenge. $\mathcal{A}$ produces two challenge messages m_0 and m_1. If $m_{0i} = m_{1i}$ we stop and guess at random a bit b. However, if $m_{0i} \neq m_{1i}$ then we set $v_i = z$. We may now set it up such that $z = h^{m_{0i}r}$ gives us the challenge $(g_{L,1}^{r}, g_{R,1}^{m_{01}m_{11}r}, h_1^{m_{01}r}, \ldots, g_{L,i}^{r}, g_{R,i}^{m_{0i}m_{1i}r}, h_i^{m_{0i}r}, \ldots, g_{L,k}^{m_{0k}m_{1k}r}, g_{R,k}^{r}, h_k^{m_{1k}r}, \prod_{i=1}^{k} u_{L,i}^{k_{L,i}+l_{L,i}\mathrm{h}(m_1)} u_{R,i}^{k_{R,i}+l_{R,i}\mathrm{h}(m_1)})$, while $z = h^{m_{1i}r}$ gives us the challenge $(g_{L,1}^{r}, g_{R,1}^{m_{01}m_{11}r}, h_1^{m_{01}r}, \ldots, g_{L,i}^{m_{0i}m_{1i}r}, g_{R,i}^{r}, h_i^{m_{1i}r}, \ldots, g_{L,k}^{m_{0k}m_{1k}r}, g_{R,k}^{r}, h_k^{m_{1k}r}, \prod_{i=1}^{k} u_{L,i}^{k_{L,i}+l_{L,i}\mathrm{h}(m_1)} u_{R,i}^{k_{R,i}+l_{R,i}\mathrm{h}(m_1)})$. Since $\mathcal{A}$ can distinguish these two challenges this means we have broken the DDH assumption. □

4 Rerandomizable RCCA Secure Encryption

The WRCCA secure cryptosystem is not RCCA secure. First, let us argue that the WRCCA secure cryptosystem from the previous section is not RCCA secure. So we are given a ciphertext $(u_{L,1}, u_{R,1}, v_1, \ldots, u_{L,k}, u_{R,k}, v_k, \alpha)$ and want to know whether it encrypts m_0 or m_1. We simply transform it into $(u_{L,1}g_{L,1}, u_{R,1}g_{R,1}, v_1 h_1^{m_{01}}, \ldots, u_{L,k}g_{L,k}, u_{R,k}g_{R,k}, v_k h_k^{m_{0k}}, \alpha c d^{\mathrm{h}(m_0)})$. We then submit this modified ciphertext to the oracle $\mathcal{O}_2$. If the encrypted message is m_0 then we have a new encryption of m_0, and $\mathcal{O}_2$ answers `test`. On the other hand, if the encrypted message is m_1, then we have messed things up and $\mathcal{O}_2$ answers `invalid`. This means that we can distinguish between encryptions of the two possible plaintexts.

Improving the cryptosystem to have RCCA security. In the following, we attempt to fix the WRCCA secure cryptosystem. The problem in the attack above is that the adversary can rerandomize the ciphertext in a way such that he depending on the message inside gets either `test` or `invalid` as the answer. To prevent this we wish for a cryptosystem where the adversary is forced to make a correct rerandomization, and if he does not then he has overwhelming probability of getting `invalid` as answer.

To accomplish this we raise α to a random value Z. Rerandomization still works by raising all parts of the ciphertext to some random r'. Assuming the receiver knows this secret Z he can decrypt the ciphertext. On the other hand an adversary that does not know Z can only modify the proof in a meaningful way by raising the proof to some exponent. The adversary is therefore forced to either make correct rerandomizations or make some garbage. In particular he cannot use the previous attack where he with 50% probability creates a rerandomization and with 50% probability makes some garbage.

For this to be a public key cryptosystem we need the sender to choose Z and transmit it to the receiver. Therefore, she encrypts Z and sends it to the receiver. Since we want to have perfect rerandomization we also need to be able to rerandomize Z and the encryption of Z. We therefore use a homomorphic cryptosystem with message space $\mathbb{Z}_n$ to transmit Z to the receiver. This could for instance be Paillier-encryption, Cramer-Shoup Lite encryption based on the decisional composite residuosity assumption or perhaps some elliptic curve based cryptosystem.

Key generation: We set up the same public private keys (pk, sk) as in the previous section. Generate also keys (pk_n, sk_n) for an additively homomorphic cryptosystem with message space $\mathbb{Z}_n$. We demand that it is infeasible to find non-trivial factors of n.

The public key is $PK = (pk, pk_n)$.

The secret key is $SK = (sk, sk_n)$.

Encryption: Input: $m \in \{-1, 1\}^k$.

$E_{PK}(m; r, R, Z) = (g_{L,1}^r, g_{R,1}^r, h_1^{m_1 r}, \ldots, g_{L,k}^r, g_{R,k}^r, h_k^{m_k r}, (cd^{\mathrm{h}(m)})^{rZ}, E_{pk_n}(Z; R))$.

Decryption: Given a ciphertext $Y = (u_{L,1}, u_{R,1}, v_1, \ldots, u_{L,k}, u_{R,k}, v_k, \beta, y)$.

Compute $Z = D_{sk_n}(y)$. Check that $Z \in \mathbb{Z}_n^*$. Set $\alpha = \beta^{Z^{-1}}$. Finally, compute $m = D_{sk}(u_{L,1}, u_{R,1}, v_1, \ldots, u_{L,k}, u_{R,k}, v_k, \alpha)$.

If all checks and computations work out return m, otherwise return `invalid`.

Rerandomization: Input: PK and a ciphertext Y.

Format Y as $(u_{L,1}, u_{R,1}, v_1, \ldots, u_{L,k}, u_{R,k}, v_k, \beta, y)$. Check that all of these elements belong to appropriate groups.

Select randomizers r', Z', R'.

Return $(u_{L,1}^{r'}, u_{R,1}^{r'}, v_1^{r'}, \ldots, u_{L,k}^{r'}, u_{R,1}^{r'}, v_k^{r'}, \beta^{r'Z'}, E_{pk_n}(0; R')y^{Z'})$.

It is straightforward to verify that the cryptosystem is rerandomizable, and WRCCA security follows from the previous section. Left is the question whether it is RCCA secure.

Speaking against this idea is the fact that the adversary does actually get access to a chosen ciphertext attack on the homomorphic cryptosystem. For instance, given a y, it may form $(g_{L,1}^r, \ldots, h_k^r, (cd^{\mathrm{h}(1^k)})^z, y)$. Giving this ciphertext to $\mathcal{O}_2$ it can learn whether y contains z or not. Of course, if the adversary can use queries like this to figure out Z of the challenge encryption, then it may use the attack on the WRCCA scheme to violate the RCCA security of the proposed cryptosystem.

The semi-generic model. We are unable to prove security of the cryptosystem directly and likewise unable to break it. We therefore try to formulate a reasonable security model that says something about the security of the cryptosystem. Since random oracles are no good with respect to rerandomizable encryption we instead turn to the generic model, which has been explored in several papers including [5, 14, 9]. In other words, we will prove that if a generic homomorphic cryptosystem over $\mathbb{Z}_n$ is used to encrypt Z, then the construction is RCCA secure.

By a generic cryptosystem, we mean the following functionality. On an input (Encrypt, z) we choose y at random and store (z, y). On a query (Add, y, y') we look up whether y, y' have already been stored. In that case we select at random y'' and store $(z + z', y'')$. On input (Decrypt, y) we look up whether (z, y) has been stored for some z, and in that case we return z. Note that both adding a known value to an encrypted message and multiplying an encrypted message by some known number can be built from these two functions. This means that we allow use of the well-known homomorphic properties of cryptosystems such as Paillier encryption, CS-Lite encryption or elliptic curve based encryption. In the following, we use the shorthand $[x]$ to denote a generic encryption of x.

Encryption and decryption work as before, except we now use this generic cryptosystem to encrypt Z. The problem in the WRCCA case was that our oracle that just used left-decryption could not tell when to answer `test` and `invalid`, and indeed we showed with a concrete attack that this difference is important. We will argue that this problem goes away in the semi-generic model.

Recall that in the intuition provided for our conjectured RCCA secure cryptosystem we imagined Z to be completely unknown to the adversary. Since the adversary has access to a chosen ciphertext attack it is not possible to use semantic security of the encryption of Z to argue RCCA security. The semi-generic model intuitively corresponds to a “perfect” encryption of Z, which at the same time has the needed homomorphic property.

Theorem 2. *The cryptosystem described above is RCCA secure against semi-generic adversaries under the DDH assumption and the collision-freeness of the hash-function.*

Proof. Consider the definition of RCCA security. We will replace the oracle $\mathcal{O}_2$ with a different oracle $\mathcal{O}'$. $\mathcal{O}'$ works like $\mathcal{O}_2$ except when seeing a ciphertext Y that left-decrypts to m_0 and right-decrypts to m_1. In this special case it will check whether the proof α is valid with either $\mathrm{h}(m_0)$ or $\mathrm{h}(m_1)$. In those two cases, $\mathcal{O}'$ returns `test`, while in all other cases it acts like $\mathcal{O}_2$.

Just as in the proof of Theorem 1 we consider probabilities $p_0, \ldots, p_6$ that we define the following way: $p_i = \Pr[(pk, sk) \leftarrow K(); (m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{O}_1}(pk); y \leftarrow \mathrm{Chal_i} : \mathcal{A}^{\mathcal{O}}(y) = 1]$, where $\mathrm{Chal_i}$ for the various probabilities gives the following challenges, and in p_0, p_6 we use $\mathcal{O} = \mathcal{O}_2$, while in p_1, p_2, p_3, p_4 we use $\mathcal{O} = \mathcal{O}'$.

p_0: $(g_{L,1}^r, g_{R,1}^r, h_1^{m_{01}r}, \ldots, g_{L,k}^r, g_{R,k}^r, h_k^{m_{0k}r}, (cd^{\mathrm{h}(m_0)})^{rZ}, [Z])$.
p_1: $(\ldots, g_{L,i}^r, g_{R,i}^{m_{0i}m_{0i}r}, h_i^{m_{0i}r}, \ldots, (\prod_{i=1}^k u_{L,i}^{k_{L,i}+l_{L,i}\mathrm{h}(m_0)} u_{R,i}^{k_{R,i}+l_{R,i}\mathrm{h}(m_0)})^Z, [Z])$.
p_2: $(\ldots, g_{L,i}^r, g_{R,i}^{m_{0i}m_{1i}r}, h_i^{m_{0i}r}, \ldots, (\prod_{i=1}^k u_{L,i}^{k_{L,i}+l_{L,i}\mathrm{h}(m_0)} u_{R,i}^{k_{R,i}+l_{R,i}\mathrm{h}(m_0)})^Z, [Z])$.
p_3: $(\ldots, g_{L,i}^r, g_{R,i}^{m_{0i}m_{1i}r}, h_i^{m_{0i}r}, \ldots, (\prod_{i=1}^k u_{L,i}^{k_{L,i}+l_{L,i}\mathrm{h}(m_1)} u_{R,i}^{k_{R,i}+l_{R,i}\mathrm{h}(m_1)})^Z, [Z])$.
p_4: $(\ldots, g_{L,i}^{m_{0i}m_{1i}r}, g_{R,i}^r, h_i^{m_{1i}r}, \ldots, (\prod_{i=1}^k u_{L,i}^{k_{L,i}+l_{L,i}\mathrm{h}(m_1)} u_{R,i}^{k_{R,i}+l_{R,i}\mathrm{h}(m_1)})^Z, [Z])$.
p_5: $(\ldots, g_{L,i}^{m_{1i}m_{1i}r}, g_{R,i}^r, h_i^{m_{1i}r}, \ldots, (\prod_{i=1}^k u_{L,i}^{k_{L,i}+l_{L,i}\mathrm{h}(m_1)} u_{R,i}^{k_{R,i}+l_{R,i}\mathrm{h}(m_1)})^Z, [Z])$.
p_6: $(g_{L,1}^r, g_{R,1}^r, h_1^{m_{11}r}, \ldots, g_{L,k}^r, g_{R,k}^r, h_k^{m_{1k}r}, (cd^{\mathrm{h}(m_1)})^{rZ}, [Z])$.

To prove that the cryptosystem is RCCA secure we need to prove that $p_0 \approx p_6$. $p_0 \approx p_1$ according to Claim 21. $p_1 \approx p_2$ according to Claim 22. $p_2 \approx p_3$ according to Claim 22. $p_3 \approx p_4$ follows from a similar argument as we gave for Claim 14 in the proof of Theorem 1. $p_4 \approx p_5$ follows from a quite similar argument as the one given for Claim 22. $p_5 \approx p_6$ likewise follows from the proof of Claim 21.

Claim 21. $p_0 \approx p_1$.

Proof. Both challenges are computed the same way. The difference between the probabilities is the oracles $\mathcal{O}_2$ and $\mathcal{O}'$. However, we will argue that it is infeasible even for a computationally unbounded adversary $\mathcal{A}$ to distinguish between the two oracles as long as it may only make a polynomial number of queries, and even if $\mathcal{A}$ is allowed to freely make decryption queries to the generic cryptosystem.

The information available to $\mathcal{A}$ about $k_{L,1}, \ldots, l_{R,k}$ is what it can tell from c and d and the challenge. Consider a query $(u'_{L,1}, u'_{R,1}, v'_1, \ldots, u'_{L,k}, u'_{R,k}, v'_k, \beta', [Z'])$. Calling the respective discrete logarithms $r_{L,1}, r_{R,1}, r_1, \ldots, r_{L,k}, r_{R,k}, r_k, Z' \log(\alpha')$ we get the following system of linear equations.

$$\begin{pmatrix} 1 & 1 & \cdots & 0 & 0 & \cdots \\ 0 & 0 & \cdots & 1 & 1 & \cdots \\ r & r & \cdots & r\mathrm{h}(m_0) & r\mathrm{h}(m_0) & \cdots \\ r_{L,1} & r_{R,1} & \cdots & r_{L,1}\mathrm{h}(m) & r_{R,1}\mathrm{h}(m) & \cdots \end{pmatrix} \begin{pmatrix} \log(g_{L,1})k_{L,1} \\ \log(g_{R,1})k_{R,1} \\ \vdots \\ \log(g_{L,1})l_{L,1} \\ \log(g_{R,1})l_{R,1} \\ \vdots \end{pmatrix} = \begin{pmatrix} \log(c) \\ \log(d) \\ \log(\alpha) \\ \log(\alpha') \end{pmatrix}.$$

If the query is to return something else than `invalid` with more than negligible probability then $\mathcal{A}$ must use $r_{L,1} = r_{R,1} = \cdots = r_{L,k} = r_{R,k}$. But on such queries $\mathcal{O}_2$ and $\mathcal{O}'$ work the same way.

Claim 22. $p_1 \approx p_2$.

Sketch of proof. Just as in Claim 11 in the proof of Theorem 1 we may argue that we can break the DDH assumption if $\mathcal{A}$ distinguishes between the two challenges. The difference between Claim 11 and Claim 22 is the oracles that are used. However, here we may also argue just as in the proof of that claim that left-decryptions work just as well as right-decryptions. This follows from Claim 23.

Claim 23. The oracles $\mathcal{O}_1^L, \mathcal{O}'^L$ that only left-decrypt ciphertexts give the same answers as $\mathcal{O}_1, \mathcal{O}'$.

Proof. We look at the difficult case, namely whether $\mathcal{O}'$ and $\mathcal{O}'^L$ answer the same. Consider the information available to an adversary regarding $k_{L,1}, \ldots, l_{R,k}$. There is c, d and possibly a fake ciphertext. From this it must create a ciphertext with "proof" β'. Since we are using a generic cryptosystem for storing Z, the adversary must store some value $f(Z)$ in the homomorphic encryption. With the generic cryptosystem $f(Z) = aZ + b \bmod n$ with a, b known to the adversary.

Defining $\delta_i = m_{0i} m_{1i}$ we get the following system of linear equations in $\mathbb{Z}_n$.

$$\begin{pmatrix} 1 & 1 & \cdots & 0 & 0 & \cdots \\ 0 & 0 & \cdots & 1 & 1 & \cdots \\ r & r\delta_1 & \cdots & r\mathrm{h}(m_0) & r\delta_1\mathrm{h}(m_0) & \cdots \\ r_{L,1} & r_{R,1} & \cdots & r_{L,1}\mathrm{h}(m) & r_{R,1}\mathrm{h}(m) & \cdots \end{pmatrix} \begin{pmatrix} \log(g_{L,1})k_{L,1} \\ \log(g_{R,1})k_{R,1} \\ \vdots \\ \log(g_{L,1})l_{L,1} \\ \log(g_{R,1})l_{R,1} \\ \vdots \end{pmatrix} = \begin{pmatrix} \log(c) \\ \log(d) \\ \frac{\log(\beta)}{Z} \\ \frac{\log(\beta')}{f(Z)} \end{pmatrix}.$$

It is immediate that for any query with noticeable chance of being valid we must have some $r_L = r_{L,i}$ for all i. Reducing the matrices we get

$$\left(\begin{array}{cccccc|c} 1 & 1 & \cdots 0 & 0 & \cdots & & \log(c) \\ 0 & 0 & \cdots 1 & 1 & \cdots & & \log(d) \\ 0 & \delta_1 - 1 & \cdots 0 & (\delta_1 - 1)\mathrm{h}(m_0) & \cdots & & \frac{\log(\beta)}{rZ} - \log(c) - \mathrm{h}(m_0)\log(d) \\ 0 & r_{R,1} - r_L & \cdots 0 & (r_{R,1} - r_L)\mathrm{h}(m) & \cdots & & \frac{\log(\beta')}{f(Z)} - r_L\log(c) - r_L\mathrm{h}(m)\log(d) \end{array}\right)$$

If we have $r_L = r_{R,i}$ for all i then the left-decryption corresponds to the real decryption and both pairs of oracles answer the same. Assuming we are not in this trivial situation we can argue that for all i with $\delta_i = 1$ we have $r_{R,i} = r_L$. Similarly we have some r_R such that for all the other i's we have $r_R = r_{R,i}$. We also see that $m = m_0$ by the collision-freeness of the hash-function. Adding $\frac{r_R - r_L}{2}$ times row three to row four we get:

$$\left(\begin{array}{cccccc|c}
1 & 1 & \cdots & 0 & 0 & \cdots & \log(c) \\
0 & 0 & \cdots & 1 & 1 & \cdots & \log(d) \\
0 & \delta_1 - 1 & \cdots & 0 & (\delta_1 - 1)\mathrm{h}(m_0) & \cdots & \frac{\log(\beta)}{rZ} - \log(c) - \mathrm{h}(m_0)\log(d) \\
0 & 0 & \cdots & 0 & 0 & \cdots & \frac{\log(\beta')}{f(Z)} - r_L \log(c) - r_L \mathrm{h}(m_0)\log(d) \\
 & & & & & & -\frac{r_L - r_R}{2}\left(\frac{\log(\beta)}{rZ} - \log(c) - \mathrm{h}(m_0)\log(d)\right)
\end{array}\right)$$

We must therefore have

$$2\frac{\log(\beta')}{f(Z)} - (r_L + r_R)(\log(c) + \mathrm{h}(m_0)\log(d)) + (r_R - r_L)\frac{\log(\beta)}{rZ} = 0 \bmod n.$$

This implies

$$2\log(\beta')rZ - (r_L + r_R)(\log(c) + \mathrm{h}(m_0)\log(d))rZf(Z) + (r_R - r_L)\log(\beta)f(Z) = 0 \bmod n.$$

Since we use a generic cryptosystem the adversary cannot produce anything but $f(Z) = aZ + b$ with a and b known. We then get a degree 2 polynomial on the left side of the equation. Since Z is unknown, the adversary can only have a chance at producing correct proofs by making sure that it is the zero-polynomial on the left side.

So if the left side of the equation is the zero-polynomial then we get $(r_R - r_L)\log(\beta)b = 0 \bmod n$. Since b cannot be a non-trivial factor of n this implies $b = 0$ or $r_R - r_L = 0$. In the latter case both right- and left-decryption is the same and we are done. We therefore continue under the assumption that $b = 0$.

Considering the Z^2-part we get $(r_L + r_R)(\log(c) + \mathrm{h}(m_0)\log(d))ra = 0 \bmod n$. This implies $a = 0$ or $r_R = -r_L$. However, $a = 0$ would mean that y' contains $0Z + 0$ which automatically leads to the response `invalid` by both the real oracles and the left-oracles. On the other hand if we have $r_R = -r_L$ then we have for all i's where $\delta_i = -1$ that $r_R = \delta_i r_L$. Since the left-decryption is m_0 then this implies a right-decryption to m_1. But also in this case we then have the left-oracles give the same answer as the real oracles.

Remark 1. It is worth noting that even if we allow other types of mauling of the generic cryptosystem, we may have security. In particular, if we allow it to be algebraically homomorphic (i.e., both addition and multiplication of plaintexts is possible) this does not break our construction. In that case $f(Z)$ becomes a polynomial in Z with a polynomial number of different roots and we can use arguments similar to the one above to show that the left-oracles works the same way as the real oracles.

Claim 24. $p_2 \approx p_3$

Proof. Assume WLOG that $\mathcal{A}$ is computationally unbounded (but may only make a polynomial number of queries to the oracles) and knows the secret keys except $k_{L,1},\ldots,l_{R,k}$. We may argue from Claim 13 in the proof of Theorem 1 that it does not have any information on $\mathrm{h}(m_b)$ from the challenge itself, and therefore cannot distinguish the two experiments without making oracle queries.

Let us consider the oracle queries that it may make. We label the discrete logarithms of a successful query $(u'_{L,1},u'_{R,1},v'_1,\ldots,u'_{L,k},u'_{R,k},v'_k,(\alpha')^{Z'},[Z'])$ with $(r_{L,1},r_{R,1},r_1,\ldots,r_{L,k},r_{R,k},r_k,-,-)$. We then have the following system of equations.

$$\begin{pmatrix} 1 & 1 & \cdots & 0 & 0 & \cdots \\ 0 & 0 & \cdots & 1 & 1 & \cdots \\ r & r\delta_1 & \cdots & r\mathrm{h}(m_b) & r\delta_1\mathrm{h}(m_b) & \cdots \\ r_{L,1} & r_{R,1} & \cdots & r_{L,1}\mathrm{h}(m) & r_{R,1}\mathrm{h}(m) & \cdots \end{pmatrix} \begin{pmatrix} \log(g_{L,1})k_{L,1} \\ \log(g_{R,1})k_{R,1} \\ \vdots \\ \log(g_{L,1})l_{L,1} \\ \log(g_{R,1})l_{R,1} \\ \vdots \end{pmatrix} = \begin{pmatrix} \log(c) \\ \log(d) \\ \log(\alpha) \\ \log(\alpha') \end{pmatrix}.$$

We see that there is an element r_L such that for all i we have $r_{L,i} = r_L$. Reducing the matrix we get.

$$\begin{pmatrix} 1 & 1 & \cdots 0 & 0 & \cdots \\ 0 & 0 & \cdots 1 & 1 & \cdots \\ 0 & \delta_1 - 1 & \cdots 0 & (\delta_1 - 1)\mathrm{h}(m_b) & \cdots \\ 0 & r_{R,1} - r_L & \cdots 0 & (r_{R,1} - r_L)\mathrm{h}(m) & \cdots \end{pmatrix}.$$

Unless $r_{R,i} = r_L$ for all i, then $\mathrm{h}(m) = \mathrm{h}(m_b)$ and $r_{R,i} = r_{R,1}$ for all i. Those two options correspond to respectively make a new ciphertext, or rerandomize the challenge. In either case, $\mathcal{A}$ does not learn anything new from $\mathcal{O}'$'s answers. □

Theorem 2 tells us is that the scheme is RCCA secure against semi-generic adversaries that only use standard group operations on the encryption of Z. We can instantiate the cryptosystems with many possible homomorphic cryptosystems, for instance Paillier encryption, CS-Lite encryption or elliptic curve encryption. We could also use a multiplicative homomorphic property instead and use standard RSA to encrypt Z. To break the scheme we would have to come up with some non-standard way of mauling these cryptosystems. We believe such a result would be highly interesting in itself.

5 Discussion

To evaluate our results we find it useful to compare them with the development of standard CCA2 secure public key encryption. In this process, Naor and Yung [12] invented a CCA1 secure encryption scheme. Dolev, Dwork and Naor [10] then suggested a CCA2 secure cryptosystem. Several years after this Cramer and

Shoup [7] suggested the first practical CCA2 secure cryptosystem. Furthermore, several schemes have been proposed that are secure in the random oracle model. A proof of security in the random oracle model is not a real proof of security, but it is better than no proof at all.

With respect to rerandomizable encryption our intuition is that WRCCA secure encryption is a step on the way. WRCCA secure encryption may have its uses, however, as CCA2 secure encryption is the standard for public key encryption we think RCCA secure encryption is the right standard for rerandomizable encryption. As stated earlier we believe coming up with a rerandomizable RCCA secure encryption scheme is a very hard task, and certainly an interesting open problem. In lack of such a scheme, we have suggested using another security paradigm, namely RCCA security against semi-generic adversaries. Just as proving CCA2 security in the random oracle model is not the same as proving CCA2 security in the standard model, proving RCCA security in the semi-generic model is not the same as proving RCCA security in the standard model, but it is better than no proof at all.

6 Acknowledgments

Moni Naor suggested the idea of rerandomizable encryption in a conversation with us and we also appreciate his encouragement during the research. Thanks goes to Alon Rosen and Jesper Buus Nielsen for discussions.

References

1. Jee Hea An, Yevgeni Dodis, and Tal Rabin. On the security of joint signature and encryption. In *proceedings of EUROCRYPT '02, LNCS series, volume 2332*, pages 83–107, 2002.
2. Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *proceedings of CRYPTO '98, LNCS series, volume 1462*, pages 26–45, 1998.
3. Mihir Bellare and Amit Sahai. Non-malleable encryption: Equivalence between two notions, and an indistinguishability-based characterization. In *proceedings of CRYPTO '99, LNCS series, volume 1666*, pages 519–536, 1999.
4. Daniel Bleichenbacher. Chosen ciphertext attacks against protocols based on the rsa encryption standard pkcs 1. In *proceedings of CRYPTO '98, LNCS series, volume 1462*, pages 1–12, 1998.
5. Dan Boneh and Richard J. Lipton. Algorithms for black-box fields and their application to cryptography. In *proceedings of CRYPTO '96, LNCS series, volume 1109*, pages 283–297, 1996.
6. Ran Canetti, Hugo Krawczyk, and Jesper Buus Nielsen. Relaxing chosen-ciphertext security. In *proceedings of CRYPTO '03, LNCS series, volume 2729*, pages 565–582, 2003.
7. Ronald Cramer and Victor Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. In *proceedings of CRYPTO '98, LNCS series, volume 1462*, pages 13–25, 1998.

8. Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *proceedings of EUROCRYPT '02, LNCS series, volume 2332*, pages 45–64, 2002.
9. Ivan Damgård and Maciej Koprowski. Generic lower bounds for root extraction and signature schemes in general groups. In *proceedings of EUROCRYPT '02, LNCS series, volume 2332*, pages 256–271, 2002.
10. Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. *SIAM J. of Computing*, 30(2):391–437, 2000. Earlier version at STOC '91.
11. Yehuda Lindell. A simpler construction of cca2-secure public-key encryption under general assumptions. In *proceedings of EUROCRYPT '03, LNCS series, volume 2656*, pages 241–254, 2003.
12. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *proceedings of STOC '90*, pages 427–437, 1990.
13. Amit Sahai. Non-malleable non-interactive zero-knowledge and adaptive chosen-ciphertext security. In *proceedings of FOCS '01*, pages 543–553, 2001.
14. Victor Shoup. Lower bounds for discrete logarithms and related problems. In *proceedings of EUROCRYPT '97, LNCS series, volume 1233*, pages 256–266, 1997.
15. Victor Shoup. A proposal for an iso standard for public key encryption. Cryptology ePrint Archive, Report 2001/112, 2001. `http://eprint.iacr.org/2001/212`.

A Appendix

Theorem 3. *The directed graph in Figure 1 describes completely the relations between our security notions. ATT1 security implies ATT2 security if there is a path from ATT1 to ATT2. If there is no path from ATT1 to ATT2, then a cryptosystem with ATT1 security implies the existence of a ATT2 secure cryptosystem, which is not ATT1 secure.*

Sketch of proof. It is trivial to follow each arrow and see that it leads to a weaker security notion.

We list the constructions that can be used to separate the security notions. To show that ATT1 $\nrightarrow$ ATT2 we assume that (K, E, D) is an ATT1 secure cryptosystem and present (K', E', D') that is ATT1 secure but not ATT2 secure. K', E' will be as follows

Key generation: K' runs $(pk, sk) \leftarrow K()$. It also selects at random a seed s for a pseudorandom function PRF and a random nonce r. It returns $(pk' = (pk, r), sk' = (sk, r, s))$.
Encryption: $E'_{pk'}(m; r) = (0, E_{pk}(m))$.

Left is to describe how D' works, which we do in the table of inputs and corresponding outputs in the table below.

RCCA+PA1 $\nrightarrow$ CCA2 :

$(0, y)$	: $D_{sk}(y)$
$(1, y)$	: $\mathrm{PRF}_s(y)$
$(2, p, y)$	: If $p = \mathrm{PRF}_s(y)$ return $D_{sk}(y)$, else return `invalid`.

$$\begin{tikzcd}
 & & \text{CCA2} \arrow[d] & & \\
 & & \text{RCCA+PA1} \arrow[dl] \arrow[dr] & & \\
 & \text{RCCA+PA0} \arrow[dl] \arrow[dr] & & \text{WRCCA+PA1} \arrow[dl] \arrow[dr] & \\
\text{RCCA} \arrow[d] & & \text{WRCCA+RPA1+PA0} \arrow[dll] \arrow[d] \arrow[drr] & & \text{PA1} \arrow[d] \\
\text{WRCCA+RPA1} \arrow[d] \arrow[drr] & & \text{WRCCA+PA0} \arrow[dll] \arrow[drr] & & \text{RPA1+PA0} \arrow[dll] \arrow[d] \\
\text{WRCCA+RPA0} \arrow[d] \arrow[drr] & & \text{RPA1} \arrow[d] & & \text{WRPA1+PA0} \arrow[dll] \arrow[d] \\
\text{WRCA} \arrow[d] & & \text{WRPA1+RPA0} \arrow[dll] \arrow[d] & & \text{CCA1+PA0} \arrow[dll] \arrow[d] \\
\text{WRPA1} \arrow[dr] & & \text{CCA1+RPA0} \arrow[dl] \arrow[dr] & & \text{PA0} \arrow[dl] \\
 & \text{CCA1+WRPA0} \arrow[dr] & & \text{RPA0} \arrow[dl] & \\
 & & \text{WRPA0} \arrow[d] & & \\
 & & \text{CPA} & &
\end{tikzcd}$$

Fig. 1. Relations between security notions.

WRCCA+ATT $\not\rightarrow$ RCCA, where ATT$\in$\{RPA1,RPA1+PA0,PA1\}

$(0, y)$: $D_{sk}(y)$

$(1, y)$: $\mathrm{PRF}_s(y)$

$(2, p, m, y)$: If $p = \mathrm{PRF}_s(y)$ and $m = D_{sk}(y)$ return m, else return `invalid`.

ATT $\not\rightarrow$ WRCCA, ATT$\in$\{PA1,RPA1+PA0,RPA1,WRPA1+RPA0,WRPA1\}

$(0, y)$: $D_{sk}(y)$

$(1, y)$: $\mathrm{PRF}_s(y)$

$(2, p, m, y)$: If $p = \mathrm{PRF}_s(y)$ and $m = D_{sk}(y)$ return `yes`, else return `invalid`.

CCA1+ATT $\not\rightarrow$ WPA1, where ATT$\in$\{PA0,RPA0,WRPA0,nothing\}

$(0, y)$: $D_{sk}(y)$

$(1, r)$: s

$(2, s, y)$: $D_{sk}(y)$.

ATT $\not\rightarrow$ CCA1, where ATT$\in$\{PA0,RPA0,WRPA0,CPA\}

$(0, y)$: $D_{sk}(y)$

$(1, r)$: s

$(2, s)$: sk.

ATT $\not\rightarrow$ WRPA0, where ATT$\in$\{CPA,CCA1\}

$(0, y)$: $D_{sk}(y)$

$(1, m, y)$: If $m = D_{sk}(y)$ then return `yes`, else return `invalid`.

ATT $\not\rightarrow$ RPA0, where ATT$\in$\{WRPA0,CCA1+WRPA0,WRPA1,WRCCA\}

$(0, y)$: $D_{sk}(y)$

$(1, m, y)$: If $m = D_{sk}(y)$ then return m, else return `invalid`.

ATT $\not\rightarrow$ PA0, where ATT$\in$\{RPA0,CCA1+RPA0,RPA1,WRCCA+RPA0\}

$(0, y)$: $D_{sk}(y)$

$(1, y)$: $D_{sk}(y)$.

ATT $\not\rightarrow$ RPA1, ATT$\in$\{WRCCA,WRCCA+PA0,WRPA1+RPA0,WRPA1+PA0\}

$(0, y)$: $D_{sk}(y)$

$(1, r)$: s

$(2, s, m, y)$: If $m = D_{sk}(y)$ then return m, else return `invalid`.

ATT $\not\rightarrow$ PA1, ATT$\in$\{WRCCA+RPA1+PA0,RPA1+PA0,RCCA+PA0\}

$(0, y)$: $D_{sk}(y)$

$(1, r)$: s

$(2, s, y)$: $D_{sk}(y)$.

□

It is interesting to note that Theorem 3 implies that a cryptosystem that is both IND-CCA1 secure and NM-CPA secure is not necessarily NM-CCA1 secure. This combination was not considered in [2].