applied sciences

MDPI

Article

Chosen-Ciphertext Secure Unidirectional Proxy Re-Encryption Based on Asymmetric Pairings

Benjamin Zengin *, Paulin Deupmann, Nicolas Buchmann and Marian Margraf

Secure Systems Engineering, Fraunhofer AISEC, 14199 Berlin, Germany
* Correspondence: benjamin.zengin@aisec.fraunhofer.de

Abstract: Proxy re-encryption (PRE) is a cryptographic primitive that extends public key encryption by allowing ciphertexts to be re-encrypted from one user to another without revealing information about the underlying plaintext. This makes it an essential privacy-enhancing technology, as only the intended recipient is able to decrypt sensitive personal information. Previous PRE schemes were commonly based on symmetric bilinear pairings. However, these have been found to be slower and less secure than the more modern asymmetric pairings. To address this, we propose two new PRE scheme variants, based on the unidirectional symmetric pairing-based scheme by Weng et al. and adapted to utilize asymmetric pairings. We employ a known automated black-box reduction technique to transform the base scheme to the asymmetric setting, identify its shortcomings, and subsequently present an alternative manual transformation that fixes these flaws. The adapted schemes retain the properties of the base scheme and are therefore CCA-secure in the adaptive corruption model without the use of random oracles, while being faster, practical, and more secure overall than the base scheme.

Keywords: proxy re-encryption; unidirectional; chosen-ciphertext security; asymmetric bilinear groups

Citation: Zengin, B.; Deupmann, P.; Buchmann, N.; Margraf, M. Chosen-Ciphertext Secure Unidirectional Proxy Re-Encryption Based on Asymmetric Pairings. *Appl. Sci.* **2024**, *14*, 11322. https://doi.org/10.3390/app142311322

Academic Editors: Yi-Fan Tseng and Chun-I Fan

Received: 15 October 2024
Revised: 29 November 2024
Accepted: 2 December 2024
Published: 4 December 2024

1. Introduction

Proxy re-encryption (PRE), first introduced by Blaze et al. [1], is a cryptographic primitive that extends public key encryption by allowing ciphertexts to be re-encrypted from one user (delegator) to another (delegatee) without revealing information about the underlying plaintext. This privacy-friendly re-encryption is performed by a semi-trusted proxy using a re-encryption key and establishes PRE as an essential privacy-enhancing technology, as only the intended recipient is able to decrypt sensitive personal information.

A PRE scheme can be classified according to different properties. In a unidirectional scheme, re-encryption keys can re-encrypt ciphertexts from the delegator to the delegatee. In a bidirectional scheme, they can also be used vice versa. A scheme can be either single-use or multi-use. Single-use schemes do not allow re-encryption of ciphertexts that have already been re-encrypted, while multi-use schemes have no such restriction. This paper considers unidirectional single-use schemes.

Many different unidirectional PRE schemes have been proposed in the past. The first unidirectional schemes were proposed by Ateniese et al. [2], but they were only secure against chosen-plaintext attacks (CPAs). Libert and Vergnaud [3] proposed a unidirectional scheme in the selective corruption model and proved its security against replayable chosen-ciphertext attacks (RCCAs)—a slightly weaker variant of chosen-ciphertext attacks (CCAs). Weng et al. [4] further extended this result and presented a scheme which they proved to be CCA-secure in the adaptive corruption model. To the best of our knowledge, the scheme by Weng et al. is the only PRE scheme in the literature achieving CCA security under adaptive corruptions in the standard model. The adaptive corruption model is considered superior to approaches that only permit selected corruptions. This is because it provides a more realistic assessment of security, as it accounts for dynamic adversarial behavior.

All these schemes have in common that they are based on bilinear pairings, which can be categorized into three main types [5]. Type 1 pairings are called symmetric; Type 2 and Type 3 pairings are called asymmetric. To the best of our knowledge, all CCA-secure unidirectional single-use PRE schemes based on pairings proposed in the literature to date use Type 1 pairings.

However, it has been shown that Type 1 pairings exhibit a significantly poorer runtime [6] than their asymmetric counterparts [7,8] and are generally considered inferior in terms of security, flexibility, and efficiency, leading to the conclusion that modern asymmetric pairings should be the default choice when designing schemes based on pairings [9]. In the asymmetric setting, in turn, Type 3 pairings are the preferred choice, since it has been demonstrated that whatever is achievable in terms of functionality and security in Type 2 can also be achieved in Type 3 [10], but with better performance.

Although CCA-secure PRE schemes have been proposed that do not rely on pairings, their complexity often introduces a higher risk of errors. Shao et al. [11] proposed the first CCA-secure unidirectional PRE scheme without pairings under adaptive corruptions. However, Chow et al. [12] demonstrated a flaw in the scheme proposed in [11] and presented an adapted scheme that achieves CCA security, but only under selective corruptions. Selvi et al. [13] subsequently demonstrated that the proof in [12] was flawed and proposed another CCA-secure PRE scheme under selective corruptions. In contrast to the previously discussed pairing-based schemes, the security proofs of the proposed pairing-free schemes require the use of the random oracle model instead of the standard model. Recent work also includes quantum-safe PRE based on lattices. In this context, Dutta et al. [14] presented novel constructions for identity-based PRE, while Susilo et al. [15] advanced the field of attribute-based PRE. Zhou et al. [16] proposed a PRE scheme that is secure under adaptive corruptions and supports fine-grained re-encryptions. They subsequently extended this scheme to a multi-use scheme [17]. However, thus far, lattice-based PRE schemes have only achieved CPA security.

Consequently, pairing-based PRE schemes exhibit a unique characteristic within the field. Currently, they are the only PRE schemes that feature CCA security in the standard model. However, to date, all pairing-based schemes are based on Type 1 pairings, as opposed to Type 3 pairings. Type 1 pairing-based schemes require supersingular elliptic curves over large characteristic fields, which are mainly relevant for academic research, but not for practical implementations [9]. In contrast, Type 3 pairing-based schemes permit the use of established pairing-friendly elliptic curves (e.g., BN and BLS curves [18]), which exhibit better security margins and are practical for implementation.

1.1. Our Contribution

The goal of this paper is to address the gap in the existing literature by proposing a practical CCA-secure PRE scheme based on Type 3 pairings. Given the unique properties of achieving CCA security under adaptive corruptions, the Type 1 pairing-based scheme by Weng et al. [4] is used as the base scheme. By transforming the base scheme, a CCA-secure Type 3 pairing-based PRE scheme is obtained.

The initial evident methodology for obtaining a Type 3 pairing-based PRE scheme is to apply a generalized scheme transformation, which was first proposed by Abe et al. [19]. This approach is pursued by applying the automated black-box reduction technique by Akinyele et al. [20] to transform the base scheme and its security proof from the Type 1 to the Type 3 setting. However, we identify that the scheme resulting from the transformation exhibits flaws that render it an impractical PRE scheme, which is contrary to the goal of our paper. To resolve these issues, we next propose a manually transformed scheme. This scheme preserves the properties of the base scheme and the CCA security in the adaptive corruption model without the use of random oracles. We refine the hardness assumption for the Type 3 setting, ensuring that it is at least as hard as in the base scheme.

1.2. Organization of This Paper

This paper begins with a review of bilinear pairings and relevant complexity assumptions in Section 2. Furthermore, the notion of unidirectional PRE and the security model is reviewed. Section 3 outlines the application of the automated transformation and the resulting scheme is analyzed. Subsequently, Section 4 presents our adapted manually transformed Type 3 pairing-based PRE scheme and the necessary adjustments to the security proof. This is followed by a performance and ciphertext size evaluation of both transformed schemes. Finally, the paper concludes in Section 5.

2. Preliminaries

This section introduces our notation and reviews the definition of bilinear pairings. Subsequently, the complexity assumption that is used to modify the security proof of the base scheme for our manually transformed scheme is described and proven to be at least as hard as the assumptions used in previous schemes. The section concludes with a review of the definition of PRE and the security model used by Weng et al. [4].

2.1. Notation

We denote drawing an element x from a finite set S uniformly at random by $x \xleftarrow{\$} S$. For a string $x \in \{0,1\}^n$, we let $[x]_\ell$ denote its first ℓ bits and $[x]^\ell$ denote its last ℓ bits.

2.2. Bilinear Pairings

For cyclic groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of large prime order p, a bilinear pairing is a function $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that maps pairs of elements in $(\mathbb{G}_1, \mathbb{G}_2)$ to elements of the group $\mathbb{G}_T$. The map e must satisfy the following properties:

- Bilinear: For all $u \in \mathbb{G}_1, v \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p$, it holds that $e(u^a, v^b) = e(u, v)^{ab}$.
- Computable: The map e is efficiently computable and so are the group operations in $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$.
- Non-degenerate: There exist $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ such that $e(g_1, g_2) \neq 1$.

There are two forms of pairings used in the cryptography literature [5]. In the symmetric setting, it holds that $\mathbb{G}_1 = \mathbb{G}_2$, whereas in the asymmetric setting $\mathbb{G}_1 \neq \mathbb{G}_2$. Besides the distinction between symmetric and asymmetric pairings, three basic types can be identified as possible pairing instantiations. Type 1 is the symmetric setting. In the asymmetric setting, a distinction is made between Type 2, where there is an efficiently computable isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$, and Type 3, where there are no efficiently computable isomorphisms between the source groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

In order to differentiate between the pairing types, we define, analogously to [10], the groups $\mathbb{G}_1 = \langle g_1 \rangle, \mathbb{G}_2' = \langle g_2' \rangle$, and $\hat{\mathbb{G}}_2 = \langle \hat{g}_2 \rangle$ of prime order p, such that there is an isomorphism $\psi : \mathbb{G}_2' \to \mathbb{G}_1$ with $g_1 = \psi(g_2')$ and an isomorphism $\rho : \mathbb{G}_2' \to \hat{\mathbb{G}}_2$ with $\hat{g}_2 = \rho(g_2')^{\frac{1}{c}}$ for an arbitrary $c \in \mathbb{Z}_p^*$. Finally, we define the Type 2 pairing $e_2 : \mathbb{G}_1 \times \mathbb{G}_2' \to \mathbb{G}_T$ and the Type 3 pairing $e_3 : \mathbb{G}_1 \times \hat{\mathbb{G}}_2 \to \mathbb{G}_T$. The relation between e_2 and e_3 is established by the following Lemma.

Lemma 1 (Chatterjee and Menezes [10] (Lemma 2)). *Let g_1, g_2', and $\hat{g}_2$ be generators of $\mathbb{G}_1, \mathbb{G}_2'$, and $\hat{\mathbb{G}}_2$ with $g_1 = \psi(g_2')$ and $\hat{g}_2 = \rho(g_2')^{\frac{1}{c}}$ for some $c \in \mathbb{Z}_n^*$. Then, $e_2(g_1, g_2') = e_3(g_1, \hat{g}_2)^{2c}$.*

This notation is maintained throughout the paper in order to facilitate the differentiation between elements $x \in \mathbb{G}_1$ and elements $\hat{x} \in \hat{\mathbb{G}}_2$.

2.3. Complexity Assumptions

We base the security of our manually transformed scheme on a variant of the 3-weak Decisional Bilinear Diffie–Hellman Inversion (3-wDBDHI) assumption, which was used by Libert and Vergnaud [3] and subsequently by Weng et al. [4] to construct their unidirectional PRE schemes.

Definition 1. *The 3-weak Decisional Bilinear Diffie–Hellman Inversion Type 3 assumption (3-wDBDHI$_3$) states that, given* $(g_1, g_1^{\frac{1}{a}}, g_1^a, g_1^{(a^2)}, g_1^b, \hat{g}_2, \hat{g}_2^{\frac{1}{a}}, \hat{g}_2^a, \hat{g}_2^{(a^2)}, \hat{g}_2^b, Q) \in \mathbb{G}_1^5 \times \hat{\mathbb{G}}_2^5 \times \mathbb{G}_T$ *with unknown* $a, b \in \mathbb{Z}_p^*$*, it is computationally infeasible to decide whether* $Q = e_3(g_1, \hat{g}_2)^{\frac{b}{a^2}}$*. A distinguisher* $\mathcal{B}$ (t, ε) *breaks the assumption if it runs in time t and*

$$\left| \Pr\left[\mathcal{B}\left(\begin{array}{c} g_1, g_1^{\frac{1}{a}}, g_1^a, g_1^{(a^2)}, g_1^b, \hat{g}_2, \hat{g}_2^{\frac{1}{a}}, \\ \hat{g}_2^a, \hat{g}_2^{(a^2)}, \hat{g}_2^b, Q = e_3(g_1, \hat{g}_2)^{\frac{b}{a^2}} \end{array} \right) = 1 \middle| a, b \xleftarrow{\$} \mathbb{Z}_p^* \right] \right.$$
$$\left. - \Pr\left[\mathcal{B}\left(\begin{array}{c} g_1, g_1^{\frac{1}{a}}, g_1^a, g_1^{(a^2)}, g_1^b, \hat{g}_2, \hat{g}_2^{\frac{1}{a}}, \\ \hat{g}_2^a, \hat{g}_2^{(a^2)}, \hat{g}_2^b, Q = e_3(g_1, \hat{g}_2)^z \end{array} \right) = 1 \middle| a, b, z \xleftarrow{\$} \mathbb{Z}_p^* \right] \right| \geq \varepsilon.$$

Lemma 2 shows that the 3-wDBDHI$_3$ problem is at least as hard as the 3-wDBDHI$_2$ problem, where the task is to distinguish $e_2(g_1, g_2')^{\frac{b}{a^2}}$ from random data given $g_1 \in \mathbb{G}_1$ and $g_2', {g_2'}^a, {g_2'}^{(a^2)}, {g_2'}^b \in \mathbb{G}_2'$. The 3-wDBDHI$_2$ problem is the same assumption that Weng et al. [4] use in their scheme, but translated from the symmetric setting into the Type 2 setting.

Lemma 2. *Let* g_1 *be a generator of* $\mathbb{G}_1$ *and* g_2' *a generator of* $\mathbb{G}_2'$ *with* $g_1 = \psi(g_2')$*. Then, 3-wDBDHI$_3$ is at least as hard as 3-wDBDHI$_2$.*

Proof. Given a 3-wDBDHI$_2$ instance $(g_1, g_2', {g_2'}^{\frac{1}{a}}, {g_2'}^a, {g_2'}^{(a^2)}, {g_2'}^b, Q) \in \mathbb{G}_1 \times {\mathbb{G}_2'}^5 \times \mathbb{G}_T$ with $a, b \in \mathbb{Z}_p^*$, we apply $\rho : \mathbb{G}_2' \to \hat{\mathbb{G}}_2$ to obtain $\hat{g}_2 = \rho(g_2')^{\frac{1}{c}}, \hat{g}_2^{\frac{1}{a}} = \rho({g_2'}^{\frac{1}{a}})^{\frac{1}{c}}, \hat{g}_2^a = \rho({g_2'}^a)^{\frac{1}{c}}, \hat{g}_2^{(a^2)} = \rho({g_2'}^{(a^2)})^{\frac{1}{c}}, \hat{g}_2^b = \rho({g_2'}^b)^{\frac{1}{c}}$. Furthermore, we apply $\psi : \mathbb{G}_2' \to \mathbb{G}_1$ to $g_1 = \psi(g_2')$, $g_1^{\frac{1}{a}} = \psi({g_2'}^{\frac{1}{a}}), g_1^a = \psi({g_2'}^a), g_1^{(a^2)} = \psi({g_2'}^{(a^2)}), g_1^b = \psi({g_2'}^b)$. The resulting 3-wDBDHI$_3$ problem instance $(g_1, g_1^{\frac{1}{a}}, g_1^a, g_1^{(a^2)}, g_1^b, \hat{g}_2, \hat{g}_2^{\frac{1}{a}}, \hat{g}_2^a, \hat{g}_2^{(a^2)}, \hat{g}_2^b, Q^{\frac{1}{2c}}) \in \mathbb{G}_1^5 \times \mathbb{G}_2^5 \times \mathbb{G}_T$ is given to the 3-wDBDHI$_3$ solver, which determines whether $Q^{\frac{1}{2c}} = e_3(g_1, \hat{g}_2)^{\frac{b}{a^2}}$ which, by Lemma 1, is equivalent to $Q = e_2(g_1, g_2')^{\frac{b}{a^2}}$. This establishes that 3-wDBDHI$_2 \leq$ 3-wDBDHI$_3$. □

2.4. Model of PRE

We recall the syntax of PRE using the following definition.

Definition 2. *A single-hop unidirectional PRE scheme consists of a tuple of algorithms* (Setup, KeyGen, ReKeyGen, Enc$_2$, Enc$_1$, ReEnc, Dec$_2$, Dec$_1$)*:*

Setup$(1^k) \to$ *param: Given the security parameter k, output the public parameters param which will be used by all parties of the scheme.*

KeyGen$(param) \to (sk_i, pk_i)$*: Given the global parameters param, output a secret/public key pair* (sk_i, pk_i)*.*

In the remaining algorithms that follow, the public parameter param will be implicitly included.

ReKeyGen$(sk_i, pk_j) \to rk_{i \to j}$*: Given the secret key* sk_i *and another public key* pk_j*, output a re-encryption key* $rk_{i \to j}$*.*

Enc$_2(pk_i, m) \to \mathrm{CT}_i$*: Given a public key* pk_i *and a message* $m \in \mathcal{M}$*, output a second-level ciphertext* CT_i *that can be re-encrypted into a first-level ciphertext using the suitable re-encryption key.*

Enc$_1(pk_j, m) \to \mathrm{CT}_j$*: Given a public key* pk_j *and a message* $m \in \mathcal{M}$*, output a first-level ciphertext* CT_j *that cannot be re-encrypted for another party.*

ReEnc$(pk_i, rk_{i \to j}, \mathrm{CT}_i) \to \mathrm{CT}_j$*: Given the public key* pk_i*, a re-encryption key* $rk_{i \to j}$*, and a second-level ciphertext* CT_i *encrypted under user i's public key, output a first-level ciphertext* CT_j *or* $\perp$ *if* CT_i *is invalid.*

Dec$_2(sk_i, \mathrm{CT}_i) \to m$*: Given a secret key* sk_i *and a second-level ciphertext* CT_i*, output either a message* $m \in \mathcal{M}$ *or* $\perp$ *if* CT_i *is invalid.*

$\mathsf{Dec}_1(sk_j, \mathrm{CT}_j) \rightarrow m$: *Given a secret key* sk_j *and a first-level ciphertext* CT_j*, output a message* $m \in \mathcal{M}$ *or* $\perp$ *if* CT_j *is invalid.*

For any common public parameters, param; for any message $m \in \mathcal{M}$*; and for any pair of secret/public key pairs,* (sk_i, pk_i), (sk_j, pk_j)*, these algorithms should satisfy the following correctness conditions:*

$$\begin{gathered}\mathsf{Dec}_1(sk_i, \mathsf{Enc}_1(pk_i, m)) = m;\\ \mathsf{Dec}_2(sk_i, \mathsf{Enc}_2(pk_i, m)) = m;\\ \mathsf{Dec}_1(sk_j, \mathsf{ReEnc}(pk_i, \mathsf{ReKeyGen}(sk_i, pk_j), \mathsf{Enc}_2(pk_i, m))) = m.\end{gathered}$$

2.5. Security Model

The transformation of the base scheme to the Type 3 setting does not alter the security model in which the scheme is defined. Therefore, a brief overview of the game-based security model used by Weng et al. [4] is provided. The security game consists of five phases.

In the setup, the challenger $\mathcal{B}$ hands over the required information to the adversary $\mathcal{A}$. Then, $\mathcal{A}$ can query oracles which the challenger responds to in the find stage. In the challenge phase, the adversary outputs two distinct messages m_0, m_1 as well as the target public key pk_{i^*}. The challenger returns the encrypted message $\mathrm{CT}^* = \mathsf{Enc}(pk_{i^*}, m_\delta)$ with $\delta \in 0, 1$. In the guess stage, $\mathcal{A}$ is given access to the oracles again, and in the output phase, they return their guess δ'. The adversary has access to the following oracles in the find and guess phase.

- Public key oracle $\mathcal{O}_{pk}(i)$: Create a key pair (pk_i, sk_i) analogously to $\mathsf{KeyGen}(param)$ and return pk_i to $\mathcal{A}$.
- Secret key oracle $\mathcal{O}_{sk}(pk_i)$: Return sk_i to $\mathcal{A}$ with respect to pk_i, which was generated by the oracle $\mathcal{O}_{pk}$ beforehand.
- Re-encryption key oracle $\mathcal{O}_{rk}(pk_i, pk_j)$: Given two public keys, pk_i and pk_j, which were generated by the oracle $\mathcal{O}_{pk}$ beforehand, run $rk_{i\rightarrow j} \leftarrow \mathsf{ReKeyGen}(sk_i, pk_j)$ and return the re-encryption key $rk_{i\rightarrow j}$ to $\mathcal{A}$.
- Re-encryption oracle $\mathcal{O}_{re}(pk_i, pk_j, \mathrm{CT}_i)$: Given a second-level ciphertext CT_i and two public keys, pk_i and pk_j, which were generated by the oracle $\mathcal{O}_{pk}$ beforehand, return the re-encrypted first-level ciphertext $\mathrm{CT}_j \leftarrow \mathsf{ReEnc}(pk_i, \mathsf{ReKeyGen}(sk_i, pk_j), \mathrm{CT}_i)$ to $\mathcal{A}$.
- First-level decryption oracle $\mathcal{O}_{1d}(pk_j, \mathrm{CT}_j)$: Given a first-level ciphertext CT_j and a public key pk_j which was generated by the oracle $\mathcal{O}_{pk}$ beforehand, return the result of $\mathsf{Dec}_1(sk_j, \mathrm{CT}_j)$ to $\mathcal{A}$.

Weng et al. proved their scheme to be CCA-secure in the adaptive corruption model. This model allows the adversary $\mathcal{A}$ to adaptively corrupt users, thereby permitting them to query arbitrary secret keys from the secret key oracle during the find and guess phase. This stands in contrast to a selective corruption model, where the attacker must commit ahead of time to a certain set of users to corrupt in the setup.

Naturally, the oracle queries are constrained in such a way that it is not possible for the adversary to trivially win the security game by, for example, corrupting the target user of the challenge, submitting the challenge ciphertext directly to the decryption oracle, or using the re-encryption oracle on the challenge ciphertext and corrupting the target user of the re-encryption.

In accordance with the security game outlined above, Weng et al. defined security notions for both types of ciphertexts. A PRE scheme is called IND-2PRE-CCA-secure if the adversary's advantage in winning the security game for a second-level challenge ciphertext is negligible in the security parameter k. The definition of IND-1PRE-CCA is analogous for first-level ciphertexts. Finally, we consider the notion of master secret security (MSS-PRE), also referred to as collusion-resistance, in [12,13]. This notion captures the requirement that it should not be possible for a dishonest proxy (holding $rk_{i\rightarrow j}$) and delegatee (holding sk_j) to reveal the delegator's secret key sk_i by colluding with each other. It can be shown that if a PRE scheme is IND-1PRE-CCA-secure, it is also MSS-PRE-secure.

Additionally, Weng et al. base the security of their scheme on target collision-resistant (TCR) hash function families and pseudorandom function families (PRFs). A hash function family $\mathcal{H}$ is said to be TCR if it is infeasible for an adversary, given a random hash function H from the family $\mathcal{H}$ and a random element x, to find another element y such that $H(x) = H(y)$. Further, a function family $\mathcal{F}$ is said to be a PRF if it is infeasible for an adversary to distinguish $\mathcal{F}$ from a true random function family.

3. Automated Transformation of the PRE Scheme

We begin this section with a brief review of the PRE scheme by Weng et al. [4], which we take as the base scheme for transforming into the asymmetric pairing setting. Next, we use the automated transformation by Akinyele et al. [20], which translates cryptographic schemes defined in the Type 1 setting to the Type 3 setting. We apply the transformation to the base scheme by using the software tool provided by the publication and obtain a variant of the scheme defined in the Type 3 setting. Finally, we analyze the resulting scheme and identify flaws, which we address in Section 4.

3.1. Symmetric Pairing-Based Scheme by Weng et al. [4]

Below (see Algorithms 1–8), we briefly review the scheme introduced by Weng et al. [4] in the symmetric setting, where k, ℓ, and ℓ_1 are security parameters.

Algorithm 1 $\mathsf{Setup}(1^k)$

1: Choose Type 1 bilinear groups $\mathbb{G}, \mathbb{G}_T$ of prime order $p > 2^k$
2: $g, g_1, u, v, w \xleftarrow{\$} \mathbb{G}$
3: $Z = e(g, g)$
4: Choose a TCR hash function
$H\colon \mathbb{G} \times \{0,1\}^{\ell} \to \mathbb{Z}_p^*$
5: Choose a PRF
$F\colon \mathbb{G}_T \times \mathbb{G} \to \{0,1\}^{\ell-\ell_1} \times \{0,1\}^{\ell_1}$
6: **return** $param = (p, \mathbb{G}, \mathbb{G}_T, g, g_1, u, v, w, Z, H, F, \ell_1, \ell)$

Algorithm 2 $\mathsf{KeyGen}(1^k)$

1: $x_i \xleftarrow{\$} \mathbb{Z}_p^*$
2: $pk_i = g^{x_i}$
3: $sk_i = x_i$
4: **return** (pk_i, sk_i)

Algorithm 3 $\mathsf{ReKeyGen}(sk_i, pk_j)$

1: $rk_{i \to j} = pk_j^{1/sk_i} = g^{x_j/x_i}$
2: **return** $rk_{i \to j}$

Algorithm 4 $\mathsf{Enc}_2(pk_i, m)$

1: $r \xleftarrow{\$} \mathbb{Z}_p^*$
2: $C_1 = g_1^r$
3: $C_2 = pk_i^r$
4: $K = Z^r$
5: $C_3 = [F(K, C_1)]_{\ell-\ell_1} || ([F(K, C_1)]^{\ell_1} \oplus m)$
6: $t \xleftarrow{\$} \mathbb{Z}_p^*$
7: $h = H(C_1, C_3)$
8: $C_4 = (u^h v^t w)^r$
9: **return** $\mathrm{CT}_i = (t, C_1, C_2, C_3, C_4)$

Algorithm 5 $\mathsf{Enc}_1(pk_j, m)$

$r \xleftarrow{\$} \mathbb{Z}_p^*$
$C_1 = g_1^r$
$C_2' = e(pk_j, g)^r$
$K = Z^r$
$C_3 = [F(K, C_1)]_{\ell-\ell_1} || ([F(K, C_1)]^{\ell_1} \oplus m)$
$t \xleftarrow{\$} \mathbb{Z}_p^*$
$h = H(C_1, C_3)$
$C_4 = (u^h v^t w)^r$
return $\mathrm{CT}_j = (t, C_1, C_2', C_3, C_4)$

Algorithm 6 $\mathsf{ReEnc}(pk_i, rk_{i\to j}, \mathrm{CT}_i)$

$(t, C_1, C_2, C_3, C_4) \leftarrow \mathrm{CT}_i$
$h = H(C_1, C_3)$
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$
if $e(C_1, pk_i^{r_1}(u^h v^t w)^{r_2}) \neq e(C_2^{r_1} C_4^{r_2}, g_1)$ **then**
    return $\perp$
$C_2' = e(C_2, rk_{i\to j})$
return $\mathrm{CT}_j = (t, C_1, C_2', C_3, C_4)$

Algorithm 7 $\mathsf{Dec}_2(sk_i, \mathrm{CT}_i)$

$(t, C_1, C_2, C_3, C_4) \leftarrow \mathrm{CT}_i$
$h = H(C_1, C_3)$
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$
if $e(C_1, pk_i^{r_1}(u^h v^t w)^{r_2}) \neq e(C_2^{r_1} C_4^{r_2}, g_1)$ **then**
    return $\perp$
$K = e(C_2, g)^{1/sk_i}$
if $[F(K, C_1)]_{\ell-\ell_1} \neq [C_3]_{\ell-\ell_1}$ **then**
    return $\perp$
return $m = [F(K, C_1)]^{\ell_1} \oplus [C_3]^{\ell_1}$

Algorithm 8 $\mathsf{Dec}_1(sk_i, \mathrm{CT}_i)$

$(t, C_1, C_2, C_3, C_4) \leftarrow \mathrm{CT}_i$
$h = H(C_1, C_3)$
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$
if $e(C_1, u^h v^t w) \neq e(C_4, g_1)$ **then**
    return $\perp$
$K = C_2'^{1/sk_j}$
if $[F(K, C_1)]_{\ell-\ell_1} \neq [C_3]_{\ell-\ell_1}$ **then**
    return $\perp$
return $m = [F(K, C_1)]^{\ell_1} \oplus [C_3]^{\ell_1}$

3.2. Applying the Automated Transformation

For the automated transformation of the scheme from the symmetric to the asymmetric setting, we used the method by Akinyele et al. [20]. The method is based on the theoretical framework introduced by Abe et al. [19]. We note that there exists an improvement to this method presented by Abe et al. [21]; however, it only speeds up the transformation process and has no effect on the resulting scheme itself. We used the method by Akinyele et al. because the authors made their proposed transformation available as an open-source

tool (https://github.com/JHUISI/auto-tools/tree/88d20b0/auto_group, (accessed on 31 January 2024)). The application of the transformation resulted in the following scheme. Differences from the base scheme are highlighted in gray (Algorithms 9–16).

Algorithm 9 $\mathsf{Setup}(1^k)$

Choose Type 3 bilinear groups $\mathbb{G}_1$, $\hat{\mathbb{G}}_2$, $\mathbb{G}_T$ of prime order $p > 2^k$
$g \xleftarrow{\$} \mathbb{G}_1$, $\hat{g} \xleftarrow{\$} \hat{\mathbb{G}}_2$
$a \xleftarrow{\$} \mathbb{Z}_p^*, g_1 = g^a, \hat{g}_1 = \hat{g}^a$
$b \xleftarrow{\$} \mathbb{Z}_p^*, u = g^b, \hat{u} = \hat{g}^b$
$c \xleftarrow{\$} \mathbb{Z}_p^*, v = g^c, \hat{v} = \hat{g}^c$
$d \xleftarrow{\$} \mathbb{Z}_p^*, w = g^d, \hat{w} = \hat{g}^d$
$Z = e(g, \hat{g})$
Choose a TCR hash function
$H\colon \mathbb{G}_1 \times \{0,1\}^\ell \to \mathbb{Z}_p^*$
Choose a PRF
$F\colon \mathbb{G}_T \times \mathbb{G}_1 \to \{0,1\}^{\ell-\ell_1} \times \{0,1\}^{\ell_1}$
return $param = (p, \mathbb{G}_1, \hat{\mathbb{G}}_2, \mathbb{G}_T, g, g_1, u, v, w, \hat{g}, \hat{g}_1, \hat{u}, \hat{v}, \hat{w}, Z, H, F, \ell_1, \ell)$

Algorithm 10 $\mathsf{KeyGen}(1^k)$

$x_i \xleftarrow{\$} \mathbb{Z}_p^*$
$pk_i = (pk_{i,1}, pk_{i,2}) = (g^{x_i}, \hat{g}^{x_i})$
$sk_i = x_i$
return (pk_i, sk_i)

Algorithm 11 $\mathsf{ReKeyGen}(sk_i, pk_j)$

$rk_{i \to j} = pk_{j,2}^{1/sk_i} = \hat{g}^{x_j/x_i}$
return $rk_{i \to j}$

Algorithm 12 $\mathsf{Enc}_2(pk_i, m)$

$r \xleftarrow{\$} \mathbb{Z}_p^*$
$C_1 = g_1^r$
$C_2 = pk_{i,1}^r$
$K = Z^r$
$C_3 = [F(K, C_1)]_{\ell-\ell_1} || ([F(K, C_1)]^{\ell_1} \oplus m)$
$t \xleftarrow{\$} \mathbb{Z}_p^*$
$h = H(C_1, C_3)$
$C_4 = (u^h v^t w)^r$
return $\mathrm{CT}_i = (t, C_1, C_2, C_3, C_4)$

Algorithm 13 $\mathsf{Enc}_1(pk_j, m)$

$r \xleftarrow{\$} \mathbb{Z}_p^*$
$C_1 = g_1^r$
$C_2' = e(pk_{j,1}, \hat{g})^r$
$K = Z^r$
$C_3 = [F(K, C_1)]_{\ell-\ell_1} || ([F(K, C_1)]^{\ell_1} \oplus m)$
$t \xleftarrow{\$} \mathbb{Z}_p^*$
$h = H(C_1, C_3)$
$C_4 = (u^h v^t w)^r$
return $\mathrm{CT}_j = (t, C_1, C_2', C_3, C_4)$

Algorithm 14 $\mathsf{ReEnc}(pk_i, rk_{i\to j}, \mathrm{CT}_i)$

$(t, C_1, C_2, C_3, C_4) \leftarrow \mathrm{CT}_i$
$h = H(C_1, C_3)$
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$
if $e(C_1, pk_{i,2}^{r_1}(\hat{u}^h \hat{v}^t \hat{w})^{r_2}) \neq e(C_2^{r_1} C_4^{r_2}, \hat{g}_1)$ **then**
    return $\perp$
$C_2' = e(C_2, rk_{i\to j})$
return $\mathrm{CT}_j = (t, C_1, C_2', C_3, C_4)$

Algorithm 15 $\mathsf{Dec}_2(sk_i, \mathrm{CT}_i)$

$(t, C_1, C_2, C_3, C_4) \leftarrow \mathrm{CT}_i$
$h = H(C_1, C_3)$
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$
if $e(C_1, pk_{i,2}^{r_1}(\hat{u}^h \hat{v}^t \hat{w})^{r_2}) \neq e(C_2^{r_1} C_4^{r_2}, \hat{g}_1)$ **then**
    return $\perp$
$K = e(C_2, \hat{g})^{1/sk_i}$
if $[F(K, C_1)]_{\ell-\ell_1} \neq [C_3]_{\ell-\ell_1}$ **then**
    return $\perp$
return $m = [F(K, C_1)]^{\ell_1} \oplus [C_3]^{\ell_1}$

Algorithm 16 $\mathsf{Dec}_1(sk_i, \mathrm{CT}_i)$

$(t, C_1, C_2, C_3, C_4) \leftarrow \mathrm{CT}_i$
$h = H(C_1, C_3)$
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$
if $e(C_1, \hat{u}^h \hat{v}^t \hat{w}) \neq e(C_4, \hat{g}_1)$ **then**
    return $\perp$
$K = C_2'^{1/sk_j}$
if $[F(K, C_1)]_{\ell-\ell_1} \neq [C_3]_{\ell-\ell_1}$ **then**
    return $\perp$
return $m = [F(K, C_1)]^{\ell_1} \oplus [C_3]^{\ell_1}$

3.3. Analysis of the Transformed Scheme

The scheme resulting from the automated transform retains the ciphertext size of the original scheme and adds a second group element to the public key. The main portion of the transformation to the asymmetric setting is realized by duplicating the generators from the public parameters into both source groups. However, for the practical application of the scheme, this leads to a problem. If we look at the public generator g_1 and its use in the ciphertext element C_1, we notice that if the discrete logarithm a of g_1 to base g is known, it can function as a backdoor since any ciphertext could be decrypted by computing $e(C_1, \hat{g})^{1/a} = e(g_1^r, \hat{g})^{1/a} = e(g^{ar}, \hat{g})^{1/a} = K$. In the base scheme, g_1 is chosen uniformly at random, which in practice could be achieved by hashing a nothing-up-my-sleeve value, e.g., digits of π or a fixed string into $\mathbb{G}$. In contrast, in the transformed scheme, a is explicitly chosen in the setup method to compute g_1 and $\hat{g}_1$, which would pose a virtually insurmountable hurdle to establishing trust in the scheme in a practical instantiation. To tackle this issue, we propose an alternative transformation of the scheme that does not exhibit this design flaw in the following section.

4. Our PRE Scheme Design

In this section, we present our manual transformation of the base scheme to the Type 3 setting. The transformation fixes the identified design flaw of the automated transformation by preventing the duplication of generators to both source groups. This introduces the need for adjustments to the security proof, which can be found in Section 4.2.

4.1. Manually Transformed Scheme

As in Section 3.2, differences to the base scheme are highlighted in gray (Algorithms 17–24).

Algorithm 17 $\mathsf{Setup}(1^k)$

1: Choose Type 3 bilinear groups $\mathbb{G}_1$, $\hat{\mathbb{G}}_2$, $\mathbb{G}_T$ of prime order $p > 2^k$
2: $g, u, v, w \xleftarrow{\$} \mathbb{G}$
3: $\hat{g}, \hat{g}_1 \xleftarrow{\$} \hat{\mathbb{G}}_2$
4: $Z = e(g, \hat{g})$
5: Choose a TCR hash function
$H\colon \hat{\mathbb{G}}_2 \times \{0,1\}^{\ell} \to \mathbb{Z}_p^*$
6: Choose a PRF
$F\colon \mathbb{G}_T \times \hat{\mathbb{G}}_2 \to \{0,1\}^{\ell-\ell_1} \times \{0,1\}^{\ell_1}$
7: **return** $param = (p, \mathbb{G}_1, \hat{\mathbb{G}}_2, \mathbb{G}_T, g, u, v, w, \hat{g}, \hat{g}_1, Z, H, F, \ell_1, \ell)$

Algorithm 18 $\mathsf{KeyGen}(1^k)$

1: $x_i \xleftarrow{\$} \mathbb{Z}_p^*$
2: $pk_i = (pk_{i,1}, pk_{i,2}) = (g^{x_i}, \hat{g}^{x_i})$
3: $sk_i = x_i$
4: **return** (pk_i, sk_i)

Algorithm 19 $\mathsf{ReKeyGen}(sk_i, pk_j)$

1: $rk_{i \to j} = pk_{j,2}^{1/sk_i} = \hat{g}^{x_j/x_i}$
2: **return** $rk_{i \to j}$

Algorithm 20 $\mathsf{Enc}_2(pk_i, m)$

$r \xleftarrow{\$} \mathbb{Z}_p^*$
$C_1 = \hat{g}_1^r$
$C_2 = pk_{i,1}^r$
$K = Z^r$
$C_3 = [F(K, C_1)]_{\ell-\ell_1} || ([F(K, C_1)]^{\ell_1} \oplus m)$
$t \xleftarrow{\$} \mathbb{Z}_p^*$
$h = H(C_1, C_3)$
$C_4 = (u^h v^t w)^r$
return $\mathrm{CT}_i = (t, C_1, C_2, C_3, C_4)$

Algorithm 21 $\mathsf{Enc}_1(pk_j, m)$

$r \xleftarrow{\$} \mathbb{Z}_p^*$
$C_1 = \hat{g}_1^r$
$C_2' = e(pk_{j,1}, \hat{g})^r$
$K = Z^r$
$C_3 = [F(K, C_1)]_{\ell-\ell_1} || ([F(K, C_1)]^{\ell_1} \oplus m)$
$t \xleftarrow{\$} \mathbb{Z}_p^*$
$h = H(C_1, C_3)$
$C_4 = (u^h v^t w)^r$
return $\mathrm{CT}_j = (t, C_1, C_2', C_3, C_4)$

Algorithm 22 $\mathsf{ReEnc}(pk_i, rk_{i\to j}, \mathrm{CT}_i)$

$(t, C_1, C_2, C_3, C_4) \leftarrow \mathrm{CT}_i$
$h = H(C_1, C_3)$
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$
if $e(pk_{i,1}^{r_1}(u^h v^t w)^{r_2}, C_1) \neq e(C_2^{r_1} C_4^{r_2}, \hat{g}_1)$ **then**
    return $\perp$
$C_2' = e(C_2, rk_{i\to j})$
return $\mathrm{CT}_j = (t, C_1, C_2', C_3, C_4)$

Algorithm 23 $\mathsf{Dec}_2(sk_i, \mathrm{CT}_i)$

$(t, C_1, C_2, C_3, C_4) \leftarrow \mathrm{CT}_i$
$h = H(C_1, C_3)$
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$
if $e(pk_{i,1}^{r_1}(u^h v^t w)^{r_2}, C_1) \neq e(C_2^{r_1} C_4^{r_2}, \hat{g}_1)$ **then**
    return $\perp$
$K = e(C_2, \hat{g})^{1/sk_i}$
if $[F(K, C_1)]_{\ell-\ell_1} \neq [C_3]_{\ell-\ell_1}$ **then**
    return $\perp$
return $m = [F(K, C_1)]^{\ell_1} \oplus [C_3]^{\ell_1}$

Algorithm 24 $\mathsf{Dec}_1(sk_i, \mathrm{CT}_i)$

$(t, C_1, C_2, C_3, C_4) \leftarrow \mathrm{CT}_i$
$h = H(C_1, C_3)$
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p^*$
if $e(u^h v^t w, C_1) \neq e(C_4, \hat{g}_1)$ **then**
    return $\perp$
$K = C_2'^{1/sk_j}$
if $[F(K, C_1)]_{\ell-\ell_1} \neq [C_3]_{\ell-\ell_1}$ **then**
    return $\perp$
return $m = [F(K, C_1)]^{\ell_1} \oplus [C_3]^{\ell_1}$

4.2. Transforming the Security Proof

The base scheme has been proven to be IND-2PRE-CCA-secure and IND-1PRE-CCA-secure. Given the analogous structure of the two proofs, the strategy for transforming the proofs to the Type 3 setting is applicable to both.

In the manually transformed scheme, $C_1 \in \mathbb{G}_1$ is chosen from a different group than $C_2, C_3, C_4 \in \mathbb{G}_2$ and the generators are not duplicated to both source groups. This prevents the implicit transformation of the security proofs from the base scheme that would otherwise occur through the automated transformation. Consequently, we present the necessary adjustments for transforming the proof to the Type 3 setting.

As indicated in Section 2.3, we replace the symmetric pairing-based assumption used by Weng et al. in the base scheme with the 3-wDBDHI$_3$ assumption (Definition 1) in the security proof. In the transformed proof, the challenger $\mathcal{B}$ is thus given a 3-wDBDHI$_3$ instance $(g, A_{-1} = g^{1/a}, A_1 = g^a, A_2 = g^{(a^2)}, B = g^b, \hat{g}, \hat{A}_{-1} = \hat{g}^{1/a}, \hat{A}_1 = \hat{g}^a, \hat{A}_2 = \hat{g}^{(a^2)}, \hat{B} = \hat{g}^b, Q) \in \mathbb{G}_1^5 \times \hat{\mathbb{G}}_2^5 \times \mathbb{G}_T$ with unknown $a, b \xleftarrow{\$} \mathbb{Z}_p^*$, which is then used in the setup phase to provide the adversary $\mathcal{A}$ with the public parameters $u = A_1^{\alpha_1} A_2^{\beta_1}, v = A_1^{\alpha_2} A_2^{\beta_2}$, $w = A_1^{\alpha_3} A_2^{\beta_3}$, and $\hat{g}_1 = \hat{A}_2^{\alpha_4}$ for random $\alpha_1, \alpha_2, \alpha_3, \alpha_4, \beta_1, \beta_2, \beta_3 \xleftarrow{\$} \mathbb{Z}_p^*$, analogous to the proof of the base scheme. The majority of the proof of the base scheme can be transformed by making use of either A_i or $\hat{A}_i$ according to the required group or by adjusting the positions of variables in a pairing to match the order of the source groups. However, in the proof of the base scheme both the re-encryption and the first-level decryption oracle recover $K = e(g, \hat{g})^r$ by computing $K = e(A_{-1}, A_1^r)$, where

$$A_1^r = \left(\frac{C_4}{C_1^{\frac{\beta_1 h + \beta_2 t + \beta_3}{\alpha_0}}} \right)^{\frac{1}{\alpha_1 h + \alpha_2 t + \alpha_3}}.$$

This is not feasible for the manually transformed scheme since $C_1 \in \hat{\mathbb{G}}_2$ and $C_4 \in \mathbb{G}_1$ are incompatible. Fortunately, there is an alternative approach to recovering K from C_1 and C_4, which leverages the bilinearity of the pairing e by computing

$$K = e(g, \hat{g})^r = \left(\frac{e(C_4, \hat{A}_{-1})}{e(A_{-1}, C_1^{1/\alpha_4})^{\beta_1 h + \beta_2 t + \beta_3}} \right)^{\frac{1}{\alpha_1 h + \alpha_2 t + \alpha_3}},$$

since

$$\begin{aligned} e(C_4, \hat{A}_{-1}) &= e((A_1^{\alpha_1 h + \alpha_2 t + \alpha_3} A_2^{\beta_1 h + \beta_2 t + \beta_3})^r, \hat{A}_{-1}) \\ &= e(g, \hat{g})^{r(\alpha_1 h + \alpha_2 t + \alpha_3)} e(g, \hat{g})^{ar(\beta_1 h + \beta_2 t + \beta_3)} \end{aligned}$$

and

$$e(A_{-1}, C_1^{1/\alpha_4}) = e(A_{-1}, \hat{A}_2^r) = e(g, \hat{g})^{ar}.$$

The application of these adjustments allows for the transformation of both the chosen-ciphertext security proofs at the first and second level of the base scheme to the Type 3 setting for the manually transformed scheme.

4.3. Performance Evaluation

We compare the performance and ciphertext size of the schemes resulting from the automated transformation (Section 3.2) and the manual transformation (Section 4.1). To that end, we implemented the schemes using mcl-wasm (https://github.com/herumi/mcl, (accessed on 1 March 2024)) [7] with the BLS12-381 curve, roughly targeting the 128-bits security level [22].

As we can see in Table 1, the manually transformed scheme outperforms the automatically transformed scheme in decryption and re-encryption at the cost of slightly slower encryption and larger ciphertexts. This can be explained by the difference that the manual transformation chooses C_1 to be in $\hat{\mathbb{G}}_2$ instead of $\mathbb{G}_1$ as in the automated transformation. This leads to the fact that the manually transformed scheme performs more computations in $\hat{\mathbb{G}}_2$ for the encryption, whereas the automatically transformed scheme carries this out in the ciphertext validation, which is required when re-encrypting and decrypting. Since in the Type 3 setting $\hat{\mathbb{G}}_2$ is usually defined over an extension field, group operations are slower in $\hat{\mathbb{G}}_2$ than in $\mathbb{G}_1$. Furthermore, due to this fact, the representation of group elements of $\hat{\mathbb{G}}_2$ is larger than in $\mathbb{G}_1$, leading to the slightly larger ciphertext sizes.

Table 1. Ciphertext sizes and benchmark of the automatically and manually transformed schemes on an Intel Core i7-6600U CPU with 16 GB of RAM.

Method/Scheme	Automated	Manual
KeyGen	1.5 ms	1.5 ms
ReKeyGen	1.0 ms	1.0 ms
Encrypt2	4.3 ms	4.7 ms
Encrypt1	9.3 ms	9.7 ms
ReEncrypt	20.1 ms	17.8 ms
Decrypt2	21.6 ms	18.8 ms
Decrypt1	13.7 ms	12.5 ms
Second-Level Ciphertext Size	208 B	256 B
First-Level Ciphertext Size	736 B	784 B

5. Conclusions

In conclusion, we presented the first unidirectional PRE scheme based on asymmetric pairings. We started with the application of the automated black-box reduction technique by Akinyele et al. [20] to transform the base scheme and its security proof from the Type 1 to the Type 3 setting. Our findings reveal that relying solely on this automated technique does not necessarily produce a scheme suitable for practical use, as it introduced flaws that rendered the resulting scheme impractical. To address these issues, we proposed an adapted manually transformed scheme. The scheme retains the properties of the base scheme, achieving CCA security under adaptive corruptions in the standard model. We refined the hardness assumption for the Type 3 setting, ensuring that it is at least as hard as in the base scheme, and outlined the necessary adjustments to the security proof. The transformed scheme enables the use of superior Type 3 pairing-friendly curves, thereby superseding the obsolete Type 1 setting of the base scheme.

The manual transformation is essential as it fixes the flaws present in the automatically transformed scheme, as discussed in Section 3.3, making it suitable for practical use. While this comes with slight computational overhead associated with encryption performance and increased ciphertext size, it also results in faster re-encryption and decryption, contributing to the scheme's overall practicality. Therefore, we argue that our manually transformed

scheme represents a preferable choice for the practical instantiation of a Type 3 pairing-based PRE scheme, while still being faster and more secure overall than the base scheme. Our scheme offers an alternative to more complex pairing-free PRE schemes, providing a practical and efficient solution for real-world applications.

Future work will focus on leveraging the distinctive properties of Type 3 pairings to construct an even more efficient asymmetric pairing-based PRE scheme or create a PRE scheme with advanced properties that is CCA-secure in the standard model.

Author Contributions: Conceptualization, B.Z. and N.B.; methodology, B.Z.; software, B.Z.; investigation, B.Z.; writing—original draft preparation, B.Z., N.B., P.D. and M.M.; writing—review and editing, B.Z., N.B., P.D. and M.M.; visualization, B.Z.; supervision, N.B.; project administration, B.Z. and N.B.; funding acquisition, M.M. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Blaze, M.; Bleumer, G.; Strauss, M. Divertible Protocols and Atomic Proxy Cryptography. In Proceedings of the EUROCRYPT, Espoo, Finland, 31 May–4 June 1998; Nyberg, K., Ed.; LNCS; Springer: Berlin/Heidelberg, Germany, 1998; Volume 1403, pp. 127–144. [CrossRef]
2. Ateniese, G.; Fu, K.; Green, M.; Hohenberger, S. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.* **2006**, *9*, 1–30. [CrossRef]
3. Libert, B.; Vergnaud, D. Unidirectional Chosen-Ciphertext Secure Proxy Re-encryption. In Proceedings of the PKC 2008, Barcelona, Spain, 9–12 March 2008; Cramer, R., Ed.; LNCS; Springer: Berlin/Heidelberg, Germany, 2008; Volume 4939, pp. 360–379. [CrossRef]
4. Weng, J.; Chen, M.; Yang, Y.; Deng, R.; Chen, K.; Bao, F. CCA-secure Unidirectional Proxy Re-Encryption in the Adaptive Corruption Model without Random Oracles. *Sci. China Inf. Sci.* **2010**, *53*, 593–606. [CrossRef]
5. Galbraith, S.D.; Paterson, K.G.; Smart, N.P. Pairings for cryptographers. *Discret. Appl. Math.* **2008**, *156*, 3113–3121. [CrossRef]
6. Zhang, X.; Wang, K. Fast Symmetric Pairing Revisited. In Proceedings of the Pairing 2013, Beijing, China, 22–24 November 2013; Cao, Z., Zhang, F., Eds.; LNCS; Springer: Berlin/Heidelberg, Germany, 2013; Volume 8365, pp. 131–148. [CrossRef]
7. Beuchat, J.L.; González-Díaz, J.E.; Mitsunari, S.; Okamoto, E.; Rodríguez-Henríquez, F.; Teruya, T. High-Speed Software Implementation of the Optimal Ate Pairing over Barreto-Naehrig Curves. In Proceedings of the Pairing 2010, Yamanaka Hot Spring, Japan, 13–15 December 2010; Joye, M., Miyaji, A., Otsuka, A., Eds.; LNCS; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6487, pp. 21–39. [CrossRef]
8. Aranha, D.F.; Karabina, K.; Longa, P.; Gebotys, C.H.; López, J. Faster Explicit Formulas for Computing Pairings over Ordinary Curves. In Proceedings of the EUROCRYPT 2011, Tallinn, Estonia, 15–19 May 2011; Paterson, K.G., Ed.; LNCS; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6632, pp. 48–68. [CrossRef]
9. Uzunkol, O.; Kiraz, M.S. Still wrong use of pairings in cryptography. *Appl. Math. Comput.* **2018**, *333*, 467–479. [CrossRef]
10. Chatterjee, S.; Menezes, A. On cryptographic protocols employing asymmetric pairings—The role of Ψ revisited. *Discret. Appl. Math.* **2011**, *159*, 1311–1322. [CrossRef]
11. Shao, J.; Cao, Z. CCA-Secure Proxy Re-encryption without Pairings. In Proceedings of the PKC 2009, Irvine, CA, USA, 18–20 March 2009; Jarecki, S., Tsudik, G., Eds.; LNCS; Springer: Berlin/Heidelberg, Germany, 2009; Volume 5443, pp. 357–376. [CrossRef]
12. Chow, S.S.M.; Weng, J.; Yang, Y.; Deng, R.H. Efficient Unidirectional Proxy Re-Encryption. In Proceedings of the AFRICACRYPT 2010, Stellenbosch, South Africa, 3–6 May 2010; Bernstein, D.J., Lange, T., Eds.; LNCS; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6055, pp. 316–332. [CrossRef]
13. Selvi, S.S.D.; Paul, A.; Pandurangan, C. A Provably-Secure Unidirectional Proxy Re-encryption Scheme Without Pairing in the Random Oracle Model. In Proceedings of the CANS 2017, Hong Kong, China, 30 November–2 December 2017; Capkun, S., Chow, S.S.M., Eds.; LNCS; Springer: Berlin/Heidelberg, Germany, 2017; Volume 11261, pp. 459–469. [CrossRef]
14. Dutta, P.; Susilo, W.; Duong, D.H.; Roy, P.S. Collusion-resistant identity-based Proxy Re-encryption: Lattice-based constructions in Standard Model. *Theor. Comput. Sci.* **2021**, *871*, 16–29. [CrossRef]

15. Susilo, W.; Dutta, P.; Duong, D.H.; Roy, P.S. Lattice-Based HRA-secure Attribute-Based Proxy Re-Encryption in Standard Model. In Proceedings of the ESORICS 2021, Darmstadt, Germany, 4–8 October 2021; Bertino, E., Schulmann, H., Waidner, M., Eds.; LNCS; Springer: Berlin/Heidelberg, Germany, 2021; Volume 12973, pp. 169–191. [CrossRef]
16. Zhou, Y.; Liu, S.; Han, S.; Zhang, H. Fine-Grained Proxy Re-encryption: Definitions and Constructions from LWE. In Proceedings of the ASIACRYPT 2023, Guangzhou, China, 4–8 December 2023; Guo, J., Steinfeld, R., Eds.; LNCS; Springer: Berlin/Heidelberg, Germany, 2023; Volume 14443, pp. 199–231. [CrossRef]
17. Zhou, Y.; Liu, S.; Han, S. Multi-hop Fine-Grained Proxy Re-encryption. In Proceedings of the PKC 2024, Sydney, NSW, Australia, 15–17 April 2024; Tang, Q., Teague, V., Eds.; LNCS; Springer: Berlin/Heidelberg, Germany, 2024; Volume 14604, pp. 161–192. [CrossRef]
18. Barbulescu, R.; Duquesne, S. Updating Key Size Estimations for Pairings. *J. Cryptol.* **2019**, *32*, 1298–1336. [CrossRef]
19. Abe, M.; Groth, J.; Ohkubo, M.; Tango, T. Converting Cryptographic Schemes from Symmetric to Asymmetric Bilinear Groups. In Proceedings of the CRYPTO 2014, Santa Barbara, CA, USA, 17–21 August 2014; Garay, J.A., Gennaro, R., Eds.; LNCS; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8616, pp. 241–260. [CrossRef]
20. Akinyele, J.A.; Garman, C.; Hohenberger, S. Automating Fast and Secure Translations from Type-I to Type-III Pairing Schemes. In Proceedings of the CCS 2015, Denver, CO, USA, 12–16 October 2015; Ray, I., Li, N., Kruegel, C., Eds.; ACM: New York, NY, USA, 2015; pp. 1370–1381. [CrossRef]
21. Abe, M.; Hoshino, F.; Ohkubo, M. Design in Type-I, Run in Type-III: Fast and Scalable Bilinear-Type Conversion Using Integer Programming. In Proceedings of the CRYPTO 2016, Santa Barbara, CA, USA, 14–18 August 2016; Robshaw, M., Katz, J., Eds.; LNCS; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9816, pp. 387–415. [CrossRef]
22. Bowe, S. BLS12-381: New zk-SNARK Elliptic Curve Construction. Available online: https://electriccoin.co/blog/new-snark-curve/ (accessed on 10 October 2024).