cryptography

MDPI

Review

On Multiple Encryption for Public-Key Cryptography

Tudor Soroceanu *, Nicolas Buchmann and Marian Margraf

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

Abstract: Using multiple, individual encryption schemes is a well-established method to increase the overall security of encrypted data. These so-called *multiple encryption*or *hybrid* schemes have regained traction in the context of public-key cryptography due to the rise of quantum computers, since it allows the combination of well-known classical encryption schemes with novel post-quantum schemes. In this paper, we conduct a survey of the state-of-the-art *public-key multiple encryption* (M-PKE) schemes. For the first time, we describe the most relevant M-PKE schemes in detail and discuss their security in a unified model, which allows better comparison between the schemes. Hence, we compare the security, efficiency, and complexity of the schemes and offer recommendations for usage based on common use cases. Our survey emphasizes the importance of being deliberate when combining encryption schemes, as small nuances can easily break security.

Keywords: multiple encryption; hybrid encryption; combiners; public-key cryptography; post-quantum cryptography; quantum resistance

Citation: Soroceanu, T.; Buchmann, N.; Margraf, M. On Multiple Encryption for Public-Key Cryptography. *Cryptography* **2023**, *7*, 49. https://doi.org/10.3390/cryptography7040049

Academic Editor: Josef Pieprzyk

Received: 8 September 2023
Revised: 2 October 2023
Accepted: 4 October 2023
Published: 6 October 2023

1. Introduction

A cornerstone of today's information technology is modern cryptography. Public-key cryptography—like the RSA system [1] or the Diffie–Hellman key-exchange [2]—is threatened by recent advances in quantum computing [3,4]. A way to counter this threat is the development and standardization of *post-quantum cryptography* (PQC). The *National Institute of Standards and Technology* (NIST) conducted a standardization competition for PQC algorithms and announced in 2022 the first group of winners of this competition [5]. Nevertheless, the novel PQC algorithms are not as thoroughly studied as the well-known classical public-key cryptography. For example, last year, the promising NIST-submissions Rainbow and SIKE were unexpectedly broken [6,7]. An established way to strengthen cryptographic primitives in such a situation is to utilize redundancy by combining multiple schemes at once in a secure way.

Multiple encryption—also known as "double encryption", "combiners", or "hybrid encryption"—allows the combination of multiple, individual cryptographic schemes in order to create a new secure scheme. Such a new combined scheme resists attacks that target only one of its components. It is possible to combine the same encryption scheme several times with different keys, but usually, schemes based on different computational assumptions are used. In the context of migrating to a fully post-quantum infrastructure, this means combining well-known classical cryptosystems with novel post-quantum cryptosystems. This increases the chances of defending against both classical and quantum cryptanalytic advances in the future. Therefore, hybrid public-key schemes have become a focus of current research in the context of PQC, with particular emphasis on key-encapsulation mechanisms (KEMs) and digital signatures. The systematic mapping study of Giron et al. [8] highlights the growing interest in hybrid PQC key-exchange. Note that the term "hybrid encryption" is more commonly used for the combination of KEMs and data encapsulation mechanisms (DEMs). In this case, both components have to be secure in order to securely exchange data. We use the terms "multiple encryption", "hybrid

encryption", or "(hybrid) combiner" interchangeable for "public-key multiple encryption" when the context is clear.

While hybrid KEMs and hybrid digital signatures have recently received increased attention in the context of PQC, the research into hybrid asymmetric encryption has been more or less neglected. This research gap is filled by the paper at hand providing an overview of plain public-key multiple encryption and its development over the past decades. We summarize the state of the art in this area, provide a deeper understanding of hybrid public-key schemes and identify open questions in the field. Besides presenting the constructions of the most relevant multiple encryption schemes in detail, we also discuss security, efficiency, and complexity of the schemes. At the end, we provide recommendations for selecting multiple encryption schemes based on common use cases. This document provides guidance for decision making while choosing hybrid encryption schemes in the design of new security protocols.

The remainder of the paper is organized as follows. The related work with emphasis on public-key multiple encryption is presented in Section 2. In Section 3, we provide the relevant background—namely definitions and security notions—to make this work self-contained. The category of sequential hybrid schemes is presented in Section 4, while parallel hybrid schemes are presented in Section 5. Section 6 discusses combined sequential–parallel constructions. The advantages and disadvantages of the individual schemes as well as recommendations are discussed in Section 7. Conclusions are drawn in Section 8, and Section 9 offers an outlook for future work.

2. Related Work

The combination of multiple security schemes dates back to Shannon [9], who proposed the use of the "product" of two secrecy systems in order to enhance the overall security. This sequential construction—sometimes called cascade cipher—was later the focus of research on block ciphers [10,11] and lead to the development and security analysis of Triple DES [12,13].

In the context of public-key cryptosystems, one of the earliest multiple encryption schemes was proposed by Asmuth and Blakley [14], who used the XOR operator to mask the plaintext message once for each individual component cipher. Herzberg [15] studied the security of multiple "folklore" combiners, such as the XOR-input combiner or the public-key sequential construction. The biggest weakness of all these natural constructions is that they are not secure against adaptive chosen-ciphertext attacks (IND-CCA). This is mainly due to the fact that the natural constructions are not able to check for malicious ciphertexts during the decryption. Consequently, Zhang et al. [16] and Dodis and Katz [17] presented independently new IND-CCA secure asymmetric multiple encryption schemes. While Zhang et al. used hash functions to construct an IND-CCA secure scheme from weaker components in the *random oracle model* (ROM), Dodis and Katz made use of additional primitives, such as secret sharing schemes and digital signatures, to provide an IND-CCA secure scheme in the standard model. Fujioka et al. [18] presented an IND-CCA secure sequential cipher that is more efficient than previous schemes and shows more similarities to the natural sequential combiner. Hohenberger et al. [19] also offered an IND-CCA secure construction from weaker components. Their focus was on defining a *detectable*-CCA (IND-dCCA) security and using this to achieve IND-CCA security. The drawback of their construction is that each component must remain secure. Recently, Goncalves and Mashatan [20] presented the first public-key multiple encryption scheme that is secure in the *quantum random oracle model* (QROM). Currently, their hybrid scheme remains the only one that has been analyzed in the ROM and in the QROM.

The concept of the combination of multiple schemes was formalized by Herzberg [15] and Harnik et al. [21] as (k, n)-robust combiners, where n is the number of all components and k is the threshold number of secure components. Their definition has the disadvantage that it is limited to using the same security notion for the entire hybrid scheme and the individual underlying encryption schemes. This does not allow the construction of

hybrid schemes from weaker components, and it is not possible to further adapt the security notions for the multiple encryption setting. Stronger security notions were defined in [17,18,22] that allow the reception of more information by the adversary or the oracle in the indistinguishability games. These notions are further discussed in Section 3.5.

Hybrid combiners are also applicable to other primitives and domains, including oblivious transfer [21], obfuscation [23] and functional encryption [24].

With the advent of quantum computers, hybrid schemes are becoming more interesting for two cryptographic primitives in particular: key encapsulation and digital signatures. Giacon et al. [25] considered hybrid KEM combiners while focusing on classical adversaries. Bindel et al. [26] considered key exchange and encapsulation against quantum and classical adversaries. They formalized the concept of quantum adversaries and their capabilities. The efficient construction of hybrid PQC-KEM combiners from weaker PKEs as a generalization of the Fujisaki–Okamoto (FO) transform was considered by Huguenin–Dumittan and Vaudenay [27]. A systematic mapping study for post-quantum hybrid key exchange was conducted by Giron et al. [8].

The development of hybrid post-quantum schemes has also been advanced by technology companies. In 2016, Google temporarily integrated a hybrid key exchange into an experimental build of its web browser Chrome [28] using elliptic curve Diffie–Hellman (ECDH) and the PQC scheme NewHope [29]. A follow-up experiment was conducted in 2018, this time using the scheme HRSS [30] for the post-quantum part and the ECDH algorithm X25519 for the classical part. The identical combination is now used by Google to secure its internal Application Layer Transport Security [31]. Other companies continue to develop hybrid schemes and their integration in real-world infrastructures, like Cloudfare [32], Microsoft Research [33], and Amazon [34].

3. Preliminaries

In this section, we introduce the necessary preliminaries for the remainder of this paper. We first cover the used notation before formally defining public-key (multiple) encryption and the corresponding security notions.

3.1. Notation

We denote by $\mathcal{A}(x_1, x_2, \ldots)$ the output of a probabilistic algorithm $\mathcal{A}$ on inputs $x_1, x_2, \ldots$ where the required randomness is determined by internal random coin tosses that are uniformly at random. If needed, the outcome r of the random coin tosses is given as an auxiliary input to $\mathcal{A}$, denoted by $\mathcal{A}(x_1, x_2, \ldots; r)$. The notation $y \leftarrow \mathcal{A}(\cdot)$ assigns the output of $\mathcal{A}$ to y. A subroutine of an algorithm $\mathcal{A}$ is called by $\mathcal{A}.\texttt{Subroutine}(\cdot)$. We let $\mathcal{S}$ be a set; then, $b \xleftarrow{\$} \mathcal{S}$ assigns to b a uniformly at-random chosen element from $\mathcal{S}$. All adversaries for the indistinguishability security are considered to be probabilistic polynomial time (PPT) algorithms in their input length. We call function $\varepsilon : \mathbb{N} \to \mathbb{R}$ *negligible* if for all $c > 0$ there exists an integer k_c such that $\varepsilon(k) \leq k^{-c}$ for all $k > k_c$.

The concatenation of two strings a, b is denoted by $a||b$. The bit-wise exclusive or (XOR) of two strings a, b is denoted by $a \oplus b$. The magnitude of a set $\mathcal{S}$ is given by $|\mathcal{S}|$, and the bit-length of a string a is given by $|a|$.

3.2. Public-Key Multiple Encryption

We first offer the definition of public-key encryption (PKE), before defining public-key multiple encryption (M-PKE).

Definition 1 (Public-Key Encryption Scheme)**.** *A* public-key encryption scheme Π *is given by a triple of algorithms* $(\mathcal{K}, \mathcal{E}, \mathcal{D})$ *defined over a key space* $\mathcal{X}$*, a message space* $\mathcal{M}$ *and a ciphertext space* $\mathcal{C}$*, where*

- $\mathcal{K}$*, the key generation algorithm, is a probabilistic algorithm that takes as the input a security parameter* $k, k \in \mathbb{N}$ *as* 1^k*, generates system parameters* $\mathcal{P}$ *for* Π *and outputs a related pair* (pk, sk) *of public and secret keys.*

- *$\mathcal{E}$, the encryption algorithm, is a probabilistic algorithm that takes as the input a public key pk and a message $m \in \mathcal{M}$ and outputs a ciphertext $c \in \mathcal{C}$. If necessary, we explicitly state the randomness r used for the encryption as $c =: \mathcal{E}(pk, m; r)$. Otherwise, the randomness is internally chosen uniformly at random.*
- *$\mathcal{D}$, the decryption algorithm, is a deterministic algorithm that takes as the input a secret key sk and a ciphertext $c \in \mathcal{C}$ and outputs a message $m \in \mathcal{M}$ or a designated rejection symbol $\perp$ to indicate a failed decryption.*

For readability, we omit the key space and the system parameters from now on.

Correctness of PKE. We consider encryption schemes whose decryption sometimes fails. This consideration is particularly relevant for PQC schemes, which typically have a small probability of decryption failure. Informally, a PKE scheme is correct if the probability that the decryption does not return the original message is negligible. More formally, we say that a PKE Π is *ε-correct* if we have

$$\Pr\Big[m \neq \Pi.\mathcal{D}(sk, c) : \quad (pk, sk) \leftarrow \Pi.\mathcal{K}(1^k); c \leftarrow \Pi.\mathcal{E}(pk, m)\Big] \leq \varepsilon \tag{1}$$

for all messages $m \in \mathcal{M}$. If $\varepsilon = 0$, we call Π perfectly correct.

Definition 2 (Public-Key Multiple Encryption Scheme)**.** *Let $\Pi_1, \ldots, \Pi_n$ be n public-key encryption schemes with corresponding message spaces $\mathcal{M}_i$ and ciphertext spaces $\mathcal{C}_i$. Call Π_i the ith component cipher. A* public-key multiple encryption scheme *Π^n composed of $\Pi_1, \ldots, \Pi_n$ is given by a triple of algorithms $(\mathcal{K}^n, \mathcal{E}^n, \mathcal{D}^n)$ defined over a message space $\mathcal{M}^n$ and a ciphertext space $\mathcal{C}^n$, where*

- *$\mathcal{K}^n = \Pi_1.\mathcal{K} \times \ldots \times \Pi_n.\mathcal{K}$, the key generation algorithm, is a probabilistic algorithm that takes as the input a security parameter $k, k \in \mathbb{N}$ as 1^k and generates n keypairs $(pk_i, sk_i) \leftarrow \Pi_i.\mathcal{K}(1^k)$ for $1 \leq i \leq n$. The output of $\mathcal{K}^n$ is the multiple encryption keypair $(PK, SK) = ((pk_1, \ldots, pk_n), (sk_1, \ldots, sk_n))$.*
- *$\mathcal{E}^n = \Pi_1.\mathcal{E} \times \ldots \times \Pi_n.\mathcal{E}$, the encryption algorithm, is a probabilistic algorithm that takes as the input a multiple encryption public key PK and a message $m \in \mathcal{M}^n$ and outputs a ciphertext $C \in \mathcal{C}^n$ using $\Pi_1.\mathcal{E}(pk_1, \cdot), \ldots, \Pi_n.\mathcal{E}(pk_n, \cdot)$. If necessary, we explicitly state the randomness r_i used for the encryption as $\Pi_i.\mathcal{E}(pk, \cdot; r_i)$. Otherwise, the randomness is internally chosen uniformly at random.*
- *$\mathcal{D}^n = \Pi_1.\mathcal{D} \times \ldots \times \Pi_n.\mathcal{D}$, the decryption algorithm, is a deterministic algorithm that takes as the input a multiple encryption key SK and a ciphertext $C \in \mathcal{C}^n$ and uses $\Pi_1.\mathcal{D}(sk_1, \cdot), \ldots, \Pi_n.\mathcal{D}(sk_n, \cdot)$ to output a message $m \in \mathcal{M}^n$ or a designated rejection symbol $\perp$ to indicate a failed decryption using.*

Correctness of M-PKE. The correctness of an M-PKE scheme depends on the failure probability of the individual component ciphers of the scheme. If the decryption of one of the components fails, then the entire decryption of the M-PKE fails. We let probability $Pr[\Pi_i \text{ fails}]$ that the decryption of the ith component fails be as in Equation 1. We say that an M-PKE scheme Π^n is *ε-correct* if we have

$$\Pr[\Pi^n \text{ fails}] = \Pr\left[\bigvee_i \Pi_i \text{ fails}\right] \leq \sum_i \varepsilon_i \stackrel{\text{def}}{=} \varepsilon \tag{2}$$

for all messages $m \in \mathcal{M}^n$, where the inequality follows from the union bound. We call Π^n perfectly correct if $\varepsilon_1 = \ldots = \varepsilon_n = 0$.

3.3. Design Principles of Multiple Encryption

The multiple encryption definition is a general definition that does not describe precisely how the individual encryption and decryption algorithms are used, since this is specific to each scheme. Nevertheless, multiple encryption schemes for public-key

encryption can be categorized into two main designs. The first design principle used to combine the different encryption schemes is to construct an encryption chain using the output of one scheme as the input for the next scheme. This is called *sequential multiple encryption* or *cascade encryption* and follows the design principle of Shannon (see Section 2). Sequential constructions are discussed in Section 4. The second design principle is to pre-process the plaintext by splitting it into multiple shares and encrypt each share in parallel using a different encryption scheme. We note that the plaintext must be split in such a way that all shares are present in order to reconstruct the plaintext during decryption. This principle is called *parallel multiple encryption* and is further discussed in Section 5. It is also possible to mix the two design principles and create combined *sequential–parallel multiple encryption* as discussed in Section 6. Throughout this paper, we talk about combining different encryption schemes into a new M-PKE scheme. We note that it is also possible to use the same encryption scheme multiple times for an M-PKE, but with different sets of key pairs, similar to the symmetric key block cipher Triple DES. The security of such M-PKE schemes then depends on the different sets of keys, and not on the different underlying hard mathematical problems.

3.4. Indistinguishability Notions for Public-Key Encryption

We now describe the standard indistinguishability notions first for PKE schemes and then for M-PKE schemes. In general, the goal of the ciphertext indistinguishability security notion of an encryption scheme is that an adversary should not learn any information from a given ciphertext besides the length of the message. Usually, three different standard indistinguishability notions are used, IND-CPA, IND-CCA1, and IND-CCA2. These notions are modeled as games in order to evaluate their security. The only difference between the notions is whether or not the attacker has access to a decryption oracle during the game phases. We note that IND-CCA2 is commonly referred to simply as IND-CCA. We use this notation for the remainder of this paper when the context is clear. First, we describe the IND-CCA notion, before highlighting the differences between the various notions.

Definition 3 (IND-CCA Security of PKE)**.** *We define the IND-CCA security for a PKE* Π *as a game between an adversary* $\mathcal{A}$ *and a challenger* $\mathcal{C}$*. The challenger generates a key pair* (pk, sk) *for* Π *and gives the public key* pk *to the adversary, and the secret key* sk *to a decryption oracle* $\mathcal{O}$*. In the first phase,* $\mathcal{A}$ *can query* $\mathcal{O}$ *with arbitrary ciphertexts* c *to obtain* $m = \Pi.\mathcal{D}(sk, c)$*. At the end of the first phase,* $\mathcal{A}$ *selects two messages* m_0, m_1 *of equal length and gives them to* $\mathcal{C}$*. Then,* $\mathcal{C}$ *chooses a random bit* $b \leftarrow \$\{0,1\}$*, encrypts* m_b *to the challenge ciphertext* c^**, and sends* c^* *to* $\mathcal{A}$*. In the second phase,* $\mathcal{A}$ *can query* $\mathcal{O}$ *again with ciphertexts* $c \neq c^*$*. At the end of the second phase,* $\mathcal{A}$ *outputs a guess* b' *to which message was encrypted to* c^**. The adversary* $\mathcal{A}$ *wins the game if* $b = b'$*.*

The advantage Adv *of* $\mathcal{A}$ *in the IND-CCA game with regard to* Π *is defined as* $\mathsf{Adv}^{IND-CCA}_{\mathcal{A},\Pi}(k) = 2 \cdot \Pr[b = b'] - 1$*, where* k *is the security parameter. We say that* Π *is secure in the sense of IND-CCA if the function* $\mathsf{Adv}^{IND-CCA}_{\mathcal{A},\Pi}(k)(\cdot)$ *is negligible for any PPT adversary* $\mathcal{A}$*.*

IND-CPA. In the *indistinguishability under chosen-plaintext attack*, the adversary is not allowed any queries to the decryption oracle.

IND-CCA1. In the *indistinguishability under non-adaptive chosen-ciphertext attack*, the adversary can query the decryption oracle on any ciphertexts in the first phase, that is, before $\mathcal{A}$ sends the two messages to the challenger.

IND-CCA2. In the *indistinguishability under adaptive chosen-ciphertext attack*, the adversary can additionally query the decryption oracle in the second phase after receiving the challenge ciphertext c^*. The only constraint is that $\mathcal{A}$ cannot send c^* to the decryption oracle.

For more formal definitions and information on standard security notions, we refer to the work of Bellare et al. [35] and Katz and Young [36]. Sometimes the IND-CCA notion is seen as too strong, allowing the adversary decryption of all ciphertexts but the challenge.

Therefore, relaxed variants of the CCA security exist, for example, *replayable CCA* (IND-rCCA) [37], *generalized CCA* (IND-gCCA) [38], or *detectable CCA* (IND-dCCA) [19]. These variants additionally reject ciphertexts in the second phase of the game that seem useful to the adversary. Adaptations and proofs exist for some multiple encryption schemes under the rCCA, gCCA, or dCCA security; see, for example, [15,22,39].

3.5. Indistinguishability Notions for Multiple Public-Key Encryption

We now extend the indistinguishability security for the multiple public-key encryption setting. When defining the security of multiple encryption schemes, it is important to take into account additional points. Most importantly, some of the used component encryption schemes can become partially or completely broken, or the corresponding keys can be exposed. Additionally, one must consider the flavor of different decryption oracles that an adversary can query and what these oracles can detect and reject. There exist different approaches for these considerations in the literature.

Harnik et al. [21] and Herzberg [15] model the breaking of schemes by defining the term (k, n)*-robust combiner*, where n is the total number of used encryption schemes and k is the number of secure schemes. They call a combiner robust if it uses k secure schemes for a security specification s and the resulting combiner also fulfills s. Zhang et al. [22] model the breaking of a scheme by using a designated key exposure oracle that can be queried by the adversary, whereas Dodis and Katz [17] and Fujioka et al. [18] hand the secret keys to the adversary in the setup phase of the indistinguishability games. In both models, the number of queries or keys handed over is limited to $n - k$.

The number of different decryption oracles and their abilities are discussed by Dodis and Katz [17]. They define three indistinguishability games for multiple encryption schemes, a weak variant IND-wMCCA, a standard variant IND-MCCA, and a strong variant IND-sMCCA, for parallel multiple encryption. Fujioka et al. [18] extend these notions for sequential multiple encryption. These notions focus on the increasing abilities of the attacker. Other literature mostly uses security notions similar to the IND-wMCCA definition, which is equivalent to IND-CCA for PKE. But sometimes the oracles reject additional ciphertexts, like the rCCA extension of Zhang et al. [22]. A similar extension of rCCA is defined by Herzberg [15], but without the rejection of component ciphertexts; see Section 4.1 for details. Herzberg himself notices the discrepancy between the different security notions and different definitions of combiners, multiple encryption, and hybrid schemes. We note that there is a research gap in the literature, in particular a unified definition or framework for the security of multiple encryption, which is still an open question.

We now review the definition of the security notions from Dodis and Katz as well as Fujioka et al. in a generalized way. Whenever possible, we map the security of other schemes in this paper to these notions for consistency. If this is not possible, we make the differences clear and refer the reader to the original definitions. The indistinguishability games for multiple encryption proceed in the same way as the games for the PKE notions, where the adversary $\mathcal{A}$ has access to one or multiple decryption oracles during the two phases. $\mathcal{A}$ wins each game by correctly guessing the bit b that indicates which of the two messages was encrypted by the challenger $\mathcal{C}$.

IND-wMCPA. In the *indistinguishability under weak multiple chosen-plaintext attack*, the adversary $\mathcal{A}$ is not allowed any queries to the decryption oracles.

IND-wMCCA1. In the *indistinguishability under weak multiple non-adaptive chosen-ciphertext attack*, the adversary $\mathcal{A}$ can send queries to a decryption oracle $\mathcal{O}$ for a multiple encryption ciphertext C in order to receive its decryption. $\mathcal{A}$ cannot see any partial or intermediate decryption results. In the second phase, $\mathcal{A}$ is not given access to any decryption oracles. This notion is equivalent to the IND-CCA1 notion for PKE.

IND-wMCCA2. In the *indistinguishability under weak multiple adaptive chosen-ciphertext attack*, the adversary $\mathcal{A}$ can send queries to a decryption oracle $\mathcal{O}$ for a multiple encryption ciphertext C. $\mathcal{A}$ then obtains the plaintext of C without seeing any partial or intermediate decryption results. In the second phase, $\mathcal{A}$ can query $\mathcal{O}$ again, but is not allowed the

sending of the challenge ciphertext C^*. This notion is equivalent to the IND-CCA2 notion for PKE.

IND-MCCA. In the *indistinguishability under (regular) multiple adaptive chosen-ciphertext attack*, the decryption oracle $\mathcal{O}$ behaves slightly differently. After sending the query ciphertext C, $\mathcal{A}$ obtains the intermediate decryption values $m_1, \ldots, m_n$ back. What exactly these values are depends on the individual multiple encryption scheme and is clarified in the respective descriptions of the schemes. In the second phase, $\mathcal{A}$ can query the oracle $\mathcal{O}$ again, but is not allowed the submission of the challenge ciphertext C^*.

IND-sMCCA. In the *indistinguishability under strong multiple adaptive chosen-ciphertext attack*, $\mathcal{A}$ gains access to additional decryption oracles $\mathcal{OD}_i$. These oracles accept queries to the decryption subroutines of the individual PKE components of the M-PKE scheme. This way, $\mathcal{A}$ obtains even more information about parts of the ciphertext. However, the second phase requires additional restrictions for the oracle queries. $\mathcal{A}$ can again query the oracles in this phase. But now $\mathcal{A}$ is forbidden from submitting C^* to the usual decryption oracle $\mathcal{O}$ and submitting intermediate ciphertexts c_i^* of C^* to $\mathcal{OD}_i$. The exact meaning of what intermediate ciphertexts are depends on the individual multiple encryption scheme.

Definition 4 (Security of M-PKE). *We let $\mathcal{I} \in \{$wMCPA,wMCCA1, wMCCA2, MCCA, sMCCA$\}$. The advantage Adv of an attacker $\mathcal{A}$ in the IND-$\mathcal{I}$ game with regard to the M-PKE Π^n is defined as $\mathsf{Adv}^{IND-\mathcal{I}}_{\mathcal{A},\Pi^n} = 2 \cdot \Pr[b = b'] - 1$. We call a multiple public-key encryption (k, n)-secure in the sense of IND-$\mathcal{I}$ if the following holds:*

1. *At least k of the n component ciphers of Π^n are secure PKEs with regard to some security notion.*
2. *The function $\mathsf{Adv}^{IND-\mathcal{I}}_{\mathcal{A},\Pi^n}$ is negligible for any PPT adversary $\mathcal{A}$.*

4. Sequential Multiple Encryption

One of the two basic designs for multiple encryption is the *sequential multiple encryption*, sometimes called the *cascade cipher*. A sequential multiple encryption is a chain of usually different ciphers, where the plaintext is encrypted by the first component cipher. The resulting ciphertext of each cipher is given as the input for the next one. The output of the last cipher in the chain is the ciphertext of the combined scheme. The individual encryption schemes of the sequential multiple encryption can be chosen independently; the only limitation is that the range of an encryption component must be compatible with the domain of the next component. The advantage of the sequential constructions is the small ciphertext, that is, just the size of the last ciphertext in the encryption chain.

In the context of PKE, the natural sequential construction is studied by Herzberg [15], who shows multiple positive and negative results for different security notions. The main disadvantage of the natural sequential construction is that it is not IND-wMCCA secure. Fujioka et al. [18] overcome this disadvantage by proposing a sequential construction that uses hash functions in order to derive pseudo-random values from a ciphertext for the next component cipher in the sequential construction. Their construction is IND-wMCCA secure in the ROM. Zhang et al. [16] propose an IND-wMCCA secure sequential scheme that encrypts random values and uses the resulting ciphertexts XORed with the plaintext.

In this section, we first present the construction of the natural sequential multiple encryption, before showing the constructions of Fujioka et al. and Zhang et al.

4.1. Natural Sequential Multiple Encryption

The *natural sequential multiple encryption* uses only the individual encryption schemes and no additional computations or cryptographic primitives. The ciphertext of the ith component cipher is given as the input for the $i + 1$th component cipher. The input for the first component is the plaintext m and the output of the last component is the ciphertext C. The structure of the natural sequential combiner is depicted in Figure 1.

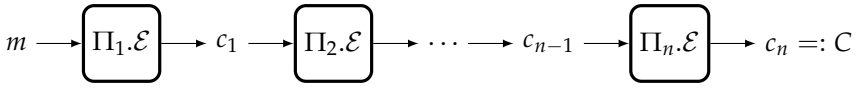

Figure 1. Scheme of the natural sequential multiple encryption procedure.

Definition 5 (Natural Sequential Multiple Encryption)**.** *Let $\Pi_1, \ldots, \Pi_n$ be n PKE schemes with corresponding message spaces $\mathcal{M}_i$ and ciphertext spaces $\mathcal{C}_i$. Let the range of Π_i be in the domain of Π_{i+1}, i.e., $\mathcal{C}_i \subseteq \mathcal{M}_{i+1}$. The* natural sequential multiple encryption scheme *defined over the message space $\mathcal{M}^n = \mathcal{M}_1$ and the ciphertext space $\mathcal{C}^n = \mathcal{C}_n$ is given by a triple of algorithms $(\mathcal{K}^n, \mathcal{E}^n, \mathcal{D}^n)$, where*

- *$\mathcal{K}^n$, the key generation algorithm, is a probabilistic algorithm that takes the security parameters $(k_1, \ldots, k_n)$ as the input. It invokes the key generation algorithm of each PKE component in order to obtain n keypairs $(pk_i, sk_i) \leftarrow \Pi_i.\mathcal{K}(1^{k_i})$. The output of $\mathcal{K}^n$ is $(PK, SK) = ((pk_1, \ldots, pk_n), (sk_1, \ldots, sk_n))$.*
- *$\mathcal{E}^n$, the encryption algorithm, is a probabilistic algorithm that takes as the input a multiple encryption public key PK and a message $m \in \mathcal{M}^n$ and computes the ciphertext $C \in \mathcal{C}^n$ as follows: Set $c_0 = m$ for $i = 1$ to n perform $c_i \leftarrow \Pi_i.\mathcal{E}(pk_i, c_{i-1})$. Output $C = c_n$.*
- *$\mathcal{D}^n$, the decryption algorithm, is a deterministic algorithm that takes as the input a multiple encryption secret key SK and a ciphertext C and decrypts the message as follows. Set $c_n = C$ and compute for $i = n$ to 1 the individual decryption of $c_{i-1} \leftarrow \Pi_i.\mathcal{D}(sk_i, c_i)$. Return $m' = c_0$.*

The security of the natural sequential multiple encryption is studied in depth by Herzberg [15]. Zhang et al. [16] also discuss different security notions of the natural sequential combiner. The natural sequential multiple encryption is not $(n-1, n)$-secure in the sense of IND-wMCCA, even if all encryption components are IND-CCA secure. Hence, a single exposed key is enough to break the IND-wMCCA security of the natural sequential multiple encryption. We suppose the secret key of the last component Π_n becomes exposed; then, a re-encryption attack can be launched. In the first step, the last ciphertext c_n^* is decrypted with the exposed secret key to c_{n-1}^*. In the second step, c_{n-1}^* is then encrypted again, resulting in a different ciphertext c_n' due to the probabilistic nature of modern PKE schemes. A decryption oracle in the IND-wMCCA game does not notice the difference between c_n^* and c_n' and returns the decryption of c_n', breaking the IND-wMCCA security of the natural sequential multiple encryption. We note that all the natural construction are vulnerable to re-encryption attacks. But the sequential multiple encryption achieves $(1, n)$ security for the IND-wMCCA1 notion, if at least one component remains IND-CCA1 secure.

For the non-standard security notions, sequential multiple encryption is IND-rCCA secure [15]. In the case of IND-gCCA, the security depends on the definition of the notion for multiple encryption. Zhang et al. [16] use a ciphertext relation function $r(\cdot, c^*)$ which calls the corresponding relation functions r_i for each component cipher. In this way, the decryption oracle rejects the decryption of a ciphertext if only one intermediate ciphertext relates to one intermediate ciphertext of the challenge ciphertext c^*. Herzberg [15], on the other hand, uses an IND-gCCA security definition, whose ciphertext relation function $r(\cdot, c^*)$ considers the sequential chain as a whole, similar to the IND-wMCCA security notion. Thus, the relation function cannot detect intermediate relations of the ciphertexts. The natural sequential multiple encryption is therefore IND-gCCA secure for the definition in [16] and it is *not* IND-gCCA secure for the definition in [15].

4.2. Hashed Sequential Multiple Encryption

The *hashed sequential multiple encryption* of Fujioka et al. [18] increases the security of the natural sequential multiple encryption by using hash functions between the encryption steps. The hashed construction is very similar to the natural construction in Section 4.1 and mimics its simplicity. In order to prevent re-encryption attacks, it uses the hash values of intermediate ciphertexts as random coins for the next encryption component. More precisely, the $i+1$th component cipher obtains the ith ciphertext c_i as the input and the

hash value of the ith ciphertext $H(c_i)$ as random coins. Only the random coins for the first encryption component are chosen uniformly at random. This idea of de-randomizing a probabilistic encryption is akin to the construction of Fujisaki and Okamoto [40] for increasing the security of PKE schemes. The structure of the hashed sequential combiner is depicted in Figure 2.

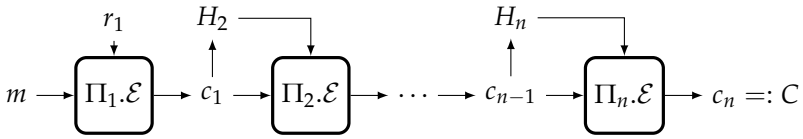

Figure 2. Scheme of the hashed sequential multiple encryption procedure.

Definition 6 (Hashed Sequential Multiple Encryption)**.** *Let $\Pi_1, \ldots, \Pi_n$ be n PKE schemes with corresponding message spaces $\mathcal{M}_i$ and ciphertext spaces $\mathcal{C}_i$. Let the range of Π_i be in the domain of Π_{i+1}, i.e., $\mathcal{C}_i \subseteq \mathcal{M}_{i+1}$. Let $H_2, \ldots, H_n$ be $n-1$ hash functions, where $H_i : \{0,1\}^* \to \{0,1\}^{rlen_i(k_i)}$ and $rlen_i(k_i)$ is the required number of random coin tosses for Π_i. The* hashed sequential multiple encryption scheme *defined over the message space $\mathcal{M}^n = \mathcal{M}_1$ and the ciphertext space $\mathcal{C}^n = \mathcal{C}_n$ is given by a triple of algorithms $(\mathcal{K}^n, \mathcal{E}^n, \mathcal{D}^n)$, where*

- *$\mathcal{K}^n$, the key generation algorithm, is a probabilistic algorithm that takes the security parameters $(k_1, \ldots, k_n)$ as the input. It invokes the key generation algorithm of each PKE component Π_i in order to obtain n keypairs $(pk_i, sk_i) \leftarrow \Pi_i.\mathcal{K}(1^{k_i})$. The output of $\mathcal{K}^n$ is $(PK, SK) = ((pk_1, \ldots, pk_n), (sk_1, \ldots, sk_n))$.*
- *$\mathcal{E}^n$, the encryption algorithm, is a probabilistic algorithm that takes as the input a multiple encryption public key PK and a message $m \in \mathcal{M}^n$ and computes the ciphertext $C \in \mathcal{C}^n$ as follows. Choose $r_1 \xleftarrow{\$} \{0,1\}^{rlen_1(k_1)}$ and compute $c_1 \leftarrow \Pi_1.\mathcal{E}(pk_1, m; r_1)$. For $i = 2$ to n, perform $c_i \leftarrow \Pi_i.\mathcal{E}(pk_i, c_{i-1}; H_i(c_{i-1}))$. Output $C = c_n$.*
- *$\mathcal{D}^n$, the decryption algorithm, is a deterministic algorithm that takes as the input a multiple encryption key pair (PK, SK) and a ciphertext C' and decrypts the message as follows: Set $c'_n = C'$ and compute for $i = n$ to 2 the individual decryption of $c'_{i-1} \leftarrow \Pi_i.\mathcal{D}(sk_i, c'_i)$. After each decryption step, check if $c'_i = \Pi_i.\mathcal{E}(pk_i, c'_{i-1}; H_i(c'_{i-1}))$ and return $\perp$ if not. Finally, decrypt the last ciphertext $c'_0 \leftarrow \Pi_1.\mathcal{D}(sk_1, c'_1)$ and return the message $m' = c'_0$.*

The work of Fujioka et al. [18] contains multiple proofs for the hashed sequential construction in the ROM. They extend the security notion from [16] to adapt it to the stronger security notions from [17]. The hashed sequential multiple encryption is shown to be $(1, n)$-IND-wMCCA secure constructed from n weaker IND-CPA secure encryption components. If the underlying encryption components are IND-CCA secure, the hashed sequential multiple encryption is even $(1, n)$-IND-sMCCA secure in the ROM. In this case, the IND-sMCCA security can be reduced to the IND-CCA security of the one remaining secure encryption component.

4.3. XOR Sequential Multiple Encryption

The *XOR sequential multiple encryption* of Zhang et al. [16] bases its encryption of the plaintext on the XOR operation of random values and the intermediate ciphertexts. In each step, a random value is encrypted with a component cipher. The hash of this random value is XORed with the ciphertext from the previous step. At the end of each step, the ciphertext consists of the ciphertext of the random value and the XOR value. Finally, the output of the last step is the ciphertext of the XOR sequential multiple encryption. The random coins for the PKE components are derived from the n random values and the message in order to assure the knowledge about these values during decryption. This measure prevents re-encryption attacks on this scheme. The decryption is then performed with a re-encryption check after the decryption of the message and all the random values. The structure of the XOR sequential multiple encryption is depicted in Figure 3.

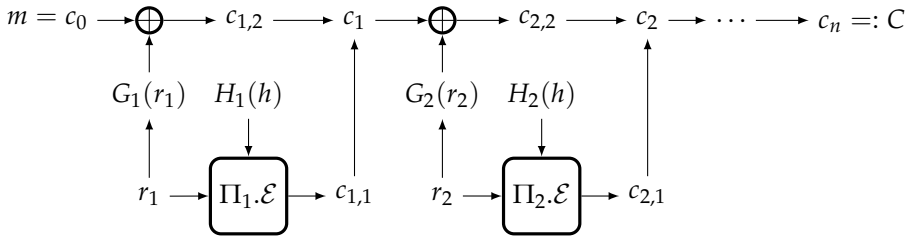

Figure 3. Scheme of the XOR sequential multiple encryption procedure, where $h = m \,||\, r_1 \,||\, \ldots \,||\, r_n$ and the final ciphertext is $C = c_n$.

Definition 7 (XOR Sequential Multiple Encryption). *Let $\Pi_1, \ldots, \Pi_n$ be n PKE schemes with corresponding message spaces $\mathcal{M}_i$ and ciphertext spaces $\mathcal{C}_i$. Let $H_1, \ldots, H_n$ and $G_1, \ldots, G_n$ be hash functions, where $G_i : \{0,1\}^* \to \{0,1\}^{|G_{i-1}|} \times \mathcal{C}_i$, $H_i : \{0,1\}^* \to \{0,1\}^{rlen_i(k_i)}$, and $rlen_i(k_i)$ is the required number of random coin tosses for Π_i. The XOR sequential multiple encryption scheme defined over the message space $\mathcal{M}^n$ of arbitrary length and the ciphertext space $\mathcal{C}^n = \{0,1\}^{|G_n|} \times \mathcal{C}_n$ is given by a triple of algorithms $(\mathcal{K}^n, \mathcal{E}^n, \mathcal{D}^n)$, where*

- *$\mathcal{K}^n$, the key generation algorithm, is a probabilistic algorithm that takes the security parameters $(k_1, \ldots, k_n)$ as the input. It invokes the key generation algorithm of each PKE component Π_i in order to obtain n keypairs $(pk_i, sk_i) \leftarrow \Pi_i.\mathcal{K}(1^{k_i})$. The output of $\mathcal{K}^n$ is $(PK, SK) = ((pk_1, \ldots, pk_n), (sk_1, \ldots, sk_n))$.*
- *$\mathcal{E}^n$, the encryption algorithm, is a probabilistic algorithm that takes as the input a multiple encryption public key PK and a message $m \in \mathcal{M}^n$ and computes the ciphertext $C \in \mathcal{C}^n$ as follows: Choose random values $r_i \xleftarrow{\$} \mathcal{M}_i, 1 \leq i \leq n$ and compute $h = m \,||\, r_1 \,||\, \ldots \,||\, r_n$. Set $c_0 = m$. For $i = 1$ to n, perform $c_{i,1} \leftarrow \Pi_i.\mathcal{E}(pk_i, r_i; H_i(h))$ and $c_{i,2} \leftarrow c_{i-1} \oplus G_i(r_i)$. Set $c_i = (c_{i,1}, c_{i,2})$. Output $C = c_n$.*
- *$\mathcal{D}^n$, the decryption algorithm, is a deterministic algorithm that takes as the input a multiple encryption key pair (PK, SK) and a ciphertext $C' = (c'_{n,1}, c'_{n,2})$ and decrypts the message as follows: Decrypt each intermediate ciphertext for $i = n$ to 1 by computing first the decryption of $r'_i \leftarrow \Pi_i.\mathcal{D}(sk_i, c'_{i,1})$ and second $c'_{i-1} \leftarrow c'_{i,2} \oplus G_i(r'_i)$. Set $m' = c'_0$. For the re-encryption check, concatenate $h' = m' \,||\, r'_1 \,||\, \ldots \,||\, r'_n$. For $i = 1$ to n, check if $c'_i = (\Pi_i.\mathcal{E}(pk_i, r'_i; H_i(h')), c'_{i-1} \oplus G_i(r'_i))$ and return $\perp$ if not. If all checks pass, return the message m'.*

Initially, the idea of this scheme was only sketched by Zhang et al. in their conference paper [22], where they presented a parallel multiple encryption scheme (see Section 5.2). They expanded this idea in the extended version [16] of the conference paper, but the description is incomplete. Fujioka et al. [18] present an updated sketch of the XOR sequential construction after correspondence with Zhang et al. Here, we presented the complete formal description of the XOR sequential multiple encryption. The XOR sequential multiple encryption has a major disadvantage regarding its ciphertext size. In each step, the overall ciphertext size increases by the ciphertext size of the current component cipher. Therefore, the resulting ciphertext is just as large as for a parallel multiple encryption.

Zhang et al. claim that their construction is $(1, n)$-IND-wMCCA secure. Unfortunately, they prove the propositions only for their improved natural parallel construction; even the extended paper does not include a proof for the XOR sequential multiple encryption. So far, there are no published security proofs for the XOR sequential multiple encryption.

5. Parallel Multiple Encryption

The second basic design principle for multiple encryption schemes is the *parallel multiple encryption*. Here, n PKE-schemes Π_i are invoked in parallel to encrypt a plaintext in a secure way. The plaintext m must be pre-processed and split into multiple shares $m_{i,1 \leq i \leq n}$, such that all shares must be present in order to reconstruct the original plaintext. Without the pre-processing, the benefit of the multiple encryption is already lost with one broken scheme. After splitting m, each share m_i is encrypted by a different encryption scheme. The

ciphertext of a parallel multiple encryption usually consists of all the individual ciphertexts $\Pi_i.\mathcal{E}(m_i), 1 \leq i \leq n$. When decrypting a parallel multiple encryption ciphertext, the receiver first decrypts the individual shares m_i and then recovers the plaintext m.

This natural approach is formally described by Zhang et al. [22], where the splitting is performed by an *all-or-nothing tranform* (AONT) [41,42]. "An AONT is an unkeyed, invertible, randomized transformation, with the property that it is hard to invert unless all of the output is known" [42], and fulfills the conditions for parallel multiple encryption message splitting. Usually, an AONT is combined with an encryption scheme to hinder brute-force key search attacks [41]. In this context, more efficient AONTs are developed, for example, in [43,44]. Since the natural parallel multiple encryption is not IND-wMCCA secure, Zhang et al. present an improved scheme that is IND-wMCCA secure in the ROM. For this improved scheme, they use additional hash values to allow a re-encryption check during decryption. It is also possible to replace the AONT transform with a secret sharing scheme [45,46]. This is described by Dodis and Katz [17] as "folklore" parallel multiple encryption. A generic construction that is IND-sMCCA secure in the standard model is presented by Dodis and Katz [17], where the AONT is replaced by an $(n-1, n)$ secret sharing scheme, and one-time signature schemes are used to bind the individual ciphertexts together.

Besides the generic schemes with pre-processing, there is a group of parallel multiple encryption schemes that use the XOR operation to mask or directly encrypt the message without any prior splitting. In fact, the earliest published M-PKE scheme is the XOR-combiner by Asmuth and Blakely [14], which directly XORs the plaintext with two random values. Only these random values are individually encrypted by the scheme's component ciphers. This scheme is vulnerable to re-encryption attacks and is not IND-wMCCA secure. A slightly different version of the XOR-combiner for two encryption components recorded by Herzberg [15]. Here, the message is XORed with a random value and then encrypted by the first component cipher. The second component cipher encrypts only the random value. Herzberg proves that this construction is IND-wMCCA1 secure but not IND-wMCCA2 secure.

The parallel design is a space-time trade-off when compared to the sequential design, where the size of the ciphertext is increased but the run time can be decreased drastically by parallel execution of the PKE components. This section describes the parallel multiple encryption schemes in the order listed above.

5.1. Natural Parallel Multiple Encryption

The *natural parallel multiple encryption* uses an AONT $\mathcal{T}$ to pre-process the message m into n shares $m_{i,1\leq i\leq n}$, such that all shares must be present to reconstruct m. If even one share is missing, it is infeasible to find out any information about m. Each share m_i is then encrypted by a PKE scheme Π_i in parallel. The ciphertext of this scheme consists of all individual ciphertexts $\Pi_i.\mathcal{E}(m_i)$ concatenated together. For the decryption, the recepient must first decrypt each share m_i and then apply the inverse AONT $\mathcal{T}^{-1}$ to all shares to obtain the message m. The structure of the natural parallel multiple encryption is shown in Figure 4.

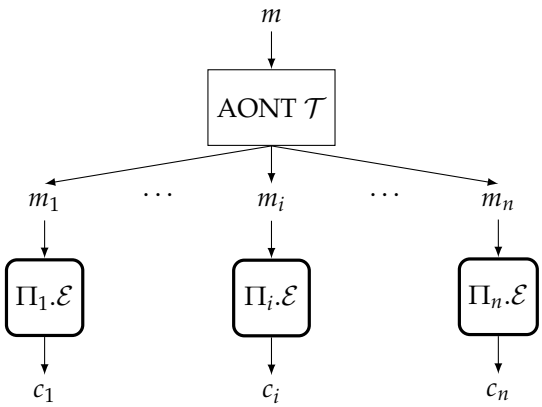

Figure 4. Scheme of the natural parallel multiple encryption procedure.

Definition 8 (Natural Parallel Multiple Encryption)**.** *Let $\Pi_1, \ldots, \Pi_n$ be n public-key encryption schemes with corresponding message spaces $\mathcal{M}_i$ and ciphertext spaces $\mathcal{C}_i$. Let $\mathcal{T} : \{0,1\}^* \to \mathcal{M}_1 \times \ldots \times \mathcal{M}_n$ be an AONT. The natural parallel multiple encryption scheme defined over the message space $\mathcal{M}^n$ of arbitrary length and the ciphertext space $\mathcal{C}^n = \mathcal{C}_1 \times \ldots \times \mathcal{C}_n$ is given by a triple of algorithms $(\mathcal{K}^n, \mathcal{E}^n, \mathcal{D}^n)$, where*

- *$\mathcal{K}^n$, the key generation algorithm, is a probabilistic algorithm that takes the security parameters $(k_1, \ldots, k_n)$ as the input. It invokes the key generation algorithm of each PKE component Π_i in order to obtain n keypairs $(pk_i, sk_i) \leftarrow \Pi_i.\mathcal{K}(1^{k_i})$. The output of $\mathcal{K}^n$ is $(PK, SK) = ((pk_1, \ldots, pk_n), (sk_1, \ldots, sk_n))$.*
- *$\mathcal{E}^n$, the encryption algorithm, is a probabilistic algorithm that takes as the input a multiple encryption public key PK and a message $m \in \mathcal{M}^n$ and computes the ciphertext $C \in \mathcal{C}^n$ as follows: Compute the shares of the message as $(m_1, \ldots, m_n) \leftarrow \mathcal{T}(m)$. For $i = 1$ to n, perform $c_i \leftarrow \Pi_i.\mathcal{E}(pk_i, m_i)$. Output $C = (c_1, \ldots, c_n)$.*
- *$\mathcal{D}^n$, the decryption algorithm, is a deterministic algorithm that takes as the input a multiple encryption key pair (PK, SK) and a ciphertext $C' = (c'_1, \ldots, c'_n)$ and decrypts the message as follows: Decrypt the individual shares by computing for $i = 1$ to n the decryption of $m'_i \leftarrow \Pi_i.\mathcal{D}(sk_i, c'_i)$. Recover the message by computing $m' \leftarrow \mathcal{T}^{-1}(m'_1, \ldots, m'_n)$ and return m'.*

Zhang et al. [22] prove that the natural parallel multiple encryption is vulnerable to re-encryption attacks and is therefore not $(n-1, n)$ IND-wMCCA secure. We note that Zhang et al. use the term "IND-ME-wMCCA" to describe a different security notion than IND-wMCCA. Their notion "IND-ME-wCCA" is a relaxed version of IND-wCCA, which applies the IND-gCCA security notion to the multiple encryption setting. They show, additionally, that the natural parallel multiple encryption is $(1, n)$-secure for the extended IND-gCCA notion, where the decryption oracle has the additional ability to check for the intermediate decrypted values of m_i.

5.2. Improved Natural Parallel Multiple Encryption

The *improved natural parallel multiple encryption* follows the natural parallel construction from the previous section, but it prevents the malleability of ciphertexts to increase the security. This is due to the fact that the randomness is explicitly part of the encryption process. Each message share m_i is XORed with the hash of a random value r_i, which itself is encrypted with a PKE component. The randomness for the PKE components is derived from the original message and the random values $r_{i,1 \leq i \leq n}$. During the decryption, it is then possible to check for the consistency of the randomness and prevent re-encryption attacks. The structure of the encryption process of the improved natural parallel multiple encryption is illustrated in Figure 5.

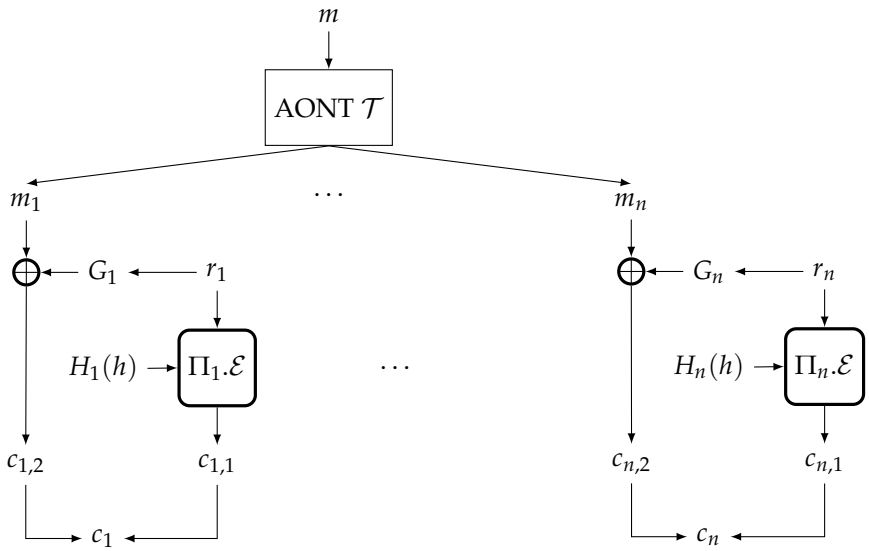

Figure 5. Scheme of the improved natural parallel multiple encryption procedure. The final ciphertext is $C = (c_1, \ldots, c_n)$.

Definition 9 (Improved Natural Parallel Multiple Encryption). *Let $\Pi_1, \ldots, \Pi_n$ be n public-key encryption schemes with corresponding message spaces $\mathcal{M}_i$ and ciphertext spaces $\mathcal{C}_i$. Let $\mathcal{T} : \{0,1\}^* \rightarrow \mathcal{M}_{AONT}^n$ be an AONT, where the range $\mathcal{M}_{AONT}$ can be chosen arbitrarily. Let $H_1, \ldots, H_n$ and $G_1, \ldots, G_n$ be hash functions, where $G_i : \{0,1\}^* \rightarrow \{0,1\}^{|\mathcal{M}_{AONT}|}$, $H_i : \{0,1\}^* \rightarrow \{0,1\}^{rlen_i(k_i)}$, and $rlen_i(k_i)$ is the required number of random coin tosses for Π_i. The improved natural parallel multiple encryption scheme defined over the message space $\mathcal{M}^n$ of arbitrary length and the ciphertext space $\mathcal{C}^n = (\{0,1\}^{|\mathcal{M}_{AONT}|}, \mathcal{C}_1) \times \ldots \times (\{0,1\}^{|\mathcal{M}_{AONT}|}, \mathcal{C}_n)$ is given by a triple of algorithms $(\mathcal{K}^n, \mathcal{E}^n, \mathcal{D}^n)$, where*

- *$\mathcal{K}^n$, the key generation algorithm, is a probabilistic algorithm that takes the security parameters $(k_1, \ldots, k_n)$ as the input. It invokes the key generation algorithm of each PKE component Π_i in order to obtain n keypairs $(pk_i, sk_i) \leftarrow \Pi_i.\mathcal{K}(1^{k_i})$. The output of $\mathcal{K}^n$ is $(PK, SK) = ((pk_1, \ldots, pk_n), (sk_1, \ldots, sk_n))$.*
- *$\mathcal{E}^n$, the encryption algorithm, is a probabilistic algorithm that takes as the input a multiple encryption public key PK and a message $m \in \mathcal{M}^n$ and computes the ciphertext $C \in \mathcal{C}^n$ as follows: Compute the shares of the message as $(m_1, \ldots, m_n) \leftarrow \mathcal{T}(m)$. Choose random values $r_i \xleftarrow{\$} \mathcal{M}_i, 1 \leq i \leq n$ and compute $h = m \mid\mid r_1 \mid\mid \ldots \mid\mid r_n$. For $i = 1$ to n, encrypt r_i as $c_{i,1} \leftarrow \Pi_i.\mathcal{E}(pk_i, r_i; H_i(h))$ and XOR the message share m_i to $c_{i,2} \leftarrow m_i \oplus G_i(r_i)$. Set $c_i \leftarrow (c_{i,1}, c_{i,2})$. Finally, output the ciphertext $C = (c_1, \ldots, c_n)$.*
- *$\mathcal{D}^n$, the decryption algorithm, is a deterministic algorithm that takes as the input a multiple encryption key pair (PK, SK) and a ciphertext $C' = (c'_1, \ldots, c'_n)$ and decrypts the message as follows: For $i = 1$ to n, perform the following steps: Parse c'_i to $(c'_{i,1}, c'_{i,2})$ and decrypt the random value $r'_i \leftarrow \Pi_i.\mathcal{D}(sk_i, c'_{i,1})$. Then, compute the message share $m'_i \leftarrow c'_{i,2} \oplus G_i(r'_i)$. After the loop, recover the message by computing $m' \leftarrow \mathcal{T}^{-1}(m'_1, \ldots, m'_n)$. Then, check the validity of the ciphertext by first computing $h' = m' \mid\mid r'_1 \mid\mid \ldots \mid\mid r'_n$. If $c'_{i,1} = \Pi_i.\mathcal{E}(pk_i, r'_i; H_i(h'))$ and $c'_{i,2} = m'_i \oplus G_i(r'_i)$ for $i = 1$ to n, return the plaintext m'. Otherwise, return $\perp$ as the ciphertext is not valid.*

Zhang et al. [22] show that their improved natural parallel multiple encryption is $(1, n)$-secure for the IND-wMCCA notion in the ROM. Notably, they also prove that the improved scheme can be constructed from weaker One-Way-CPA (OW-CPA) component ciphers.

5.3. Secret Sharing Signature Parallel Multiple Encryption

The *secret sharing signature multiple encryption* from Dodis and Katz [17] is an extension of the natural parallel scheme that utilizes multiple additional cryptographic primitives

to increase the security in the standard model. It uses a secret sharing scheme to split the message and then uses a signature scheme to bind the individual ciphertexts together. The ciphertext of the scheme consists of the component ciphertext, the signature of these ciphertexts, and the corresponding verification key. In addition, the component ciphers must be able to use labels during their operations. This binds the signature verification key to the ciphertext. During decryption, the attached signature is verified to ensure the validity of the ciphertext.

We shortly recall the functionality of secret sharing and signature schemes to the extent necessary to understand this particular multiple encryption scheme. A *secret sharing scheme* $\mathcal{SS}$ is given by a tuple of algorithms $(\mathcal{S}, \mathcal{R})$. The probabilistic sharing algorithm $\mathcal{S}$ takes as the input message m and splits it into n shares $(m_1, \ldots, m_n) \leftarrow \mathcal{SS}.\mathcal{S}(m)$. The deterministic recovery algorithm $\mathcal{R}$ takes as the input n shares $m'_1, \ldots, m'_n$ and outputs some message $m' \leftarrow \mathcal{SS}.\mathcal{R}(m'_1, \ldots, m'_n)$. The correctness property states that $\mathcal{SS}.\mathcal{R}(\mathcal{SS}.\mathcal{S}(m)) = m$, for all m. A (t, n)-secure secret sharing scheme splits the message into n shares, where t is the maximum number of shares that do not reveal any information about m. Secret sharing schemes were introduced by Shamir [46] and Blakley [45]. More efficient secret sharing schemes can be found, for example, in [47,48]. Ding et al. [49] presented a scheme that reduces the decoding bandwidth and operates with smaller share sizes.

A *signature scheme* Σ consists of a triple of algorithms $(\mathcal{K}, \mathcal{S}, \mathcal{V})$. The probabilistic key generation algorithm $\Sigma.\mathcal{K}$ takes as the input a security parameter 1^k and outputs a related pair (sk, vk), consisting of a signing-key sk and a verification-key vk. The probabilistic signing algorithms $\Sigma.\mathcal{S}$ take as the input a signing-key sk and a message m and output a signature σ. The deterministic verification algorithm $\Sigma.\mathcal{V}$ takes as the input a verification-key vk, a message m, and a signature σ. It outputs 1 if and only if σ is a valid signature for m. The signature scheme Σ must satisfy $\Sigma.\mathcal{V}(vk, m, \Sigma.\mathcal{S}(sk, m)) = 1$, for all m.

The structure of the encryption procedure is shown in Figure 6. The verification key of the signature scheme is used as label during encryption. A label contains data that may be implicit from the context, but should be bound to the ciphertext in a non-malleable way [50].

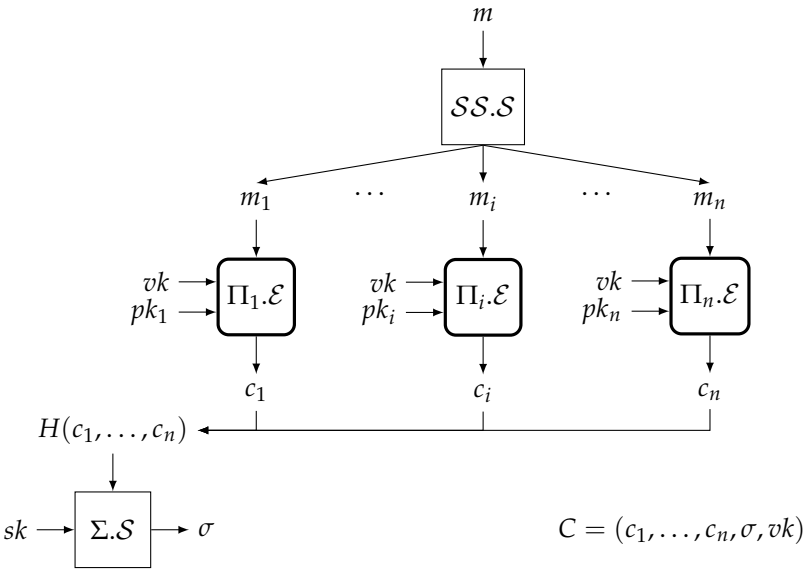

Figure 6. Scheme of the secret sharing signature parallel multiple encryption procedure. It uses a secret sharing scheme $\mathcal{SS}$, a signature scheme Σ with corresponding verification-key vk, and a hash function H, in addition to the usual component ciphers. The key vk is also used as a label during the individual encryptions. The final ciphertext is $C = (c_1, \ldots, c_n, \sigma, vk)$.

Definition 10 (Secret Sharing Signature Parallel Multiple Encryption). *Let $\Pi_1, \ldots, \Pi_n$ be n PKE schemes with corresponding message spaces $\mathcal{M}_i$ and ciphertext spaces $\mathcal{C}_i$. Let $\mathcal{SS}$ be a secret sharing function, Σ a one-time signature scheme, and H a hash function. The secret sharing signature parallel multiple encryption scheme defined over the message space $\mathcal{M}^n$ of arbitrary length and the ciphertext space $\mathcal{C}^n$ is given by a triple of algorithms $(\mathcal{K}^n, \mathcal{E}^n, \mathcal{D}^n)$, where*

- *$\mathcal{K}^n$, the key generation algorithm, is a probabilistic algorithm that takes the security parameters $(k_1, \ldots, k_n)$ as the input. It invokes the key generation algorithm of each PKE component Π_i in order to obtain n keypairs $(pk_i, sk_i) \leftarrow \Pi_i.\mathcal{K}(1^{k_i})$. The output of $\mathcal{K}^n$ is $(PK, SK) = ((pk_1, \ldots, pk_n), (sk_1, \ldots, sk_n))$.*
- *$\mathcal{E}^n$, the encryption algorithm, is a probabilistic algorithm that takes as the input a multiple encryption public key PK and a message $m \in \mathcal{M}^n$ and computes the ciphertext $C \in \mathcal{C}^n$ as follows: Compute the shares of the message as $(m_1, \ldots, m_n) \leftarrow \mathcal{SS}.\mathcal{S}(m)$. Generate a signature-key pair $(sk, vk) \leftarrow \Sigma.\mathcal{K}(1^{k_1}, \ldots, 1^{k_n})$. For $i = 1$ to n, perform $c_i \leftarrow \Pi_i.\mathcal{E}^{vk}(pk_i, m_i)$. Then, compute the hash of the individual ciphertexts $h = H(c_1, \ldots, c_n)$ and sign that value as $\sigma = \Sigma.\mathcal{S}(sk, h)$. Output the ciphertext $C = (c_1, \ldots, c_n, \sigma, vk)$.*
- *$\mathcal{D}^n$, the decryption algorithm, is a deterministic algorithm that takes as the input a multiple encryption key pair (PK, SK) and a ciphertext $C' = (c'_1, \ldots, c'_n, \sigma', vk')$ and decrypts the message as follows: Compute $h' = H(c'_1, \ldots, c'_n)$ and verify the attached signature $\Sigma.\mathcal{V}(vk', h', \sigma')$. If the signature is not valid, return $\perp$. Otherwise, continue by decrypting the individual shares $m'_i \leftarrow \Pi_i.\mathcal{D}^{vk'}(sk_i, c'_i)$, for $i = 1$ to n. Recover the message by computing $m' \leftarrow \mathcal{SS}.\mathcal{R}(m'_1, \ldots, m'_n)$ and return m'.*

Dodis and Katz [17] introduced multiple security notions for multiple encryption. The strongest one is the IND-sMCCA security, where the adversary also has access to the decryption values of the individual shares m'_i. They proved that their secret sharing signature multiple encryption is $(n - t, n)$-secure for the IND-sMCCA notion, if $\mathcal{SS}$ is a (t, n)-secure secret sharing scheme, Σ is a secure one-time signature, and Π_i is IND-CCA secure, for $i = 1, \ldots, n$. Notably, the secret sharing signature scheme is the only multiple encryption scheme that is IND-sMCCA secure in the standard model.

5.4. Natural XOR Parallel Multiple Encryption

The earliest known combiner for public-key encryption is the *natural XOR parallel multiple encryption* by Asmuth and Blakely [14], which uses only the component PKEs and the XOR operation. In the original paper, the message m is XORed with two random values r_1, r_2. The random values are then encrypted with the component ciphers and the XOR value is transmitted directly to the receiver. We define a generalized version of the natural XOR combiner for n component ciphers. The structure of the natural XOR parallel multiple encryption is depicted in Figure 7.

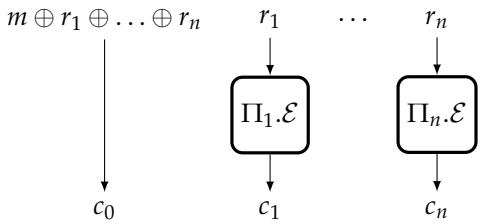

Figure 7. Scheme of the natural XOR parallel multiple encryption procedure. The values $r_1, \ldots, r_n$ are chosen at random and the final ciphertext is $C = (c_0, c_1, \ldots, c_n)$.

Definition 11 (Natural XOR Parallel Multiple Encryption). *Let $\Pi_1, \ldots, \Pi_n$ be n public-key encryption schemes with corresponding message spaces $\mathcal{M}_i$ and ciphertext spaces $\mathcal{C}_i$. The natural XOR parallel multiple encryption scheme defined over the message space $\mathcal{M}^n \subseteq \{0,1\}^{\min\{\log_2 |\mathcal{M}_i| : 1 \le i \le n\}}$ and the ciphertext space $\mathcal{C}^n = \mathcal{M}^n \times \mathcal{C}_1 \times \ldots \times \mathcal{C}_n$ is given by a triple of algorithms $(\mathcal{K}^n, \mathcal{E}^n, \mathcal{D}^n)$, where*

- $\mathcal{K}^n$, *the key generation algorithm, is a probabilistic algorithm that takes the security parameters* $(k_1, \ldots, k_n)$ *as the input. It invokes the key generation algorithm of each PKE component* Π_i *in order to obtain n keypairs* $(pk_i, sk_i) \leftarrow \Pi_i.\mathcal{K}(1^{k_i})$. *The output of* $\mathcal{K}^n$ *is* $(PK, SK) = ((pk_1, \ldots, pk_n), (sk_1, \ldots, sk_n))$.
- $\mathcal{E}^n$, *the encryption algorithm, is a probabilistic algorithm that takes as the input a multiple encryption public key PK and a message* $m \in \mathcal{M}^n$ *and computes the ciphertext* $C \in \mathcal{C}^n$ *as follows: Choose n random values* $r_i \xleftarrow{\$} \mathcal{M}_i, 1 \leq i \leq n$. *Encrypt the XOR value of the message m with all* r_i *as* $c_0 \leftarrow m \oplus r_1 \oplus \ldots \oplus r_n$. *For* $i = 1$ *to n, perform* $c_i \leftarrow \Pi_i.\mathcal{E}(pk_i, r_i)$. *Output* $C = (c_0, c_1, \ldots, c_n)$.
- $\mathcal{D}^n$, *the decryption algorithm, is a deterministic algorithm that takes as the input a multiple encryption key pair* (PK, SK) *and a ciphertext* $C' = (c'_0, c'_1, \ldots, c'_n)$ *and decrypts the message as follows: Decrypt the random values by computing for* $i = 1$ *to n the decryption of* $r'_i \leftarrow \Pi_i.\mathcal{D}(sk_i, c'_i)$. *Recover the message by computing* $m' \leftarrow c'_0 \oplus r'_1 \oplus \ldots \oplus r'_n$ *and return* m'.

Since the natural XOR parallel multiple encryption is vulnerable to re-encryption attacks if one secret key is leaked like the other natural constructions, it is not IND-wMCCA secure. Other than that, there are no known security proofs for indistinguishability notions.

5.5. XOR-Input Parallel Multiple Encryption

The *XOR-input parallel multiple encryption* is a parallel multiple encryption that utilizes the XOR operation to mask the plaintext before encryption. Its construction is similar to the natural XOR parallel multiple encryption form Section 5.4. The difference is that the XORed plaintext is not transmitted directly, but first encrypted using a component cipher. The random values used to mask the plaintext are encrypted using the other $n - 1$ component ciphers. The structure of the XOR-input parallel multiple encryption is depicted in Figure 8. We note that the literature often treats the XOR-input parallel multiple encryption and the XOR parallel multiple encryption as the same construction, even though this is not the case.

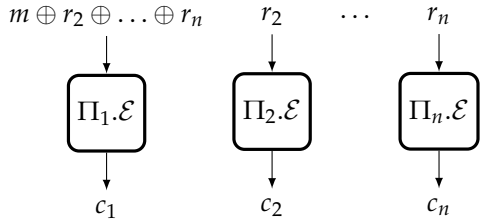

Figure 8. Scheme of the XOR-input parallel multiple encryption procedure. The values $r_2, \ldots, r_n$ are chosen uniformly at random and the final ciphertext is $C = (c_1, \ldots, c_n)$.

Definition 12 (XOR-Input Parallel Multiple Encryption)**.** *Let* $\Pi_1, \ldots, \Pi_n$ *be n public-key encryption schemes with corresponding message spaces* $\mathcal{M}_i$ *and ciphertext spaces* $\mathcal{C}_i$. *The XOR-input parallel multiple encryption scheme defined over the message space* $\mathcal{M}^n \subseteq \mathcal{M}_1$ *and the ciphertext space* $\mathcal{C}^n = \mathcal{C}_1 \times \ldots \times \mathcal{C}_n$ *is given by a triple of algorithms* $(\mathcal{K}^n, \mathcal{E}^n, \mathcal{D}^n)$, *where*

- $\mathcal{K}^n$, *the key generation algorithm, is a probabilistic algorithm that takes the security parameters* $(k_1, \ldots, k_n)$ *as the input. It invokes the key generation algorithm of each PKE component* Π_i *in order to obtain n keypairs* $(pk_i, sk_i) \leftarrow \Pi_i.\mathcal{K}(1^{k_i})$. *The output of* $\mathcal{K}^n$ *is* $(PK, SK) = ((pk_1, \ldots, pk_n), (sk_1, \ldots, sk_n))$.
- $\mathcal{E}^n$, *the encryption algorithm, is a probabilistic algorithm that takes as the input a multiple encryption public key PK and a message* $m \in \mathcal{M}^n$ *and computes the ciphertext* $C \in \mathcal{C}^n$ *as follows: Choose* $n - 1$ *random values* $r_i \xleftarrow{\$} \mathcal{M}^n, 2 \leq i \leq n$. *Encrypt the XOR value of the message m with all* r_i *as* $c_1 \leftarrow \Pi_1.\mathcal{E}(pk_1, m \oplus r_2 \oplus \ldots \oplus r_n)$. *For* $i = 2$ *to n, perform* $c_i \leftarrow \Pi_i.\mathcal{E}(pk_i, r_i)$. *Output* $C = (c_1, \ldots, c_n)$.
- $\mathcal{D}^n$, *the decryption algorithm, is a deterministic algorithm that takes as the input a multiple encryption key pair* (PK, SK) *and a ciphertext* $C' = (c'_1, \ldots, c'_n)$ *and decrypts the message as follows: Decrypt the random values by computing for* $i = 2$ *to n the decryption*

of $r'_i \leftarrow \Pi_i.\mathcal{D}(sk_i, c'_i)$. *Recover the message by computing* $h' \leftarrow \Pi_1.\mathcal{D}(sk_1, c'_1)$ *and return* $m' = h' \oplus r'_2 \oplus \ldots \oplus r'_n$.

Herzberg [15] studied the security of the XOR-input combiner in depth. He showed that the combiner is $(1,2)$-IND-wMCPA and $(1,2)$-IND-wCCA1 secure but not (n,n)-IND-wMCCA2 secure. This is the case even if all of the component ciphers are IND-CCA secure and there is no leakage of secret keys. Herzberg also provec that the XOR-input combiner is *not* secure under his non-standard notions of IND-rCCA and IND-gCCA security in the multiple encryption setting. In contrast, Zhang et al. [16] used a slightly different definition for IND-gCCA, in which the decryption oracle was given the ability to also detect intermediate malicious queries (see security discussion in Section 4.1). With this extended oracle ability, the attacks of Herzberg cannot be applied to the IND-gCCA game of Zhang et al. Hence, the proofs of Zhang et al. for the natural parallel construction with the AONT can be used for the XOR-input combiner. Therefore, the XOR-combiner is secure under the IND-gCCA notion of Zhang et al. and implicitly under a corresponding IND-rCCA notion. Another non-standard security notion was proven by Zhang et al. [39]. They showed that the XOR-combiner is also IND-dCCA secure, in which an additional *detection function* can detect malicious queries to the oracle. The IND-dCCA secure scheme can then be used to construct an IND-wMCCA secure scheme. However, this approach has the disadvantage that additional PKEs with different security notions must be used and, additionally, all components must remain secure.

6. Sequential–Parallel Multiple Encryption

Sequential–parallel multiple encryption combines the two basic design principles of sequential (Section 4) and parallel (Section 5) multiple encryption into new combined schemes. This allows for more flexibility when designing new multiple encryption schemes, since one can balance between run time and ciphertext size. One can imagine the structure of a sequential– scheme as a tree, where each node represents an encryption component. A sequential part is added to the tree if a node has only one child. If a node has multiple children, a parallel part is added.

In this section, we describe two sequential–parallel multiple encryption schemes. The first one is by Hohenberger et al. [19], which is based on PKE components with different security notions. The second one is by Goncalves and Mashatan [20], which is secure in the QROM model and is based on different cryptographic primitives.

6.1. Detectable Sequential–Parallel Multiple Encryption

The *detectable sequential–parallel multiple encryption* of Hohenberger et al. [19] uses three component ciphers with different security notions. The authors introduce a new security notion of *detectable CCA* (IND-dCCA). In this relaxed version of IND-CCA, the decryption oracle is able to detect "dangerous" queries and reject them. First, an IND-dCCA PKE is used to encrypt the message m and two random values, which are used as random coins for the subsequent encryptions. The resulting ciphertext is encrypted two times, once with an *1-bounded-IND-CCA* (1b-INC-CCA) secure scheme and once with an IND-CPA secure scheme. During decryption, a re-encryption check is performed to assure the validity of the ciphertext. The structure of the detectable sequential–parallel multiple encryption is depicted in Figure 9.

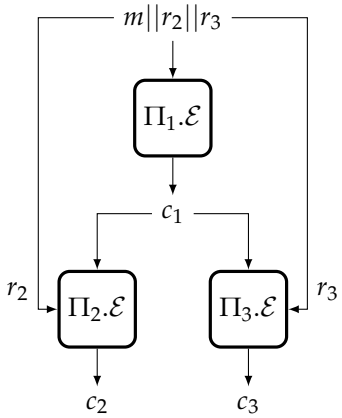

Figure 9. Scheme of the detectable sequential–parallel multiple encryption procedure. The final ciphertext is $C = (c_2, c_3)$.

Definition 13 (Detectable Sequential–Parallel Multiple Encryption)**.** *Let* Π_1 *be an IND-dCCA-secure PKE,* Π_2 *an 1b-IND-CCA-secure PKE, and* Π_3 *an IND-CPA-secure PKE with corresponding message spaces* $\mathcal{M}_i$ *and ciphertext spaces* $\mathcal{C}_i$*,* $1 \leq i \leq 3$*. The detectable sequential–parallel multiple encryption scheme defined over the message space* $\mathcal{M}^3 = \mathcal{M}_1$ *and the ciphertext space* $\mathcal{C}^3 = \mathcal{C}_2 \times \mathcal{C}_3$ *is given by a triple of algorithms* $(\mathcal{K}^3, \mathcal{E}^3, \mathcal{D}^3)$*, where*

- $\mathcal{K}^3$*, the key generation algorithm, is a probabilistic algorithm that takes the security parameters* (k_1, k_2, k_3) *as the input. It invokes the key generation algorithm of each PKE component* Π_i *in order to obtain three keypairs* $(pk_i, sk_i) \leftarrow \Pi_i.\mathcal{K}(1^{k_i})$*. The output of* $\mathcal{K}^3$ *is* $(PK, SK) = ((pk_1, pk_2, pk_3), (sk_1, sk_2, sk_3))$*.*
- $\mathcal{E}^3$*, the encryption algorithm, is a probabilistic algorithm that takes as the input a multiple encryption public key* PK *and a message* $m \in \mathcal{M}^3$ *and computes the ciphertext* $C \in \mathcal{C}^3$ *as follows: Choose two random values* $r_i \leftarrow \{0,1\}^{rlen_i(k_i)}$ *for* $i \in \{2,3\}$*, where* $rlen_i(\cdot)$ *returns the required number of random coin tosses for* Π_i*. First, compute* $c_1 \leftarrow \Pi_1.\mathcal{E}(pk_1, m||r_2||r_3)$*. Then, for* $i \in \{2,3\}$*, perform* $c_i \leftarrow \Pi_i.\mathcal{E}(pk_i, c_1; r_i)$*. Output* $C = (c_2, c_3)$*.*
- $\mathcal{D}^3$*, the decryption algorithm, is a deterministic algorithm that takes as the input a multiple encryption key pair* (PK, SK) *and a ciphertext* $C' = (c_2', c_3')$ *and decrypts the message as follows: Decrypt the first ciphertext by computing the decryption of* $c_1' \leftarrow \Pi_2.\mathcal{D}(sk_2, c_2')$*. Recover the message and the random coins by computing* $(m', r_2', r_3') \leftarrow \Pi_1.\mathcal{D}(sk_1, c_1')$*. Verify the validity of the ciphertext by checking if* $c_i' = \Pi_i.\mathcal{E}(pk_i, c_1'; r_i')$ *for* $i \in \{2,3\}$*. If both checks pass, return the message* m'*. Otherwise, return* $\perp$*.*

Hohenberger et al. proved that their construction is $(3,3)$-IND-wMCCA secure in the ROM, designed from weaker component ciphers. But for this to be the case, all component ciphers must remain secure with respect to their required security notion.

6.2. Quantum Augmented KEM-DEM Multiple Encryption

The *quantum augmented KEM-DEM multiple encryption* from Goncalves and Mashatan [20] is the first combiner whose security has been proven in the QROM. It expands the KEM-DEM paradigm for constructing secure PKEs of Cramer and Shoup [51]. In this classical KEM-DEM construction, the asymmetric key-pair of the KEM is used as the key-pair for the resulting PKE. The KEM is used to derive a symmetric key for the DEM, which then encrypts the message. The resulting ciphertext consists of the encapsulated KEM-key and the DEM-ciphertext. This paradigm is extended in the quantum augmented KEM-DEM by encrypting the DEM-ciphertext a second time with a PKE component. Similar to the other M-PKE schemes, a random value is used as random coins for the KEM and the PKE. This allows checking of the validity of the ciphertext during the decryption and prevents re-encryption attacks.

We shortly review the functionality of KEMs for completeness. A KEM κ is given by a triple of algorithms $(\mathcal{K}, \mathcal{E}, \mathcal{D})$. The probabilistic key generation algorithm $\kappa.\mathcal{K}$ takes as the input a security parameter 1^k and outputs a decapsulation and an encapsulation key $(dk, ek) \leftarrow \kappa.\mathcal{K}(1^k)$. The probabilistic key encapsulation algorithm $\kappa.\mathcal{E}$ takes as the input an encapsulation key ek and outputs a shared secret k and its encapsulation c. The decapsulation algorithm $\kappa.\mathcal{D}$ takes as the input a decapsulation key dk and an encapsulated key c and outputs the shared secret k. An IND-CCA secure KEM can be constructed from weaker PKEs [52,53].

The encryption structure of the quantum augmented KEM-DEM multiple encryption is depicted in Figure 10.

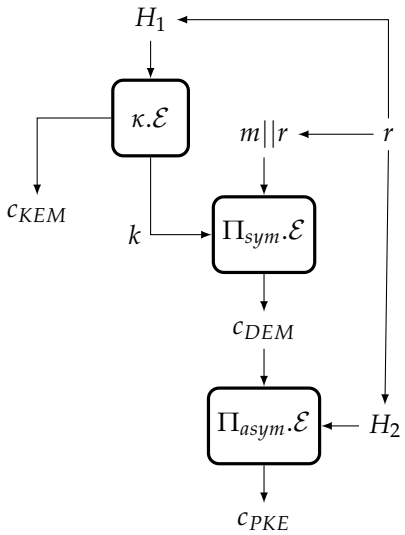

Figure 10. Scheme of the quantum augmented KEM-DEM multiple encryption procedure. The final ciphertext is $C = (c_{KEM}, c_{PKE})$.

Definition 14 (Quantum Augmented KEM-DEM Multiple Encryption)**.** *Let κ be a KEM, Π_{asym} a PKE, and Π_{sym} a DEM, with corresponding message spaces $\mathcal{M}_i$ and ciphertext spaces $\mathcal{C}_i$, $i \in \{KEM, asym, sym\}$. Let $H_1 : \{0,1\}^l \rightarrow \{0,1\}^{rlen_{KEM}(k_{KEM})}$ and $H_2 : \{0,1\}^l \rightarrow \{0,1\}^{rlen_{asym}(k_{asym})}$ be two hash functions, where $rlen(\cdot)$ returns the required number of random coin tosses for the KEM and PKE, respectively. The quantum augmented KEM-DEM multiple encryption defined over the message space $\mathcal{M}^2 = \mathcal{M}_{sym}$ and the ciphertext space $\mathcal{C}^2 = \mathcal{C}_{KEM} \times \mathcal{C}_{asym}$ is given by a triple of algorithms $(\mathcal{K}^2, \mathcal{E}^2, \mathcal{D}^2)$, where*

- *$\mathcal{K}^2$, the key generation algorithm, is a probabilistic algorithm that takes the security parameters (k_{KEM}, k_{asym}) as the input. It invokes the key generation algorithm of the KEM and the PKE component in order to obtain two keypairs $(pk, sk) \leftarrow \Pi_{asym}.\mathcal{K}(1^{k_{asym}})$ and $(dk, ek) \leftarrow \kappa.\mathcal{K}(1^{k_{KEM}})$. The output of $\mathcal{K}^2$ is $(PK, SK) = ((pk, ek), (sk, dk))$.*
- *$\mathcal{E}^2$, the encryption algorithm, is a probabilistic algorithm that takes as the input a multiple encryption public key PK and a message $m \in \mathcal{M}^2$ and computes the ciphertext $C \in \mathcal{C}^2$ as follows: Choose a random value $r \xleftarrow{\$} \{0,1\}^l$ and call the key encapsulation function to obtain $(k, c_{KEM}) \leftarrow \kappa.\mathcal{E}(ek; H_1(r))$. Use the generated shared secret to encrypt the message m and r with the DEM, $c_{DEM} \leftarrow \Pi_{sym}(k, m||r)$. Then, encrypt the ciphertext from the DEM again with the PKE, $c_{PKE} \leftarrow \Pi_{asym}.\mathcal{E}(pk, c_{DEM}; H_2(r)$. Output as ciphertext $C = (c_{KEM}, c_{PKE})$.*
- *$\mathcal{D}^2$, the decryption algorithm, is a deterministic algorithm that takes as the input a multiple encryption key pair (PK, SK) and a ciphertext $C' = (c'_{KEM}, c'_{PKE})$ and decrypts the message as follows: Decrypt the asymmetric ciphertext and obtain $c'_{DEM} \leftarrow \Pi_{asym}.\mathcal{D}(sk, c'_{PKE})$. Decapsulate the shared secret $k' \leftarrow \kappa.\mathcal{D}(dk, c'_{KEM})$. Recover the message and the random coins by computing $(m'||r') \leftarrow \Pi_{sym}.\mathcal{D}(k', c'_{DEM})$. Verify the validity of the ciphertext by*

checking if $c'_{KEM} = \kappa.\mathcal{E}(ek; H_1(r'))$ *and* $c'_{PKE} = \Pi_{asym}.\mathcal{E}(pk, c'_{DEM}; H_2(r'))$. *If both checks pass, return the message* m'. *Otherwise, return* $\perp$.

The quantum augmented KEM-DEM multiple encryption is proven by Goncalves and Mashatan [20] to be an M-PKE that is $(1,2)$-secure for the IND-wMCCA notion in the ROM *and* in the QROM. For this, the KEM or the PKE component must be IND-CCA secure in the ROM or in the QROM, respectively. A drawback of this scheme is that it uses three different cryptographic primitives (KEM, DEM, and PKE). We refer the reader to Bindel et al. [26] for more information on the QROM and how adversaries and attacks are modeled therein.

7. Discussion and Recommendations

In this section, we provide an evaluation of the M-PKE schemes covered in this paper. We focus on discussing the properties of security, efficiency, and complexity of the schemes. We also provide recommendations for M-PKE schemes based on some common use cases. A general overview of the M-PKE schemes discussed in this paper is provided in Table 1.

7.1. Security

Simply encrypting a message multiple times with different encryption algorithms does not necessarily mean that the result is more secure. This "careless" combination can even break a combined scheme altogether, if only one secret key is leaked or one algorithm is broken. Therefore, it is important to analyze M-PKE schemes based on cryptographic standards. The most secure notion in the context of multiple encryption is the $(1,n)$-IND-sMCCA notion. Here, all but one of the encryption components are broken or the keys are leaked, and the adversary additionally has access to intermediate decryption values from the decryption oracle. Even in this "worst case", the confidentiality of the plaintext must be preserved. Two of the presented schemes, the hashed sequential M-PKE (Section 4.2) and the secret sharing signature M-PKE (Section 5.3), achieve this highest security level.

Unfortunately, there are no strict established standards for M-PKE security notions in the research community. It is therefore possible that other M-PKE schemes may also achieve this level of security. The standard security level for PKE schemes is the IND-CCA2 security notion. In the M-PKE setting, this translates to the IND-wMCCA2 notion. There are several schemes that achieve this level of security: the detectable M-PKE (Section 6.1), which can be constructed from weaker components, but has the disadvantage that all components must remain secure; the improved parallel M-PKE (Section 5.2), which can even be constructed from much weaker components; and the quantum augmented M-PKE (Section 6.2), which uses IND-CCA2 components.

Most of the ciphers are proven to be secure in the ROM, with the exception being the secret sharing signature M-PKE (Section 5.3), which is proven to be secure in the standard model. As is usually the case with schemes in the standard model, this combiner is more complex than others and requires additional cryptographic primitives. As we move into the quantum era and transition to post-quantum schemes, it is also important to study the security of schemes in the QROM. Such schemes remain secure as long as the underlying components remain secure against quantum algorithms. So far, the only M-PKE scheme that is secure in the QROM is the quantum augmented M-PKE (Section 6.2). Which of the other M-PKE schemes can be proven to remain secure in the QROM remains an open question.

Table 1. Overview of the presented M-PKE schemes.

M-PKE Scheme	Ref.	Design	Security Implication	Proof Model	Ciphertext Size	Additional Primitives
Natural Sequential	Herzberg [15] and Zhang et al. [16]	Sequential	IND-CCA1 $\Rightarrow$ $(1,n)$-IND-wMCCA1	Standard	$\|c\|$	-
Hashed Sequential	Fujioka et al. [18]	Sequential	IND-CCA2 $\Rightarrow$ $(1,n)$-IND-sMCCA	ROM	$\|c\|$	Hash functions
XOR Sequential	Zhang et al. [16]	Sequential	OW-CPA $\Rightarrow$ $(1,n)$-IND-wMCCA2 *	ROM	$n \cdot \|c\| + \|m\|$	AONT and hash functions
Natural Parallel	Zhang et al. [22]	Parallel	IND-CCA1 $\Rightarrow$ $(1,n)$-IND-wMCCA1 *	Standard	$n \cdot \|c\|$	AONT
Improved Natural Parallel	Zhang et al. [22]	Parallel	OW-CPA $\Rightarrow$ $(1,n)$-IND-wMCCA2	ROM	$n \cdot (\|c\| + \|m_i\|)$	AONT and hash functions
Secret Sharing Signature	Dodis and Katz [17]	Parallel	IND-CCA2 $\Rightarrow$ $(n-t,n)$-IND-sMCCA	Standard	$n \cdot \|c\| + \|\sigma\| + \|k\|$	(t,n) secret sharing scheme and signature scheme
Natural XOR	Asmuth and Blakely [14]	Parallel	-	-	$n \cdot \|c\| + \|m\|$	-
XOR-Input	Herzberg [15]	Parallel	IND-CCA1 $\Rightarrow$ $(n-1,n)$-IND-wMCCA1	Standard	$n \cdot \|c\|$	-
Detectable	Hohenberger et al. [19]	Sequential–Parallel	IND-dCCA, 1b-IND-CCA, IND-CPA $\Rightarrow$ $(3,3)$-IND-wMCCA2	ROM	$2 \cdot \|c\|$	PKEs with different securities
Quantum Augmented KEM-DEM	Goncalves and Mashatan [20]	Sequential–Parallel	IND-CCA2 $\Rightarrow$ $(1,2)$-IND-wMCCA2	ROM and QROM	$2 \cdot \|c\|$	Hash functions, KEM and DEM

* Without explicit proof.

7.2. Efficiency and Complexity

When discussing efficiency of M-PKE schemes, the two most relevant properties are the asymptotic run time and the size of the ciphertext. In general, the sequential schemes have a longer run time, but allow for a short ciphertext, while the parallel schemes have a shorter run time, but increase the size of the ciphertext. Other properties, such as actual run time, the memory consumption, and the size of the implementation, depend on the algorithms chosen, the implementation, and the underlying hardware. The asymptotic run time of the sequential schemes is simply the sum of the individual run times of the component ciphers, since each encryption step is completely dependent on the result of the previous step and no parallelization is possible. The parallel schemes have the advantage that they can be executed in parallel for the most part. Only some pre-computations, such as choosing random values and splitting the message, cannot be parallelized. In addition, the secret sharing signature scheme computes a signature of all the ciphertexts. Therefore, the run time of the parallel schemes depends on the parallelization capabilities of the executing hardware and on the slowest algorithm. The fastest parallel scheme would be the improved natural parallel M-PKE (Section 5.2), while achieving IND-wMCCA2 security. The two combined sequential–parallel schemes have different run times. While the detectable M-PKE scheme can execute the second layer in parallel, the quantum augmented M-PKE scheme must run all computations sequentially.

The ciphertext size of the M-PKE schemes is depended on the underlying designs of the schemes (see Table 1). The sequential schemes have the advantage of their resulting ciphertext being only the ciphertext of the last encryption step. Therefore, they offer the smallest possible ciphertext of all M-PKE schemes. In contrast, the parallel schemes must transmit all ciphertexts of the component ciphers to the receiver. In addition, the natural XOR M-PKE (Section 5.4) transmits the XOR of the message, and the more secure schemes require additional information. These can be encrypted random values, XOR values, or signatures. An overview of the expected ciphertext size of the schemes is given in Table 1, where $|c|$ is the ciphertext size of a component cipher, $|m|$ is the message size, and $|m_i|$ is the size of a message share after pre-computation. The ciphertext of the secret sharing signature scheme also transmits a signature (size $|\sigma|$) and a corresponding verification key (size $|k|$).

The overall complexity of the schemes differs significantly. While the natural schemes do not require additional functions, the more secure schemes use additional primitives, such as AONT and signatures. This is important to keep in mind when choosing M-PKE schemes, especially for computationally restricted devices.

7.3. Recommendations

When it comes to suitable M-PKE algorithms, considering the following properties or use cases can guide the selection process. In general, we consider schemes with proven $(1, n)$-IND-wMCCA security or above (IND-MCCA and IND-sMCCA) to be secure enough. This corresponds to the standard IND-CCA security for PKE schemes. The following recommendations refer to asymptotic behavior and do not consider specific PKE selections and implementations.

- *Ciphertext size*. If the size of the ciphertext is relevant, we recommend using the hashed sequential scheme (Section 4.2). It is secure in the sense of $(1, n)$-IND-sMCCA from IND-CCA PKE components and only uses hash functions as an additional cryptographic primitive.
- *Run time*. If the overall run time of the M-PKE scheme is important, we recommend using the improved natural parallel scheme (Section 5.2). Note that this advantage is lost if the PKE components cannot be parallelized, e.g., due to hardware limitations. If it is not possible to parallelize the computation, the hashed sequential scheme can be used. In this case, the hashed sequential scheme is just as fast as the improved natural parallel scheme, but offers smaller ciphertexts.

- *Additional primitives*. We recommend the hashed sequential scheme (Section 4.2) when only a limited selection of cryptographic primitives is available. Especially lightweight cryptographic libraries may lack the support of AONTs or secret sharing schemes.
- *Quantum resistance*. The only proven M-PKE scheme in the QROM is the quantum augmented KEM-DEM scheme (Section 6.2). One should be careful when using this scheme since there is a patent by the authors describing their scheme [54]. It is also possible to use schemes that are proven to be secure in the standard model, when they do not rely on cryptographic hash functions being a truly random function, like the secret sharing signature scheme (Section 5.3). Unfortunately, this scheme uses digital signatures to ensure the validity of the ciphertext. Therefore, (hybrid) quantum-resistant digital signature schemes must be used to guarantee the quantum resistance, which complicates the scheme and may affect the existing security proofs. Currently, we cannot recommend any M-PKE schemes for quantum-safe use; this is still an open research question (see Section 9).

8. Conclusions

Multiple encryption is an established method of increasing the security of encryption schemes by using more than one scheme for the encryption process. This option becomes more relevant for PKE schemes with the advent of quantum computers, which threaten public-key cryptography in particular. In this way, novel post-quantum cryptography can already be integrated into existing systems without compromising current security levels.

In this work, we studied the combination of multiple encryption schemes in the public key setting. We first provided an overview of the security notions for M-PKE schemes and covered the current state of M-PKE research for the sequential, parallel, and sequential–parallel constructions. The schemes were discussed regarding their security, efficiency, and complexity. In addition to the description and analysis, we identified the most relevant research directions in order to apply M-PKE schemes to the quantum world.

9. Future Directions

While M-PKE schemes offer an appealing solution for the transition into a quantum-resistant infrastructure, there are still ongoing challenges to make M-PKE suitable for the quantum world. Comparing different multiple encryption schemes remains challenging due to the fact that there are many different security notions for M-PKE. In this paper, we compared and adapted the various security notions for the presented schemes. Nevertheless, several published schemes have not been studied by their authors regarding these indistinguishability notions, and comparison remains difficult since non-standard PKE security notions are used for the M-PKE proofs. We believe that a unified model for the indistinguishability notions for multiple encryption would help comparing existing and future schemes.

Going forward, the security of the hybrid schemes against quantum adversaries should be studied in depth. This applies, first of all, to the schemes that are proven to be secure in the ROM, whose security now must be proven in the QROM. But it can also affect schemes in the standard model that use additional primitives that are affected by quantum computers. This is an ongoing research to which all cryptographic primitives and protocols must adapt.

Although the NIST has announced the first group of PQC winners, there are still many viable PQC schemes to choose from, especially since the fourth round of the NIST PQC Standardization Competition is still ongoing. It can also be expected that other standardization bodies will evaluate and standardize additional PQC schemes. This increases the number of possible PQC schemes depending on use cases and security expectation. Therefore, it will be necessary to evaluate real-world implementations and applications of M-PKE schemes that use a variety of combinations of classical and PQC schemes. Of consideration should be, among other things, the overall run times, ciphertext sizes, and implementation sizes of the combined schemes based on different use cases.

In order to speed up the runtimes of M-PKE schemes even further, the evaluation of hardware implementations of M-PKE schemes will become increasingly important. To the best of our knowledge, there has not yet been any significant research on specific hardware implementations of M-PKE schemes. Nevertheless, hardware implementations of classical asymmetric schemes (e.g., [55–58]) and asymmetric PQC encryption schemes already exist (e.g., [59,60]). Combining and utilizing the efficiency and speedup of these existing hardware implementations in future research will be useful to enhance M-PKE schemes further, especially for computationally constrained devices.

Author Contributions: Conceptualization, T.S. and N.B.; methodology, T.S.; validation, N.B. and M.M.; investigation, T.S.; writing—original draft preparation, T.S., N.B. and M.M.; writing—review and editing, N.B. and M.M.; visualization, T.S.; supervision, N.B.; project administration, T.S. 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.

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

References

1. Rivest, R.L.; Shamir, A.; Adleman, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. Assoc. Comput. Mach.* **1978**, *21*, 120–126. [CrossRef]
2. Diffie, W.; Hellman, M.E. New Directions in Cryptography. *IEEE Trans. Inf. Theory* **1976**, *22*, 644–654. [CrossRef]
3. Arute, F.; Arya, K.; Babbush, R.; Bacon, D.; Bardin, J.C.; Barends, R.; Biswas, R.; Boixo, S.; Brandao, F.G.S.L.; Buell, D.A.; et al. Quantum Supremacy Using a Programmable Superconducting Processor. *Nature* **2019**, *574*, 505–510. [CrossRef] [PubMed]
4. IBM. IBM Unveils 400 Qubit-Plus Quantum Processor and Next-Generation IBM Quantum System Two. 2022. Available online: https://newsroom.ibm.com/2022-11-09-IBM-Unveils-400-Qubit-Plus-Quantum-Processor-and-Next-Generation-IBM-Quantum-System-Two (accessed on 2 September 2023).
5. NIST. Selected Algorithms 2022—Post-Quantum Cryptography | CSRC | CSRC. 2022 Available online: https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022 (accessed on 25 August 2023).
6. Beullens, W. Breaking Rainbow Takes a Weekend on a Laptop. In *Advances in Cryptology—CRYPTO 2022*; Dodis, Y., Shrimpton, T., Eds.; Lecture Notes in Computer Science; Springer Nature: Cham, Switzerland, 2022; pp. 464–479. [CrossRef]
7. Castryck, W.; Decru, T. An Efficient Key Recovery Attack on SIDH. In *Advances in Cryptology–EUROCRYPT 2023*; Hazay, C., Stam, M., Eds.; Lecture Notes in Computer Science; Springer Nature: Cham, Switzerland, 2023; pp. 423–447. [CrossRef]
8. Giron, A.A.; Custódio, R.; Rodríguez-Henríquez, F. Post-Quantum Hybrid Key Exchange: A Systematic Mapping Study. *J. Cryptogr. Eng.* **2022**, *13*, 71–88. [CrossRef]
9. Shannon, C.E. Communication Theory of Secrecy Systems. *Bell Syst. Tech. J.* **1949**, *28*, 656–715. [CrossRef]
10. Even, S.; Goldreich, O. On the Power of Cascade Ciphers. In *Advances in Cryptology: Proceedings of Crypto 83*; Chaum, D., Ed.; Springer: Boston, MA, USA, 1984; pp. 43–50. [CrossRef]
11. Maurer, U.M.; Massey, J.L. Cascade Ciphers: The Importance of Being First. *J. Cryptol.* **1993**, *6*, 55–61. [CrossRef]
12. Aiello, W.; Bellare, M.; Di Crescenzo, G.; Venkatesan, R. Security Amplification by Composition: The Case of Doubly-Iterated, Ideal Ciphers. In *Advances in Cryptology—CRYPTO '98*; Krawczyk, H., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1998; pp. 390–407. [CrossRef]
13. Merkle, R.C.; Hellman, M.E. On the Security of Multiple Encryption. *Commun. Acm* **1981**, *24*, 465–467. [CrossRef]
14. Asmuth, C.; Blakley, G. An Efficient Algorithm for Constructing a Cryptosystem Which Is Harder to Break than Two Other Cryptosystems. *Comput. Math. Appl.* **1981**, *7*, 447–450. [CrossRef]
15. Herzberg, A. Folklore, Practice and Theory of Robust Combiners. *J. Comput. Secur.* **2009**, *17*, 159–189. [CrossRef]
16. Zhang, R.; Hanaoka, G.; Shikata, J.; Imai. On the Security of Multiple Encryption or CCA-security+CCA-security=CCA-security? 2003. Available online: https://eprint.iacr.org/2003/181 (accessed on 6 July 2023).
17. Dodis, Y.; Katz, J. Chosen-Ciphertext Security of Multiple Encryption. In *Theory of Cryptography*; Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J.M., Mattern, F., Mitchell, J.C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., et al., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3378, pp. 188–209. [CrossRef]
18. Fujioka, A.; Okamoto, Y.; Saito, T. Security of Sequential Multiple Encryption. In *Progress in Cryptology—LATINCRYPT 2010*; Abdalla, M., Barreto, P.S.L.M., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; pp. 20–39. [CrossRef]
19. Hohenberger, S.; Lewko, A.; Waters, B. Detecting Dangerous Queries: A New Approach for Chosen Ciphertext Security. In *Advances in Cryptology—EUROCRYPT 2012*; Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J.M., Mattern, F., Mitchell, J.C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., et al., Eds.; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7237, pp. 663–681. [CrossRef]

20. Goncalves, B.; Mashatan, A. Tightly Secure PKE Combiner in the Quantum Random Oracle Model. *Cryptography* **2022**, *6*, 15. [CrossRef]
21. Harnik, D.; Kilian, J.; Naor, M.; Reingold, O.; Rosen, A. On Robust Combiners for Oblivious Transfer and Other Primitives. In *Advances in Cryptology—EUROCRYPT 2005*; Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J.M., Mattern, F., Mitchell, J.C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., et al., Eds.; Springer: Berlin/Heidelberg, Germany, 2005; Volume 3494, pp. 96–113. [CrossRef]
22. Zhang, R.; Hanaoka, G.; Shikata, J.; Imai, H. On the Security of Multiple Encryption or CCA-security+CCA-security=CCA-security? In *Public Key Cryptography—PKC 2004*; Bao, F., Deng, R., Zhou, J., Eds.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2004; pp. 360–374. [CrossRef]
23. Fischlin, M.; Herzberg, A.; Bin-Noon, H.; Shulman, H. Obfuscation Combiners. In *Advances in Cryptology— CRYPTO 2016*; Robshaw, M.; Katz, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2016; Volume 9815, pp. 521–550. [CrossRef]
24. Jain, A.; Manohar, N.; Sahai, A. Combiners for Functional Encryption, Unconditionally. In *Advances in Cryptology— EUROCRYPT 2020*; Canteaut, A.; Ishai, Y., Eds.; Springer International Publishing: Cham, 2020; Volume 12105, pp. 141–168. [CrossRef]
25. Giacon, F.; Heuer, F.; Poettering, B. KEM Combiners. In *Public-Key Cryptography— PKC 2018*; Abdalla, M.; Dahab, R., Eds.; Springer International Publishing: Cham, 2018; Volume 10769, pp. 190–218. [CrossRef]
26. Bindel, N.; Brendel, J.; Fischlin, M.; Goncalves, B.; Stebila, D. Hybrid Key Encapsulation Mechanisms and Authenticated Key Exchange. In *Post-Quantum Cryptography*; Ding, J., Steinwandt, R., Eds.; Springer International Publishing: Cham, Switzerland 2019; Volume 11505, pp. 206–226. [CrossRef]
27. Huguenin-Dumittan, L.; Vaudenay, S. FO-like Combiners and Hybrid Post-Quantum Cryptography. In *Cryptology and Network Security*; Conti, M., Stevens, M., Krenn, S., Eds.; Springer International Publishing: Cham, Switzerland, 2021; Volume 13099, pp. 225–244. [CrossRef]
28. Braithwaite, M. Experimenting with Post-Quantum Cryptography. Available online: https://security.googleblog.com/2016/07/experimenting-with-post-quantum.html (accessed on 30 January 2023).
29. Alkim, E.; Ducas, L.; Pöppelmann, T.; Schwabe, P. Post-Quantum Key Exchange: A New Hope. In Proceedings of the 25th USENIX Conference on Security Symposium, SEC'16, Austin, TX, USA, 10–12 August 2016; USENIX Association: Berkeley, CA, USA, 2016; pp. 327–343.
30. Hülsing, A.; Rijneveld, J.; Schanck, J.; Schwabe, P. High-Speed Key Encapsulation from NTRU. In *Cryptographic Hardware and Embedded Systems —CHES 2017*; Fischer, W.; Homma, N., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2017; pp. 232–252. [CrossRef]
31. Kölbl, S.; Misoczki, R.; Schmieg, S. Why Google Now Uses Post-Quantum Cryptography for Internal Comms. Available online: https://cloud.google.com/blog/products/identity-security/why-google-now-uses-post-quantum-cryptography-for-internal-comms (accessed on 30 January 2023).
32. Kwiatkowski, K.; Valenta, L. TLS Post-Quantum Experiment. 2019. Available online: http://blog.cloudflare.com/the-tls-post-quantum-experiment/ (accessed on 23 April 2023).
33. Easterbrook, K.; Paquin, C. Post-Quantum TLS. Available online: https://www.microsoft.com/en-us/research/project/post-quantum-tls (accessed on 26 April 2023).
34. Anastasova, M.; Kampanakis, P.; Massimo, J. PQ-HPKE: Post quantum hybrid public key encryption. In Proceedings of the ICMC 2022, Limerick, Ireland, 3–9 July 2022.
35. Bellare, M.; Desai, A.; Pointcheval, D.; Rogaway, P. Relations among Notions of Security for Public-Key Encryption Schemes. In *Advances in Cryptology—CRYPTO '98*; Goos, G.; Hartmanis, J.; van Leeuwen, J.; Krawczyk, H., Eds.; Springer: Berlin/Heidelberg, Germany, 1998; Volume 1462, pp. 26–45. [CrossRef]
36. Katz, J.; Yung, M. Characterization of Security Notions for Probabilistic Private-Key Encryption. *J. Cryptol.* **2006**, *19*, 67–95. [CrossRef]
37. Canetti, R.; Krawczyk, H.; Nielsen, J.B. Relaxing Chosen-Ciphertext Security. In *the Advances in Cryptology—CRYPTO 2003*; Boneh, D., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2003; pp. 565–582. [CrossRef]
38. An, J.H.; Dodis, Y.; Rabin, T. On the Security of Joint Signature and Encryption. In *Advances in Cryptology—EUROCRYPT 2002*; Goos, G., Hartmanis, J., van Leeuwen, J., Knudsen, L.R., Eds.; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2332, pp. 83–107. [CrossRef]
39. Zhang, C.; Cash, D.; Wang, X.; Yu, X.; Chow, S.S.M. Combiners for Chosen-Ciphertext Security. In *Computing and Combinatorics*; Dinh, T.N., Thai, M.T., Eds.; Springer International Publishing: Cham, Switzerland, 2016; Volume 9797, pp. 257–268. [CrossRef]
40. Fujisaki, E.; Okamoto, T. Secure Integration of Asymmetric and Symmetric Encryption Schemes. *J. Cryptol.* **2013**, *26*, 80–101. [CrossRef]
41. Rivest, R.L. All-or-Nothing Encryption and the Package Transform. In *Fast Software Encryption*; Biham, E., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 1997; pp. 210–218. [CrossRef]
42. Boyko, V. On the Security Properties of OAEP as an All-or-Nothing Transform. In *Advances in Cryptology—CRYPTO' 99*; Goos, G.; Hartmanis, J., van Leeuwen, J., Wiener, M., Eds.; Springer: Berlin/Heidelberg, Germany, 1999; Volume 1666, pp. 503–518. [CrossRef]

43. Desai, A. The Security of All-or-Nothing Encryption: Protecting against Exhaustive Key Search. In *Advances in Cryptology—CRYPTO 2000*; Bellare, M., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2000; pp. 359–375. [CrossRef]
44. McEvoy, R.P.; Murphy, C.C. Efficient All-or-Nothing Encryption Using CTR Mode. In *E-Business and Telecommunication Networks*; Filipe, J., Obaidat, M.S., Eds.; Communications in Computer and Information Science; Springer: Berlin, Heidelberg, Germany, 2008; pp. 92–106. [CrossRef]
45. Blakley, G.R. Safeguarding Cryptographic Keys. In Proceedings of the 1979 International Workshop on Managing Requirements Knowledge (MARK), New York, NY, USA, 4–7 June 1979; pp. 313–318. [CrossRef]
46. Shamir, A. How to Share a Secret. *Commun. ACM* **1979**, *22*, 612–613. [CrossRef]
47. Lv, C.; Jia, X.; Lin, J.; Jing, J.; Tian, L.; Sun, M. Efficient Secret Sharing Schemes. In *Secure and Trust Computing, Data Management and Applications*; Park, J.J.; Lopez, J., Yeo, S.S., Shon, T., Taniar, D., Eds.; Communications in Computer and Information Science; Springer: Berlin/Heidelberg, Germany, 2011; pp. 114–121. [CrossRef]
48. Wang, Y.; Desmedt, Y. Efficient Secret Sharing Schemes Achieving Optimal Information Rate. In Proceedings of the 2014 IEEE Information Theory Workshop (ITW 2014), Hobart, TAS, Australia, 2–5 November 2014; pp. 516–520. [CrossRef]
49. Ding, J.; Lin, C.; Wang, H.; Xing, C. Communication Efficient Secret Sharing With Small Share Size. *IEEE Trans. Inf. Theory* **2022**, *68*, 659–669. [CrossRef]
50. *ISO/IEC 18033-2:2006*; Information Technology—Security Techniques—Encryption Algorithms—Part 2: Asymmetric Ciphers. ISO: Geneva, Switzerland, 2006.
51. Cramer, R.; Shoup, V. Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack. *SIAM J. Comput.* **2004**, *33*, 167–226. [CrossRef]
52. Dent, A.W. A Designer's Guide to KEMs. In *Cryptography and Coding*; Goos, G., Hartmanis, J., van Leeuwen, J., Paterson, K.G., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2898, pp. 133–151. [CrossRef]
53. Hofheinz, D.; Hövelmanns, K.; Kiltz, E. A Modular Analysis of the Fujisaki-Okamoto Transformation. In *Theory of Cryptography*; Kalai, Y., Reyzin, L., Eds.; Springer International Publishing: Cham, Switzerland, 2017; Volume 10677, pp. 341–371. [CrossRef]
54. Goncalves, B.; Mashatan, A.; Fallah, J.; Byrne, K.; Siddavaatam, P. Quantum-Augmentable Hybrid Encryption System and Method. U.S. Patent 11,431,498, 30 August 2022.
55. Rahman, M.; Rokon, I.R.; Rahman, M. Efficient Hardware Implementation of RSA Cryptography. In Proceedings of the And Identification in Communication 2009 3rd International Conference on Anti-Counterfeiting, Security, Hong Kong, China, 20–22 August 2009; pp. 316–319. [CrossRef]
56. Thabah, S.D.; Sonowal, M.; Ahmed, R.U.; Saha, P. Fast and Area Efficient Implementation of RSA Algorithm. *Procedia Comput. Sci.* **2019**, *165*, 525–531. [CrossRef]
57. Zhang, Y.; Chen, D.; Choi, Y.; Chen, L.; Ko, S.B. A High Performance ECC Hardware Implementation with Instruction-Level Parallelism over GF(2163). *Microprocess. Microsystems* **2010**, *34*, 228–236. [CrossRef]
58. MuthuKumar, B.; Jeevananthan, S. High Speed Hardware Implementation of an Elliptic Curve Cryptography (ECC) Co-Processor. In Proceedings of the Trendz in Information Sciences & Computing(TISC2010), Chennai, India, 17–19 December 2010; pp. 176–180. [CrossRef]
59. Xing, Y.; Li, S. A Compact Hardware Implementation of CCA-Secure Key Exchange Mechanism CRYSTALS-KYBER on FPGA. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2021**, *2021*, 328–356. [CrossRef]
60. Jati, A.; Gupta, N.; Chattopadhyay, A.; Sanadhya, S.K. A Configurable CRYSTALS-Kyber Hardware Implementation with Side-Channel Protection. *ACM Trans. Embed. Comput. Syst.* **2023**. [CrossRef]