Chosen-Ciphertext Security of Multiple Encryption

Yevgeniy Dodis[1⋆] and Jonathan Katz[2⋆⋆]

[1] Dept. of Computer Science, New York University.
[2] Dept. of Computer Science, University of Maryland.

Abstract. Encryption of data using multiple, independent encryption schemes ("multiple encryption") has been suggested in a variety of contexts, and can be used, for example, to protect against partial key exposure or cryptanalysis, or to enforce threshold access to data. Most prior work on this subject has focused on the security of multiple encryption against *chosen-plaintext attacks*, and has shown constructions secure in this sense based on the chosen-plaintext security of the component schemes. Subsequent work has sometimes assumed that these solutions are also secure against *chosen-ciphertext attacks* when component schemes with stronger security properties are used. Unfortunately, this intuition is false for all existing multiple encryption schemes.

Here, in addition to formalizing the problem of chosen-ciphertext security for multiple encryption, we give simple, efficient, and generic constructions of multiple encryption schemes secure against chosen-ciphertext attacks (based on *any* component schemes secure against such attacks) in the standard model. We also give a more efficient construction from any (hierarchical) identity-based encryption scheme secure against selective-identity *chosen plaintext* attacks. Finally, we discuss a wide range of applications for our proposed schemes.

1 Introduction

Encrypting data using multiple, independent instantiations of a basic encryption scheme (or schemes) is a simple — yet powerful — approach which can be used both to improve security as well as to provide additional functionality not present in any of the underlying schemes. The security implications of *multiple encryption* (as we refer to it here) were noted as early as Shannon [38], who proposed using "product ciphers" to enhance the security of symmetric-key primitives. This idea was further explored and rigorously formalized in a number of subsequent works (e.g., [32, 21, 31]) analyzing the security of *cascade ciphers* (in which a message m is encrypted via $\mathcal{E}_{k_1}(\mathcal{E}'_{k_2}(m))$, where k_1, k_2 are two independent keys and $\mathcal{E}, \mathcal{E}'$ are symmetric-key encryption schemes) in the symmetric-key setting. The approach can be applied to public-key encryption as

⋆ This work was supported by the NSF under CAREER Award CCR-0133806 and Trusted Computing Grant CCR-0311095.

⋆⋆ This work was supported by NSF Trusted Computing Grant CCR-0310751.

well; for example, "cascaded encryption" (which we will call *sequential encryption*) was advocated as part of the NESSIE recommendation [34]: "[f]or very high level security we note that double encryption... gives a good range of security".

Multiple encryption, of which sequential encryption is but one example, offers at least two potential security advantages: First, the resulting scheme may be secure as long as *any one of* the component schemes is secure (indeed, sequential encryption is secure against chosen-plaintext attacks as long as either of the component schemes are). Thus, multiple encryption offers a way to "hedge one's bets" about the security of any particular scheme (see also the recent work of Herzberg [28]). This is especially important when the security of different schemes depends upon different, and incomparable, cryptographic assumptions. A second potential advantage of multiple encryption is that the resulting encryption scheme may in fact be *more secure* than any of the component schemes; this is the rationale, for example, behind using triple-DES (see also [1]).

Beyond the security-oriented advantages listed above, multiple encryption schemes potentially offer *functionality* not present in any of the component schemes. We briefly highlight two applications of multiple encryption, and defer a more detailed discussion of these and other applications to Section 6:

Threshold encryption. In a threshold encryption scheme [16], the data is encrypted in such a way that only particular sets of users can recover it; typically, a scheme requires any t-out-of-n users in order to decrypt, but more general access structures can also be considered. Multiple encryption gives *generic* constructions of threshold encryption in either the private- or public-key settings. For example, to enforce n-out-of-n decryption in the private-key setting, one may provide each user i with an independent key k_i and encrypt a message M via $\mathcal{E}^1_{k_1}(M_1), \ldots, \mathcal{E}^i_{k_i}(M_i)$, where the M_i are chosen at random subject to $\oplus_{i=1}^n M_i = M$ (and the $\mathcal{E}^i$ may, in general, be different schemes). Let J (with $|J| < n$) represent the set of corrupted players; i.e., if $j \in J$ then the adversary has the key k_j. The above scheme, which we will refer to as *parallel encryption*, informally satisfies the following level of security against chosen-plaintext attacks: as long as *any* encryption scheme $\mathcal{E}^i$ with $i \notin J$ is secure, the message remains secret. Thus, in addition to enabling threshold access to the data, this scheme also allows one again to "hedge one's bets" about the security of any particular scheme (as in the case of sequential encryption, discussed earlier).

(Strong) key-insulated encryption. Multiple encryption has also been used to give a generic construction of a key-insulated public-key encryption scheme secure against chosen-plaintext attacks [20]. Without going into the full details — and omitting some details unimportant for the present discussion — in this case a message M is encrypted by first splitting the message into *shares* $M_1, \ldots, M_i$ and then encrypting each share M_i with respect to a particular public key PK_i. (This general technique is similar to the parallel encryption discussed above; indeed, parallel encryption is obtained if the shares constitute an n-out-of-n sharing of M.) If the message is "split" a second time (before the sharing described above), and one of these shares is encrypted with a public key whose secret key is known only to the user, it is possible to obtain a generic construction of *strong* key-insulated encryption [20].

Other applications. We remark that multiple encryption is applicable to many other domains as well, including anonymous routing [14, 26], broadcast encryption [22], proxy encryption (see [19]), and certificate-based encryption [24]. We defer a more detailed discussion to Section 6.

1.1 Motivation for our Work

Chosen-ciphertext security ("CCA security") is as much of a concern in each of the above settings as it is in the case of standard encryption. One might hope to achieve CCA security for any of the above settings by simply "plugging in" an appropriate CCA-secure multiple encryption scheme. However (with one recent exception; see below), *we are unaware of any previous work which considers chosen-ciphertext security for multiple encryption.* To be clear: there has been much work aimed at giving solutions for *specific* applications using *specific* number-theoretic assumptions: for example, in the context of CCA-secure threshold encryption [40, 13, 30], broadcast encryption [20], and key-insulated encryption [18]. However, this type of approach suffers from at least two drawbacks: first, it does not provide *generic* solutions, but instead only provides solutions based on very specific assumptions. Second, the derived solutions are application-dependent, and must be constantly "re-invented" and modified each time one wants to apply the techniques to a new domain. Although solutions based on specific assumptions are often more efficient than generic solutions, it is important to at least be aware that a generic solution exists so that its efficiency can be directly compared with a solution based on specific assumptions. Indeed, we argue in Section 6 that for some applications, a generic solution may be roughly as efficient as (or may offer reasonable efficiency tradeoffs as compared to) the best currently-known solutions based on specific assumptions.

Making the problem even more acute is that currently-known schemes for multiple encryption are demonstrably *insecure* against chosen-ciphertext attacks (this holds even with respect to the weakest definition considered here; see Section 3.1). Zhang, et al. [41] have also recently noticed this problem, and appear to be the first to have considered chosen-ciphertext security for multiple encryption. We compare our work to theirs in the following section.

1.2 Our Contributions

Our results may be summarized as follows:

Definitions of security. We provide formal definitions of chosen-ciphertext security for multiple encryption. Interestingly, multiple definitions make sense in this context, and we introduce three such definitions and briefly comment on the relationships between them. We also which of these definitions is the "right" one for a number of different applications.

CCA-secure multiple encryption. We show two constructions of CCA-secure multiple encryption schemes which are *generic* (i.e., they may be constructed based on *any* CCA-secure standard encryption scheme) and are proven secure *in the standard model*. Our first construction achieves a "basic" level of security which suffices for many (but not all!) applications of multiple encryption. Our second construction satisfies the strongest notion of security proposed here, and suffices for all applications we consider. We also show a more efficient construction based on any (hierarchical) identity-based encryption scheme secure against selective-identity *chosen plaintext* attacks.

Applications. As mentioned earlier, our work was motivated by the applications of CCA-secure multiple encryption to a variety of settings; we therefore conclude the paper by sketching a number of applications of the constructions given here. Our resulting schemes are, for most cases, the first known *generic* constructions achieving CCA security in the given setting. Furthermore, in some cases the solutions we give are roughly as efficient as (or even more efficient than) previous solutions which were based on very specific assumptions. As one example, we show a CCA-secure threshold encryption scheme with completely *non-interactive* decryption (and a proof of security in the standard model); for the two-party case, our solution is roughly as efficient as the only previous solution [30].

Comparison to previous work. Our definitions differ from those given by Zhang, et al. [41], and the definitions given in their work are weaker than those given here. In fact, the best construction given by Zhang, et al. only satisfies the weakest of our definitions; therefore, their constructions are not sufficient for certain applications such as threshold encryption. (Indeed, they concentrate primarily on the application to key-insulated encryption, while we consider a much wider range of applications.) Finally, their constructions require the random oracle model whereas our results all hold in the standard model.

2 Preliminaries

We begin by introducing some notation. A (standard) public-key encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ consists of three PPT algorithms: the key-generation algorithm Gen takes as input security parameter 1^k and outputs a encryption key EK and a decryption key DK. The randomized encryption algorithm Enc takes as input EK, a label ℓ, and a message m, and outputs a ciphertext C; for brevity, we sometimes omit EK and write this as $C \leftarrow \mathsf{Enc}^{\ell}(m)$. The decryption algorithm Dec takes as input DK, a ciphertext C, and a label ℓ; it outputs a message m, or $\perp$ if C is "invalid". We write this as $m \leftarrow \mathsf{Dec}^{\ell}(C)$ (where we again sometimes omit DK). We assume $\mathsf{Dec}^{\ell}(\mathsf{Enc}^{\ell}(m)) = m$ for any message m and label ℓ. Security for encryption is defined following [3, 39]. In particular, we use "CPA-secure" to refer to what is called $\mathsf{IND\text{-}CPA}$ security in [3], and "CCA-secure" to refer to what is called $\mathsf{IND\text{-}CCA2}$ in [3] (modified to take labels into account as in [39]).

A signature scheme $\Sigma = (\mathsf{Sig\text{-}Gen}, \mathsf{Sig}, \mathsf{Ver})$ consists of three PPT algorithms: the key-generation algorithm $\mathsf{Sig\text{-}Gen}$ takes as input a security parameter 1^k and outputs a signing key SK and a verification key VK. The signing algorithm Sig takes as input SK and a message m, and outputs a signature σ; we will sometimes omit SK and write $\sigma \leftarrow \mathsf{Sig}(m)$. The verification algorithm Ver takes as input

VK, a message m, and a signature σ; it outputs 1 iff the signature is valid. We write this as $a \leftarrow \mathsf{Ver}(m, \sigma)$ (again, sometimes omitting VK). We require that $\mathsf{Ver}(m, \mathsf{Sig}(m)) = 1$, for all m.

Unless specified otherwise, the notion of security we consider for signature schemes is that of strong unforgeability under adaptive chosen-message attacks, following [27, 4]. We also use the notion of *one-time signature schemes* which satisfy an analogous definition of security except that an adversary is only allowed to request a signature on a single message.

Secret sharing schemes. A secret sharing scheme is a pair of transformations $\mathcal{SSS} = (\mathsf{Share}, \mathsf{Rec})$.[1] $\mathsf{Share}(\cdot)$ is a probabilistic transformation which takes a message M and outputs n secret shares $s_1, \ldots, s_n$ and possibly one public share pub. Rec is a deterministic transformation which takes n shares $s'_1, \ldots, s'_n$ (some of which might be $\perp$) and (if present) the public share pub, and outputs some message M' (possibly $\perp$). The basic correctness property states that $\mathsf{Rec}(\mathsf{Share}(M)) = M$. Security may be quantified by the following thresholds:

- t_p — the *privacy* threshold. Determines the maximum number of shares which (together with pub) reveal "no information" about the message.
- t_f — the *fault-tolerance* threshold. Determines the minimum number of correct shares which (together with pub) suffice to recover the message, when the other shares are *missing*.
- t_r — the *robustness* threshold. Determines the minimum number of correct shares which (together with pub) suffice to recover the message, when the other shares are *adversarially set*.
- t_s — the *soundness* threshold. Determines the minimum number of correct shares which (together with pub) ensure that it is impossible to recover an *incorrect* message $M' \notin \{M, \perp\}$, when the other shares are *adversarially set*.

The above must satisfy $t_p + 1 \leq t_f \leq t_r \leq n$ and $t_s \leq t_r$. The security properties corresponding to the thresholds above can all be formalized in a straightforward way, so we omit them. In a basic secret sharing scheme, only privacy and fault-tolerance are addressed. This is useful when all the parties holding the corresponding shares are trustworthy, but some shares may have been leaked to an adversary and/or some parties may be (temporarily) unavailable. Shamir's scheme [37] is the classical example; this scheme achieves information-theoretic privacy, has no public share, and achieves $t_f = t_p + 1$ and $|M| = |s_i|$. Generalizing this idea [23], one can achieve arbitrary $t_f > t_p$. Krawczyk [29] extended Shamir's scheme to the computational setting by using the scheme to share a short symmetric key k, and then encrypting the message M using k. The resulting ciphertext can either be stored publicly, or shared among the servers using an *information dispersal scheme* [35] (i.e., a secret sharing scheme which achieves fault-tolerance and/or robustness, but has $t_p = 0$). In fact, this approach can be applied to any information-theoretic secret sharing scheme to obtain a computational scheme with share size proportional to the security parameter and public

[1] Sometimes, we may also have a setup procedure which prepares public parameters. For simplicity, we omit this from our description.

part proportional to the message length. When fault-tolerance is not needed, one can also use computational all-or-nothing transforms (AONTs) [36, 11] to achieve extremely short shares.

Sometimes, basic secret sharing schemes already enjoy certain robustness properties. For example, Shamir's scheme achieves $t_r = (n + t_f)/2$. Moreover, there are several simple methods to transform any (t_p, t_f, n)-secret sharing scheme into a *robust* (t_p, t_f, t_r, t_s, n)-secret sharing scheme (in a computational sense), achieving optimal values $t_s = 0$ and $t_r = t_f$. We describe two such methods now. In both methods, the dealer first computes the regular sharing $(s_1, \ldots, s_n, \mathsf{pub})$ of M. In the first method, the dealer then generates signing/verification keys $(\mathsf{SK}, \mathsf{VK})$ for a signature scheme, and sets $s'_i = (s_i, \mathsf{Sig}_{\mathsf{SK}}(i, s_i))$, $\mathsf{pub}' = (\mathsf{pub}, \mathsf{VK})$. To reconstruct, users apply the original reconstruction algorithm only to shares whose signatures are correct. In the second method, the dealer uses a commitment scheme to commit to (i, s_i); let c_i (resp., d_i) be the corresponding commitment (resp., decommitment). The dealer then sets $s'_i = (s_i, d_i)$, $\mathsf{pub}' = (\mathsf{pub}, c_1, \ldots, c_n)$. As before, users will only use those shares whose proper decommitment is revealed. In this second method the size of the public information is $O(n)$, but using, e.g., Merkle trees this storage can be reduced considerably at the expense of slightly increasing the share size.

3 Multiple Encryption

We now define a multiple encryption scheme.

Definition 1. *A (non-interactive) public-key* multiple encryption scheme *is a tuple of* PPT *algorithms* $\mathcal{TE} = (\mathsf{TGen}, \mathsf{TEnc}, \mathsf{Split}, \mathsf{TDec}, \mathsf{Combine})$ *such that:*

- TGen*, the* key generation algorithm*, is a probabilistic algorithm which takes as input a security parameter* 1^k *and outputs a public key* TEK *along with* n *secret keys* $\mathbf{TDK} = (\mathsf{TDK}_1, \ldots, \mathsf{TDK}_n)$*.*
- TEnc*, the* encryption algorithm*, is a probabilistic algorithm which takes as input a public key* TEK*, a message* M*, and a label* L*. It outputs a ciphertext* $C \leftarrow \mathsf{TEnc}^L(M)$*.*
- Split*, the* splitting algorithm*, is a deterministic algorithm which takes as input a public key* TEK*, a ciphertext* C *and a label* L*. It either outputs* $\perp$*, or* n ciphertext shares $\mathbf{C} = (C_1, \ldots, C_n)$ *and some auxiliary info* aux*.*
- TDec*, the* partial decryption algorithm*, takes as input* $i \in \{1, \ldots, n\}$*, a secret key* TDK_i*, and a ciphertext share* C_i*; it outputs the message share* M_i *or the distinguished symbol* $\perp$*. We denote the output of this algorithm by* $\mathsf{TDec}^i(C_i)$*. We also let* $\mathbf{DEC}(\mathbf{C}, \mathsf{aux}) \stackrel{\text{def}}{=} (\mathsf{TDec}^1(C_1), \ldots, \mathsf{TDec}^n(C_n), \mathsf{aux})$*.*
- $\mathsf{Combine}$*, the* combining algorithm*, takes as input shares* $\mathbf{M} = (M_1, \ldots, M_n)$ *and the auxiliary info* aux*, and outputs a message* M *or* $\perp$*.*

Correctness (refined later) requires that for all $\mathsf{TEK}, \mathbf{TDK}$ *output by* TGen*, all messages* M *and labels* L*, we have:* $\mathsf{Combine}(\mathbf{DEC}(\mathsf{Split}^L(\mathsf{TEnc}^L(M)))) = M$*.*

Before discussing security, a few remarks are in place. It is important to recognize that multiple encryption might be used in a number of different scenarios.

In one scenario, the set of decryption keys $\mathbf{TDK}$ (or some subset of these keys) are co-located, so a single user receiving a ciphertext C would perform the splitting, partial decryption, and combining by itself. In another scenario, there are a set of n servers and server i stores TDK_i. Here, a user receiving a ciphertext C would perform the splitting himself to obtain $\mathbf{C}, \mathsf{aux}$, would keep aux, and would send the ciphertext share C_i to server i for decryption. Server i would respond with M_i and the various message shares would be combined by the user to recover the original message. These different ways of thinking about the decryption process are each appropriate for different applications of multiple encryption.

When decryption keys TDK_i are stored at different locations (i.e., on different servers), the above definition implies that servers do not communicate with each other and do not keep any intermediate state. Also, we remark that we could have ignored the splitting algorithm altogether and simply have TDK_i operate on the entire ciphertext C (performing any splitting itself, as necessary). The reason for not doing so is that C might contain information which is not "relevant" to server i, and thus sending the entire ciphertext to each server might be wasteful. In fact, our solutions achieve $\sum |C_i| = O(|C|)$, so the total communication between the user and *all* servers is proportional to the size of the original ciphertext.

In either of the above scenarios (i.e., whether the decryption keys are co-located or stored at different servers), it is possible for some of the decryption keys to be compromised by an adversary. This raises the first security issue, which is that of *message privacy*. When keys are stored on separate servers, there is also the possibility that some servers may be compromised in their entirety; this raises the additional issue of *decryption robustness*. Since the security issues in the latter case are stronger than those in the former case, for the remainder of this section we will speak in terms of a central user running the splitting/combining algorithm and n servers performing the partial decryption of each share.

Message privacy. We assume that the adversary may learn $t_p < n$ decryption keys, where t_p is the *privacy threshold*. Formally, given a set $I = \left\{i_1, \ldots, i_{t_p}\right\}$, an adversary is given a randomly-generated public key TEK, the set of secret keys $\mathsf{TDK}_I = \left\{\mathsf{TDK}_{i_1}, \ldots, \mathsf{TDK}_{i_{t_p}}\right\}$, and oracle access to some oracle $\mathcal{O}$ whose meaning will be clarified shortly. $\mathcal{B}$ outputs two messages M_0, M_1 (along with some label L), and receives a challenge ciphertext $C \leftarrow \mathsf{TEnc}(M_b)$ for a randomly-chosen b. The adversary succeeds if it correctly guesses b, and the adversary's advantage is defined as the absolute value of the difference between its success probability and 1/2. If the oracle $\mathcal{O}$ is "empty", we say that $\mathcal{B}$ is performing a *(multiple) chosen-plaintext attack* (MCPA). As for the *(multiple) chosen-ciphertext attack*, there are several meaningful flavors described below in the order of increasing adversarial power.

In the weakest such attack, denoted wMCCA ("weak MCCA"), we have $\mathcal{O} = \mathsf{Combine}(\mathbf{DEC}(\mathsf{Split}^{(\cdot)}(\cdot)))$ (where the adversary is prohibited from submitting (C, L) to this oracle). Namely, $\mathcal{B}$ only gets access to the entire decryption process without seeing any partial decryption results and without being able to ask questions to the decryption servers directly. While this notion already suffices for some applications, it assumes that the adversary can never see the intermediate decryption shares. In a (regular) MCCA attack, we let $\mathcal{O} = \mathbf{DEC}(\mathsf{Split}^{(\cdot)}(\cdot))$ (as before, we forbid the adversary from submitting (C, L) to this oracle). Namely, we still assume that the ciphertext gets passed through a proper splitting procedure but $\mathcal{B}$ also learns the intermediate decryption results $M_1, \ldots, M_n$. As we shall see, this attack is sufficient for most applications of multiple encryption.

However, sometimes we need to consider an even stronger attack denoted sMCCA (for "strong MCCA"), where we have $\mathcal{O} = \mathsf{TDec}^{(\cdot)}(\cdot)$. Namely, we allow $\mathcal{B}$ to ask arbitrary and questions to the individual decryption servers. Of course, to make sense of this attack, we need to add some restrictions. First and most obvious, for a challenge ciphertext C (with label L) we disallow questions (i, C_i), where C_i is the ciphertext share for server i that results from "splitting" C using label L. Second and less obvious, we assume (for all i) that the mapping Split_i from (C, L) to C_i is *weakly collision-resistant*. This means that no PPT adversary $\mathcal{A}$ can succeed with non-negligible probability in the following game: $\mathcal{A}(\mathbf{TDK})$ supplies some pair (M, L) to the encryption oracle, and gets back a ciphertext $C \leftarrow \mathsf{TEnc}^L(M)$. $\mathcal{A}$ succeeds if it can output a pair $(C', L') \neq (C, L)$ and an index i such that $\mathsf{Split}_i(C, L) = \mathsf{Split}_i(C', L')$. Indeed, without this latter condition it seems unnecessarily restrictive to prohibit the adversary $\mathcal{B}$ in the sMCCA game from asking questions $(i, C_i = \mathsf{Split}_i(C, L))$. This is because there is a chance such a question might have "legally" come from a different ciphertext $(C', L') \neq (C, L)$. We further observe that when the Split procedure *does* satisfy this condition, the sMCCA attack is at least as strong as the MCCA attack,[2] and it is easy to see that this conclusion does *not* hold without weak collision resistance. Therefore, we will insist on weak collision-resistance when talking about sMCCA attacks.

Definition 2. *Let $X \in \{\mathsf{MCPA}, \mathsf{wMCCA}, \mathsf{MCCA}, \mathsf{sMCCA}\}$. We say multiple encryption scheme $\mathcal{TE}$ is X-*secure *with privacy threshold t_p, if the advantage of any* PPT *adversary $\mathcal{B}$ performing attack X with any set I of size t_p is negligible.*

Decryption robustness. The correctness property of Definition 1 only ensures correct decryption when all algorithms are honestly and correctly executed. Just as in the case of secret sharing, however, one may often desire fault-tolerance, robustness, and/or soundness. As in the case of secret sharing, these are parameterized by thresholds t_f, t_r, t_s, whose meaning is completely analogous to their meaning in the case of secret sharing (described earlier). Our solutions can achieve optimal $t_s = 0$, $t_r = t_f$, and any $t_p < t_f$.

3.1 Insecurity of Known Multiple Encryption Schemes

It is instructive to note that known constructions of multiple encryption schemes (even when instantiated with a CCA-secure standard encryption scheme) are insecure under the weakest definition of chosen-ciphertext security considered above. We briefly illustrate this for the simplest case of $n = 2$.

[2] This is so since one can simulate (with all but negligible probability) any "allowed" call to $\mathbf{DEC}(\mathsf{Split}^{(\cdot)}(\cdot))$ by n "allowed" calls to $\mathsf{TDec}^{(\cdot)}(\cdot)$.

In sequential encryption, M is encrypted via $C \leftarrow \mathsf{Enc}_{\mathsf{EK}_1}(\mathsf{Enc}_{\mathsf{EK}_2}(M))$. An adversary, when given the decryption key DK_1 and a challenge ciphertext C, can break the encryption scheme as follows: decrypt C using DK_1 to obtain $C' \in \mathsf{Enc}_{\mathsf{EK}_2}(M)$ and then *re-encrypt* C' using EK_1; this results in a second, different ciphertext $\tilde{C}$. Now, by submitting $\tilde{C}$ to its decryption oracle, the adversary will receive in return the original message M.

Attacks are also possible for the case of parallel encryption. Here, a message M is encrypted as $C = \langle C_1, C_2 \rangle$, where $C_1 \leftarrow \mathsf{Enc}_{\mathsf{EK}_1}(s_1)$, $C_2 \leftarrow \mathsf{Enc}_{\mathsf{EK}_2}(s_2)$, and s_1 and s_2 are chosen at random subject to $s_1 \oplus s_2 = M$. Now, even without being given any decryption keys, an adversary given a challenge ciphertext C can compute $\tilde{C}_1 \leftarrow \mathsf{Enc}_{\mathsf{EK}_1}(0)$ and $\tilde{C}_2 \leftarrow \mathsf{Enc}_{\mathsf{EK}_2}(0)$, and then submit the ciphertexts $\langle \tilde{C}_1, C_2 \rangle$ and $\langle C_1, \tilde{C}_2 \rangle$. Note that the adversary thus obtains both s_1 and s_2 separately, from which it can recover the original message $M = s_1 \oplus s_2$.

4 Generic Constructions

In this section we describe how to build MCCA- and sMCCA-secure multiple encryption schemes from any (standard) CCA-secure encryption scheme $\mathcal{E}$. In our schemes, the decryption keys will simply be decryption keys DK_i independently-generated by $\mathcal{E}$, and partial decryption will essentially require only a single decryption with this key. Our results achieve: (1) ciphertext length linear in the length of the plaintext message; (2) communication with each server *independent* of the number of servers and the length of the message. We also stress that when the decryption keys are held by several servers, no interaction between servers is required. A drawback is that the ciphertext and public-key lengths in our solutions are proportional to the number of decryption keys n (of course, for small n, such as the important case of $n = 2$, this is not a problem). We believe that this dependence in unavoidable if we are not willing to assume any algebraic structure on $\mathcal{E}$. Indeed, in the following section we show how this dependence can be avoided when starting from (hierarchical) identity-based encryption.

For the remainder of this section, let $\mathcal{SSS} = (\mathsf{Share}, \mathsf{Rec})$ be a (t_p, t_f, t_r, t_s, n)-secret sharing scheme. All multiple encryption schemes we construct will inherent the same thresholds t_p, t_f, t_r, t_s, which elegantly allows us to push all the privacy and robustness constraints onto the much simpler secret sharing primitive.

4.1 Achieving Chosen-Ciphertext Security

Recall that in parallel encryption the message M is first shared using $\mathcal{SSS}$, and then each share is separately encrypted using an independent key. As noted earlier, this folklore scheme is not secure against chosen-ciphertext attacks (even against a weak MCCA attack and with no corrupted keys). We show a simple and elegant way to extend parallel encryption so as to solve this problem, without introducing much extra complexity. In brief, we use a secure one-time signature scheme $\Sigma = (\mathsf{Sig\text{-}Gen}, \mathsf{Sig}, \mathsf{Ver})$ to bind all the local ciphertexts to each other (and to the label L). The main twist which makes this work is that we also bind the verification key of Σ to each of the ciphertexts.

Before giving the formal description of our solution, we illustrate our construction for the case $n = 2$ (with $t_f = t_r = t_s = 2, t_p = 1$, and no labels). The public key consists of two independently-generated keys $\mathsf{EK}_1, \mathsf{EK}_2$, and the secret key contains the corresponding decryption keys $\mathsf{DK}_1, \mathsf{DK}_2$. Let $\mathsf{Enc}_1 \stackrel{\text{def}}{=} \mathsf{Enc}_{\mathsf{EK}_1}$ and similarly for Enc_2. To encrypt M, a sender first "splits" M by choosing random s_1 and setting $s_2 = M \oplus s_1$. The sender then generates a key pair $(\mathsf{VK}, \mathsf{SK})$ for a one-time signature scheme, and computes $C_1 \leftarrow \mathsf{Enc}_1^{\mathsf{VK}}(s_1)$ and $C_2 \leftarrow \mathsf{Enc}_2^{\mathsf{VK}}(s_2)$. Finally, the sender computes $\sigma = \mathsf{Sig}_{\mathsf{SK}}(C_1, C_2)$; the complete ciphertext is $\langle \mathsf{VK}, C_1, C_2, \sigma \rangle$. Decryption is done in the obvious way: if σ is not a valid signature on C_1, C_2 with respect to VK, the ciphertext is invalid. Otherwise, DK_1 and DK_2 are used to obtain s_1 and s_2 from which the original message $M = s_1 \oplus s_2$ can be recovered.

We now generalize this solution to arbitrary n and using an arbitrary secret sharing scheme $\mathcal{SSS} = (\mathsf{Share}, \mathsf{Rec})$.

- $\underline{\mathsf{TGen}(1^k)}$: For $i = 1, \ldots, n$, let $(\mathsf{EK}_i, \mathsf{DK}_i) \leftarrow \mathsf{Gen}(1^k)$ and set $\mathsf{TEK} = \underline{(\mathsf{EK}_1, \ldots, \mathsf{EK}_n)}$, $\mathsf{TDK}_i = \mathsf{DK}_i$, so that $\mathbf{TDK} = (\mathsf{DK}_1, \ldots, \mathsf{DK}_n)$. Below, let $\mathsf{Enc}_i \stackrel{\text{def}}{=} \mathsf{Enc}_{\mathsf{EK}_i}$ and $\mathsf{Dec}_i \stackrel{\text{def}}{=} \mathsf{Dec}_{\mathsf{DK}_i}$.
- $\underline{\mathsf{TEnc}^L(M)}$: Let $(s_1, \ldots, s_n, \mathsf{pub}) \leftarrow \mathsf{Share}(M)$, and $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Sig\text{-}Gen}(1^k)$. Set $C_i = \mathsf{Enc}_i^{\mathsf{VK}}(s_i)$ (for $i = 1, \ldots, n$) and then compute the signature $\sigma = \mathsf{Sig}_{\mathsf{SK}}(C_1, \ldots, C_n, \mathsf{pub}, L)$. Output $C = (C_1, \ldots, C_n, \mathsf{pub}, \mathsf{VK}, \sigma)$.
- $\underline{\mathsf{Split}^L(C)}$: Parse C as $(C_1, \ldots, C_n, \mathsf{pub}, \mathsf{VK}, \sigma)$, and reject if verification fails; i.e., if $\mathsf{Ver}_{\mathsf{VK}}((C_1, \ldots, C_n, \mathsf{pub}, L), \sigma) = 0$. Otherwise, set ciphertext share $\hat{C}_i = (C_i, \mathsf{VK})$ and $\mathsf{aux} = \mathsf{pub}$.
- $\underline{\mathsf{TDec}^i(C_i, \mathsf{VK})}$: Output $s_i' = \mathsf{Dec}_i^{\mathsf{VK}}(C_i)$.
- $\underline{\mathsf{Combine}(s_1', \ldots, s_n', \mathsf{pub})}$: Output $\mathsf{Rec}(s_1', \ldots s_n', \mathsf{pub})$.

As with the folklore scheme, each decryption server simply performs a single regular (now CCA-secure) decryption, but here using a label which is the verification key of a one-time signature scheme (and which is used to bind all the ciphertexts together). We claim:

Theorem 1. *If $\mathcal{E}$ is CCA-secure, $\mathcal{SSS}$ is a (t_p, t_f, t_r, t_s, n)-secret sharing scheme, and Σ is a secure one-time signature scheme, then $\mathcal{TE}$ is MCCA-secure with thresholds t_p, t_f, t_r, t_s.*

Proof. Robustness thresholds t_f, t_r, t_s follow immediately from those of the secret sharing scheme, due to the definition of $\mathsf{Combine} = \mathsf{Rec}$. We now argue message privacy.

Assume there exists some PPT adversary $\mathcal{B}$ attacking MCCA-security who has some non-negligible advantage. Recall, this $\mathcal{B}$ has oracle access to $\mathcal{O}(\cdot, \cdot) = \mathbf{DEC}(\mathsf{Split}^{(\cdot)}(\cdot))$, chooses some messages M_0, M_1 and a label L, gets an unknown ciphertext $C = (C_1, \ldots, C_n, \mathsf{pub}, \mathsf{VK}, \sigma)$, and tries to guess whether this corresponds to the encryption of M_0 or M_1 (with label L). Let X denote the event that $\mathcal{B}$ asks $\mathcal{O}$ a query $(C', L') \neq (C, L)$, where C' includes the same verification key $\mathsf{VK}' = \mathsf{VK}$ as the challenge, but σ' is a new, valid signature (with respect to VK) of the corresponding "message" $(C'_1, \ldots, C'_n, \mathsf{pub}', L')$. It is immediate that $\Pr[X] = \mathsf{negl}(k)$, or else an easy argument (omitted) shows that we can use $\mathcal{B}$ to construct a PPT adversary breaking the security of the one-time signature scheme Σ with non-negligible advantage.

We can therefore construct an adversary $\mathcal{B}'$ who never makes a query to $\mathcal{O}$ using the same verification key as in the challenge ciphertext, yet whose advantage is negligibly close to the advantage of $\mathcal{B}$. Let ε_0 denote the advantage of $\mathcal{B}'$, and assume w.l.o.g. that $\mathcal{B}'$ corrupts servers $\{n - t_p + 1, \ldots, n\}$. We refer to this game involving $\mathcal{B}'$ as G_0, and now gradually change this game into games $G_1, \ldots, G_{n-t_p}$. In general, G_i is identical to G_0, except for one step in the computation of the challenge ciphertext C. Recall, in G_0 we have $C_j \leftarrow \mathsf{Enc}_j^{\mathsf{VK}}(s_j)$, where s_j is the j-th share of the secret sharing scheme. In game G_i we instead do this only for $j > i$, but set $C_i \leftarrow \mathsf{Enc}_i^{L}(0)$ for $j \leq i$ (where 0 is some arbitrary fixed message in our space). In other words, G_{i-1} and G_i are identical except G_{i-1} sets $C_i \leftarrow \mathsf{Enc}_i^{\mathsf{VK}}(s_i)$, while G_i sets $C_i \leftarrow \mathsf{Enc}_i^{\mathsf{VK}}(0)$ Denote by ε_i the advantage of $\mathcal{B}'$ in predicting the challenge bit b in game G_i. We claim that for every $1 \leq i \leq n - t_p$ we have $|\varepsilon_i - \varepsilon_{i-1}| = \mathsf{negl}(k)$.

To show the claim, using $\mathcal{B}'$ we construct an adversary $\mathcal{A}_i$ who succeeds in breaking CCA-security of $\mathcal{E}$ with advantage $\delta_i = \frac{1}{2}|\varepsilon_{i-1} - \varepsilon_i|$. Since $\mathcal{E}$ is assumed to be CCA-secure, the claim follows. $\mathcal{A}_i$ gets an encryption key EK for $\mathcal{E}$, sets $\mathsf{EK}_i = \mathsf{EK}$, and generates the remaining $(n-1)$ public/secret keys by himself. These public keys, as well as the last t_p secret keys, are given to $\mathcal{B}'$. Adversary $\mathcal{A}_i$ then honestly simulates the run of G_{i-1}/G_i until $\mathcal{B}'$ submits the challenge (M_0, M_1, L). At this point, $\mathcal{A}_i$ chooses a random bit b, generates $(\mathsf{SK}, \mathsf{VK})$, computes the shares $(s_1, \ldots, s_n, \mathsf{pub}) \leftarrow \mathsf{Share}(M_b)$, and prepares C_j for $j \neq i$ just as in G_{i-1} and G_i. Furthermore, $\mathcal{A}'$ outputs the challenge $(s_i, 0, \mathsf{VK})$ in its own CCA game. Upon receiving the challenge ciphertext C, it sets $C_i = C$, signs whatever is needed, and passes the resulting challenge ciphertext to $\mathcal{B}'$. It only remains to specify how $\mathcal{A}_i$ deals with oracle queries of $\mathcal{B}'$. Notice that $\mathcal{A}_i$ can decrypt all ciphertexts C'_j for $j \neq i$ by himself, since the appropriate decryption keys are known. As for C'_i, since (by construction) $\mathcal{B}'$ does not reuse the challenge value VK, this means that $\mathcal{A}_i$ can always submit C'_i to its decryption oracle using the label $\mathsf{VK}' \neq \mathsf{VK}$. Finally, $\mathcal{A}_i$ outputs 1 iff $\mathcal{B}'$ correctly predicts b. This completes the description of $\mathcal{A}_i$, and it is not hard to see that $\mathcal{A}_i$ gives a perfect simulation of either game G_{i-1} or G_i depending on which of s_i or 0 was encrypted. The claim regarding $|\varepsilon_{i-1} - \varepsilon_i|$ follows easily.

Now, since $(n - t_p)$ is polynomial in k and ε_0 is assumed to be non-negligible, we get that ε_{n-t_p} is non-negligible as well. But let us now examine the game G_{n-t_p} more closely. When encrypting the challenge M_b, only $t = t_p$ shares (and the value pub) are used in creating the ciphertext. But then the privacy of the secret sharing scheme implies that ε_{n-t_p} must be negligible, a contradiction. (This is not hard to see, and we omit the obvious details.)

Replacing signatures with MACs. At the expense of settling for (weaker) wMCCA-security, we can use the recent technique of Boneh and Katz [10] to replace the one-time signature scheme by the more efficient combination of a message authentication code (MAC) and a weak form of commitment. The idea is to commit to a MAC key τ, then encrypt both the message M and the decommitment d using the secret sharing technique above, but with the public verification key VK replaced by the commitment c. Finally, τ is used to compute a message authentication code on the entire resulting ciphertext. In brief, the reason this only yields wMCCA-security is that the message authentication code computed over the ciphertext (as opposed to the one-time signature computed above) can only be verified *after* all the shares are collected. More details are given in Appendix A.

4.2 Achieving Strong Chosen-Ciphertext Security

The scheme above does not enjoy sMCCA-security since, in particular, the mapping Split_i from (C, L) to C_i is not weakly collision-resistant; indeed, it ignores all ciphertexts other than C_i. A natural first thought is to simply append a hash α of the entire ciphertext C to each of the local decryption shares C_i (and let each server simply ignore α). While this may make each Split_i weakly collision-resistant, it will *not* achieve sMCCA-security: Since the servers ignore α anyway, the adversary can simply replace α by "garbage" while keeping the rest of the C_i the same; this will result in a "valid" decryption request to each server, but will result in a proper decryption of C_i to the adversary.

A natural way to fix this is to let each server check the hash α by sending to the server the entire ciphertext C. In fact, if we are willing to send the entire ciphertext to each server, we no longer need α: each server can just perform the corresponding splitting procedure on C by itself. In fact, doing so will trivially give sMCCA-security. However, sending all of C (and having each server perform the splitting procedure) may be wasteful in some scenarios; it therefore remains interesting to explore improved solutions with lower user-server communication and in which more of the work is shifted to the user rather than the servers.

For the case of the *particular* MCCA-secure scheme $\mathcal{TE}_{cca}$ of the previous section, sMCCA-security can be achieved at a very small additional cost. Let $\mathcal{H} = \{H\}$ be a family of collision-resistant hash functions. We now describe the modified scheme $\mathcal{TE}_{scca}$.

- $\underline{\mathsf{TGen}(1^k)}$. Sample $H \leftarrow \mathcal{H}$ and for $i = 1, \ldots, n$, let $(\mathsf{EK}_i, \mathsf{DK}_i) \leftarrow \mathsf{Gen}(1^k)$. Set $\mathsf{TEK} = (\mathsf{EK}_1, \ldots, \mathsf{EK}_n, H)$, $\mathsf{TDK}_i = \mathsf{DK}_i$. Below, denote $\mathsf{Enc}_i = \mathsf{Enc}_{\mathsf{EK}_i}$, $\mathsf{Dec}_i = \mathsf{Dec}_{\mathsf{DK}_i}$.
- $\underline{\mathsf{TEnc}^L(M)}$. Let $(s_1, \ldots, s_n, \mathsf{pub}) \leftarrow \mathsf{Share}(M)$, and $(\mathsf{VK}, \mathsf{SK}) \leftarrow \mathsf{Sig\text{-}Gen}(1^k)$. Set $C_i = \mathsf{Enc}_i^{\mathsf{VK}}(s_i)$ for $i = 1, \ldots, n$; then compute $\alpha = H(C_1, \ldots, C_n, \mathsf{pub}, L)$ and $\sigma = \mathsf{Sig}_{\mathsf{SK}}(\alpha)$. Output $C = (C_1, \ldots, C_n, \mathsf{pub}, \mathsf{VK}, \sigma)$.
- $\underline{\mathsf{Split}^L(C)}$. Parse $C = (C_1, \ldots, C_n, \mathsf{pub}, \mathsf{VK}, \sigma)$, set $\alpha = H(C_1, \ldots, C_n, \mathsf{pub}, L)$, and reject if $\mathsf{Ver}_{\mathsf{VK}}(\alpha, \sigma) = 0$. Otherwise, set the ciphertext share to be $\hat{C}_i = (C_i, \mathsf{VK}, \alpha, \sigma)$ and set $\mathsf{aux} = \mathsf{pub}$.
- $\underline{\mathsf{TDec}^i(C_i, \mathsf{VK}, \alpha, \sigma)}$. Output $\mathsf{Dec}_i^{\mathsf{VK}}(C_i)$ if $\mathsf{Ver}_{\mathsf{VK}}(\alpha, \sigma) = 1$, and $\perp$ otherwise.

– $\underline{\mathsf{Combine}(s'_1, \ldots, s'_n, \mathsf{pub})}$. Output $\mathsf{Rec}(s'_1, \ldots s'_n, \mathsf{pub})$.

Thus, the only effective change is to force each server to verify a signature (of a one-time signature scheme) before performing the decryption. The cost of this will typically be small compared to the cost of decryption.

We now consider the security of the above. On an intuitive level, when an adversary makes a decryption query, either: (1) the adversary reuses a previous VK, which implies that it uses a previous α (due to unforgeability of the signature scheme), which in turn implies that the query is illegal (since H is collision-resistant); or (2) the adversary uses a new VK, in which case the chosen-ciphertext security of the underlying encryption schemes (which use VK as a label) implies that the resulting ciphertexts are unrelated to the challenge. Notice, there is no need for the server to check that α is the correct hash; having a valid signature of α implicitly assures the server that either this decryption query is unrelated to the challenge, or α is indeed correct due to the unforgeability of the one-time signature scheme. Notice also that once again the communication between the user and each server is independent of n. The above intuition can in fact be used to prove the following theorem:

Theorem 2. *If $\mathcal{E}$ is* CCA*-secure, $\mathcal{SSS}$ is a (t_p, t_f, t_r, t_s, n)-secret sharing scheme, Σ is a secure one-time signature scheme, and $\mathcal{H}$ is collision-resistant, then $\mathcal{TE}_{scca}$ is* sMCCA*-secure with thresholds t_p, t_f, t_r, t_s.*

Proof. As before, robustness thresholds t_f, t_r, t_s follow immediately from those of the secret sharing scheme since $\mathsf{Combine} = \mathsf{Rec}$. We now argue message privacy. Here we need to argue two things: indistinguishability of the scheme against sMCCA attack and weak collision resistance of the splitting procedure.

We start with the second part. Take any adversary $\mathcal{A}$ attacking weak collision resistance of $\mathcal{TE}_{scca}$. $\mathcal{A}$ gets the entire secret key $\mathbf{TDK}$, produces a pair (M, L), gets $C \leftarrow \mathsf{TEnc}^L(M)$, and outputs $(C', L') \neq (C, L)$ and an index i. If it is the case that $\mathsf{Split}_i(C, L) = \mathsf{Split}_i(C', L')$ then (by definition of Split) this means that $(C_i, \mathsf{VK}, \alpha, \sigma) = (C'_i, \mathsf{VK}', \alpha', \sigma')$. But then $H(C_1 \ldots C_n, \mathsf{pub}, L) = \alpha = \alpha' = H(C'_1 \ldots C'_n, \mathsf{pub}', L')$ and this violates collision-resistance of $\mathcal{H}$.

Next, we show security against sMCCA attack. Assume there exists some adversary $\mathcal{B}$ attacking sMCCA-security who has some non-negligible advantage. Recall, $\mathcal{B}$ has oracle access to $\mathcal{O}(\cdot, \cdot) = \mathsf{TDec}^{(\cdot)}(\cdot)$, chooses some messages M_0, M_1 and a label L, gets a challenge ciphertext $C = (C_1, \ldots, C_n, \mathsf{pub}, \mathsf{VK}, \sigma)$, and tries to predict whether this ciphertext corresponds to an encryption of M_0 or of M_1 (with label L). Let X denote the event that $\mathcal{B}$ asks $\mathcal{O}$ a query $(i, (C'_i, \mathsf{VK}, \alpha', \sigma'))$, where VK is the same verification key as the one used in the challenge ciphertext but σ' is a *new, valid* signature with respect to VK of the corresponding message α'. Namely, σ' is a valid signature of α', but $(\alpha', \sigma') \neq (\alpha, \sigma)$. Clearly, $\Pr(X) = \mathsf{negl}(k)$ or otherwise $\mathcal{B}$ can be used to break the security of the one-time signature scheme Σ.

We thus assume that X never occurs in the run of $\mathcal{B}$, yet $\mathcal{B}$ still has non-negligible advantage. Since $\mathcal{B}$ is forbidden to ask any challenge query of the form $(i, (C_i, \mathsf{VK}, \alpha, \sigma))$, this means that every query $(i, (C'_i, \mathsf{VK}', \alpha', \sigma'))$ that $\mathcal{B}$ makes satisfies one of three conditions: (1) $\mathsf{Ver}_{\mathsf{VK}'}(\alpha', \sigma) = 0$, in which case the response is automatically $\perp$ (and so we can assume that $\mathcal{B}$ never makes such a query); (2) $(\mathsf{VK}', \alpha', \sigma') = (\mathsf{VK}, \alpha, \sigma)$, but $C'_i \neq C_i$ (recall, we proved that $\mathsf{VK}' = \mathsf{VK}$ implies $(\alpha', \sigma') = (\alpha, \sigma)$, so the only way for this query to be legal while keeping $\mathsf{VK}' = \mathsf{VK}$ is to have $C'_i \neq C_i$); (3) $\mathsf{VK}' \neq \mathsf{VK}$. Since we excluded queries of type (1), we combine cases (2) and (3) to conclude that every query of $\mathcal{B}$ must have $(C'_i, \mathsf{VK}') \neq (C_i, \mathsf{VK})$.

Given this observation, the rest of the proof is almost identical to the proof of Theorem 1 (with obvious syntactic modifications). Namely, we create hybrid games in which encryptions of the shares of M_b are gradually replaced by encryptions of 0. As in the proof of the previous theorem, we show that any such change cannot be noticed by $\mathcal{B}$ since the corresponding encryption scheme $\mathcal{E}_i$ is CCA-secure. The only new aspect of this proof is the description of how $\mathcal{A}_i$ handles $\mathcal{B}$'s queries $(j, (C'_j, \mathsf{VK}', \alpha', \sigma'))$. When $j \neq i$, then $\mathcal{A}_i$ can simply decrypt by itself, as before. For $j = i$, $\mathcal{A}_i$ first checks the validity of the signature, and then asks its own decryption oracle to decrypt (C'_i, VK'). So all we need to argue is that this query is different from $\mathcal{A}_i$'s own challenge (C_i, VK) (which $\mathcal{A}_i$ is forbidden to ask). But this is precisely what we argued about $\mathcal{B}$'s behavior in the previous paragraph.

Remark 1. The existence of collision-resistant hash functions does not seem to follow from the existence of CCA-secure encryption schemes. However, by slightly sacrificing the efficiency of our construction, we can rely on universal one-way hash functions (UOWHFs) (which *are* implied by the existence of CCA-secure encryption) thus making our construction completely generic. Briefly, instead of using a single $H \in \mathcal{H}$ in the public key, the sender will choose a new $H \leftarrow \mathcal{H}$ for every encryption. The description of H will then be included as part of the ciphertext, signed together with α, and be included as part of each server's share. Since one can achieve $|H| \sim \log n$ [6], this still keeps the user-server communication very low.

5 Direct Constructions From Selective Identity IBE/HIBE Schemes

We assume the reader is familiar with the basic terminology of identity-based encryption (IBE) and hierarchical identity-based encryption (HIBE); see [8, 25]. Recently, Canetti et al. [12] gave a simple and elegant construction transforming a "weak" (so called selective-identity-secure) IBE scheme secure against CPA attacks into a CCA-secure (standard) public-key encryption scheme. Their transformation uses a secure one-time signature scheme, by first encrypting the message M with the identity VK (for newly chosen keys (SK, VK)), and then signing the resulting ciphertext with SK. The receiver, who stores the master secret key for the IBE scheme, can then decrypt the ciphertext if the signature is valid.

We could then use the resulting CCA-secure encryption schemes in our transformations to get CCA-secure multiple encryption schemes, where each server would store a master key for an independent IBE scheme. However, this will result in generating $(n+1)$ one-time keys and signatures per ciphertext, which is wasteful. Instead, we notice that the same verification key VK can be used as the identity for all n IBE schemes, and then used to sign the concatenation of n ciphertexts (or its hash). This gives a much more efficient direct construction with only a single one-time signature per ciphertext.

However, just like our original scheme, the public key of the resulting multiple encryption is still proportional to the number of parties n. We now show that using a *two-level hierarchical* IBE scheme (secure against selective-identity CPA-attack), we can make the first relatively generic multiple encryption scheme whose public key is *independent* of the number of players (although the ciphertext size still is). Specifically, the public key pk is simply the mater public key of the two-level HIBE. The i-th decryption key TDK_i consists of level-1 identity-based secret key corresponding to identity i. To encrypt a message M, the sender (as before) generates a key pair $(\mathsf{SK}, \mathsf{VK}) \leftarrow \mathsf{Sig\text{-}Gen}(1^k)$ and applies a secret sharing scheme to the message M resulting in shares $s_1 \ldots s_n$ (and pub). Now, however, the sender encrypts s_i "to" the level-2 identity (i, VK), and then signs the resulting ciphertexts (or their hash) using SK. Each server i can still decrypt its share since it knows the level-1 secret key for the parent identity i, while the collusion-resistance of the HIBE easily implies that no other coalition of servers can get any information from this share. We omit a formal proof in this abstract.

We remark that Boneh and Boyen [7] have recently constructed simple and efficient selective-identity IBE/HIBE schemes, which immediately give rise to simple and efficient multiple encryption schemes using our paradigm. In particular, using their HIBE scheme we get an efficient multiple encryption scheme with a constant-size public key. We also notice that the technique of replacing signatures by MACs [10] also applies here to obtain more efficient wMCCA-secure multiple encryption.

6 Applications

We outline in brief a number of applications of multiple encryption.

CCA-secure threshold encryption. In the generally-considered model for threshold encryption, there is a *combiner* who receives a ciphertext and sends some information to various servers who may then potentially interact, either with each other or with the combiner. The information sent to the servers is typically assumed to be the ciphertext itself, but in general (and in our case in particular) it is possible to transmit a smaller amount of information to each server. In either case, the servers then send decryption shares back to the combiner, who uses these to recover the original message. In a chosen-ciphertext attack on a threshold encryption scheme (see, e.g., [40, 13, 30]), an adversary can expose the decryption shares stored at some number of servers, observe a ciphertext C, and send ciphertexts $C' \neq C$ to the combiner. When it does so, in addition to receiving the decryption of C', it is also typically assumed that the adversary can observe all communication in the network, both between the servers and the combiner as well as between the servers themselves.

It is not hard to see that the adversarial model thus described corresponds exactly to a MCCA-attack. Moreover, if the combiner itself is untrusted (and can send what it likes to the servers), we effectively have a sMCCA-attack. Thus, any MCCA/sMCCA-secure multiple encryption scheme with privacy threshold t_p immediately gives a threshold encryption scheme with the same privacy threshold. Furthermore, a MCCA/sMCCA-secure multiple encryption scheme with robustness threshold t_r immediately gives a threshold encryption scheme in which the ciphertext can be correctly decrypted as long as t_r servers remain uncorrupted. Thresholds t_f and t_s can be interpreted similarly.

Our techniques thus give the first *generic* construction for CCA-secure threshold encryption (note that no previous generic solution existed even in the random oracle model). We remark further that for small values of n, our schemes are competitive with previous threshold schemes. For example, when $n = 2$ and we use the Cramer-Shoup [15] encryption scheme as our building block, we obtain a CCA-secure two-party public-key encryption scheme (in the standard model) which has more efficient decryption than the scheme recently proposed by MacKenzie [30]. In fact, although this construction increases the encryption time and ciphertext size by (roughly) a factor of two as compared to [30], the time required for decryption (by each server) is actually a factor of 10 *more* efficient; furthermore decryption in our case is completely non-interactive. As another example, if we use RSA-OAEP as our building block we obtain a very efficient solution for CCA-secure, RSA-based threshold encryption with completely non-interactive decryption (in the random oracle model).

CCA-secure key-insulated and strong key-insulated encryption. We assume the reader is somewhat familiar with the key-insulated model, as well as with the generic constructions of [20] (which achieve only CPA security). In a key-insulated public-key encryption scheme there is a *server* and a *user*; at the beginning of each time period, the user communicates with the server to update the user's secret key. Ciphertexts sent during any time period can be decrypted by the user alone, without any further communication with the server. The main property of such schemes is that exposing the secret information stored by the user during many time periods leaves *all* non-exposed periods secure.

At a high level, in the generic solution of [20] the server stores n secret keys for a standard encryption scheme (and the n corresponding public keys constitute the public key of the key-insulated scheme). At the beginning of each time period, some ℓ of these secret keys are given to the user. To encrypt a message during a particular time period, the sender first splits the message into ℓ shares using a secret-sharing scheme, and then encrypts each of these shares using one of the ℓ keys associated with the current time period. The keys are chosen in such a way so that multiple exposures of the user do not compromise "enough" of the ℓ keys associated with any other time periods. (In [20], it is shown how to "tune" n and ℓ to achieve the desired level of security in a reasonably-efficient way.)

It is immediately apparent that the above encryption process (namely, splitting the message and then encrypting each share with an independent key) corresponds exactly to multiple encryption. For this particular application, a single user stores all ℓ keys that are used to decrypt during a given time period; therefore, a chosen-ciphertext attack against a key-insulated cryptosystem is equivalent to a wMCCA attack on a multiple encryption scheme (that is, an adversary does not get to see the individual shares output by each partial decryption algorithm). Thus, any wMCCA-secure multiple encryption scheme can be used to achieve CCA-secure key-insulated encryption. We remark that robustness is not needed for this particular application since all keys are stored by a single entity (namely, the user).

Dodis, et al. [20] also show a generic conversion from any CPA-secure key-insulated scheme to a CPA-secure *strong* key-insulated scheme (where in a strong key-insulated scheme, encrypted messages are kept confidential even from the server itself). In their conversion, they split the plaintext message into two shares, encrypt one share using any "basic" key-insulated scheme, and encrypt the second share using a key that is stored (at all times) only by the user. Again, it can be seen that this solution corresponds to "double" encryption; thus, the techniques outlined in this paper suffice to construct generic CCA-secure *strong* key-insulated schemes from any CCA-secure key insulated scheme (thereby answering a question left open by [5]).

CCA-secure certificate-based encryption. The notion of certificate-based encryption (CBE) was recently introduced by Gentry [24]. In this model, a certificate — or, more generally, a signature — acts not only as a "certification" of the public key of a particular entity, but serves also as a decryption key. In particular, to decrypt a message a key-holder needs *both* its secret key and an up-to-date certificate from its certification authority (CA). Certificate-based encryption combines the aspects of identity-based encryption (IBE) and public-key encryption (PKE). Specifically, the sender of the message does not need to check whether the user is properly certified before sending the message, and the user can decrypt the message *only if* he has been certified (this is called *implicit certification*, a feature of IBE but not of PKE). Additionally, (1) the certificates from the CA can be sent to the user in the clear (as in PKE but unlike IBE), and (2) the CA cannot decrypt messages sent to to the user since he does not know the user's private key (i.e., there is no *escrow*, again like PKE but unlike IBE).

From the above description, one would expect that it should be possible to construct a CBE scheme using a simple combination of any IBE and regular PKE. In fact, this was the intuitive description of CBE as presented by Gentry [24], and this approach achieves security against chosen-plaintext attacks. Unfortunately, this does not suffice to achieve security against chosen-ciphertext attacks. As a result, [24] only constructed a CCA-secure CBE scheme based on specific assumptions, and left open the problem of designing a *generic* CCA-secure CBE scheme. Using the techniques from this paper with $n = 2$, but applying them to an IBE and a PKE (instead of two PKEs), we can easily resolve this open question. Note that ones only needs a wMCCA-secure multiple encryption scheme with no robustness in this case, since the user holds both keys and never reveals any intermediate results.

Our technique also applies to most of the CBE extensions presented by Gentry, such as hierarchical CBE (which combines CCA-secure hierarchical IBE and PKE) and the general technique (based on subset covers) to reduce CA computation in a multi-user environment.

CCA-secure broadcast encryption. A *broadcast encryption* scheme allows the sender to securely distribute data to a dynamically changing set of users over an insecure channel, with the possibility of "revoking" users when they are no longer "qualified". One of the most challenging settings for this problem is that of *stateless receivers*, where each user is given a fixed set of keys which cannot be updated for the lifetime of the system. This setting was considered by Naor, Naor, and Lotspiech [33], who also present a general "subset cover framework" for this problem. Although originally used in the symmetric-key setting, Dodis and Fazio [17] extended the subset cover framework to the public-key setting, where anybody can encrypt the data using a single public key of the system.

Without getting into technical details, each user (more or less) stores a certain, user-specific subset of secret keys, while all the public keys are freely available to everybody (specifically, are efficiently derived from a single "global public key"; in the case of [17] this is done by using an appropriate identity-based mechanism whose details are not important for the present discussion). When one wants to revoke a certain subset of users, one cleverly chooses a small subset P of public keys satisfying the following two properties: (1) every non-revoked user possesses at least one secret key corresponding to some key in P; but (2) every revoked user possesses no secret keys in P. Once this is done, a message is simply encrypted in parallel using every key in P.

Clearly, the above corresponds exactly to a multiple encryption scheme with $t_p = 0$ and $t_f = 1$. However, as acknowledged in [33, 17], the resulting broadcast encryption scheme is at best secure against "lunch-time" chosen-ciphertext attacks even if the underlying encryption scheme being used is CCA-secure. Using the techniques of this paper, we can resolve this problem and extend the subset-cover framework to achieve CCA-security (provided, of course, that the corresponding basic encryption schemes are CCA-secure). This results in the first generic CCA-secure broadcast encryption scheme. When instantiated with any of the two subset cover methods given in [33, 17], we obtain two "semi-generic" constructions of CCA-secure broadcast encryption: from any regular (e.g. [8]) or any hierarchical (e.g. [25]) identity-based encryption scheme, respectively. Each of these schemes, when properly instantiated, will offer several advantages over the only previously known CCA-secure broadcast encryption scheme [18] (which was based on specific assumptions), including a fixed public-key size, an unbounded number of revocations, and qualitatively stronger traitor-tracing capabilities.

We remark that although wMCCA-security is already enough for this application, a more communication-efficient solution can be achieved using our sMCCA-secure scheme (since each user can then simply "ignore" the majority of the ciphertext which is "not relevant" to him).

Cryptanalysis-tolerant CCA-secure encryption. As discussed in the Introduction, a multiple encryption scheme may be viewed as achieving "cryptanalysis-tolerance" for public-key encryption: namely, a message can be encrypted with respect to multiple encryption schemes (using independent keys) such that the message remains confidential as long as any *one* of these schemes remains secure (see [28] for further discussion of this concept). Herzberg [28] shows constructions of cryptanalysis-tolerant CPA-secure encryption schemes; the techniques outlined here resolve the question of constructing cryptanalysis-tolerant CCA-secure encryption schemes.

CCA-secure proxy encryption. Proxy encryption [19] may be viewed as non-interactive, two-party, threshold encryption, where one server is the end-user and the other server is called the *proxy*. The proxy receives the ciphertext C, partially decrypts it into some ciphertext C', and forwards C' to the end-user. The user stores the second part of the decryption key and can now recover the message M from C'. In [19], the authors give a formal treatment of proxy encryption but left open the question of constructing a generic, CCA-secure scheme. The generic 2-party multiple encryption scheme presented in this paper resolves this open question in the natural way. We remark that we require MCCA-security for this application, since the attacker (who is one of the servers) has full oracle access to the other server.

Other applications. We believe that multiple encryption schemes will find even more uses; we highlight two. One interesting direction is to apply multiple encryption to the construction of "anonymous channels" [14] using, e.g., "onion routing" [26]. It would be interesting to see if our methods can be extended to give CCA-secure constructions in this setting. For the second application, we mention recent work of Boneh, et al. [9] on searchable public-key encryption. Here, one wants to design an encryption scheme for which one can encrypt some keyword W as a ciphertext C such that that: (1) given some trapdoor T_W one can test whether C is an encryption of W; (2) without such trapdoor, one gets no information about W, even when given many other trapdoors T_X for $X \neq W$ (except that W is not one of these X's). It is not hard to see that this concept is also related to anonymous IBE, where the ciphertext should not reveal anything about the identity of the recipient of the message. Alternately, it is also related to key-insulated encryption in which the ciphertext does not reveal the time period for which the ciphertext was encrypted. In all these cases, one can adapt the generic construction of key-insulated encryption from [20], discussed earlier in this section, to obtain a CPA-secure version of the corresponding primitive, provided that the regular encryption $\mathcal{E}$ is *key-indistinguishable* [2]. Indeed, one of the constructions in [9] exactly follows this route. Using the techniques in this paper, we can obtain generic CCA-secure searchable encryption, recipient-anonymous IBE, or time-anonymous key-insulated encryption, provided one uses a CCA-secure, key-indistinguishable encryption scheme (such as the Cramer-Shoup encryption scheme [15], shown to be key-indistinguishable by [2]).

References

1. B. Aiello, M. Bellare, G. Di Crescenzo, and R. Venkatesan. Security Amplification by Composition: the Case of Doubly-Iterated, Ideal Ciphers. Crypto '98.
2. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-Privacy in Public-Key Encryption. Asiacrypt 2001.
3. M. Bellare, A. Desai, D. Pointcheval, and P. Rogaway. Relations among Notions of Security for Public-Key Encryption Schemes. Crypto '98.
4. M. Bellare and C. Namprempre. Authenticated Encryption: Relations Among Notions and Analysis of the Generic Composition Paradigm. Asiacrypt 2000.
5. M. Bellare and A. Palacio. Protecting against Key Exposure: Strongly Key-Insulated Encryption with Optimal Threshold. Available at http://eprint.iacr.org/2002/064.
6. M. Bellare and P. Rogaway. Collision-Resistant Hashing: Towards Making UOWHFs Practical. Crypto '97.
7. D. Boneh and X. Boyen. Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles. Eurocrypt 2004.
8. D. Boneh and M. Franklin. Identity-Based Encryption From the Weil Pairing. Crypto 2001.
9. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Searchable Public Key Encryption. Eurocrypt 2004.
10. D. Boneh and J. Katz. Improved Efficiency for CCA-Secure Cryptosystems Built Using Identity Based Encryption. RSA — Cryptographers' Track 2005, to appear.
11. R. Canetti, Y. Dodis, S. Halevi, E. Kushilevitz, and A. Sahai. Exposure-Resilient Functions and All-or-Nothing Transforms. Eurocrypt 2000.
12. R. Canetti, S. Halevi, and J. Katz. Chosen-Ciphertext Security from Identity-Based Encryption. Eurocrypt 2004.
13. R. Canetti and S. Goldwasser. An Efficient Threshold Public-Key Cryptosystem Secure Against Adaptive Chosen-Ciphertext Attack. Eurocrypt '99.
14. D. Chaum. Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. *Comm. ACM* 24(2): 84–88 (1981).
15. R. Cramer and V. Shoup. A Practical Public Key Cryptosystem Provably Secure Against Chosen Ciphertext Attack. Crypto '98.
16. Y. Desmedt. Society and Group-Oriented Cryptography: a New Concept. Crypto '87.
17. Y. Dodis and N. Fazio. Public Key Broadcast Encryption for Stateless Receivers. *ACM Workshop on Digital Rights Management*, 2002.
18. Y. Dodis and N. Fazio. Public Key Broadcast Encryption Secure Against Adaptive Chosen Ciphertext Attack. PKC 2003.
19. Y. Dodis and A. Ivan. Proxy Cryptography Revisited. NDSS 2003.
20. Y. Dodis, J. Katz, S. Xu, and M. Yung. Key-Insulated Public-Key Cryptosystems. Eurocrypt 2002.
21. S. Even and O. Goldreich. On the Power of Cascade Ciphers. *ACM Trans. Comp. Systems* 3: 108–116 (1985).
22. A. Fiat and M. Naor. Broadcast Encryption. Crypto '93.
23. M. Franklin and M. Yung. Communication Complexity of Secure Computation. STOC '92.
24. C. Gentry. Certificate-Based Encryption and the Certificate Revocation Problem. Eurocrypt 2003.
25. C. Gentry and A. Silverberg. Hierarchical Id-Based Cryptography. Asiacrypt 2002.

26. D. Goldschlag, M. Reed, and P. Syverson. Onion Routing. *Comm. ACM* 42(2): 39–41 (1999).
27. S. Goldwasser, S. Micali, and R. Rivest. A Digital Signature Scheme Secure Against Adaptive Chosen-Message Attacks. *SIAM J. Computing* 17(2): 281–308, 1988.
28. A. Herzberg. On Tolerant Cryptographic Constructions. Available at http://eprint.iacr.org/2002/135/.
29. H. Krawczyk. Secret Sharing Made Short. Crypto '93.
30. P. MacKenzie. An Efficient Two-Party Public Key Cryptosystem Secure Against Adaptive Chosen Ciphertext Attack. PKC 2003.
31. U. Maurer and J. Massey. Cascade Ciphers: the Importance of Being First. *J. Crypto* 6(1): 55–61 (1993).
32. R. Merkle and M. Hellman. On the Security of Multiple Encryption. *Comm. ACM* 24(7): 465–467 (1981).
33. D. Naor, M. Naor, and J. Lotspiech. Revocation and Tracing Schemes for Stateless Receivers. Crypto 2001.
34. NESSIE consortium. Portfolio of Recommended Cryptographic Primitives. Manuscript, Feb. 2003. Available at http://www.cosic.esat.kuleuven.ac.be/nessie/deliverables/decision-final.pdf.
35. M. Rabin. Efficient Dispersal of Information for Security, Load Balancing, and Fault Tolerance. *J. ACM* 36(2): 335–348 (1989).
36. R. Rivest. All-or-Nothing Encryption and the Package Transform. FSE '97.
37. A. Shamir. How to Share a Secret. *Comm. ACM* 22(11): 612–613 (1979).
38. C. Shannon. Communication Theory of Secrecy Systems. *Bell System Technical Journal*, vol. 28, Oct. 1949.
39. V. Shoup. A Proposal for an ISO Standard for Public-Key Encryption, version 2.1. Available at `http://eprint.iacr.org/2001/112/`
40. V. Shoup and R. Gennaro. Securing Threshold Cryptosystems Against Chosen Ciphertext Attack. *J. Crypto* 15(2): 75–96 (2002).
41. R. Zhang, G. Hanaoka, J. Shikata, and H. Imai. On the Security of Multiple Encryption, or CCA-security+CCA-security=CCA-security? Public Key Cryptography (PKC) 2004. Also available at http://eprint.iacr.org/2003/181.

A Replacing Signatures by MACs

Recall, a message authentication code (MAC) is given by a deterministic algorithm Tag which outputs an "existentially unforgeable" tag $T = \mathsf{Tag}_\tau(M)$ for a given message M using a secret key τ. In fact, a "one-time" message authentication code (defined analogously to a one-time signature scheme) is sufficient for our purposes. We define a relaxed commitment scheme $\mathcal{C} = (\mathsf{Setup}, \mathsf{Commit}, \mathsf{Open})$ (termed *encapsulation* in [10]) as follows: $\mathsf{Setup}(1^k)$ outputs the public commitment key CK, which is always input to both Commit and Open and is omitted for brevity. Commit takes no inputs and produces a triple of values (τ, c, d), where τ is a (random) key, c is the commitment to this key, and d is the corresponding decommitment. $\mathsf{Open}(c, d)$ should produce τ under normal circumstances. The hiding property states that τ "looks random" given c (i.e., one cannot efficiently distinguish (CK, c, τ) from (CK, c, r) for random r). The relaxed binding property states that given a random triple (τ, c, d) output by Commit, it is infeasible to produce $d' \neq d$ such that $\mathsf{Open}(c, d') \notin \{\tau, \perp\}$. It is easy to construct simple and efficient MACs and relaxed commitment schemes (see [10]).

Given the above, we construct $\mathcal{TE}_{wcca}$ as follows:

- $\underline{\mathsf{TGen}(1^k)}$. Let $\mathsf{CK} \leftarrow \mathsf{Setup}(1^k)$, and for $i = 1 \ldots n$, let $(\mathsf{EK}_i, \mathsf{DK}_i) \leftarrow \mathsf{Gen}(1^k)$. Set $\mathsf{TEK} = (\mathsf{EK}_1 \ldots \mathsf{EK}_n, \mathsf{CK})$, $\mathsf{TDK}_i = \mathsf{DK}_i$, so that $\mathbf{TDK} = (\mathsf{DK}_1 \ldots \mathsf{DK}_n)$. Below, denote $\mathsf{Enc}_i = \mathsf{Enc}_{\mathsf{EK}_i}$, $\mathsf{Dec}_i = \mathsf{Dec}_{\mathsf{DK}_i}$.
- $\underline{\mathsf{TEnc}^L(M)}$. Let $(\tau, c, d) \leftarrow \mathsf{Commit}(1^k)$ and $(s_1, \ldots, s_n, \mathsf{pub}) \leftarrow \mathsf{Share}(M, d)$. Set $C_i = \mathsf{Enc}_i^c(s_i)$ $(i = 1 \ldots n)$ and compute $\sigma = \mathsf{Tag}_\tau(C_1, \ldots, C_n, \mathsf{pub}, L)$. Output $C = (C_1, \ldots, C_n, \mathsf{pub}, c, \sigma)$.
- $\underline{\mathsf{Split}^L(C)}$. Parse $C = (C_1, \ldots, C_n, \mathsf{pub}, c, \sigma)$, and let ciphertext share $\hat{C}_i = (C_i, c)$, and $\mathsf{aux} = (\mathsf{pub}, c, L)$.
- $\underline{\mathsf{TDec}^i(C_i, c)}$. Output $s'_i = \mathsf{Dec}_i^c(C_i)$.
- $\underline{\mathsf{Combine}(s'_1, \ldots, s'_n, (\mathsf{pub}, c, L))}$. Let $(M, d) = \mathsf{Rec}(s'_1, \ldots s'_n, \mathsf{pub})$ (if invalid, reject). Let $\tau = \mathsf{Open}(c, d)$. Reject if $\sigma \neq \mathsf{Tag}_\tau(C_1, \ldots, C_n, \mathsf{pub}, L)$. Otherwise, output M.

Theorem 3. *$\mathcal{TE}_{wcca}$ is* wMCCA*-secure with thresholds t_p, t_f, t_r, t_s, provided $\mathcal{E}$ is* CCA*-secure, $\mathcal{SSS}$ is (t_p, t_f, t_r, t_s, n)-robust, $\mathcal{C}$ is a relaxed commitment scheme, and* MAC *is a one-time message authentication code.*

We give the complete proof in the full version, here only briefly sketching our argument (which is based on [10]). The problem is the apparent circularity in the usage of the MAC as Tag is applied to data which depends on the MAC key τ. Intuitively, what saves us here is the relaxed binding property which holds even when the adversary *knows* d. This means that when the attacker is given the challenge ciphertext C, it has to either (1) try to use new value c (which does not help due to the CCA-security of the underlying encryption scheme which uses c as a label); or (2) reuse the same c and cause an invalid $d' \neq d$ to be recovered (which leads to rejection anyway); or (3) reuse the same pair (c, d), which results in the same τ and then also to rejection due to the one-time security of the MAC. The latter argument is the most delicate, and its proof in fact requires several sub-arguments. See [10] for further details.