Concealment and its Applications to Authenticated Encryption

Yevgeniy Dodis[*] Jee Hea An[†]

March 13, 2003

Abstract

We introduce a new cryptographic primitive we call *concealment*, which is related, but quite different from the notion of commitment. A concealment is a publicly known randomized transformation, which, on input m, outputs a *hider* h and a *binder* b. Together, h and b allow one to recover m, but separately, (1) the hider h reveals "no information" about m, while (2) the binder b can be "meaningfully opened" by at most one hider h. While setting $b = m$, $h = \emptyset$ is a trivial concealment, the challenge is to make $|b| \ll |m|$, which we call a "non-trivial" concealment. We show that non-trivial concealments are equivalent to the existence of collision-resistant hash functions. Moreover, our construction of concealments is extremely simple, optimal, and yet very general, giving rise to a multitude of efficient implementations.

We show that concealments have natural and important applications in the area of *authenticated encryption*. Specifically, let $\mathcal{AE}$ be an authenticated encryption scheme (either public- or symmetric-key) designed to work on short messages. We show that concealments are *exactly* the right abstraction allowing one to use $\mathcal{AE}$ for encrypting long messages. Namely, to encrypt "long" m, one uses a concealment scheme to get h and b, and outputs authenticated ciphertext $\langle \mathcal{AE}(b), h\rangle$. More surprisingly, the above paradigm leads to a very simple and general solution to the problem of *remotely keyed (authenticated) encryption* (RKAE) [12, 13]. In this problem, one wishes to split the task of high-bandwidth authenticated encryption between a secure, but low-bandwidth/computationally limited device, and an insecure, but computationally powerful host. We give formal definitions for RKAE, which we believe are simpler and more natural than all the previous definitions. We then show that our composition paradigm satisfies our (very strong) definition. Namely, for authenticated encryption, the host simply sends a short value b to the device (which stores the actual secret key for $\mathcal{AE}$), gets back $\mathcal{AE}(b)$, and outputs $\langle \mathcal{AE}(b), h\rangle$ (authenticated decryption is similar). Finally, we also observe that the particular schemes of [13, 18] are all special examples of our general paradigm.

1 Introduction

AUTHENTICATED ENCRYPTION. The notions of privacy and authenticity are well understood in the cryptographic community. Interestingly, until very recently they have been viewed and analyzed as important but *distinct* building blocks of various cryptographic systems. When both were needed, the folklore wisdom was to "compose" the standard solutions for two. Recently, however, the area of *authenticated encryption* has received considerable attention. This was caused by many related reasons. First, a "composition" paradigm might not always work [7, 20, 2], at least if not used appropriately [2, 26]. Second, a tailored solution providing both privacy and authenticity might be noticeably more efficient (or have other advantages) than a straightforward composition [17, 27, 32, 2, 6]. Third, the proper modeling of authenticated encryption is not so obvious, especially in the public-key setting [2, 3]. Finally, viewing authenticated encryption as a

[*]Department of Computer Science, New York University, 251 Mercer Street, New York, NY 10012, USA. Email: dodis@cs.nyu.edu

[†]SoftMax Inc., San Diego, USA. Email: jeehea@cs.ucsd.edu.

separate *primitive* may conceptually simplify the design of complex protocols which require both privacy and authenticity.

OUR MAIN QUESTION. Despite the recent attention to authenticated encryption, the area is so new that many fundamental questions remain open. In this work, we study and completely resolve one such fundamental question, which has several important applications. Specifically, assume we have a secure authenticated encryption (either symmetric- or public-key) $\mathcal{AE}$ which works on "short" messages. How do we build a secure authenticated encryption $\mathcal{AE}'$ on "long" messages out of $\mathcal{AE}$? (Throughout, we should interpret "short" as having very small length, like 256 bits; "long" stands for fixed, but considerably larger length, possibly on the order of gigabytes.) While our question was not previously studied in the context of authenticated encryption, it clearly has rich history in the context of many other cryptographic primitives. We briefly review some of this work, since it will suggest the first solutions to our problem too.

First, in the context of regular chosen plaintext secure (CPA-secure) encryption, we can simply split the message into blocks and encrypt it "block-by-block". Of course, this solution multiplicatively increases the size of the ciphertext, so a lot of work has been developed into designing more efficient solutions. In the public-key setting, the classical "hybrid" encryption solution reduces the problem into that in the symmetric-key setting. Namely, one encrypts, using the public-key, a short randomly chosen symmetric key τ, and uses τ to symmetrically encrypt the actual message m. As for the symmetric-key setting, one typically uses one of many secure *modes of operations* on block ciphers (such as CBC; see [23]), which typically (and necessarily) add only one extra block of redundancy when encrypting a long message m. For authentication, a different flavor of techniques is usually used. Specifically, a common method is to utilize a *collision-resistant hash function* [14] H[1] which maps a long input m into a short output such that it is hard to find a "collision" $H(m_0) = H(m_1)$ for $m_0 \neq m_1$. Then one applies the given authentication mechanism for short strings to $H(m)$ to authenticate much longer m. This works, for example, for digital signatures (this is called "hash-then-sign"), message authentication codes (MACs), and pseudorandom functions (for the latter two, other methods are possible; see [5, 4, 11, 1] and the references therein).

FIRST SOLUTION ATTEMPT. One way to use this prior work is to examine generic constructions of authenticated encryption using some of the above primitives, and apply the above "compression" techniques to each basic primitive used. For example, in the symmetric-key setting we can take the "encrypt-then-mac" solution [7] for authenticated encryption, the CBC mode for encryption, the CBC-MAC [5] for message authentication, and build a specific authenticated encryption on long messages using only a fixed-length block cipher. Even better, in this setting we could utilize some special purpose, recently designed modes of operation for *authenticated* encryption, such as IACBC [17] or OCB [27]. Similar techniques could be applied in the public-key setting using the "hybrid" technique for encryption, "hash-then-sign" for signatures, and any of the three generic signature/encryption compositions presented by [2].

In other words, prior work already gives us some tools to build "long" authenticated encryption, without first reducing it to "short" authenticated encryption.

WHY SOLVING OUR PROBLEM THEN? The first reason is in its theoretical value. It is a very interesting structural question to design an elegant amplification from "short" to "long" authenticated encryption, without building the "long" primitive from scratch. For example, in the public-key setting especially, it is curious to see what is the common generalization of such differently looking methods as "hybrid" encryption and "hash-then-sign" authentication. Indeed, we shall see that this generalization yields a very elegant new primitive, certainly worth studying on its own. The second reason is that it gives one more *option* to designing "long-message" authenticated encryption. Namely, instead of solving the problem by using *other* "long-message" primitives, and implementing these separately, we directly reduce it to the *same*, but "short-message" primitive, and implement it separately. And this may bring other advantages (e.g. efficiency,

[1]Or, when possible, a weaker class of hash functions, such as various types of universal hash functions.

ease of implementation, etc.), depending on its application and implementation. Consider, for example, the public-key setting, where authenticated encryption is usually called *signcryption* [32]. With any of the generic signature-encryption compositions [2], signcryption of a long messages will eventually reduce to a regular signature plus a regular encryption on some short messages. With our paradigm, it will reduce to a single signcryption on a short message, which can potentially be faster than doing a separate signature and encryption. Indeed, this potential efficiency gain was the main motivation of Zheng [32] to introduce signcryption in the first place! Finally, our technique has important applications on its own. In particular, we show that it naturally leads to a very general, yet simple solution to the problem of *remotely keyed authenticated encryption* [12, 21, 13] (RKAE), discussed a bit later. None of the other techniques we mentioned seem to yield the solution to this problem.

OUR MAIN CONSTRUCTION AND A NEW PRIMITIVE. In our solution method, we seek to amplify a given "short" authenticated encryption $\mathcal{AE}$ into a "long" $\mathcal{AE}'$ as follows. First, we somehow split the long message m into two parts $(h, b) \leftarrow T(m)$, where $|b| \ll |m|$, and then define $\mathcal{AE}'(m) = \langle \mathcal{AE}(b), h \rangle$. Which transformations T suffice in order to make $\mathcal{AE}'$ a "secure" authenticated encryption if $\mathcal{AE}$ is such? We completely characterize such transformations T, which we call *concealments*. Specifically, we show that $\mathcal{AE}'$ is secure if and only if T is a (relaxed) concealment scheme.

Our new notion of concealments is remarkably simple and natural, and defines a new cryptographic primitive of independent interest. Intuitively, a concealment T has to be invertible, and also satisfy the following properties: (1) the *hider* h reveals no information about m; and (2) the *binder* b "commits" one to m in a sense that it is hard to find a valid (h', b) where $h' \neq h$. Property (2) has two formalizations leading to the notions of regular and relaxed concealment schemes. Relaxed concealments suffice for the composition purposes above, but we will need (strong) regular concealments for the problem of RKAE, briefly mentioned earlier and discussed shortly. We remark that concealments look very similar to *commitment schemes* at first glance, but there are few crucial differences, making these notions quite distinct. This comparison will be discussed in Section 2.

Finally, we are left with the question of constructing concealment schemes. First, we show that *non-trivial* (i.e., $|b| < |m|$) concealment schemes are equivalent to the existence of collision-resistant hash functions (CRHFs). In particular, our construction from CRHFs is very simple, efficient and general, giving rise to many optimal implementations. Specifically, $|h| \approx |m|$, while $|b|$ is only proportional to the security parameter. In fact, one special case of our construction looks very similar to the famous *Optimal Asymmetric Encryption Padding* (OAEP) [8]. Our construction replaces two random oracles G and H used in this variant of OAEP by a pseudorandom generator and a collision-resistant hash function, respectively. Thus, having a well established goal in mind, we essentially found an application of (slightly modified) OAEP, where we can provably eliminate random oracles in the analysis. More from a theoretical point of view, we also give a useful, but slightly less efficient construction of *relaxed* concealments from a somewhat weaker notion of *universal one-way hash functions* (UOWHF) [25]. In principle, this shows that relaxed concealments can be constructed even from regular one-way functions [28], thus separating them from regular concealments by the result of Simon [31].

To summarize, we show that concealments are very natural cryptographic gadgets, and can be efficiently built from standard assumptions. In particular, they give an efficient way to implement "long" authenticated encryption from a "short" one. Finally, we describe a powerful application of concealments and our amplification technique to the problem of RKAE, which deserves a separate introduction.

REMOTELY KEYED AUTHENTICATED ENCRYPTION: HISTORY. The problem of "remotely keyed encryption" (RKE) was first introduced by Blaze [12] in the symmetric-key setting. Intuitively, RKE is concerned with the problem of "high-bandwidth encryption with low bandwidth smartcards". Essentially, one would like to store the secret key in a secure, but computationally bounded and low bandwidth Card, while to have an insecure, but powerful Host perform most of the operations for encryption/decryption. Of course, the communication between the Host and the Card should be minimal as well. The original work of Blaze lacked formal modeling of the problem, but inspired a lot of subsequent research. The first formal modeling of RKE was done by Lucks [21], who chose to interpret the question as that of implementing a remotely key *pseudorandom permutation* (or block cipher), which we will call RKPRP. Lucks' paper was further improved —both in terms of formal modeling and constructions— by an influential work of Blaze, Feigenbaum and Naor [13]. For one thing, they observed that the PRP's length-preserving property implies that it *cannot* be semantically secure when viewed as encryption. Thus, in addition to RKPRP, which they called a "length-preserving RKE", they introduced the notion of a "length-increasing RKE", which is essentially meant to be the notion of remotely keyed *authenticated* encryption, so we will call it RKAE. In other words, the informal notion of "RKE" was really formalized into two very distinct notions of RKPRP and RKAE, none of which is really a plain encryption. Blaze et al. [13] gave formal definitions and constructions of RKAE and RKPRP, and the latter's construction was subsequently improved by [22].

While the RKAE definition of [13] was an important and the first step towards properly formalizing this new notion (as opposed to the notion of RKPRPs), their definition is convoluted and quite non-standard (it involves an "arbiter" who can fool any adversary). For example, it looks nothing like the formal, universally accepted notion of regular (not remotely keyed) authenticated encryption [19, 10, 7]. Of course, this has a very objective reason in that the above formal definition appeared *after* the work of [13]. Additionally, at the time Blaze et al. perhaps tried to make their definition of "length-increasing RKE" look as close as possible to their definition of "length-preserving RKE" (i.e., RKPRP) also studied in that paper, since the latter was the previously considered notion. Still, we believe that the definition of RKAE should be based on the definition of regular authenticated encryption, rather than try mimicking the definition of a somewhat related, but different concept. Thus, we will give what we feel is a simpler and more natural such definition, which looks very close to the definition of regular authenticated encryption. Additionally, we naturally extend the whole concept of RKAE to the *public-key* setting, since it is equally applicable in this case too.[2] Notice, in this setting the notion of RKPRP makes no sense, which additionally justifies our choice to base our definition on that of regular authenticated encryption.

Another closely related work is that of Jakobsson et al. [18], who also effectively studied the problem of RKAE (even though still calling it RKE despite considering authentication as part of the requirement). We note that the definition of [18] looks much closer to our new formalization. However, there are still significant differences that make our notion stronger.[3] For example, [18] do not support chosen ciphertext attack in its full generality (i.e., no Card access is given to the adversary after the challenge is received), and also require the adversary to "know" the messages corresponding to forged ciphertexts. Finally, we mention that their main scheme uses an "OAEP"-like transform, and their security analyses critically use random oracles. As we show, using another (in fact, simpler!) variant of OAEP for RKAE, we can eliminate random oracles from the analysis. Thus, a special case of our construction gives an equally simple and efficient scheme, which is provably secure in the standard model.

Finally, we mention the recent work Joux et al. [16]. Form our perspective, it showed that naive "remotely-keyed" implementation of many natural block cipher modes of operations for (authenticated) encryption, such as CBC or IACBC, are completely insecure from the perspective of RKE/RKAE. In such naive implementations, the Card stores the key to the block cipher, while the Host does everything by itself except when it needs to evaluate the block cipher (or its inverse), it which case it calls the Card. We notice that this means that to perform a single (authenticated) encryption/decryption, the Host needs to adaptively access the Card for a number of times proportional to the length of the (long) message. Perhaps not surpris-

[2]In this abstract, though, we will restrict ourselves to the symmetric-key setting.

[3]Except both [18] and [13] insist on achieving some kind of pseudorandomness of the output. Even though our constructions achieve it as well, we feel this requirement is not crucial for any application of RKAE, and was mainly put to make the definition look similar to RKPRPs.

ingly, this gives too much power to the "blockwise-adaptive" adversary, allowing him to easily break the security of such naive RKE/RKAE implementations. In contrast, in our RKAE solutions the Host accesses the Card once and on a very short input, irrespective of the length of the message it actually processes. In fact, in one of our solutions (see "extensions" paragraph below), all the Card does is a single block cipher call per invocation!

As a corollary, the work of [16] strongly supports our prior claim that direct "long" authenticated encryption schemes, such as IACBC [17], do not seem to be naturally suited for RKAE.

OUR CONTRIBUTION TO RKAE. As we mentioned, we give a simple and natural definition of RKAE, which we feel improves upon the previous definitions. In addition, we show that our construction of "long-message" authenticated encryption from that of "short-message" authenticated encryption provides a very natural, general, and provably secure solution to the problem of RKAE. Recall, we had $\mathcal{AE}'(m) = \langle \mathcal{AE}(b), h \rangle$, where (h, b) was output by some transformation T, and $|b| \ll |m|$. This immediately suggests the following protocol for RKAE. The Host computes (h, b) and sends short b to the Card, which stores the secret key. The Card computes short $c = \mathcal{AE}(b)$ and sends it to the Host, which outputs $\langle c, h \rangle$. Authenticated decryption is similar. Again, we ask the question which transformations T will suffice to make this simple scheme secure. Not surprisingly, we get that concealment schemes are necessary and sufficient, even though in this case we do need regular ("non-relaxed") concealments. We believe that our result gives a general and intuitively simple solution to the problem. Also, it generalizes the previous, so "differently looking" solutions of [13, 18], both of which can be shown to use some particular concealment and/or "short" authenticated encryption.

EXTENSIONS. All our techniques naturally support authenticated encryption *with associated data* [26], which we explain in the sequel. In fact, this distinction makes our composition paradigm even slightly more efficient. Also, we remark again that all our results apply to both the public- and the symmetric-key authenticated encryption. The only exception is the following extension that makes sense only in the symmetric-key setting. We study the question of whether we can replace our "short" authenticated encryption $\mathcal{AE}$ by a (strong) pseudorandom permutation (i.e., a block cipher, since $\mathcal{AE}$ is applied on short inputs), which would enhance the practical usability of our composition even more. We show that while arbitrary concealments are generally not enough to ensure the security of thus constructed $\mathcal{AE}'$, some mild extra restrictions —enjoyed by our main concealment constructions— make them sufficient for this purpose.

2 Definition of Concealment

Intuitively, a concealment scheme efficiently transforms a message m into a pair (h, b) such that: (1) (h, b) together reveal m; (2) the *hider* h reveals no information about m; and (3) the *binder* b "commits" one to m in a sense that it is hard to find a valid (h', b) where $h' \neq h$. Below is a formal description.

SYNTAX. A concealment scheme consists of three efficient algorithms: $\mathcal{C} = (\mathsf{Setup}, \mathsf{Conceal}, \mathsf{Open})$. The setup algorithm $\mathsf{Setup}(1^k)$, where k is the security parameter, outputs a public concealment key CK (possibly empty, but often consisting of public parameters for $\mathcal{C}$). Given a message m from the corresponding message space $\mathcal{M}$ (e.g., $\{0,1\}^k$), the randomized concealment algorithm $\mathsf{Conceal}_{\mathsf{CK}}(m; r)$ (where r is the randomness) outputs a concealment pair (h, b), where h is the *hider* of m and b is the *binder* to m. For brevity, we will usually omit CK and/or r, writing $(h, b) \leftarrow \mathsf{Conceal}(m)$. Sometimes we will write $h(m)$ (resp. $b(m)$) to denote the hider (resp. binder) part of a randomly generated (h, b). The deterministic open algorithm $\mathsf{Open}_{\mathsf{CK}}(h, b)$ outputs m if (h, b) is a "valid" pair for m (i.e. could have been generated by $\mathsf{Conceal}(m)$), or $\perp$ otherwise. Again, we will usually write $x \leftarrow \mathsf{Open}(h, b)$, where $x \in \{m, \perp\}$. The *correctness* property of concealment schemes says that $\mathsf{Open}_{\mathsf{CK}}(\mathsf{Conceal}_{\mathsf{CK}}(m)) = m$, for any m and CK.

SECURITY OF CONCEALMENT. Just like commitment schemes, concealment schemes have two security properties called *hiding* and *binding*. However, unlike commitment schemes, these properties apply to different parts of concealment, which makes a significant difference.

- **Hiding**. Having the knowledge of CK, it is computationally hard for the adversary $\mathcal{A}$ to come up with two messages $m_1, m_2 \in \mathcal{M}$ such that $\mathcal{A}$ can distinguish $h(m_1)$ from $h(m_2)$. That is, $h(m)$ reveals no information about m. Formally, for any PPT (probabilistic polynomial time) adversary $\mathcal{A}$, which runs in two stages find and guess, we require that the probability below is at most $\frac{1}{2} + \mathsf{negl}(k)$ (where $\mathsf{negl}(k)$ denotes some negligible function):

$$\Pr\left[\, \sigma = \tilde{\sigma} \;\middle|\; \begin{array}{c} \mathsf{CK} \leftarrow \mathsf{Setup}(1^k),\ (m_0, m_1, \alpha) \leftarrow \mathcal{A}(\mathsf{CK}, \mathsf{find}),\ \sigma \leftarrow_r \{0,1\}, \\ (h, b) \leftarrow \mathsf{Conceal}_{\mathsf{CK}}(m_\sigma),\ \tilde{\sigma} \leftarrow \mathcal{A}(h;\ \alpha, \mathsf{guess}) \end{array} \right]$$

  where α is some state information. We will also denote this by $h(m_0) \approx h(m_1)$.

- **Binding**. Having the knowledge of CK, it is computationally hard for the adversary $\mathcal{A}$ to come up with b, h, h', where $h \neq h'$ such that (b, h) and (b, h') are both valid concealment pairs (i.e., $\mathsf{Open}_{\mathsf{CK}}(h, b) \neq \bot$ and $\mathsf{Open}_{\mathsf{CK}}(h', b) \neq \bot$). That is, $\mathcal{A}$ cannot find a binder b which it can open with two different hiders.[4]

We immediately remark that setting $b = m$ and $h = \emptyset$ satisfies the definition above. Indeed, the challenge is to construct concealment schemes with $|b| \ll |m|$ (we call such schemes *non-trivial*). Since $|b| + |h| \geq |m|$, achieving a very good concealment scheme implies that $|h| \approx |m|$.

RELAXED CONCEALMENTS. We will also consider *relaxed* concealment schemes, where the (strict) binding property above is replaced by the **Relaxed Binding** property, which states that $\mathcal{A}$ cannot find binder collisions for a *randomly generated* binder $b(m)$, even if $\mathcal{A}$ can choose m. Formally, for any PPT $\mathcal{A}$, which runs in two stages find and $\mathsf{collide}$, the following probability is at most $\mathsf{negl}(k)$:

$$\Pr\left[\begin{array}{c} h \neq h' \wedge \\ m' \neq \bot \end{array} \;\middle|\; \begin{array}{c} \mathsf{CK} \leftarrow \mathsf{Setup}(1^k),\ (m, \alpha) \leftarrow \mathcal{A}(\mathsf{CK}, \mathsf{find}),\ (h, b) \leftarrow \mathsf{Conceal}_{\mathsf{CK}}(m), \\ h' \leftarrow \mathcal{A}(h, b;\ \alpha, \mathsf{collide}),\ m' \leftarrow \mathsf{Open}_{\mathsf{CK}}(h', b) \end{array} \right]$$

To justify this distinction, we will see later that non-trivial (strong) concealments will be equivalent to collision-resistant hash functions (CRHFs), while relaxed concealments can be built from universal one-way hash functions (UOWHFs). By the result of Simon [31], UOWHFs are strictly weaker primitives than CRHFs (in particular, they can be built from regular one-way functions [25]), which implies that relaxed concealments form a weaker cryptographic assumption than regular concealments.

COMPARISON TO COMMITMENT. At first glance, concealment schemes look extremely similar to commitment schemes. Recall, commitments also transform m into a pair (c, d), where c is the "commitment", and d is the "decommitment". However, in this setting the commitment c is *both* the hider and the binder, while in our setting b is a binder and h is a hider. This seemingly minor distinction turns out to make a very big difference. For example, irrespective of parameter settings, commitment always implies one-way functions, while there are trivial concealments when $|b| = |m|$. On the other hand, when $|b| < |m|$, we will show that concealments immediately require CRHFs, while quite non-trivial commitments can be built from one-way functions [24]. Not surprisingly, the two primitives have very different applications and constructions. In particular, commitments are not useful for our applications to authenticated encryption (even though they are useful for others; see [2]).

[4]We could have allowed $\mathcal{A}$ to find $h \neq h'$ as long as (h, b), (h', b) do not open to distinct messages $m \neq m'$. However, we will find the stronger notion more convenient.

3 Constructing Concealment Schemes

In this section, we give very simple and general constructions of strong (resp. relaxed) concealment schemes based on any family of CRHFs (resp. UOWHFs) and any symmetric one-time encryption scheme. Recall, both CRHFs and UOWHFs are defined by some family $\mathcal{H} = \{H\}$ for which it is hard to find a colliding pair $x \neq x'$ such that $H(x) = H(x')$, where H is a (compressing) function randomly chosen from $\mathcal{H}$. However, with CRHFs, we first select the function H, and for UOWHFs the adversary has to select x before H is given to it. We first observe the following simple lemma, which shows the necessity of using CRHFs (resp. UOWHFs) in our constructions.

Lemma 1 *Let $\mathcal{C} = (\mathsf{Setup}, \mathsf{Conceal}, \mathsf{Open})$ be a strong (resp. relaxed) concealment scheme where the binder b is shorter than the message m. Define a shrinking function family $\mathcal{H}$ by the following generation procedure: pick a random r, run $\mathsf{CK} \leftarrow \mathsf{Setup}(1^k)$, and output $\langle \mathsf{CK}, r\rangle$ as a description of a random function $H \in \mathcal{H}$. To evaluate such H on input m, run $(h, b) = \mathsf{Conceal}_{\mathsf{CK}}(m;\ r)$, and set $H(m) = b$ (so that $|H(m)| < |m|$). Then $\mathcal{H}$ is a family of CRHFs (resp. UOWHFs).*

Proof: If $\mathcal{C}$ is a strong concealment, finding $m_0 \neq m_1$ such that $H(m_0) = H(m_1) = b$ implies finding $h_0 = h(m_0;\ r)$, $h_1 = h(m_1;\ r)$ such that $\mathsf{Open}_{\mathsf{CK}}(h_0, b) = m_0 \neq \bot$, $\mathsf{Open}_{\mathsf{CK}}(h_1, b) = m_1 \neq \bot$ and $h_0 \neq h_1$ since $m_0 \neq m_1$. This clearly contradicts the binding property of concealment. Similarly, if one has to choose m_0 beforehand, choosing random $H \in \mathcal{H}$ involves choosing a random r. Thus, when evaluating $H(m_0)$, we effectively computed a *random* concealment $(h_0, b) \leftarrow \mathsf{Conceal}_{\mathsf{CK}}(m_0)$ and gave it to the adversary, as required by the definition of relaxed concealment. The rest of the proof is the same as for strong concealments. $\square$

In the following, we show the converse of the above observation. Even though it is quite simple, we will crystallize it even further by splitting it into several clean steps.

ACHIEVING HIDING. We first show how to achieve the hiding property so that $|b| \ll |m|$. Later we will utilize CRHFs/UOWHFs to add strong/relaxed binding property to any scheme which already enjoys hiding.

Recall that a symmetric encryption scheme $\mathcal{SE} = (\mathsf{K}, \mathsf{E}, \mathsf{D})$ consists of the key generation algorithm K, encryption algorithm E, and decryption algorithm D. Of course, if $\tau \leftarrow \mathsf{K}(1^k)$, we require that $\mathsf{D}_\tau(\mathsf{E}_\tau(m)) = m$. For our purposes we will need the most trivial and minimalistic notion of *one-time security*. Namely, for any m_0, m_1 we require $\mathsf{E}_\tau(m_0) \approx \mathsf{E}_\tau(m_1)$, where $\tau \leftarrow \mathsf{K}(1^k)$ and $\approx$ denotes computational indistinguishability. More formally, for any m_0, m_1 and any PPT A, we require

$$\Pr\left[\ \sigma = \tilde{\sigma}\ \middle|\ \tau \leftarrow \mathsf{K}(1^k),\ \sigma \leftarrow_r \{0,1\},\ c \leftarrow \mathsf{E}_\tau(m_b),\ \tilde{\sigma} \leftarrow \mathcal{A}(c)\ \right] \leq \frac{1}{2} + \mathsf{negl}(k)$$

Of course, regular one-time pad satisfies this notion. However, for our purposes we will want the secret key to be much shorter than the message: $|\tau| \ll |m|$. For the most trivial such scheme, we can utilize any pseudorandom generator (PRG) $G : \{0,1\}^k \to \{0,1\}^n$ where $k \ll n$. The secret key is a random $\tau \in \{0,1\}^k$, and to encrypt $m \in \{0,1\}^n$ we compute $\mathsf{E}_\tau(m) = G(\tau) \oplus m$ (to decrypt, compute $\mathsf{D}_\tau(c) = G(\tau) \oplus c$). Of course, any stronger encryption (possibly probabilistic, such as any chosen plaintext secure encryption) will suffice for our purposes too.

Now, let $b = \tau$ and $h \leftarrow \mathsf{E}_\tau(m)$, so that $\mathsf{Open}(b, h) = \mathsf{D}_b(h)$. It is easy to see that this scheme satisfies the hiding (but not yet the binding) property of concealment, and also that $|b| \ll |m|$ if a good one-time secure encryption is used, such as the PRG-based scheme above.

ADDING STRONG BINDING. Assume $\mathcal{C} = (\mathsf{Setup}, \mathsf{Conceal}, \mathsf{Open})$ already achieves hiding, and let $\mathcal{H} = \{H\}$ be a family of CRHFs (the lengths of inputs and outputs needed will be clear soon). We turn $\mathcal{C}$ into $\mathcal{C}' = (\mathsf{Setup}', \mathsf{Conceal}', \mathsf{Open}')$ which is a full fledged concealment scheme:

- $\mathsf{Setup}'(1^k)$: run $\mathsf{CK} \leftarrow \mathsf{Setup}(1^k)$, $H \leftarrow \mathcal{H}$ and output $\mathsf{CK}' = \langle \mathsf{CK}, H \rangle$.
- $\mathsf{Conceal}'(m)$: let $(h, b) \leftarrow \mathsf{Conceal}(m)$, $h' = h$, $b' = b \| H(h)$, and output $\langle h', b' \rangle$.
- $\mathsf{Open}'(h', b')$: parse $b' = b \| t$, $h' = h$ and output $\perp$ if $H(h) \neq t$; otherwise, output $m = \mathsf{Open}(h, b)$.

We remark that $\mathcal{H}$ should have input size equal to the hider size $|h|$. Recall that in our schemes we will always have $|h| \approx |m|$ (in fact, exactly equal in the PRG-based scheme). And the output size should be small (say, $O(k)$, where k is the security parameter), as it directly contributes to the binder length which we aim to minimize.

Lemma 2 *If $\mathcal{C}$ satisfies the hiding property and $\mathcal{H}$ is a* CRHF*, then $\mathcal{C}'$ is a (strong) concealment scheme.*

Proof: Since $h' = h$, we get hiding for free. As for binding, if some $\mathcal{A}$ outputs $b' = b \| t, h_0, h_1$ causing "collision", then $H(h_0) = H(h_1) = t$, which contradicts the collision resistance of $\mathcal{H}$. □

ADDING RELAXED BINDING. Assume $\mathcal{C} = (\mathsf{Setup}, \mathsf{Conceal}, \mathsf{Open})$ already achieves hiding, and let $\mathcal{H} = \{H\}$ be a family of UOWHFs (the lengths of inputs and outputs needed will be clear soon). We turn $\mathcal{C}$ into $\mathcal{C}'' = (\mathsf{Setup}'', \mathsf{Conceal}'', \mathsf{Open}'')$ which is a full fledged *relaxed* concealment scheme:

- $\mathsf{Setup}'' = \mathsf{Setup}$.
- $\mathsf{Conceal}''(m)$: pick $H \leftarrow \mathcal{H}$, compute $(h, b) \leftarrow \mathsf{Conceal}(m)$, set $h'' = h$, $b'' = b \| H(h) \| H$, and output $\langle h'', b'' \rangle$.
- $\mathsf{Open}''(h'', b'')$: parse $b'' = b \| t \| H$, $h'' = h$ and output $\perp$ if $H(h) \neq t$; otherwise, output $m = \mathsf{Open}(h, b)$.

We see that the construction is similar to the CRHF-based construction, except we pick a new hash function *per each call*, and append it to the binder b''. This ensures that H is always selected independently of the input h it is applied to, as required by the definition of UOWHFs. Unfortunately, it also means that the construction is less attractive than the previous, more economical CRHF-based construction. Thus, the value of this construction is mainly theoretical, since it shows that efficient *relaxed* concealments, unlike strong concealments, can be built from regular one-way functions. In practice, one should certainly use the more economical CRHF-based construction.

Lemma 3 *If $\mathcal{C}$ satisfies the hiding property and $\mathcal{H}$ is a* UOWHF*, then $\mathcal{C}''$ is a relaxed concealment scheme.*

Proof: Since $h'' = h$, we get hiding for free. As for binding, if some $\mathcal{A}$ chooses m_0, gets back $b'' = b \| t \| H$ and h_0 and then successfully outputs $h_1 \neq h_0$ such that $H(h_0) = H(h_1) = t$, this $\mathcal{A}$ immediately breaks the relaxed collision resistance of $\mathcal{H}$. □

As earlier, $\mathcal{H}$ should have input size equal to the hider size $|h|$, which is roughly $|m|$. Also, the output size should be small (say, $O(k)$, where k is the security parameter), as it directly contributes to the binder length which we aim to minimize. Now, however, we also need the description of a UOWHF H to be small, as it is also part of the binder. Unfortunately, the best known constructions of UOWHFs for long messages [9, 29] have $|H| \approx O(k \log |m|)$, where k is the security parameter and $|m| \approx |h|$ is the length of the input to H. While the logarithmic dependence on the message length is not bad in theory — in particular, we still get $|b''| \ll |m|$ — this is a big drawback as compared to the previous CRHF-based construction, which achieved $|b'| = O(k)$ in addition to its stronger binding property.

COLLECTING PIECES TOGETHER. Unifying the previous discussion, and noticing that the existence of CRHFs or UOWHFs implies the existence of a one-time secure symmetric encryption [25], we get:

Theorem 1 *Non-trivial strong (resp. relaxed) concealment schemes exist iff* CRHF*s (resp.* UOWHF*s) exist.*

In terms of a particular simple and efficient construction, we get $h \leftarrow \mathsf{E}_\tau(m)$, $b = \tau \| H(h)$, where H is a CRHF, and E is any one-time symmetric encryption. Specifically, if we set $\mathsf{E}_\tau(m) = G(\tau) \oplus m$, where G is a PRG, we get a construction which looks amazingly similar to the famous *Optimal Asymmetric Encryption Padding* (OAEP) [8],[5] but we do not need to assume G and H as random oracles in the analyses.

4 Applications to Authenticated Encryption

We now study applications of concealment to *authenticated encryption*. Recall, the latter provides means for private, authenticated communication between the sender and the receiver. Namely, an eavesdropper cannot understand anything from the transmission, while the receiver is sure that any successful transmission indeed originated from the sender, and has not been "tampered with". The intuitive idea of using concealments for authenticated encryption is simple. If $\mathcal{AE}$ is an authenticated encryption working on short $|b|$-bit messages, and $(h, b) \leftarrow \mathsf{Conceal}(m)$, we can define $\mathcal{AE}'(m) = \langle \mathcal{AE}(b), h \rangle$. Intuitively, sending the hider h "in the clear" preserves privacy due to the hiding property, while authenticated encryption of the binder b provides authenticity due to the binding property.

We formalize this intuition by presenting two applications of the above paradigm. First, we argue that it indeed yields a secure authenticated encryption on long messages from that on short messages. And this holds even if relaxed concealments are used (in fact, they are necessary and sufficient). Second, we show that this paradigm also gives a very simple and general solution to *remotely keyed* authenticated encryption. Here, the full power of (strong) concealments is needed.

We remark that our applications hold for both the symmetric- and the public-key notions of authenticated encryption (the latter is historically called *signcryption* [32]). In terms of usability, the long message authenticated encryption is probably much more useful in the public-key setting, since signcryption is typically expensive. However, even in the symmetric-key setting our approach is very fast, and should favorably compare with alternative direct solutions such as "encrypt-then-mac" [7]. For remotely keyed setting, both public- and symmetric-key models seem equally useful and important. In fact, symmetric-key is perhaps more relevant, since smartcards are currently much better suited for symmetric-key operations. Indeed, prior work on "remotely keyed encryption" focused on the symmetric setting only.

4.1 Definition of Authenticated Encryption

We remark that formal modeling of authenticated encryption in the public-key setting is somewhat more involved than that in the symmetric-key setting due to issues such as multi-user security and "identity fraud" (see [2]). However, the proofs we present are really identical despite these extra complications of the public-key setting. Intuitively, the point is that we are constructing the *same* primitive on longer messages as the primitive we are given on shorter messages. Thus, whatever (complicated) security properties were present, will remain to be present in our composition scheme. For conciseness, we chose to concentrate on a simpler symmetric setting for the remainder of this abstract. We stress, however, that this is done *for simplicity only*, our proofs translate to the public-key setting completely and trivially, except that the syntax is slightly more complex and the following minor technicality is observed. The definition of authenticated encryption naturally has two components: privacy and authenticity. In the symmetric setting only, it turns out that the authenticity notion together with a rather weak privacy notion of "chosen plaintext" security imply a stronger

[5]Except OAEP sets $b = \tau \oplus H(h)$. This lack of "redundancy" makes it fail to yield a concealment scheme. Indeed, OAEP decoding never outputs $\perp$, since it is a permutation over m and τ; thus, does not achieve any binding.

(and desired) privacy notion of “chosen ciphertext” security. Thus, in this setting it is customary to define privacy in terms of only “chosen plaintext” attack. Since the above implication is false in the public-key setting (see [2]), and we want to present a single proof template for both settings, we will define privacy using a seemingly redundant notion of “chosen ciphertext” security even in the symmetric-key setting.

SYNTAX. An authenticated encryption scheme consists of three algorithms: $\mathcal{AE} = (\mathsf{KG}, \mathsf{AE}, \mathsf{AD})$. The randomized key generation algorithm $\mathsf{KG}(1^k)$, where k is the security parameter, outputs a shared secret key K, and possibly a public parameter pub. Of course, pub can always be part of the secret key, but this might unnecessarily increase the secret storage. In the description below, all the algorithms (including the adversary's) can have access to pub, but we omit this dependence for brevity. The randomized *authencryption* (authenticate/encrypt) algorithm AE takes as input the key K and a message m from the associated message space $\mathcal{M}$, and internally flips some coins and outputs a ciphertext c; we write $c \leftarrow \mathsf{AE}_K(m)$ or $c \leftarrow \mathsf{AE}(m)$, omitting the key K for brevity. The deterministic *authdecryption* (verify/decrypt) algorithm AD takes as input the key K, and outputs $m \in \mathcal{M} \cup \{\perp\}$, where $\perp$ indicates that the input ciphertext c is ”invalid”. We write $m \leftarrow \mathsf{AD}_K(c)$ or $m \leftarrow \mathsf{AD}(c)$ (again, omitting the key). We require that $\mathsf{AD}(\mathsf{AE}(m)) = m$, for any $m \in \mathcal{M}$.

SECURITY OF AUTHENTICATED ENCRYPTION. Fix the sender S and the receiver R. Following the standard security notions [7], we define the attack models and goals of the adversary for both authenticity (i.e. sUF-CMA)[6] and privacy (IND-CCA2)[7] as follows. We first model our adversary $\mathcal{A}$. $\mathcal{A}$ has oracle access to the functionalities of both S and R. Specifically, it can mount a chosen message attack on S by asking S to produce a ciphertext C of an arbitrary message m, i.e. $\mathcal{A}$ has access to the *authencryption oracle* $\mathsf{AE}_K(\cdot)$. Similarly, it can mount a chosen ciphertext attack on R by giving R any candidate ciphertext C and receiving back the message m (where m could be $\perp$), i.e. $\mathcal{A}$ has access to the *authdecryption oracle* $\mathsf{AD}_K(\cdot)$.

To break the sUF-CMA security of the authenticated encryption scheme, $\mathcal{A}$ has to be able to produce a “valid” ciphertext C (i.e., $\mathsf{AD}_K(C) \neq \perp$), which was not returned earlier by the authencryption oracle.[8] Notice, $\mathcal{A}$ is not required to “know” $m = \mathsf{AD}_K(C)$ when producing C. The scheme is sUF-CMA-secure if for any PPT $\mathcal{A}$, $\Pr[\mathcal{A} \text{ succeeds}] \leq \mathsf{negl}(k)$.

To break the IND-CCA2 security of the authenticated encryption scheme, $\mathcal{A}$ first has to to come up with two messages m_0 and m_1. One of these will be authencrypted at random, the corresponding ciphertext $C^* \leftarrow \mathsf{AE}_K(m_\sigma)$ (where σ is a random bit) will be given to $\mathcal{A}$, and $\mathcal{A}$ has to guess the value σ. To succeed in the CCA2 attack, $\mathcal{A}$ is only disallowed to ask R to authdecrypt the challenge C^*.[9] The scheme is IND-CCA2-secure if for any PPT $\mathcal{A}$, $\Pr[\mathcal{A} \text{ succeeds}] \leq \frac{1}{2} + \mathsf{negl}(k)$. We also remark that IND-CPA-security is the same, except $\mathcal{A}$ is not given access to the authdecryption oracle.

4.2 Authenticated Encryption of Long Messages

Assume $\mathcal{AE} = (\mathsf{KG}, \mathsf{AE}, \mathsf{AD})$ is a secure authenticated encryption on $|b|$-bit messages. We would like to build an authenticated encryption $\mathcal{AE}' = (\mathsf{KG}', \mathsf{AE}', \mathsf{AD}')$ on $|m|$-bit messages, where $|m| \gg |b|$. More specifically, we seek to employ the following *canonical* composition paradigm. The key K for $\mathcal{AE}'$ is the same as that for $\mathcal{AE}$. To authencrypt m, first split it into two pieces (h, b) (so that the transformation is invertible), and output $\mathsf{AE}'_K(m) = \langle \mathsf{AE}_K(b), h \rangle$. The question we are asking is what are the necessary and sufficient conditions on the transformation $m \to (h, b)$ so that the resulting authenticated encryption

[6]Meaning “strong unforgeability against chosen message attack.”

[7]Meaning “indistinguishability against chosen ciphertext attack.”

[8]A slightly weaker notion of UF-CMA requires C to correspond to “new” message m not submitted to $\mathsf{AE}_K(\cdot)$.

[9][2] define a slightly weaker but more syntactically sound notion of gCCA2 attack. Our results apply here as well.

is secure? In this section we show that the necessary and sufficient condition is to have the transformation above be a *relaxed concealment*.

More formally, assume $\mathcal{C} = (\mathsf{Setup}, \mathsf{Conceal}, \mathsf{Open})$ satisfies the syntax, but not yet the security properties of a concealment scheme. We assume that $\mathsf{CK} \leftarrow \mathsf{Setup}(1^k)$ forms a public parameter pub of $\mathcal{AE}'$. We define $\mathcal{AE}'$ as stated above. Namely, $\mathsf{AE}'(m)$ outputs $\langle \mathsf{AE}(b), h\rangle$, where $(h, b) \leftarrow \mathsf{Conceal}(m)$, and $\mathsf{AD}'(c, h)$ outputs $\mathsf{Open}(h, \mathsf{AD}(c))$. The proof of the following theorem is in Appendix A.1.

Theorem 2 *If $\mathcal{AE}$ is secure, then $\mathcal{AE}'$ is secure if and only if $\mathcal{C}$ is a relaxed concealment scheme.*

4.3 Remotely Keyed Authenticated Encryption

SYNTAX. A one-round remotely-keyed authenticated encryption (RKAE) scheme consists of seven efficient algorithms: $\mathcal{RKAE} = (\mathsf{RKG}, \mathsf{Start\text{-}AE}, \mathsf{Card\text{-}AE}, \mathsf{Finish\text{-}AE}, \mathsf{Start\text{-}AD}, \mathsf{Card\text{-}AD}, \mathsf{Finish\text{-}AD})$ and involves two parties called the *Host* and the *Card*. The Host is assumed to be powerful, but insecure (subject to break-in by an adversary), while the Card is secure but has limited computational power and low bandwidth. The randomized key generation algorithm $\mathsf{KG}(1^k)$, where k is the security parameter, outputs a secret key K, and possibly a public parameter pub. In the description below, all the algorithms (including the adversary's) can have access to pub, but we omit this dependence for brevity. This key K is stored at the Card. The process of authenticated encryption is split into the following 3 steps. First, on input m, the Host runs probabilistic algorithm $\mathsf{Start\text{-}AE}(m)$, and gets (b, α). The value b should be short, as it will be sent to the Card, while α denotes the state information that the Host needs to remember. We stress that $\mathsf{Start\text{-}AE}$ involves no secret keys and can be run by anybody. Next, the Card receives b, and runs probabilistic algorithm $\mathsf{Card\text{-}AE}_K(b)$, using its secret key K. The resulting (short) value c will be sent to the host. Finally, the host runs another randomized algorithm $\mathsf{Finish\text{-}AE}(c, \alpha)$ and outputs the resulting ciphertext C as the final authencryption of m. Again, $\mathsf{Finish\text{-}AE}$ involves no secret keys. The sequential composition of the above 3 algorithms induces an authencryption algorithm, which we will denote by AE'_K.

Similarly, the process of authenticated decryption is split into 3 steps as well. First, on input C, the Host runs deterministic algorithm $\mathsf{Start\text{-}AD}(C)$, and gets (u, β). The value u should be short, as it will be sent to the Card, while β denotes the state information that the Host needs to remember. We stress that $\mathsf{Start\text{-}AD}$ involves no secret keys and can be run by anybody. Next, the Card receives u, and runs deterministic algorithm $\mathsf{Card\text{-}AD}_K(u)$, using its secret key K. The resulting (short) value v will be sent to the host. We note that on possible value for v will be $\perp$, meaning that the Card found some inconsistency in the value of u. Finally, the host runs another randomized algorithm $\mathsf{Finish\text{-}AD}(v, \beta)$ and outputs the resulting plaintext m if $v \neq \perp$, or $\perp$, otherwise. Again, $\mathsf{Finish\text{-}AD}$ involves no secret keys. The sequential composition of the above 3 algorithms induces an authdecryption algorithm, which we will denote by AD'_K. We also call the value C *valid* if $\mathsf{AD}'_K(C) \neq \perp$.

The correctness property states for any m, $\mathsf{AD}'(\mathsf{AE}'(m)) = m$.

SECURITY OF RKAE. As we pointed out, RKAE in particular induces a regular authenticated encryption scheme, if we combine the functionalities of the Host and the Card. Thus, at the very least we would like to require that the induced scheme $\mathcal{AE}' = (\mathsf{RKG}, \mathsf{AE}', \mathsf{AD}')$ satisfies the $\mathsf{IND\text{-}CCA2}$ and $\mathsf{sUF\text{-}CMA}$ security properties of regular authenticated encryption. Of course, this is not a sufficient guarantee in the setting of RKAE. Indeed, such security only allows the adversary oracle access to the *combined* functionality of the Host and the Card. In the setting of RKAE, the Host is anyway insecure, so the adversary should have *oracle access to the functionality of the Card*. Specifically, we allow our adversary $\mathcal{A}'$ to have oracle access to the Card algorithms $\mathsf{Card\text{-}AE}_K(\cdot)$ and $\mathsf{Card\text{-}AD}_K(\cdot)$.

Just like regular authenticated encryption, RKAE has security notions for privacy and authenticity, which we denote by $\mathsf{RK\text{-}IND\text{-}CCA}$ and $\mathsf{RK\text{-}sUF\text{-}CMA}$, respectively.

To break the $\mathsf{RK\text{-}sUF\text{-}CMA}$ security of RKAE, $\mathcal{A}'$ has to be able to produce a "one-more forgery" when interacting with the Card. Namely, $\mathcal{A}'$ tries to output $t+1$ valid ciphertexts $C_1 \ldots C_{t+1}$ after making at most t calls to $\mathsf{Card\text{-}AE}_K(\cdot)$ (where t is any polynomial in k). Again, we remark that $\mathcal{A}'$ is not required to "know" the plaintext values $m_i = \mathsf{AD}'_K(C_i)$. The scheme is $\mathsf{RK\text{-}sUF\text{-}CMA}$-secure if for any PPT $\mathcal{A}'$, $\Pr[\mathcal{A}' \text{ succeeds}] \leq \mathsf{negl}(k)$. We note that this is the only meaningful authenticity notion in the setting of RKAE. This is because the values $c \leftarrow \mathsf{Card\text{-}AE}_K(b)$ returned by the Card have no "semantic" meaning of their own. So it makes no sense to require $\mathcal{A}'$ to produce a new "valid" string c. On the other hand, it is trivial for $\mathcal{A}'$ to compute t valid ciphertexts $C_1 \ldots C_t$ with t oracle calls to $\mathsf{Card\text{-}AE}$, by simply following to honest authencryption protocol on arbitrary messages $m_1 \ldots m_t$. Thus, security against "one-more forgery" is the most ambitious goal we can try to meet in the setting of RKAE.

To break the $\mathsf{RK\text{-}IND\text{-}CCA}$ security of RKAE, $\mathcal{A}'$ first has to come up with two messages m_0 and m_1. One of these will be authencrypted at random, the corresponding ciphertext $C^* \leftarrow \mathsf{AE}_K(m_\sigma)$ (where σ is a random bit) will be given to $\mathcal{A}'$, and $\mathcal{A}'$ has to guess the value σ. To succeed in the $\mathsf{CCA2}$ attack, $\mathcal{A}'$ is only disallowed to call the Card authdecryption oracle $\mathsf{Card\text{-}AD}_K(\cdot)$ on the well-defined value u^*, where we define $\mathsf{Start\text{-}AD}(C^*) = (u^*, \beta^*)$ (recall, $\mathsf{Start\text{-}AD}$ is a deterministic algorithm). The latter restriction is to prevent $\mathcal{A}'$ from trivially authdecrypting the challenge. The scheme is $\mathsf{RK\text{-}IND\text{-}CCA}$-secure if for any PPT $\mathcal{A}'$, $\Pr[\mathcal{A}' \text{ succeeds}] \leq \frac{1}{2} + \mathsf{negl}(k)$. We briefly remark that $\mathsf{RK\text{-}IND\text{-}CPA}$-security is the same, except we do not give $\mathcal{A}'$ access to the Card authdecryption oracle.

CANONICAL RKAE. A natural implementation of RKAE would have the Card perform regular authenticated encryption/decryption on short messages, while the Host should do the special (to be discussed) preprocessing to produce the short message for the Card from the given long message. Specifically, in this case we start from some auxiliary authenticated encryption $\mathcal{AE} = (\mathsf{KG}, \mathsf{AE}, \mathsf{AD})$ which works on "short" $|b|$-bit messages, and require that $\mathsf{Card\text{-}AE} = \mathsf{AE}$, $\mathsf{Card\text{-}AD} = \mathsf{AD}$. Moreover, we would like the Card to authdecrypt the same value c that it produced during authencryption. In our prior notation, $u = c$ and $v = b$, where $c \leftarrow \mathsf{AE}_K(b)$. Finally, it is natural to assume that the Host outputs c as part of the final (long) ciphertext. Putting these together, we come up with the following notion of *canonical* RKAE.

First, the Host runs $\mathsf{Start\text{-}AE}(m)$, which we conveniently rename $\mathsf{Conceal}(m)$, and produces (h, b), where h will be part of the final ciphertext and b is "short". Then it sends b to the Card, and gets back $c \leftarrow \mathsf{AE}_K(b)$. Finally, it outputs $C = \langle c, h \rangle$ as the resulting authencryption of m. Similarly, to authdecrypt $C = \langle c, h \rangle$, it sends c to the Card, gets $b = \mathsf{AD}_K(c)$, and outputs $\mathsf{Finish\text{-}AD}(h, b)$, which we conveniently rename $\mathsf{Open}(h, b)$. Thus, the canonical RKAE is fully specified by an auxiliary authenticated encryption $\mathcal{AE}$ and a triple $\mathcal{C} = (\mathsf{Setup}, \mathsf{Conceal}, \mathsf{Open})$ (where Setup is run at key generation and outputs the key which is part of pub).

The fundamental question we address is this: what security properties of $\mathsf{Conceal}$ and Open are needed in order to achieve a secure canonical RKAE (provided the auxiliary $\mathcal{AE}$ is secure)? As we show, the necessary and sufficient condition is to employ a secure (strong) concealment scheme. We remark that the final *induced* scheme $\mathcal{AE}'$ we construct is *exactly* the composition scheme we discussed in Section 4.2. However, in that application the entire authenticated encryption was performed honestly — in particular, b was chosen by properly running $\mathsf{Conceal}(m)$, — so relaxed concealments were sufficient. Here, an untrusted Host can ask the Card to authencrypt any value b it wishes, so we need the full binding power of strong concealments.

The following theorem states this more formally and its proof is in Appendix A.2.

Theorem 3 *If $\mathcal{AE}$ is secure, and a canonical $\mathcal{RKAE}$ is constructed from $\mathcal{AE}$ and $\mathcal{C}$, then $\mathcal{RKAE}$ is secure if and only if $\mathcal{C}$ is a (strong) concealment scheme.*

COMPARISON TO PREVIOUS RKAES. We briefly compare our scheme with those of [13, 18]. First, both schemes could be put into our framework by extracting appropriate concealment schemes. In fact, the concealment we extract from [13] is essentially the same as our construction $b = \tau \| H(h)$, $h = \mathsf{E}_\tau(m)$ (they model one-time encryption slightly differently, but this is minor)! On the other hand, instead of applying arbitrary authenticated encryption to the value of b, they build a very specific one based on block ciphers and pseudorandom functions. In fact, their construction implicitly achieves a specific authenticated encryption of τ with "associated data" [26] $H(h)$ (see our extension in Section 5). Actually, this authenticated encryption construction could be viewed as an example of the recent "ciphertext translation" method of [26] applied to the "encrypt-then-mac" paradigm of [7]. To summarize, the construction of [13] is quite good and efficient, but focuses on a specific ad-hoc implementations for both concealment and authenticated encryption. We believe that our generality provides many more options, as well as gives better understanding towards designing RKAE, since our general description is much simpler than the specific scheme of [13]. As for the scheme of [18], one can also extract an "OAEP"-like concealment out of it, making it a special case of our framework too. However, the specific choices made by the authors make it very hard to replace the random oracles by some provable implementation. On the other hand, our "OAEP"-like construction (based on a PRG and a CRHF) is equally simple, but achieves provable security without the random oracles.

5 Extensions

USING A BLOCK CIPHER IN PLACE OF $\mathcal{AE}$. First, we briefly touch upon amplification paradigm of the form $\mathcal{AE}'(m) = \langle P_K(b), h\rangle$, where P is a (strong) PRP. Namely, we replace the "inner" authenticated encryption by a block cipher. Although this is applicable only in the symmetric setting, it is likely to be quite useful in practice, where PRP is typically the main building block of most other primitives. We note that a strong PRP is "almost" an authenticated encryption except it does not provide semantic security (but gives at least one-wayness). We ask the same question as before: what are the conditions on the transformation $m \to (h, b)$ for $\mathcal{AE}'$ to be secure? In the following, we just state our results, leaving the proofs to the full version [15].

It turns out that four conditions are needed, the first two of which are subsumed by any relaxed concealment. The last two conditions are stated as follows: (1) for any h, $\Pr_b[\mathsf{Open}(h, b) \neq \bot] = \mathsf{negl}(k)$. This is needed to prevent a "lucky" forgery of the form $\langle v, h\rangle$, where v is arbitrary. This condition always holds for our specific concealments, since the value of b corresponding to any h includes $H(h)$. Thus, the chance that a random b will include the same string as $H(h)$ is negligible, since the output of a CRHF (i.e. H) must be sufficiently long to avoid easy collisions. (2) having oracle access to $\mathsf{Conceal}(\cdot)$, it is hard to ever make it output the same value b. This is needed to ensure the authencryption oracle never evaluates the PRP on the same input, since the adversary will notice it. Again, this is trivially true for our concealments, since the value b always includes a random key τ for one-time encryption. By birthday bound, the chance of collision after q queries is at most $q^2/2^{|b|}$, which must be negligible. To summarize, $\langle P_K(\tau \| H(h)),\ h = \mathsf{E}_\tau(m)\rangle$ is a secure authenticated encryption. We also note that here P_K does not need to be a *strong* PRP; a regular PRP suffices.

Finally, we briefly argue when using a strong PRP suffices for our RKAE application. Here the adversary has direct oracle access to both P_K and P_K^{-1}, so we need at least a strong PRP. It turns out that the following two conditions should hold on the concealment scheme in addition to its regular hiding and binding properties (and properties (1)-(2) above). (1') given a random string b, it is hard to find h such that $\mathsf{Open}(h, b) \neq \bot$. This is needed to prevent the adversary from getting a forgery $\langle v, h\rangle$, where it previously learned $P_s^{-1}(v) = b$. In our case, b includes $H(h)$, so one needs to "invert" H on a random string b. It is easy to see that any CRHF with $|H(h)| < |h| - \omega(\log k)$ must satisfy the needed property. As for the second condition, it states: (2') for any m, if $(h, b) \leftarrow \mathsf{Conceal}(m)$, then it is hard to recover the value b

when given only m and h. This is needed so that the adversary cannot determine the value b corresponding to the challenge, and then check its guess using an oracle call to $P_s(\cdot)$. In our case, given $\mathsf{E}_\tau(m)$ and m, it should be hard to find the correct value of key τ. This property is false for general one-time encryptions (i.e., for one-time pad), but holds for the ones we have in mind here. In particular, if $\mathsf{E}_\tau(m) = G(\tau) \oplus m$, where G is a PRG, finding τ involves inverting $G(\tau)$ on a random τ. And it is well known that a PRG is a one-way function provided $|G(\tau)| > |\tau| + \omega(\log k)$. To summarize, the following scheme is safe to use for RKAE, provided $|G(\tau)| > |\tau| + \omega(\log k)$, $|H(h)| < |h| - \omega(\log k)$ and P is a strong PRP: $\mathcal{AE}'(m) = \langle P_K(\tau \| H(h)),\ G(\tau) \oplus m \rangle$. This remarkably simple scheme means that we can let the Card perform a single block cipher operation per call!

ASSOCIATED DATA. Finally, we briefly discuss extensions to supporting associated data [30, 26]. Intuitively, associated data allows one to "bind" a public label to the message. Viewing the label as part of the message is a possible solution, but the generalized view can bring non-trivial efficiency gains, as was shown by [26]. This extension is presented in more detail in Appendix B.

References

[1] J. AN AND M. BELLARE, "Constructing VIL-MACs from FIL-MACs: Message authentication under weakend assumptions," In *Crypto '99*, pp. 252–269, LNCS Vol. 1666, 1999.

[2] J. AN, Y. DODIS, AND T. RABIN, "On the Security of Joint Signature and Encryption," In *Eurocrypt '02*, pp. 83–107, LNCS Vol. 2332, 2002.

[3] J. BAEK, R. STEINFELD, AND Y. ZHENG, "Formal proofs for the security of signcryption," In *PKC '02*, pp. 80–98, LNCS Vol. 2274, 2002.

[4] M. BELLARE, R. CANETTI AND H. KRAWCZYK, "Keying hash functions for message authentication," In *Crypto '96*, pp. 1–15, LNCS Vol. 1109, 1996.

[5] M. BELLARE, J. KILIAN AND P. ROGAWAY, "The security of the cipher block chaining message authentication code," In *Journal of Computer and System Sciences*, pp. 362–399, Vol. 61, No. 3, Dec 2000.

[6] M. BELLARE, T. KOHNO, C. NAMPREMPRE, "Provably Fixing the SSH Binary Packet Protocol," In *Proc. 9th CCS*, pp. 1–11, ACM, 2002.

[7] M. BELLARE AND C. NAMPREMPRE, "Authenticated Encryption: Relations among Notions and Analysis of the Generic Composition Paradigm," In *Asiacrypt '00*, pp. 531–545, LNCS Vol. 1976, 2000.

[8] M. BELLARE AND P. ROGAWAY, "Optimal asymmetric encryption – How to encrypt with RSA," In *Eurocrypt '94*, pp. 92–111, LNCS Vol. 950, 1994.

[9] M. BELLARE AND P. ROGAWAY, "Collision-Resistant Hashing: Towards Making UOWHFs Practical," In *Crypto '97*, pp. 470–484, LNCS Vol. 1294, 1997.

[10] M. BELLARE, P. ROGAWAY, "Encode-Then-Encipher Encryption: How to Exploit Nonces or Redundancy in Plaintexts for Efficient Cryptography," In *Asiacrypt '00*, pp. 317–330, LNCS Vol 1976, 2000.

[11] J. BLACK, S. HALEVI, H. KRAWCZYK, T. KROVETZ AND P. ROGAWAY, "UMAC: Fast and secure message authentication," In *Crypto '99*, pp. 216–233, LNCS Vol. 1666, 1999.

[12] M. BLAZE, "High-Bandwidth Encryption with Low-Bandwidth Smartcards," In *Fast Software Encryption (FSE) '96*, pp. 33–40, LNCS Vol. 1039, 1996.

[13] M. BLAZE, J. FEIGENBAUM, M. NAOR, "A Formal Treatment of Remotely Keyed Encryption," In *Eurocrypt '98*, pp. 251–265, LNCS Vol. 1403, 1998.

[14] I. DAMGÅRD, "Collision free hash functions and public key signature schemes," In *Eurocrypt '87*, pp. 203–216, LNCS Vol. 304, 1987.

[15] Y. DODIS AND J. AN, "Concealment and its applications to authenticated encryption," Full version of this paper. Preliminary version appeared in *Eurocrypt 03*, pp. 306–323, 2003.

[16] A. JOUX, G. MARTINET, F. VALETTE, "Blockwise-Adaptive Attackers: Revisiting the (In)Security of Some Provably Secure Encryption Models: CBC, GEM, IACBC," In *Crypto '02*, pp. 17–30, LNCS Vol. 2442, 2002.

[17] C. JUTLA, “Encryption modes with almost free message integrity,” In *Eurocrypt ’01*, pp. 529–544, LNCS Vol. 2045, 2001.

[18] M. JAKOBSSON, J. STERN, AND M. YUNG, “Scramble All, Encrypt Small,” In *Fast Software Encryption (FSE) ’99*, pp. 95–111, LNCS Vol. 1636, 1999.

[19] J. KATZ AND M. YUNG, “Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation,” In *FSE ’00*, pp. 284–299, LNCS Vol. 1978, 2000.

[20] H. KRAWCZYK, “The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?),” In *Crypto ’01*, pp. 310–331, LNCS Vol. 2139, 2001.

[21] S. LUCKS, “On the Security of Remotely Keyed Encryption,” In *Fast Software Encryption (FSE) ’97*, pp. 219–229, LNCS Vol. 1267, 1997.

[22] S. LUCKS, “Accelerated Remotely Keyed Encryption,” In *Fast Software Encryption (FSE) ’99*, pp. 112–123, LNCS Vol. 1636, 1999.

[23] A. MENEZES, P. VAN OORSHOT AND S. VANSTONE, “Handbook of applied cryptography,” CRC Press LLC, 1997.

[24] M. NAOR, “Bit Commitment Using Pseudorandomness,” In *Journal of Cryptology*, 4(2):151–158, 1991.

[25] M. NAOR AND M. YUNG, “Universal One-Way Hash Functions and their Cryptographic Applications,” In *Proc. 21st STOC*, pp. 33–43, ACM, 1989.

[26] P. ROGAWAY, “Authenticated-Encryption with Associated-Data,” In *Proc. 9th CCS*, pp. 98–107, ACM, 2002.

[27] P. ROGAWAY, M. BELLARE, J. BLACK, AND T. KROVETZ, “OCB: A Block-Cipher Mode of Operation for Efficient Authenticated Encryption,” In *Proc. 8th CCS*, pp. 196–205, ACM, 2001.

[28] J. ROMPEL, “One-way functions are necessary and sufficient for secure signatures,” In *Proc. 22nd STOC*, pp. 387–394, ACM, 1990.

[29] V. SHOUP, “A composition theorem for universal one-way hash functions,” In *Eurocrypt ’00*, pp. 445–452, LNCS Vol. 1807, 2000.

[30] V. SHOUP, “A proposal for an ISO standard for public key encryption (version 2.1),” IACR E-Print Archive, 2001/112, `http://eprint.iacr.org/2001/112/`, 2001.

[31] D. SIMON, “Finding Collisions on a One-Way Street: Can Secure Hash Functions Be Based on General Assumptions?,” In *Eurocrypt ’98*, pp. 334–345, LNCS Vol. 1403, 1998.

[32] Y. ZHENG, “Digital Signcryption or How to Achieve Cost(Signature & Encryption) $\ll$ Cost(Signature) + Cost(Encryption),” In *Crypto ’97*, pp. 165–179, LNCS Vol. 1294, 1997.

A Proofs

A.1 Proof of Theorem 2

For one direction, we show that if $\mathcal{C}$ does not satisfy the hiding property, then $\mathcal{AE}'$ cannot even be IND-CPA-secure, let alone IND-CCA2-secure. Indeed, if some adversary $\mathcal{A}$ can find m_0, m_1 s.t. $h(m_0) \not\approx h(m_1)$, then obviously $\mathsf{AE}'(m_0) \equiv (\mathsf{AE}(b(m_0)), h(m_0)) \not\approx (\mathsf{AE}(b(m_1)), h(m_1)) \equiv \mathsf{AE}'(m_1)$, contradicting IND-CPA-security.

Similarly, if $\mathcal{C}$ does not satisfy the relaxed binding property, then $\mathcal{AE}'$ cannot be sUF-CMA-secure. Indeed, assume some adversary $\mathcal{A}$ can produce m such that when $(h, b) \leftarrow \mathsf{Conceal}(m)$ is generated and given to $\mathcal{A}$, $\mathcal{A}$ can find (with non-negligible probability ε) a value $h' \neq h$ such that $\mathsf{Open}(h', b) \neq \perp$. We build a forger $\mathcal{A}'$ for $\mathcal{AE}'$ using $\mathcal{A}$. $\mathcal{A}'$ gets m from $\mathcal{A}$, and asks its authencryption oracle the value $\mathcal{AE}'(m)$. $\mathcal{A}'$ gets back (h, c), where c is a valid authencryption of b, and (h, b) is a random concealment pair for m. $\mathcal{A}'$ gives (h, b) to $\mathcal{A}$, and gets back (with probability ε) the value $h' \neq h$ such that $\mathsf{Open}(h', b) \neq \perp$. But then (h', c) is a valid authencryption (w.r.t. $\mathcal{AE}'$) different from (h, c), contradicting the sUF-CMA-security of $\mathcal{AE}$.

PROOF OF $\mathsf{IND\text{-}CCA2}$-SECURITY. We start with $\mathsf{IND\text{-}CCA2}$-security of $\mathcal{AE}'$. Let Env_1 denote the usual environment where we place any adversary $\mathcal{A}'$ for $\mathcal{AE}'$. In particular, (1) it honestly answers all the oracle queries of $\mathcal{A}'$ throughout the run of $\mathcal{A}'$; and (2) when (m_0, m_1) are selected, Env_1 picks a random σ, sets $(h^*, b^*) \leftarrow \mathsf{Conceal}(m_\sigma)$, $c^* \leftarrow \mathcal{AE}(b^*)$ and returns $C^* = \mathcal{AE}'(m_\sigma) = \langle h^*, c^* \rangle$. We let $\mathsf{Succ}_1(\mathcal{A}')$ denote the probability $\mathcal{A}'$ succeeds in predicting σ in Env_1. We next define the following slightly modified environment Env_2. It is identical to Env_1 modulo one respect. If $\mathcal{A}'$ submits a ciphertext $\langle h, c^* \rangle$ to its authdecryption oracle, where c^* is part of the challenge $C^* = \langle h^*, c^* \rangle$ and $h \neq h^*$, then Env_2 responds with $\perp$ without even trying to verify if this is correct. We let $\mathsf{Succ}_2(\mathcal{A}')$ denote the probability $\mathcal{A}'$ succeeds in predicting σ in Env_2. Next, we modify Env_2 into a related Env_3 as follows. When Env_3 prepares the challenge C^*, it also picks some fixed message, call it 0, and outputs $C^* = \langle h^*, \mathcal{AE}(b(0)) \rangle$ instead previously output $C^* = \langle h^*, \mathcal{AE}(b^*) \rangle$. We let $\mathsf{Succ}_3(\mathcal{A}')$ denote the probability $\mathcal{A}'$ succeeds in predicting σ in Env_3.

We make three claims: (a) using the relaxed binding property of $\mathcal{C}$, no PPT adversary $\mathcal{A}'$ can distinguish Env_1 from Env_2, i.e. $|\mathsf{Succ}_1(\mathcal{A}') - \mathsf{Succ}_2(\mathcal{A}')| \leq \mathsf{negl}(k)$;[10] (b) using $\mathsf{IND\text{-}CCA2}$-security of $\mathcal{AE}$, no PPT adversary $\mathcal{A}'$ can distinguish Env_2 from Env_3, i.e. $|\mathsf{Succ}_2(\mathcal{A}') - \mathsf{Succ}_3(\mathcal{A}')| \leq \mathsf{negl}(k)$; (c) using the hiding property of $\mathcal{C}$, $\mathsf{Succ}_3(\mathcal{A}') < \frac{1}{2} + \mathsf{negl}(k)$, for any PPT $\mathcal{A}'$. Combined, claims (a)-(c) imply the $\mathsf{IND\text{-}CCA2}$-security of $\mathcal{AE}'$.

PROOF OF CLAIM (A). Notice, the only way some A' can see the difference between Env_1 and Env_2, if in Env_1 it was able to produce a valid ciphertext (h, c^*), where $h \neq h^*$, as otherwise Env_1 and Env_2 are identical. But this means that $\mathsf{Open}(h, b^*) \neq \perp$, and $h \neq h^*$, where $(h^*, b^*) \leftarrow \mathsf{Conceal}(m_\sigma)$. It is straightforward to see that this contradicts the relaxed binding property of $\mathcal{C}$, since we can construct $\mathcal{A}_1$ which prepares K by itself, submits m_σ after the find phase, and simply runs $\mathcal{A}'$ until the collision with b^* happens.

PROOF OF CLAIM (B). If for some $\mathcal{A}'$, $|\mathsf{Succ}_2(\mathcal{A}') - \mathsf{Succ}_3(\mathcal{A}')| > \varepsilon$ for non-negligible ε, we create $\mathcal{A}_2$ which will break $\mathsf{IND\text{-}CCA2}$-security of $\mathcal{AE}$. It simulates the run of $\mathcal{A}'$ by generating a concealment key CK by itself, and using its own authencryption/authdecryption oracle to answer the oracle queries of $\mathcal{A}'$. For example, $\mathcal{A}_2$ can simulate the authdecryption query $C' = (h', c')$ of $\mathcal{A}'$ by asking its own authdecryption oracle to decrypt $b' = \mathsf{AD}(c')$, and returning $\mathsf{Open}(h', b')$. Simulating authencryption queries is done similarly. When $\mathcal{A}'$ outputs m_0 and m_1, $\mathcal{A}_2$ chooses a random $\sigma \in \{0,1\}$, sets $(h_\sigma, b_\sigma) \leftarrow \mathsf{Conceal}(m_\sigma)$, $\tilde{b} = b(0)$ and claims to distinguish b_σ and $\tilde{b}$. When given challenge c^* which is either $\mathsf{AE}(b_\sigma)$ or $\mathsf{AE}(\tilde{b})$, $\mathcal{A}_2$ gives $\mathcal{A}'$ the challenge $C^* = (h_\sigma, c^*)$. Next, $\mathcal{A}_2$ uses its own authdecryption oracle to answer all decryption queries $C' = (h', c')$ as before (authencryption queries stay the same too). Notice, there is no need to worry about the case when $c' = c^*$, since both Env_2 and Env_3 are supposed to respond with $\perp$. Finally, when $\mathcal{A}'$ output its guess σ', A_2 guesses the message was b_σ (i.e., it ran $\mathcal{A}'$ in Env_2) if $\sigma' = \sigma$, and $\tilde{b}$ (i.e., it ran $\mathcal{A}'$ in Env_3) otherwise. It is easy to see that the advantage of A_2 is exactly ε.

PROOF OF CLAIM (C). If for some $\mathcal{A}'$, $\mathsf{Succ}_3(\mathcal{A}') > \frac{1}{2} + \varepsilon$ for non-negligible ε, we create $\mathcal{A}_3$ that will break the hiding property of $\mathcal{C}$. $\mathcal{A}_3$ simply picks the key $K \leftarrow \mathsf{KG}(1^k)$ by itself and runs $\mathcal{A}'$ until it outputs (m_0, m_1). It claims to distinguish the hiders of m_0 and m_1 and well, and gets a challenge $h^* = h(m_\sigma)$ for unknown σ. It gives $\mathcal{A}$ the challenge $C^* = \langle \mathsf{AE}(b(0)), h^* \rangle$, and keeps running $\mathcal{A}$ till the end outputting the same guess σ'. It is obvious it wins if and only if $\mathcal{A}'$ wins in Env_3, a contradiction.

PROOF OF $\mathsf{sUF\text{-}CMA}$-SECURITY. Finally, we show $\mathsf{sUF\text{-}CMA}$-security of $\mathcal{AE}'$. Assume some forger $\mathcal{A}'$ breaks the $\mathsf{sUF\text{-}CMA}$-security of $\mathcal{AE}'$ with non-negligible probability ε. Assume $\mathcal{A}'$ made (wlog exactly) $t = t(k)$ oracle queries to AE' for some polynomial $t(k)$. For $1 \leq i \leq t$, we let m_i be the i-th message $\mathcal{A}'$ asked to authencrypt, and (h_i, c_i) be its authencryption (where $(h_i, b_i) \leftarrow \mathsf{Conceal}(m_i)$ and $c_i \leftarrow \mathsf{AE}(b_i)$). We also let m, h, b, c have similar meaning for the ciphertext that $\mathcal{A}'$ forged. Finally, let Forged denote the

[10] As mentioned, this part is unnecessary to show in the symmetric-key setting, but is needed in the public-key setting.

event that $c \notin \{c_1, \ldots, c_t\}$. Notice,

$$\varepsilon < \Pr(\mathcal{A}' \text{ succeeds}) = \Pr(\mathcal{A}' \text{ succeeds } \wedge \ \mathsf{Forged}) + \Pr(\mathcal{A}' \text{ succeeds } \wedge \ \overline{\mathsf{Forged}})$$

Thus, at least one of the probabilities above is $\geq \varepsilon/2$. We show that the first case contradicts the sUF-CMA-security of $\mathcal{AE}$, while the second case contradicts the relaxed binding property of $\mathcal{C}$.

CASE 1: $\Pr(\mathcal{A}'$ SUCCEEDS $\wedge \mathsf{Forged}) \geq \varepsilon/2$. We construct a forger $\mathcal{A}_1$ for $\mathcal{AE}$. It simulates the run of $\mathcal{A}'$ by generating a concealment key CK by itself, and using its own authencryption/authdecryption oracle to answer the oracle queries of $\mathcal{A}'$. For example, $\mathcal{A}_1$ can simulate the authencryption query m_i of $\mathcal{A}'$ by setting $(h_i, b_i) \leftarrow \mathsf{Conceal}(m_i)$, getting $c_i \leftarrow \mathsf{AE}(b_i)$ from the oracle, and returning (c_i, h_i). When $\mathcal{A}'$ forges a ciphertext (c, h) w.r.t. $\mathcal{AE}'$, $\mathcal{A}_1$ forges ciphertext c (of b) w.r.t. $\mathcal{AE}$. Notice, c is a "new forgery" in $\mathcal{AE}$ iff Forged happens. Hence, $\mathcal{A}_1$ succeeds with probability at least $\varepsilon/2$, a contradiction to sUF-CMA-security of $\mathcal{S}$.

CASE 2: $\Pr(\mathcal{A}'$ SUCCEEDS $\wedge \ \overline{\mathsf{Forged}}) \geq \varepsilon/2$. We construct an adversary $\mathcal{A}_2$ contradicting the relaxed binding property of $\mathcal{C}$. $\mathcal{A}_2$ will generate its own key $K \leftarrow \mathsf{KG}(1^k)$, and will also pick a random index $1 \leq i \leq t$. It simulates the run of $\mathcal{A}'$ in a standard manner (same way as $\mathcal{A}_1$ above) up to the point where $\mathcal{A}'$ asks its i-th query m_i. At this stage $\mathcal{A}_2$ outputs m_i as its output to the find stage. When receiving back random $(h_i, b_i) \leftarrow \mathsf{Conceal}(m_i)$, it uses them to authencrypt m_i as before (i.e., returns $\langle c_i = \mathsf{AE}(b_i), h_i \rangle$ to $\mathcal{A}'$), and keeps simulating the run of $\mathcal{A}'$ in the usual manner. When $\mathcal{A}$ outputs the forgery (c, h) of a message m, $\mathcal{A}_2$ checks if $c_i = c$ and $h_i \neq h$. If this fails, it fails as well. Otherwise, it outputs h as its final output to the collide stage. We note that when Forged does not happen, i.e. $c \in \{c_1 \ldots c_t\}$, we have $c = c_i$ with probability at least $1/t$. Thus, with overall non-negligible probability $\varepsilon/(2t)$ we have that: (1) $c_i = c$ (Forged did not happen and $\mathcal{A}_2$ correctly guessed i such that $c_i = c$), so that $b_i = b$; (2) $h \neq h_i$ (since $\mathcal{A}'$ has to output a "new" forgery); (3) $\mathsf{Open}(h, b) \neq \perp$. But this exactly means that $\mathcal{A}_2$ broke the relaxed binding property of $\mathcal{C}$, a contradiction.

A.2 Proof of Theorem 3

The necessity of hiding is obvious, since h is given in the clear. Similarly, if $\mathcal{C}$ does not satisfy the binding property, then some $\mathcal{A}$ can find b, $h \neq h'$ such that $\mathsf{Open}(h, b)$ and $\mathsf{Open}(b.h')$ are both valid. But then it can call the Card's authencryption oracle $\mathsf{AE}_K(\cdot)$ once on input b, get the value c back, and output two valid ciphertexts $\langle h, c \rangle$, $\langle h', c \rangle$, thus successfully producing a "one-more forgery" on $\mathcal{RKAE}$.

PROOF OF IND-CCA2-SECURITY. The RK-IND-CCA-security of $\mathcal{RKAE}$ is very similar to the proof of IND-CCA2 security of $\mathcal{AE}'$ given in Theorem 2. In fact, the proof is even simpler since the adversary $\mathcal{A}'$ has oracle access to the actual oracles AE and AD. Moreover, we do not even have to use the (relaxed) binding property here, since the adversary is already forbidden to ask AD oracle on the challenge value $c^* = \mathsf{AE}(m_\sigma)$.

For completeness, brief details follow. Let Env_1 denote the usual environment where we place any adversary $\mathcal{A}'$ for $\mathcal{RKAE}$. In particular, (1) it honestly answers all the oracle queries to AE, AD throughout the run of $\mathcal{A}'$; and (2) when (m_0, m_1) are selected, Env_1 picks a random σ, sets $(h^*, b^*) \leftarrow \mathsf{Conceal}(m_\sigma)$, $c^* \leftarrow \mathcal{AE}(b^*)$ and returns $C^* = \mathcal{AE}'(m_\sigma) = \langle h^*, c^* \rangle$. We let $\mathsf{Succ}_1(\mathcal{A}')$ denote the probability $\mathcal{A}'$ succeeds in predicting σ in Env_1. Notice, $\mathcal{A}'$ is not allowed to submit c^* to AD after it gets C^*. We next define the following slightly modified environment Env_2. When Env_3 prepares the challenge C^*, it also picks some fixed message, call it 0, and outputs $C^* = \langle h^*, \mathcal{AE}(b(0)) \rangle$ instead previously output $C^* = \langle h^*, \mathcal{AE}(b^*) \rangle$. We let $\mathsf{Succ}_2(\mathcal{A}')$ denote the probability $\mathcal{A}'$ succeeds in predicting σ in Env_1.

We make three claims: (a) using IND-CCA2-security of $\mathcal{AE}$, no PPT adversary $\mathcal{A}'$ can distinguish Env_1 from Env_2, i.e. $|\mathsf{Succ}_1(\mathcal{A}') - \mathsf{Succ}_2(\mathcal{A}')| \leq \mathsf{negl}(k)$; (c) using the hiding property of $\mathcal{C}$, $\mathsf{Succ}_2(\mathcal{A}') < \frac{1}{2} + \mathsf{negl}(k)$, for any PPT $\mathcal{A}'$. Combined, claims (a)-(b) imply the RK-IND-CCA-security of $\mathcal{RKAE}$.

PROOF OF CLAIM (A). If for some $\mathcal{A}'$, $|\mathsf{Succ}_1(\mathcal{A}') - \mathsf{Succ}_2(\mathcal{A}')| > \varepsilon$ for non-negligible ε, we create $\mathcal{A}_1$ which will break IND-CCA2-security of $\mathcal{AE}$. It simulates the run of $\mathcal{A}'$ by generating a concealment key CK by itself, and using its own authencryption/authdecryption oracle to answer the oracle queries of $\mathcal{A}'$ throughout the run of $\mathcal{A}'$. In fact, these oracle are *identical* to what $\mathcal{A}'$ expects from the Card. When $\mathcal{A}'$ outputs m_0 and m_1, $\mathcal{A}_1$ chooses a random $\sigma \in \{0,1\}$, sets $(h_\sigma, b_\sigma) \leftarrow \mathsf{Conceal}(m_\sigma)$, $\tilde{b} = b(0)$ and claims to distinguish b_σ and $\tilde{b}$. When given challenge c^* which is either $\mathsf{AE}(b_\sigma)$ or $\mathsf{AE}(\tilde{b})$, $\mathcal{A}_1$ gives $\mathcal{A}'$ the challenge $C^* = (h_\sigma, c^*)$. Notice, $\mathcal{A}'$ is not allowed to submit c^* to AE_k, so A_1 never has to do it either. Finally, when $\mathcal{A}'$ output its guess σ', A_2 guesses the message was b_σ (i.e., it ran $\mathcal{A}'$ in Env_1) if $\sigma' = \sigma$, and $\tilde{b}$ (i.e., it ran $\mathcal{A}'$ in Env_2) otherwise. It is easy to see that the advantage of A_1 is exactly ε.

PROOF OF CLAIM (B). If for some $\mathcal{A}'$, $\mathsf{Succ}_2(\mathcal{A}') > \frac{1}{2} + \varepsilon$ for non-negligible ε, we create $\mathcal{A}_2$ that will break the hiding property of $\mathcal{C}$. $\mathcal{A}_2$ simply picks the key $K \leftarrow \mathsf{KG}(1^k)$ by itself and runs $\mathcal{A}'$ until it outputs (m_0, m_1). It claims to distinguish the hiders of m_0 and m_1 and well, and gets a challenge $h^* = h(m_\sigma)$ for unknown σ. It gives $\mathcal{A}$ the challenge $C^* = \langle \mathsf{AE}(b(0)), h^* \rangle$, and keeps running $\mathcal{A}$ till the end outputting the same guess σ'. It is obvious it wins if and only if $\mathcal{A}'$ wins in Env_2, a contradiction.

PROOF OF sUF-CMA-SECURITY. The proof of RK-sUF-CMA-security is quite simple too. Assume some forger $\mathcal{A}'$ asks t queries to AE_K and gets responses $c_1 \ldots c_t$. assume also that it outputs $t+1$ distinct valid ciphertexts $C_i = \langle c_i', h_i' \rangle$ for $1 \leq i \leq t+1$. There are two cases. Either all c_i' are distinct, or at least two of them are the same. In the former case, by the pigeon-hole principle at least one $c_i' \notin \{c_1 \ldots c_t\}$, but this means $\mathcal{A}'$ output a "new" valid ciphertext for $\mathcal{AE}$, contradicting its sUF-CMA-security. Otherwise, some $c_i' = c_j' = c$. Let $b = \mathsf{AD}_K(c)$. But then $\mathsf{Open}(h_i, b) \neq \perp$, $\mathsf{Open}(h_j, b) \neq \perp$, and h_i must be different from h_j since $C_i \neq C_j$ and $c_i = c_j$. And this clearly contradicts the strong binding property of $\mathcal{C}$.

B Supporting Associated Data

In this section, we show how to extend our methods to support associated data [30, 26]. Following the prior terminology of [30], we will refer to associated data as a *label* ℓ. Intuitively, labels do not have to be hidden, but should be "bound" to the corresponding message.

CONCEALMENT WITH ASSOCIATED DATA. Now, both algorithms Conceal and Open will take both the message m and the label ℓ. For future convenience, concealment can now also output some *public part* p, in addition to the hider and the binder. Formally, $\mathsf{Conceal}^{\ell}_{\mathsf{CK}}(m)$ outputs a triple (h, b, p), and $\mathsf{Open}^{\ell}_{\mathsf{CK}}(h, b, p)$ recovers m. As for security, the hiding property says that for any m_0, m_1, ℓ, if $(h_i, b_i, p_i) \leftarrow \mathsf{Conceal}^{\ell}(m_i)$ (where $i \in \{0,1\}$), then it is hard to distinguish (h_0, p_0) from (h_1, p_1). On the other hand, binding now says that it is hard to find (ℓ, b, p, h_0, h_1) such that $h_0 \neq h_1$ and both (h_0, b, p), (h_1, b, p) open successfully with ℓ. Relaxed binding is similar. Notice, public part p participates in both the hiding and the binding properties. Of course, our previous definition corresponds to $\ell = p = \emptyset$, while our new goal is to have $|b| + |p| \ll |m| + |\ell|$, where minimizing $|b|$ is more important.

CONSTRUCTION WITH ASSOCIATED DATA. We show that the constructions in Section 3 nicely extend to support labels. Hiding only is done as before via $h = \mathsf{E}_\tau(m)$, $b = \tau$. To add binding using CRHFs, we could set $h' = h$, $b' = b \| H(h \| \ell)$. However, we can move $H(h \| \ell)$ into the public part p'. Thus, we set $h' = h$, $b' = b$ and $p' = H(h \| \ell)$. In particular, we get a scheme with $h' = \mathsf{E}_\tau(m)$, $b' = \tau$ and $p' = H(h' \| \ell)$. Similar discussion holds for getting relaxed concealments using UOWHFs. In the final scheme, we get $h'' = \mathsf{E}_\tau(m)$, $b'' = \tau$ and $p'' = H(h'' \| \ell) \| H$. Notice, we moved a slightly expensive description of H from the binder into the public part.

AUTHENTICATED ENCRYPTION WITH ASSOCIATED DATA (AEAD). Following [26], we briefly describe the syntax and security of authenticated encryption with associated data. Essentially, the only thing that changes is that both AE and AD are augmented to take ℓ in addition to m: $c \leftarrow \mathsf{AE}^{\ell}_K(m)$, $m = \mathsf{AD}^{\ell}_K(c)$.

Also, the adversary $\mathcal{A}$ can now submit a pair (m, ℓ) or (c, ℓ) to its oracles. For $\mathsf{IND\text{-}CCA2}$ security, $\mathcal{A}$ has to come up with m_0, m_1, ℓ, gets a challenge $c^* \leftarrow \mathsf{AE}_K^\ell(m_\sigma)$ (for random σ) and has to predict σ, as before, provided it did not call $\mathsf{AD}^\ell(c^*)$ (but using other label is allowed). $\mathsf{sUF\text{-}CMA}$-security does not change as well except the entire pair (c, ℓ) has to be "new". Rogaway [26] demonstrated several authenticated encryption schemes, where the distinction between the message and the label (or "header") indeed leaves to significantly improved efficiency.

SUPPORTING ASSOCIATED DATA. We show that the composition paradigm above naturally supports AEAD. Namely, assume $\mathcal{AE} = (\mathsf{KG}, \mathsf{AE}, \mathsf{AD})$ supports messages of some "short" length $|b|$ and associated labels of "short" length $|p|$. Assume also that the concealment $\mathcal{C}$ supports messages of "long" length $|m|$ and labels of "long" length $|\ell|$. We define the composed authenticated encryption scheme $\mathcal{AE}' = (\mathsf{KG}', \mathsf{AE}', \mathsf{AD}')$ as follows. $\mathsf{KG}' = \mathsf{KG}$ except we also publish public information $\mathsf{CK} \leftarrow \mathtt{Setup}(1^k)$. ${\mathsf{AE}'}_K^{\ell}(m)$ first runs $(h, b, p) \leftarrow \mathsf{Conceal}^\ell(m)$ and outputs $\langle \mathsf{AE}_K^p(b), h, p \rangle$. It is a simple extension of our prior discussion that the resulting scheme $\mathcal{AE}'$ is a secure AEAD if and only if $\mathcal{C}$ is a relaxed concealment with associated data. Finally, notice that our constructions of such concealments based on CRHFs achieved $|p|, |b| = O(k)$, irrespective of the length of m and ℓ. Also, in our case outputting p is redundant since $p = H(h\|\ell)$ and can be computed from the ciphertext and the label. Thus, our particular AEAD scheme outputs $\left\langle \mathsf{AE}_K^{H(h\|\ell)}(\tau), h = \mathsf{E}_\tau(m) \right\rangle$.

We remark that similar construction applies to RKAE with associated data, where the Host sends b, p to the Card, and gets back $\mathsf{AE}_K^p(b)$. For authencryption, it sends c, p and gets back $\mathsf{AD}_K^p(c)$. As earlier, relaxed concealments (with associated data) no longer suffice, so we will need strong concealments.