Concealment and its Applications to Authenticated Encryption

Yevgeniy Dodis

Department of Computer Science, New York University, USA. dodis@cs.nyu.edu

Summary. In this survey article we will study a recent cryptographic primitive called *concealment*, which was introduced by Dodis and An [17] because of its natural applications to authenticated encryption. 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 will examine necessary and sufficient assumptions for building various flavors of concealment, and give simple, general and efficient constructions of concealments giving rise to a multitude of efficient implementations.

We also discuss two main applications of concealments to the area of authenticated encryption. First, following [17, 1], we show that concealment is the right cryptographic primitives enabling one to extend the domain of authenticated encryption. Specifically, let $\mathcal{AE}$ be an authenticated encryption scheme (either public- or symmetric-key)[1] designed to work on short messages. Using concealments, we can transform $\mathcal{AE}$ into a new authenticated encryption scheme $\mathcal{AE}'$ on longer messages as follows. To encrypt a longer message m, one uses a concealment scheme to get h and b, and then outputs authenticated ciphertext $\mathcal{AE}'(m) = \langle \mathcal{AE}(b), h \rangle$.

Second, the above paradigm leads to a very simple and general solution to the problem of *remotely keyed (authenticated) encryption* (RKAE) [14, 15], so far primarily studied in the symmetric-key setting. 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. Following [17], we show that the composition paradigm above gives a provably secure solution for RKAE: for authenticated encryption of m, 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).

[1] We note that authenticated encryption in the public-key setting is typically called *signcryption* [36]. However, since all our applications of concealments will work, with minor adjustments, in both in the symmetric- and in the public-key settings, we will use the term *authenticated encryption* throughout.

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 [8, 24, 3], at least if not used appropriately [3, 30]. Second, a tailored solution providing both privacy and authenticity might be noticeably more efficient (or have other advantages) than a straightforward composition [21, 31, 36, 3, 7]. Third, the proper modeling of authenticated encryption is not so obvious, especially in the public-key setting [3, 4]. Finally, viewing authenticated encryption as a separate *primitive* may conceptually simplify the design of complex protocols which require both privacy and authenticity.

Domain Extension of Authenticated Encryption. In this article we study a natural question of securely extending the domain of authenticated encryption. Specifically, assume we have a secure authenticated encryption (either symmetric- or public-key; see Footnote 1) $\mathcal{AE}$ 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.) In the context of authenticated encryption, this question was formally studied by Dodis and An [17] (whose work we closely follow here). However, 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 as well.

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 [27]), 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*[2] [16] H which maps a long input m into a short output such that it is hard to find a "collision"

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

$H(m_0) = H(m_1)$ 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 [6, 5, 13, 2] 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 [8] for authenticated encryption, the CBC mode for encryption, the CBC-MAC [6] 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 [21] or OCB [31]. 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 [3].

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, ease of implementation, etc.), depending on its application and implementation. Consider, for example, the public-key setting, where authenticated encryption is usually called *signcryption* [36] (see Footnote 1). With any of the generic signature-encryption compositions [3], 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 [36] 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* [14, 25, 15] (RKAE), discussed a bit later. None of the other techniques we mentioned seem to yield the solution to this problem.

Main Construction and a New Primitive: Concealment. Following [17], 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? [17, 1] completely characterize such transformations T, which are called *concealments*. Specifically, they show that $\mathcal{AE}'$ is secure if and only if T is an "appropriate" concealment scheme, where "appropriate" depends on the exact setting we consider, as discussed later.

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 three formalizations leading to the notions of regular, relaxed and super-relaxed concealment schemes. Super-relaxed concealments will turn out to be necessary and sufficient [17, 1] for the symmetric-key setting and the so called *"outside-secure"* public-key setting [3]. Relaxed concealments will be necessary and sufficient [17, 1] for the stronger and more desirable *"insider-secure"* public-key setting [3]. Finally, regular concealments will be necessary and sufficient [17] for the problem of RKAE (in either the symmetric- or the public-key settings), briefly mentioned earlier and discussed shortly. We also 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 require the existence of one-way functions. Additionally, ensuring regular binding property requires the existence of collision-resistant hash functions (CRHFs). From a positive perspective, we give a very efficient general construction of (all kinds of) concealments matching the necessary requirements stated above. Our construction uses any one-time secure symmetric key encryption (which can be built efficiently from pseudorandom generators or standard block ciphers) to ensure message hiding, and a certain family of hash functions: almost universal hash functions (AUHFs) [35] for super-relaxed binding, universal one-way hash function (UOWHFs) [29] for relaxed binding, and collision-resistant hash functions (CRHFs) [16] for regular binding. When instantiated with standard components, our constructions have a binder b whose length is only proportional to the security parameter and is independent of the message length, while the length of the hider h is roughly equal to the length of the message. In fact, one special case of our construction looks very similar to

the famous *Optimal Asymmetric Encryption Padding* (OAEP) [9], although without relying on random oracles!

To summarize, 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 [14] 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 [25], 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 [15]. 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. [15] gave formal definitions and constructions of RKAE and RKPRP, and the latter's construction was subsequently improved by [26].

While the RKAE definition of [15] 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 [23, 11, 8]. Of course, this has a very objective reason in that the above formal definition appeared *after* the work of [15]. 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 follow the work of Dodis and An [17] who gave a simpler and more natural such definition, which looks very close to the definition of regular authenticated encryption. Additionally, [17] naturally extend the whole concept of RKAE to the *public-key* setting, since it is equally applicable in this case too.[3] Notice, in the public-key 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. [22], 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 [22] looks much closer to the one of [17]. However, there are still significant differences that make the latter notion stronger.[4] For example, [22] 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. We also 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. [20]. From 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 surprisingly, 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 the solutions of [17] (see "extensions" paragraph below), all the Card does is a single block cipher call per invocation!

[3] In this abstract, though, we will concentrate on the more popular symmetric-key setting, only briefly mentioning the simple extension to the public-key setting.

[4] Except both [22] and [15] 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.

As a corollary, the work of [20] strongly supports our prior claim that direct "long" authenticated encryption schemes, such as IACBC [21], do not seem to be naturally suited for RKAE.

RKAE Constructions. In addition to giving a simple and natural definition of RKAE, Dodis and An [17] showed 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, one can ask the question which transformations T will suffice to make this simple scheme secure. Not surprisingly, [17] showed that concealment schemes were necessary and sufficient, even though in this case one needs regular binding property of concealments, and must utilize CRHFs. Overall, the above result gives a general and intuitively simple solution to the problem of RKAE. Also, it generalizes the previous, so "differently looking" solutions of [15, 22], both of which can be shown to use some particular concealment and/or "short" authenticated encryption.

Extensions. All the techniques mentioned above naturally support authenticated encryption *with associated data* [30]. 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 [30]. As shown by [17], these gains carry over to the questions studied in this survey. However, we omit the details here and instead refer to [17].

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 from [17] that makes sense only in the symmetric-key setting. They asked the question if one can replace the given "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. As shown by [17], although arbitrary concealments are generally *not* enough to ensure the security of thus constructed $\mathcal{AE}'$, some mild extra restrictions —enjoyed by the natural concealment constructions— make them sufficient for this purpose as well![5] Again, we refer to [17] for more details.

[5] Unfortunately, the shortest length of the binder b which we can currently achieve is roughly 300 bits. This means that most popular block ciphers, such as AES, cannot be used in this setting. However, any block cipher with a 512-bit block seems to more than sufficient.

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., $\mathcal{M} = \{0,1\}^n$ for some parameter $n = n(k)$), 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 of the security parameter k):

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

  where α is some state information. Sometime, we will write $h(m_0) \approx h(m_1)$ to indicate that $h(m_0)$ is computationally indistinguishable from $h(m_1)$.
- **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., $\mathsf{Open}_{\mathsf{CK}}(h, b) \neq \perp$ and

$\mathsf{Open}_{\mathsf{CK}}(h', b) \neq \perp$). That is, $\mathcal{A}$ cannot find a binder b which it can open with two different hiders.[6]

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 $|h| \approx |m|$.

As we shall see, for some applications of concealment two slightly weaker forms of binding will be enough. For the lack of better names, we call them *relaxed binding* and *super-relaxed binding*.

Relaxed Concealments. We 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 before learning $(h(m), b(m))$. Formally, for any PPT $\mathcal{A}$, which runs in two stages find and $\mathsf{collide}$, the following probability is at most $\mathsf{negl}(k)$:

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

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 [34], UOWHFs are strictly weaker primitives than CRHFs (in particular, they can be built from regular one-way functions [29]), which implies that relaxed concealments form a weaker cryptographic assumption than regular concealments.

Super-Relaxed Concealments. Finally, we will consider an even weaker form of **Super-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 before learning *only* $h(m)$. Namely, $\mathcal{A}$ has to find a collision for $b(m)$ without even knowing $b(m)$! Formally, for any PPT $\mathcal{A}$, which runs in two stages find and $\mathsf{collide}$, the following probability is at most $\mathsf{negl}(k)$:

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

As we shall see, we will be able to achieve super-relaxed binding *unconditionally*; namely, without even relying on one-way functions (which was essential for relaxed and regular binding).

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

[6] 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.

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 [28]. 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; for example, see [3] on how to use commitments to build "parallel" authenticated encryption from regular signature and encryption schemes).

3 Constructing Concealment Schemes

In this section, we give very simple and general constructions of concealment schemes based on some "appropriate" family of hash functions (see below) and any symmetric one-time encryption scheme.

Our construction will be split into two phases. First, we show how to achieve only hiding using symmetric one-time encryption, and then we show how to use hash functions to add binding to any scheme which already enjoys hiding. We will conclude the section with the observation that all the assumption we utilize in our constructions are not only sufficient, but also necessary. Thus, our constructions are tight.

3.1 Achieving Hiding

We first show how to achieve the hiding property so that $|b| \ll |m|$. 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

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

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.

Lemma 1. *If $\mathcal{SE}$ is a one-time secure encryption scheme, then the above concealment scheme satisfies hiding. Moreover, the scheme is non-trivial if and only if the key τ is shorter than the message m (which, by the result of [19], requires the existence of one-way functions).*

3.2 Achieving Binding

Next, we show how to add regular/relaxed/super-relaxed binding property using any family of collision-resistant/universal one-way/almost universal hash functions (CRHFs/UOWHFs/AUHFs). Recall, CRHFs/UOWHFs/AUHFs are defined by some family $\mathcal{H} = \{H\}$ of compressing functions for which no computationally bounded attacker can find, with non-negligible probability, a colliding pair $x \neq x'$ such that $H(x) = H(x')$, where H is a function randomly chosen from $\mathcal{H}$. However,

- with CRHFs, we first select the function H and let the attacker find (x, x') based on H.
- with UOWHFs, the attacker selects x before seeing H, and only then finds x' based on H.
- with AUHFs, the attacker has to select both (x, x') before seeing H.

We will comment on the known constructions of such hash families later, here only mentioning that AUHFs can be built unconditionally, the existence of UOWHFs is equivalent to the existence of one-way functions [32], while the existence of CRHFs seems to require strictly stronger computational assumptions than one-way functions [34]. Instead, now we see how to utilize such hash functions for our purposes of achieving the corresponding form of binding. In all the constructions we assume $\mathcal{C} = (\mathsf{Setup}, \mathsf{Conceal}, \mathsf{Open})$ already achieves hiding, and let $\mathcal{H} = \{H\}$ be some hash family whose input size equals to the input size of the hider h of $\mathcal{C}$. Recall that in our schemes we will always have $|h| \approx |m|$ (in fact, exactly equal in the PRG-based scheme), so we expect the input length of $\mathcal{H}$ to be roughly equal to the input length of our message m.

Regular Binding. Here we assume that $\mathcal{H} = \{H\}$ is a family of CRHFs. We turn the given "hiding" concealment $\mathcal{C}$ into $\mathcal{C}' = (\mathsf{Setup}', \mathsf{Conceal}', \mathsf{Open}')$ which is a full fledged concealment scheme as follows:

- $\mathsf{Setup}'(1^k)$: 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 $m = \mathsf{Open}(h, b)$.

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

Proof: Since $h' = h$, we get hiding for free. As for binding, if some $\mathcal{A}$ outputs $b' = b\|t$ and $h_0 \neq h_1$ such that $H(h_0) = H(h_1) = t$, then, in particular, $\mathcal{A}$ outputs a collision (h_0, h_1) for $\mathcal{H}$, contradicting the collision resistance of $\mathcal{H}$. □

As we can see, the output size of H directly contributes to the size of our binder b'. In practical constructions, this output size is proportional to the security parameter k and is independent of the input length n. Since the same is true for the key length of practical symmetric encryption schemes, we get that the size of the binder is optimally proportional to the security parameter k. For example, using AES-based encryption and SHA1-based hash function, $|b'| = 128 + 160 = 288$ bits.

Relaxed Binding. Here we assume that $\mathcal{H} = \{H\}$ is a family of UOWHFs. We turn the given "hiding" concealment $\mathcal{C}$ into $\mathcal{C}'' = (\mathsf{Setup}'', \mathsf{Conceal}'', \mathsf{Open}'')$ which is a full fledged *relaxed* concealment scheme as follows:

- $\mathsf{Setup}'' = \mathsf{Setup}$.
- $\mathsf{Conceal}''(m)$: 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 $m = \mathsf{Open}(h, b)$.

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, assume 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 $H(h_0) = H(h_1) = t$. Since $h_0 = h(m_0)$ is computed independently of H, we can immediately turn this $\mathcal{A}$ into an attacker $\mathcal{A}'$ breaking the UOWHF security of $\mathcal{H}$. $\mathcal{A}'$ will use $\mathcal{A}$ to get m_0, will compute $(h_0, b) \leftarrow \mathsf{Conceal}(m_0)$, and will output the message h_0 as the first colliding message. Upon learning random H, $\mathcal{A}'$ will run $\mathcal{A}$ on inputs $b'' = b\|H(h_0)\|H$ and h_0 to produce the second colliding message $h_1 \neq h_0$. □

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. In theory, this shows that efficient *relaxed* concealments, unlike regular concealments (see Lemma 6), can be built from one-way functions. In practice, the message is less clear. On the one hand, best theoretical constructions of UOWHFs from one-way functions (or even

fixed-length UOWHFs) have key length roughly proportional to $O(k \log |m|)$ [32, 10, 33], which is slightly superlinear in the security parameter k when hashing long messages. For example, hashing 1Gb message would require a binder of at least several kilobytes, which is less desirable. On the other hand, it might be much more reasonable to assume that a given "practical" hash family $\mathcal{H}$ is universal one-way as opposed to collision-resistant. For example, Halevi and Krawczyk [18] gave several efficient methods to construct UOWHFs with short keys (say, 160 bits) using building blocks which are not required (and unlikely!) to be collision-resistant. Thus, it seems reasonable that one might be able to construct a UOWHF family whose output *plus key size* might be comparable to (or perhaps only slightly larger than) the best reasonable output of a CRHF, and yet rely on a provably weaker assumption!

Super-Relaxed Binding. To construct super-relaxed concealments, we use exactly the same construction $\mathcal{C}''$ as above, except we only need to assume that the hash family $\mathcal{H} = \{H\}$ is a family of AUHFs.

Lemma 4. *If $\mathcal{C}$ satisfies the hiding property and $\mathcal{H}$ is a* AUHF*, then $\mathcal{C}''$ is a super-relaxed concealment scheme.*

Proof: The proofs is the same as of Lemma 3, except we observe that the attacker $\mathcal{A}$ never learns the value H when breaking the super-relaxed binding. This is because $\mathcal{A}$ is only given the value $h'' = h$ which does not include H. Thus, $\mathcal{A}$ effectively produces a collision pair (h_0, h_1) without having any information about H, contradicting the AUHF security of $\mathcal{H}$. □

It is known that one can construct AUHFs unconditionally. For example, the classical polynomial interpolation construction (see [12] for some history) splits the n-bit message into blocks of size v, and evaluates the resulting degree n/v polynomial at a random point in $GF[2^v]$ (where this point is the key for H). This construction achieves binding security $n/(v2^v)$,[7] and has the key and output size equal to v. For example, if the length of the message and the hider h is 1Gb, to achieve security 2^{-80} it is sufficient to set $v = 106$. Thus, using this construction with AES-based encryption, the final length of the binder $b'' = \tau \| H(h) \| H$ is $128 + 106 + 106 = 340$ bits, which is quite reasonable for a 1Gb message, and is only 52 bits longer than the SHA1-based construction (which required to assume the collision-resistance of SHA1, and certainly did not achieve even "conditional" binding security 2^{-80}).

Collecting Pieces Together. To summarize, we achieved the following constructions of concealment schemes. For the hider, all schemes set $h = \mathsf{E}_\tau(m)$, where E is a one-time secure encryption scheme (such as $\mathsf{E}_\tau(m) = m \oplus G(\tau)$, where G is a PRG, or any block-cipher based semantically secure encryption, such as CBC or CFB). For the binder, for regular binding we set $b = \tau \| H(h)$, where H is chosen from a family of CRHFs, while for the

[7] Meaning that the maximal probability two unequal messages collide under a random H is at most $\frac{n}{v2^v}$.

relaxed/super-relaxed binding we could make weaker assumptions by setting $b = \tau \| H(h) \| H$ and assuming H is chosen from a family of UOWHFs/AUHFs. In particular, using the fact the existence of CRHFs or UOWHFs implies the existence of one-way functions, and, hence, of one-time secure symmetric encryption, we get:

Theorem 1. *The hiding property of of regular/relaxed/super-relaxed concealment can be based on the existence of one-way functions (which is implied by the existence of* CRHF*). The binding property of regular/relaxed/super-relaxed concealments can be based on the existence of* CRHF*/one-way functions/no assumptions.*

As we will see in the next subsection, the above theorem is tight in terms of the minimal assumptions required.

Comparing to OAEP. Recall, the *Optimal Asymmetric Encryption Padding* (OAEP) [9] is a popular padding scheme used in designing various encryption and signature schemes based on trapdoor permutations. It picks a random value τ and sets $h = G(\tau) \oplus m$, $b = \tau \oplus H(h)$, where G and H are hash function typically modeled as random oracles in the analysis. This construction is very similar to the particular concealment construction we had above, except we set $b = \tau \| H(h)$. Namely, our construction is slightly more "redundant" in terms of the binder b. However, this "redundancy" is precisely makes it a secure concealment scheme. Indeed, OAEP decoding never outputs $\perp$, since it is a permutation over m and τ; thus, OAEP does not achieve any binding. What is interesting, though, is that our construction — which is so similar to the OAEP — does not need to assume G and H as random oracles in the analysis!

3.3 Necessity of Assumptions

We show that the assumptions of Lemma 1, Lemma 2 and Lemma 3 (and hence those of Theorem 1) are not only sufficient, but also necessary. We start by showing that achieving non-trivial hiding requires one-way functions, as stated in Lemma 1.

Lemma 5. *If $\mathcal{C}$ is non-trivial (i.e., $|b| < |m|$) and satisfies the correctness and hiding properties of concealment, then one-way functions exist.*

Proof: We use the result of Impagliazzo and Luby [19] who showed that "non-trivial, one-time secure interactive encryption" (NOTE) implies the existence of one-way functions. Here NOTE refers to any interactive protocol between Alice and Bob, who are connected by a public channel P and a secure channel S such that: (a) at the end of the protocol Alice transmits the message m to Bob; (b) Eve, who is passive and only observes all the communication over the public channel P, gets no information (in the usual sense of semantic security) about m; and (c) the total length of messages exchanged over the

secure channel S (not observed by Eve) is strictly less than the length of the message m.

Thus, it suffices to show that non-trivial concealment satisfying correctness and hiding imply a NOTE scheme. But this is simple. Alice, on input m, runs the concealment scheme to obtain $(h, b) \leftarrow \mathsf{Conceal}(m)$, and send b over the secure channel S, and h over the public channel P. Bob can recover m from b and h (by correctness), the length of b is shorter than the length of m (by non-triviality of $\mathcal{C}$), and h observed by Eve reveals no information about m (by hiding). □

Next, we show the necessity of using CRHFs/UOWHFs for ensuring regular/relaxed binding of our constructions.

Lemma 6. *Let $\mathcal{C} = (\mathsf{Setup}, \mathsf{Conceal}, \mathsf{Open})$ be a regular (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* CRHF*s (resp.* UOWHF*s).*

Proof: If $\mathcal{C}$ is a regular 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 $\mathsf{Open}_{\mathsf{CK}}(h_0, b) = m_0 \neq \perp$, $\mathsf{Open}_{\mathsf{CK}}(h_1, b) = m_1 \neq \perp$ 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 a 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. □

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 of concealments, while authenticated encryption of the binder b provides authenticity due to the binding property. (The exact type of the required binding will depend on the particular setting, as explained later.)

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 (super-)relaxed concealments are used (see below for the details). Second, we show that this paradigm also gives a very simple and general solution to *remotely keyed* authenticated encryption. Here, the full power of regular binding 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* [36]). 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" [8]. 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, before [17] 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, "insider attacks" and "identity fraud" (see [3]). Therefore, we first give the details of the symmetric-key setting, and then briefly sketch the changes required in the public-key setting.

Symmetric-Key Syntax. A symmetric-key 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 $\mathcal{M}$, 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 $m \in \mathcal{M}$.

Symmetric-Key Security. Fix the sender S and the receiver R. Following the standard security notions [8], we define the attack models and goals of the

adversary for both authenticity (i.e. sUF-CMA)[8] and privacy (IND-CCA2)[9] 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.[10] 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^*. The scheme is IND-CCA2-secure if for any PPT $\mathcal{A}$, $\Pr[\mathcal{A} \text{ succeeds}] \leq \frac{1}{2} + \mathsf{negl}(k)$.

Remark 1. We also remark that IND-CPA-security[11] is the same, except $\mathcal{A}$ is not given access to the authdecryption oracle. Moreover, in the symmetric-key setting it is known that IND-CPA+sUF-CMA security implies IND-CCA2-security [8]. However, since this implication does not hold in the public-key setting, discussed next, we do not follow this route in the symmetric-key setting.

Public-Key Syntax. For convenience, we will use almost the same syntax as before, with the following modifications. The key generation algorithm $\mathsf{KG}(1^k)$ run by user U now outputs the public verification/encryption key VEK_U and the secret signing/decryption key SDK_U for U. The randomized *authencryption* (authenticate/encrypt) algorithm AE, run by the sender S to compose a ciphertext to the receiver R, takes as input the secret key SDK_S of S, the public key VEK_R of R and a message m from the associated message space $\mathcal{M}$, internally flips some coins and outputs a ciphertext c; we write $c \leftarrow \mathsf{AE}_{\mathsf{SDK}_S}(m, \mathsf{VEK}_R)$ or simply $c \leftarrow \mathsf{AE}(m)$, when the identities of S and R are clear. The deterministic *authdecryption* (verify/decrypt) algorithm AD takes as input the secret key SDK_R or R, and the public key VEK_S of S and outputs $m \in \mathcal{M} \cup \{\perp\}$, where $\perp$ indicates that the input ciphertext c is "invalid". We

[8] Meaning "strong unforgeability against chosen message attack."
[9] Meaning "indistinguishability against chosen ciphertext attack."
[10] A slightly weaker notion of UF-CMA requires C to correspond to "new" message m not submitted to $\mathsf{AE}_K(\cdot)$.
[11] Meaning "indistinguishability against chosen plaintext attack."

write $m \leftarrow \mathsf{AD}_K(c)$ or simply $m \leftarrow \mathsf{AD}(c)$ (again, omitting the keys of S and R, when clear). We require that $\mathsf{AD}_{\mathsf{SDK}_R}(\mathsf{AE}_{\mathsf{SDK}_S}(m, \mathsf{VEK}_R), \mathsf{VEK}_S) = m$, for any $m \in \mathcal{M}$.

Public-Key Security. The security is defined similarly to the symmetric-key setting, except there are several flavors now because the sender S and the receiver R now have difference secret keys. We refer the reader to [3] for the discussing of some of those flavors, here only discussing the distinction between *outsider-security* and *insider security*. Informally, in the outsider-security setting the attacker tries to break privacy or authenticity of two honest users S and R communicating between each other, by posing as a legitimate outsider party to either S or R. In contrast, in the insider-security setting the attacker tries to break privacy or authenticity or an honest user U by "posing" as a valid sender or recipient to U. Thus, insider security is a stronger notion, but may not be required in some applications.

- *Outsider Security.* This setting is very similar to the definition in the symmetric-key setting. We fix sender S and receiver R, and give their public keys VEK_S and VEK_R to the attacker $\mathcal{A}$. As before, $\mathcal{A}$ has oracle access to the functionalities of both S and R. Specifically, it can mount a chosen message attack on S (or, similarly, R) by asking S to produce a ciphertext C of an arbitrary message m to *an arbitrary public key* $\mathsf{VEK}_{R'}$ (possibly, $R' = R$, but this is not necessary!); i.e. $\mathcal{A}$ has access to the *authencryption oracle* $\mathsf{AE}_{\mathsf{SDK}_S}(\cdot, \cdot)$. Similarly, it can mount a chosen ciphertext attack on R (or, similarly, on S) by giving R any candidate ciphertext C for *an arbitrary public key* $\mathsf{VEK}_{S'}$ (possibly $S' = S$, but this is not necessary!) and receiving back the message m or $\perp$; i.e. $\mathcal{A}$ has access to the *authdecryption oracle* $\mathsf{AD}_{\mathsf{SDK}_R}(\cdot, \cdot)$.

  To break the sUF-CMA security in the outsider setting, $\mathcal{A}$ has to be able to produce a "valid" ciphertext C (i.e., $\mathsf{AD}_{\mathsf{SDK}_R}(C, \mathsf{VEK}_S) \neq \perp$), which was not returned earlier by the authencryption oracle of S. Notice, $\mathcal{A}$ is not required to "know" the corresponding message m 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 in the outsider setting, $\mathcal{A}$ first has to to come up with two messages m_0 and m_1. One of these will be authencrypted at random from S to R, the corresponding ciphertext $C^* \leftarrow \mathsf{AE}_{\mathsf{SDK}_S}(m_\sigma, \mathsf{VEK}_R)$ (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^*, VEK_S). The scheme is IND-CCA2-secure if for any PPT $\mathcal{A}$, $\Pr[\mathcal{A} \text{ succeeds}] \leq \frac{1}{2} + \mathsf{negl}(k)$.
- *Insider Security.* In this setting the attacker attacks a single user U. As usual, $\mathcal{A}$ is given VEK_U and can mount chosen message attack (with arbitrary public keys VEK_R) and chosen ciphertext attack (with arbitrary public keys VEK_R) on U. To break the sUF-CMA security in the insider setting, $\mathcal{A}$ has to be able to produce a presumed recipient R, by providing a valid key pair $(\mathsf{SDK}_R, \mathsf{VEK}_R)$, and a "valid" ciphertext C (i.e.,

$\mathsf{AD}_{\mathsf{SDK}_R}(C, \mathsf{VEK}_U) \neq \perp$), which was not returned earlier by the authencryption oracle of U. Notice, $\mathcal{A}$ is not required to "know" the corresponding message m when producing C. Also, $\mathcal{A}$ only needs to "know" the secret key SDK_R for the forged ciphertext C, but not during his chosen message attack. The scheme is sUF-CMA-secure if for any PPT $\mathcal{A}$, $\Pr[\mathcal{A} \text{ succeeds}] \leq \mathsf{negl}(k)$. Similarly, to break the IND-CCA2 security in the outsider setting, $\mathcal{A}$ first has to to come up with a presumed recipient S, by providing a valid key pair $(\mathsf{SDK}_S, \mathsf{VEK}_S)$, and two messages m_0 and m_1. One of these will be authencrypted at random from S to U (using the provided SDK_S), the corresponding ciphertext $C^* \leftarrow \mathsf{AE}_{\mathsf{SDK}_S}(m_\sigma, \mathsf{VEK}_U)$ (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 U to authdecrypt the challenge (C^*, VEK_S). Also, during his chosen ciphertext attack, $\mathcal{A}$ only needs to know the secret key for the challenge sender S, but not for the other senders. The scheme is IND-CCA2-secure if for any PPT $\mathcal{A}$, $\Pr[\mathcal{A} \text{ succeeds}] \leq \frac{1}{2} + \mathsf{negl}(k)$.

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 $|m| \gg |b|$. We start with the symmetric-key setting, and later generalize to the public-key setting.

Symmetric-Key Setting. We will employ the following 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 $m \rightarrow (h, b)$ so that the resulting authenticated encryption is secure? As shown by Alt [1] (correcting the prior claim of [17]), the necessary and sufficient condition is to have the transformation above be a *super-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 $\mathsf{Open}(h, \mathsf{AD}(c))$. Then

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

Proof Sketch: For one easy 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 super-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 h is 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 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 $\mathcal{AE}$.

The other (interesting) direction was formally proven in [1, 17]. Here, we only give an informal intuition. For sUF-CMA-security, by the assumed sUF-CMA-security of $\mathcal{AE}$, the only way $\mathcal{A}$ can break sUF-CMA-security of $\mathcal{AE}'$ is by "reusing" some prior ciphertext $c = \mathsf{AE}(b)$ returned (together with h) by the authencryption oracle. Since $\mathcal{AE}$ is semantically secure, the value c does not give $\mathcal{A}$ any more information about b than what $\mathcal{A}$ can deduce from h alone.[12] Thus, if $\mathcal{A}$ outputs a forgery (c, h'), for some $h' \neq h$, then $\mathcal{A}$ effectively broke the relaxed binding property of $\mathcal{C}$. The IND-CCA2-security is proven similarly. First, sUF-CMA-security above implies that only IND-CPA-security of $\mathcal{AE}'$ needs to be proven [8]. The the latter trivially follows from the IND-CPA-security of $\mathcal{AE}$ and the hiding property of $\mathcal{C}$. □

Public-Key Setting. We generalize the above composition paradigm as follows. $\mathcal{AE}'$ for user U will utilize the same public/secret-key pair $(\mathsf{VEK}_U, \mathsf{SDK}_U)$ as $\mathcal{AE}$. To authencrypt m from S to R, first split it into two pieces (h, b) (so that the transformation is invertible), and output $\mathsf{AE}'_{\mathsf{SDK}_S}(m, \mathsf{VEK}_R) = \langle \mathsf{AE}_{\mathsf{SDK}_S}(b, \mathsf{VEK}_R), h \rangle$. As earlier, we are asking is what are the necessary and sufficient conditions on the transformation $m \to (h, b)$ so that the resulting public-key authenticated encryption is secure? We get a slightly different answer depending on whether we are interested in the outsider or the insider security. As shown by Alt [1] (slightly correcting the prior claim of [17]), the necessary and sufficient condition for outsider/insider security is to have the transformation above be a *super-relaxed/relaxed concealment*.

Theorem 3. *If $\mathcal{AE}$ is secure, then $\mathcal{AE}'$ is outsider/insider secure if and only if $\mathcal{C}$ is a super-relaxed/relaxed concealment scheme.*

Proof Sketch: The outsider security proof is essentially identical to the symmetric-key setting considered in Theorem 2, because the outsider public-key security is very similar to the symmetric-key security.

As for the insider security, we only sketch why *relaxed* binding is required, referring to [1, 17] for the full proof. This is because, when trying to forge a ciphertext from the target user user U to some receiver R, $\mathcal{A}$ can know the secret key SDK_R of R. More precisely, if $\mathcal{C}$ does not satisfy the relaxed

[12] The formalization of this claim is somewhat subtle; see [1].

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 (h, b) is 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 $\mathcal{A}'$ (attacking user U) for $\mathcal{AE}'$ using $\mathcal{A}$. First, $\mathcal{A}$ honestly generates keys $(\mathsf{SDK}_R, \mathsf{VEK}_R)$ for some receiver R. Then, $\mathcal{A}'$ gets m from $\mathcal{A}$, and asks its authencryption oracle the value $\mathsf{AE}'(m, \mathsf{VEK}_R)$. $\mathcal{A}'$ gets back (h, c), where c is a valid authencryption of b, and (h, b) is a random concealment pair for m. Using SDK_R, $\mathcal{A}$ retrieves the value b from c (this is the key difference from the outsider setting!), 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 "fresh" (different from (h, c)) authencryption of some valid message from U to R, contradicting the sUF-CMA-security of $\mathcal{AE}$. □

4.3 Remotely Keyed Authenticated Encryption

Similarly to the previous section, we first consider the symmetric-key setting, and then briefly sketch the extension to the public-key setting.

Symmetric-Key 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 Start-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, Finish-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 Start-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, Finish-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 IND-CCA2 and sUF-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 RK-IND-CCA and RK-sUF-CMA, respectively.

To break the RK-sUF-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 $C_1 \dots C_{t+1}$ 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 RK-sUF-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 $C_1 \dots C_t$ with t oracle calls to Card-AE, by simply following to honest authencryption protocol on arbitrary messages $m_1 \dots m_t$. Thus, security against "one-more forgery" is the most ambitious goal we can try to meet in the setting of RKAE.

To break the RK-IND-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 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, Start-AD is a deterministic algorithm). The latter restriction is to prevent $\mathcal{A}'$ from trivially authdecrypting the challenge. The scheme is RK-IND-CCA-secure if for any PPT $\mathcal{A}'$, $\Pr[\mathcal{A}' \text{ succeeds}] \leq \frac{1}{2} + \mathsf{negl}(k)$. We briefly remark

that RK-IND-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 natural question we address is what security properties of Conceal and Open are needed in order to achieve a secure canonical RKAE (provided the auxiliary $\mathcal{AE}$ is secure)? As shown by Dodis and An [17], the necessary and sufficient condition is to employ a secure (regular) 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 that (super-)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 concealments.

Theorem 4. *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 (regular) concealment scheme.*

Proof Sketch: The proof of this result is very similar to that of Theorem 2, and is omitted. We only mention why regular binding is necessary. If $\mathcal{A}$ can come up with a triple (b, h, h') such that $\mathsf{Open}(h, b) \neq \perp$, $\mathsf{Open}(h', b) \neq \perp$ and $h \neq h'$, then we can construct $\mathcal{A}'$ breaking RK-sUF-CMA-security of $\mathcal{AE}'$ as follows. $\mathcal{A}'$ asks the card to authencrypt the value b, gets back the ciphertext c, and outputs two valid ciphertexts $\langle c, h \rangle \neq \langle c, h' \rangle$. □

Comparison to Previous RKAEs. We briefly compare our scheme with those of [15, 22]. First, both schemes could be put into our framework by ex-

tracting appropriate concealment schemes. In fact, the concealment we extract from [15] 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. To summarize, the construction of [15] 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 [15]. As for the scheme of [22], 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.

Using A Block Cipher in Place of $\mathcal{AE}$. So far we considered schemes of the form $\langle \mathsf{AE}(b), h \rangle$, where AE is an authenticated encryption of short messages. While authenticated encryption is gaining popularity, in the symmetric-key setting block ciphers are much more popular. Moreover, a secure block cipher, — formally known as a (strong) pseudorandom permutation, — is "almost" a secure authenticated encryption, except it does not provide semantic security (but gives at least one-wayness). Therefore, in the symmetric setting it is natural to consider constructions of the form $\mathsf{AE}'(m) = \langle P_K(b), h \rangle$, where we replace the "inner" authenticated encryption AE by a block cipher P. This is especially relevant in the setting of RKAE, where the above scheme would mean that the Card is simply an implementation of a block cipher! As shown by [17], while general concealments might not be enough for such a replacement, the main scheme from Section 3 works in the cases of interest!

Specifically, consider the scheme $h = \mathsf{E}_\tau(m)$, $b = \tau \| H(h)$, where H is collision-resistant and E is one-time secure. Assume also that H is *preimage-resistant* — meaning that it is hard to find a preimage $v \in H^{-1}(r)$ of a random value r. We note that any CRHF $\{H\}$ with $|H(h)| < |h| - \omega(\log k)$ must be preimage-resistant. However, preimage-resistance is usually required anyway when constructing practical hash functions. Finally, assume E is *key-one-way*, meaning that for *any* message m, it is hard to recover the key τ from the ciphertext $\mathsf{E}_\tau(m)$. Once again, this property holds for the PRG-based scheme $\mathsf{E}_\tau(m) = G(\tau) \oplus m$ as long as $|m| = |G(\tau)| > |\tau| + \omega(\log k)$, and also for standard block-cipher-based schemes, such as CBC. Then

Theorem 5. *If (P, P^{-1}) is a strong pseudorandom permutation,* E *is one-time secure and key-one-way, and H comes from a family of preimage-resistant*

$\mathsf{CRHF}s$, *then* $\mathsf{AE}'_K(m) = \langle P_K(\tau \| H(h)), h = \mathsf{E}_\tau(m) \rangle$ *gives rise to a secure* RKAE.[13]

Extension to the Public-Key Setting. This extension (with the exception of replacing authenticated encryption by a block cipher, which makes no sense in the public-key setting) is pretty straightforward. In fact, unlike the question of domain extension studied in Section 4.2, no new subtleties arise in the public-key setting. Namely, when building "long" authenticated encryption from a "short" authenticated encryption, regular concealments are necessary and sufficient to maintain either the insider, or the outsider security.

References

1. S. Alt, "Authenticated Hybrid Encryption for Multiple Recipient," Available at *Eprint Archive*, `http://eprint.iacr.org/2006/029`, 2006.
2. 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.
3. 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.
4. J. Baek, R. Steinfeld, and Y. Zheng, "Formal proofs for the security of signcryption," In *J. of Cryptology*, 20(2):203–235, 2007. Conference version in *PKC '02*, pp. 80–98, LNCS Vol. 2274, 2002.
5. M. Bellare, R. Canetti and H. Krawczyk, "Keying hash functions for message authentication," In *Crypto '96*, pp. 1–15, LNCS Vol. 1109, 1996.
6. 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.
7. M. Bellare, T. Kohno, C. Namprempre, "Provably Fixing the SSH Binary Packet Protocol," In *Proc. 9th CCS*, pp. 1–11, ACM, 2002.
8. 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.
9. M. Bellare and P. Rogaway, "Optimal asymmetric encryption – How to encrypt with RSA," In *Eurocrypt '94*, pp. 92–111, LNCS Vol. 950, 1994.
10. M. Bellare and P. Rogaway, "Collision-Resistant Hashing: Towards Making UOWHFs Practical," In *Crypto '97*, pp. 470–484, LNCS Vol. 1294, 1997.
11. 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.
12. D. Bernstein, "The Poly1305-AES Message-Authentication Code," In *FSE*, pp. 32–49, 2005.

[13] Clearly, this also means that this is a secure way to build a "long" authenticated encryption from a *single call* to a block cipher. In fact, preimage-resistance of H and key-one-wayness of E are not needed in this case.

13. 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.
14. M. Blaze, "High-Bandwidth Encryption with Low-Bandwidth Smartcards," In *Fast Software Encryption (FSE) '96*, pp. 33–40, LNCS Vol. 1039, 1996.
15. M. Blaze, J. Feigenbaum, M. Naor, "A Formal Treatment of Remotely Keyed Encryption," In *Eurocrypt '98*, pp. 251–265, LNCS Vol. 1403, 1998.
16. I. Damgård, "Collision free hash functions and public key signature schemes," In *Eurocrypt '87*, pp. 203–216, LNCS Vol. 304, 1987.
17. 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.
18. S. Halevi and H. Krawczyk, "Strengthening Digital Signatures Via Randomized Hashing," In *Crypto '06*, pp. 41–59, 2006, LNCS Vol. 4117.
19. R. Impagliazzo, M. Luby, "One-way Functions are Essential for Complexity Based Cryptography," In *FOCS '89*, pp. 230–235, 1989.
20. 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.
21. C. Jutla, "Encryption modes with almost free message integrity," In *Eurocrypt '01*, pp. 529–544, LNCS Vol. 2045, 2001.
22. M. Jakobsson, J. Stern, and M. Yung, "Scramble All, Encrypt Small," In *Fast Software Encryption (FSE) '99*, pp. 95–111, LNCS Vol. 1636, 1999.
23. J. Katz and M. Yung, "Unforgeable Encryption and Chosen Ciphertext Secure Modes of Operation," In *FSE '00*, pp. 284–299, LNCS Vol. 1978, 2000.
24. 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.
25. S. Lucks, "On the Security of Remotely Keyed Encryption," In *Fast Software Encryption (FSE) '97*, pp. 219–229, LNCS Vol. 1267, 1997.
26. S. Lucks, "Accelerated Remotely Keyed Encryption," In *Fast Software Encryption (FSE) '99*, pp. 112–123, LNCS Vol. 1636, 1999.
27. A. Menezes, P. van Oorshot and S. Vanstone, "Handbook of applied cryptography," CRC Press LLC, 1997.
28. M. Naor, "Bit Commitment Using Pseudorandomness," In *Journal of Cryptology*, 4(2):151–158, 1991.
29. M. Naor and M. Yung, "Universal One-Way Hash Functions and their Cryptographic Applications," In *Proc. 21st STOC*, pp. 33–43, ACM, 1989.
30. P. Rogaway, "Authenticated-Encryption with Associated-Data," In *Proc. 9th CCS*, pp. 98–107, ACM, 2002.
31. 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.
32. J. Rompel, "One-way functions are necessary and sufficient for secure signatures," In *Proc. 22nd STOC*, pp. 387–394, ACM, 1990.
33. V. Shoup, "A composition theorem for universal one-way hash functions," In *Eurocrypt '00*, pp. 445–452, LNCS Vol. 1807, 2000.
34. 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.

35. D. Stinson, "Universal Hashing and Authentication Codes," *Designs, Codes and Cryptography*, 4:369–380, 1994.
36. 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.