Folklore, Practice and Theory of Robust Combiners
Draft of November 29, 2007

Amir Herzberg

Computer Science Department, Bar Ilan University, Ramat Gan, Israel
herzbea@cs.biu.ac.il
http://AmirHerzberg.com
November 29, 2007

Abstract. Cryptographic schemes are often designed as a combination of multiple component cryptographic modules. Such a combiner design is *robust* for a (security) specification if it meets the specification, provided that a sufficient subset of the components meet their specifications. A folklore combiner for encryption is *cascade*, i.e. $c = \mathcal{E}''_{e''}(\mathcal{E}'_{e'}(m))$. We show that cascade is a robust combiner for cryptosystems, under three important indistinguishability specifications: chosen plaintext attack (IND-CPA), non-adaptive chosen ciphertext attack (IND-CCA1), and replayable chosen ciphertext attack (IND-rCCA). We also show that cascade is not robust for the important specifications adaptive CCA (IND-CCA2) and generalized CCA (IND-gCCA). The IND-rCCA and IND-gCCA specifications are closely related, and this is an interesting difference between them. All specifications are defined within.
We also analyze few other basic and folklore combiners. In particular, we show that the following are robust combiners: the *parallel combiner* $f(x) = f''(x)||f'(x)$ for one-way functions , the *XOR-Input combiner* $c = (\mathcal{E}''_{e''}(m \oplus r), \mathcal{E}'_{e'}(r))$ for cryptosystems, and the *copy combiner* $f_{k'',k'}(m) = f''_{k''}(m)||f'_{k'}(m)$ for integrity tasks such as Message Authentication Codes (MAC) and signature schemes. Cascade is also robust for the hiding property of commitment schemes, and the copy combiner is robust for the binding property, but neither is a robust combiner for both properties. We present (new) robust combiners for commitment schemes; these new combiners can be viewed as a composition of the cascade and the copy combiners. Our combiners are simple, efficient and practical.

Keywords: applied cryptography, robust combiners, foundations of cryptography, encryption schemes, cascade ciphers, hash functions, commitment schemes

1 Introduction

The security of a cryptographic scheme is often established by the accumulated evidence of failure of experts to cryptanalyze the scheme, i.e. to find vulnerabilities of the scheme. However, cryptanalysis is an expensive, time-consuming and fallible process. In particular, since a seemingly-minor change in a cryptographic function may allow an attack which was previously impossible, cryptanalysis allows only validation of specific functions and development of engineering principles and attack methodologies and tools, but does not provide a solid theory for designing cryptographic functions. Indeed, estimating the rate of future cryptanalytical successes is difficult, see e.g. [LV01]. Prudent designers use safety margins to allow for unexpected breakthroughs; however, there is often resistance to replace widely deployed standards which were not broken yet, 'just' since the safety margins were eroded.

This motivates the usage, and study, of *robust combiners* for cryptographic mechanisms. A *robust combiner*[1] combines several cryptographic modules, such that the combined mechanisms is secure even if some of the modules turn out to be insecure. In particular, the robust

[1] In earlier versions of this document [Her02], we used the term *tolerant design* instead of *robust combiner*. In this version, we adopted *robust combiner*, coined by Harnik et al. in [HKN+05].

combiner remains secure following successful cryptanalysis of some of its modules, refutation of one or few of the assumptions underlying its security (e.g. the assumption that factoring is a hard problem), or attempts to exploit a vulnerability in some of the modules, due to implementation error, design errors, or even an intentional trapdoor in the design or implementation. Using a robust-combiner does not guarantee security; however, it would hopefully provide sufficient advance-warning time to replace the broken cryptographic modules.

Many cryptographic systems combine redundant components in hope of achieving robustness (tolerance). There are several known (often folklore) combiners of cryptographic modules, often with the stated or implicit goal of achieving robustness (tolerance) or even improving security. Possibly the most well known combiner is the *cascade combiner*, applied e.g. to block ciphers and encryption schemes. Cascading of cryptosystems and ciphers has been a common practice in cryptography for hundreds of years.

However, there were not many publications, prior to this work, analyzing robust cryptographic combiners. We believe the earliest paper proposing a robust combiner is [AB81], where Asmuth and Blakely present the simple 'XOR-Input' combiner of two cryptosystems, $c = \left(\mathcal{E}''_{e''}(m \oplus r), \mathcal{E}'_{e'}(r)\right)$. In section 5.3, we show that this combiner is robust for the IND-CPA and IND-CCA1 specifications, but not for the IND-CCA2 specification.

Even before [AB81], *cascade* was known and used, as 'folklore', with the hope of creating more secure cryptosystems. Cascade encryption refers to the sequential application of two cryptosystems $\mathcal{E}''$ and $\mathcal{E}'$ as in $c = \mathcal{E}''_{e''}(\mathcal{E}'_{e'}(m))$. Cascade was also studied in several works, however mostly for block ciphers rather than for a 'complete' cryptosystem. Even and Goldreich [EG85] showed that cascade of block ciphers is a robust combiner for security against message recovery attacks. Damgard and Knudsen [DK94] proved that cascade of block ciphers is also a robust combiner for security against chosen-plaintext key-recovery attack. Maurer and Massey [MM93] showed that, in a weaker attack model, cascade of block ciphers is at least as strong as the first cipher in the cascade. They also demonstrate that in their (weaker) attack model, cascade is *not* robust combiner of block ciphers, by demonstrating a cascade which is weaker than the second cipher; we show that cascade is a robust combiner, but for encryption schemes rather than block ciphers, and using the standard indistinguishability specifications. Other works investigated and showed improvement in security due to cascading of block ciphers; in particular Bellare and Rogaway [BR06] showed that in the ideal-cipher model, cascading of three block cipers ('triple encryption') significantly improves security.

We return to the 'folklore' cascade of cryptosystems, rather than of block ciphers. We show that cascade is a robust combiner of cryptosystems, for a wide range of indistinguishability (confidentiality) specifications, including indistinguishability under chosen-plaintext attack (IND-CPA), non-adaptive chosen ciphertext attack (IND-CCA1), and Replayable CCA attack (IND-rCCA[2]) [CKN03]. This confirms the folklore belief, that cascade is a robust combiner for encryption.

On the other hand, we observe that cascade of cryptosystems is *not a robust combiner* for indistinguishability under adaptive chosen ciphertext attack (IND-CCA2). Furthermore, we show that cascade is *not a robust combiner* also for indistinguishability under the generalized CCA (IND-gCCA) [ADR02], which, like IND-rCCA, is weaker than IND-CCA2 but arguably sufficient for most applications. This is an interesting difference between the IND-rCCA and IND-gCCA specifications, which have similar motivations.

[2] IND-rCCA is weaker than IND-CCA2, but it is arguably a sufficient requirement for most applications. In particular it allows the practical 'feedback only CCA' attacks of [Ble98,Kra01].

Other robust combiners for cryptosystems are known. Dodis and Katz [DK05] showed a combiner for cryptosystems, which is robust for the IND-CCA2 specification. Note also that Shoup presented in [Sho00] a cryptosystem, which is secure as long as one of a pair of assumptions hold; this is an alternative approach to gain confidence in a cryptographic scheme, via redundancy of assumptions (rather than redundancy of schemes, as with robust combiners).

Cascade and other combiners for encryption were also studied by Zhang et al. [ZHSI04]. Instead of using the general definition of a robust combiner, they considered the combined encryption schemes as implementations of a new cryptographic primitive they defined, *multiple encryption*. Cascade of IND-CCA2 cryptosystems is not necessarily 'multiple-encryption CCA2-secure'; however, cascade of IND-gCCA cryptosystems is 'multiple-encryption gCCA-secure', as defined in [ZHSI04]. However, we show that cascade is *not* a robust combiner for IND-gCCA. This is an interesting difference between the approach of using the robust combiner definition, and the approach of adapting the security definition as in the multiple-encryption specifications of [ZHSI04].

Robust combiners are useful for any cryptographic scheme, not just for block ciphers and encryption. In particular, we show that the 'folklore' *copy* combiner $g(x)\|f(x)$, using the same input x to both functions, is robust for several integrity properties, such as (several variants of) collision-resistant hashing, Message Authentication Codes (MAC) and digital signatures. The copy combiner is also used, as 'folklore', in practical designs and standards, e.g. in the W3C XML-DSIG specifications and in the TLS protocol [DA99].

Efficiency is critical for practical robust combiners; implementors will rarely be willing to tolerate significant performance loss, 'just' in order to tolerate potential vulnerabilities in a cryptographic function. Most combiners we present and analyze are efficient and practical.

Ignoring efficiency, there is a simple robust combiner for many cryptographic schemes, by using provable combiners of cryptographic mechanisms from few 'basic' cryptographic mechanisms such as one-way function, which have simple robust designs. Specifically, the simple *parallel* combiner $f||f'(< x, x' >) = f(x)||f'(x')$ is robust for the one-way function specification. Namely, it is sufficient that one of $\{f, f'\}$ is a one-way function, to ensure that $f||f'$ is also a one-way function. Furthermore, there are reductions, provably secure under asymptotic (poly-time) definitions, of many cryptographic mechanisms from one-way functions and vice versa, e.g. pseudo-random generators [Gol01,HILL99] and signature schemes [Rom90,NY89]. It follows that there are robust combiners for all of these tasks. However, these combiners are robust only for asymptotic (poly-time) definitions, and involve substantial degradation in efficiency and security parameters, e.g., require absurd key and/or block sizes [Gol01,HL92].

Subsequent work. Following earlier versions of this work [Her02], robust combiners were found for IND-CCA2 encryption in [DK05]. Also, robust combiners were found for other tasks, including oblivious transfer (OT) and key agreement by Harnik et al. [HKN+05]. Meier et al. present an improved robust combiner for OT in [MPW07]. In [MP06], Meier and Przydatek presented robust combiners for private information retrieval (PIR), and for PIR-to-OT and PIR-to-BC (bit commitment). Boneh and Boyen [BB06] showed that non-trivial black-box robust combiners for collision-resistant hash functions (CRHF) do not exist; this was improved by Pietrzak [Pie07]. Robust combiners for commitment schemes, OT and interactive proof systems were also studied by Sommer [Som06].

Our contributions. We identify and define cryptographic tolerance as a criteria for cryptographic specifications, and define robust combiners. We analyze the security of several 'folklore' combiners for several basic cryptographic primitives, including encryption, signa-

tures, MAC, hash functions and commitment schemes. Finally, we present efficient, practical combiners for commitment schemes.

2 Robust Combiners: Definitions

Let $\mathbb{P}$ denote the set of all programs, with a fixed encoding and machine model, e.g. Turing machines. A combiner (of plurality n) is an algorithm c, whose input is a set of n programs $P_1, \ldots, P_n \in \mathbb{P}^n$, and whose output is a single program $c(P_1, \ldots, P_n)$. We say that $c : \mathbb{P}^n \to \mathbb{P}$ is a (k, n)-robust combiner of $\mathbb{P}$ for specification (predicate) $s : \mathbb{P} \to \{0, 1\}$, if:

$$\sum_{i=1}^{n} s(P_i) \geq k \;\Rightarrow s(c(P_1, \ldots, P_n)) = 1$$

When c is a $(n-1, n)$-robust combiner for s, we say that c is a robust combiner for s. When c is a (n, n)-robust combiner for specification s, we say that c is a preserving combiner for s; this can be useful property, e.g. we use it in Section 6.2 to build a robust combiner for both the hiding and the binding specifications of commitment schemes (out of one combiner that is robust for hiding and preserves binding, and another that is robust for binding and preserves hiding).

Harnik et al. [HKN+05] define black-box robust combiners, to exclude designs which ignore the underlying schemes and directly implement the required functionality. All of our constructions are black-box combiners, however, since we only present positive results, we do not present definition and analysis of the black-box property. There are additional possible extensions of the robustness concept, e.g. to consider side channels (e.g. timing, power), to allow a mix of different specifications for the components, or to support a general access structure (rather than threshold as above). We do not discuss such extensions in this paper.

3 Cascade Encryption

A triple $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ of polynomial-time algorithms (for key generation, encryption and decryption, respectively) is an *encryption scheme*, if for every pair $(e, d) \leftarrow \mathcal{K}(1^k)$ and every m in the domain of $\mathcal{E}$ holds: $m = \mathcal{D}_d(\mathcal{E}_e(m))$. For simplicity, restrict $\mathcal{D}$ to be deterministic, while $\mathcal{K}$ and $\mathcal{E}$ are probabilistic polynomial time (PPT) algorithms.

In this section, we investigate the tolerance of *cascade encryption*. Cascade encryption is an ancient, widely-deployed technique. Let $\Pi'' = (\mathcal{K}'', \mathcal{E}'', \mathcal{D}'')$, $\Pi' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ be two encryption schemes, s.t. the domain of $\mathcal{E}''$ contains the range of $\mathcal{E}'$. Without loss of generality, split the random bits evenly between Π'' and Π'.

The *cascade* of $(\mathcal{K}'', \mathcal{E}'', \mathcal{D}'')$ and $(\mathcal{K}', \mathcal{E}', \mathcal{D}')$, denoted $(\mathcal{K}, \mathcal{E}, \mathcal{D}) = (\mathcal{K}'', \mathcal{E}'', \mathcal{D}'') \circ (\mathcal{K}', \mathcal{E}', \mathcal{D}')$, is defined as:

- Given 1^k and $r'', r' \in \{0,1\}^{\rho(k)}$ for some $\rho(k) \in \mathbb{N}$, let $(e', d') = \mathcal{K}'(1^k; r')$, $(e'', d'') = \mathcal{K}''(1^k; r'')$. Then $\mathcal{K}(1^k; r''||r') = ((e'', e'), (d'', d'))$.
- For $r'', r' \in \{0,1\}^{\rho(k)}$, let $\mathcal{E}_{e'',e'}(m; r''||r') = \mathcal{E}''_{e''}(\mathcal{E}'_{e'}(m; r'); r'')$
- $\mathcal{D}_{d'',d'}(c) = \mathcal{D}'_{d'}(\mathcal{D}''_{d''}(c))$

Above and throughout, we indicate random coins used by a probabilistic algorithm only when relevant to the discussion, by listing them following the 'regular inputs', as in $\mathcal{E}_e(m; r)$. We use $a||b$ to denote the concatenation of a and b.

Intuitively, cascading may seem to improve the secrecy of encryption schemes, or at least to ensure robustness (protect confidentiality as the more secure of the cascaded schemes). Is cascading really a robust combiner for encryption schemes? We show that this depends on the security specifications, and mainly the adversary capabilities ('attack model').

In particular, cascade encryption ${\mathcal{E}''}_{e''}({\mathcal{E}'}_{e'}(m))$, is *not robust* for adaptive chosen ciphertext attacks (CCA2). This follows from the following simple example, given by An, Dodis and Rabin [ADR02]: let ${\mathcal{E}''}_{e''}(m) \stackrel{R}{\leftarrow} \{m||0, m||1\}$. The cascade of any encryption scheme $\mathcal{E}'$ with this $\mathcal{E}''$, is not secure against $CCA2$, since given ciphertext $c^* = {\mathcal{E}''}_{e''}({\mathcal{E}'}_{e'}(m)) = {\mathcal{E}'}_{e'}(m)||\beta$ of plaintext message m, where $\beta \in \{0,1\}$, attacker can find m via the legitimate query ${\mathcal{D}'}_{d'}({\mathcal{D}''}_{d''}(c))$, where $c = {\mathcal{E}'}_{e'}(m)||\overline{\beta}$; this query clearly returns m.

An et al. argued in [ADR02], that this example indicates that CCA2 is overly restrictive. Intuitively, encrypting the ciphertext again, even with a weak scheme, cannot expose it. Hence, they defined *generalized CCA (gCCA)*, which is weaker than CCA2 (but is satisfied by ${\mathcal{E}''}_{e''}({\mathcal{E}'}_{e'}(m))$ as in the example above). Other 'relaxed' forms of adaptive chosen ciphertext attacks were proposed, most notably, the weaker notion of *replayable CCA (rCCA)*, by Canetti, Krawczyk and Nielsen [CKN03]. The differences between gCCA and rCCA are somewhat subtle. In this section we show an interesting difference: cascade is a robust combiner for rCCA, but not for gCCA.

Trivially, the cascade $\Pi = \Pi'' \circ \Pi'$ of two encryption schemes $\Pi'' = (\mathcal{K}'', \mathcal{E}'', \mathcal{D}''), \Pi' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ does not satisfy even the (weak) IND-CPA specification, as defined later, if Π' may leak information about the plaintext via the *length* of the ciphertext. Suppose, for example, that for every key e' holds $|{\mathcal{E}'}_{e'}(0)| \neq |{\mathcal{E}'}_{e'}(1)|$. Assume that for every k, m, e'' holds $|{\mathcal{E}''}_{e''}(m)| = |m| + \alpha(k)$, namely the length of the ciphertext is the same as the length of the plaintext, plus a number of bits which depends only on the security parameter k; this holds for most encryption schemes. Clearly, Π does not satisfy IND-CPA. To eliminate this form of 'side channel', we focus on cascade of *length-uniform encryption schemes* $(\mathcal{K}, \mathcal{E}, \mathcal{D})$, where for every security parameter k, keys $(e, d) \in \mathcal{K}(1^k)$ and messages m_0, m_1 s.t. $|m_0| = |m_1|$, holds: $|\mathcal{E}_e(m_0)| = |\mathcal{E}_e(m_1)|$.

In the first subsection below, we unify and extend the game-based definitions of [KY06,BDPR98], in particular extending them to cover the rCCA and gCCA specifications. In the second subsection, we investigate the tolerance of cascade encryption.

3.1 Indistinguishability experiment and specifications

We defined encryption schemes without distinguishing between shared (private) key schemes and public key schemes. In fact, most of our results hold for both types of schemes. The difference between the two, is only in the indistinguishability specifications they satisfy. Essentially, public-key encryption should satisfy indistinguishability, even when the encryption key is given to the attacker. We present a single definition of indistinguishability specifications, which covers both public key and shared key specifications. Our definition combines, therefore, the indistinguishability definitions of [KY06] and [BDPR98].

Specifically, we extend the definition of [KY06]. They already allowed substantial flexibility, by defining the attack as IND-Px-Cy, where $x, y \in \{0, 1, 2\}$ signal the phases at which the attacker is allowed an encryption (plaintext) query, for x, or a decryption (ciphertext) query, for y. We add the value $x = \mathrm{K}$, to signal public-key encryption, which simply means that the adversary is given the encryption key e. Using $x = \mathrm{K}$, gives attack values of IND-

PK-Cy, for $y = 0, 1$ and 2, corresponding to the public-key indistinguishability specifications of IND-CPA, IND-CCA1 and IND-CCA2, respectively.

We further extend the definition, adding more possible values to y, to include two other specifications, which were proposed as alternatives to CCA2. The first is *replayable CCA (rCCA)* [CKN03], which we identify by $y = \mathrm{R}$.

The second CCA2-alternative specification that we consider is *generalized CCA (gCCA)* [ADR02], which is called *benign malleability* in [Sho01]. For gCCA, we use $y = (\mathrm{G}, r)$, where $r_e(c, c')$ is an efficient, keyed relation, which is decryption-preserving for $(\mathcal{K}, \mathcal{E}, \mathcal{D})$. Namely, for every pair of ciphertexts c, c' and every key pair $(e, d) \leftarrow \mathcal{K}(\cdot)$, holds

$$r_e(c, c') = \mathrm{TRUE} \Rightarrow \mathcal{D}_d(c) = \mathcal{D}_d(c')$$

Let $\mathcal{R}(\Pi)$ denote the set of efficiently computable (PPT), keyed decryption-preserving relations $r_e(c, c')$, for $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$. This definition of gCCA and $\mathcal{R}(\Pi)$ makes sense only for public-key cryptosystems. We extend it, to cover also symmetric (shared-key) cryptosystems, by using $r(c, c')$ instead of $r_e(c, c')$. For simplicity, we abuse notations and use $\mathcal{R}(\Pi)$ to refer both to the asymmetric (public-key) variant, as defined above, and to the symmetric (shared-key) variant, where the relation $r(c, c')$ does not depend on the encryption key. Krawczyk [Kra01] defined this symmetric-key variant, referring to it as *loose ciphertext unforgeability.*

Like [KY06,BDPR98], we state the definition with respect to an adversary A which is a pair of PPT algorithms, $A = (A_1, A_2)$. We use $\mathcal{DO}_i$ and $\mathcal{EO}_i$, for $i \in \{1, 2\}$, to denote the decryption and the encryption oracle functions, respectively, for the two phases of the experiment (select phase and guess phase) . When $\mathcal{DO}_i = \epsilon$ or $\mathcal{EO}_i = \epsilon$, the corresponding function returns an empty string ϵ on any input.

Definition 1 (Indistinguishability experiment and security specifications). *Let $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ be an encryption scheme and let $A = (A_1, A_2)$ be a pair of PPT algorithms. For $x \in \{0, 1, 2, \mathrm{K}\}$ and $y \in \{0, 1, 2, \mathrm{R}\} \cup \{\mathrm{G} \times \mathcal{R}(\Pi)\}$ and $k \in \mathbb{N}$, let*

$$\mathbf{Adv}^{\mathrm{IND-P}x\mathrm{-C}y}_{\Pi,A}(k) \equiv 2 \cdot \mathbf{Pr}\left[\mathbf{Exp}^{\mathrm{IND-P}x\mathrm{-C}y-b}_{\Pi,A}(k) = b\right] - 1$$

where the probability is over a random choice of $b \in \{0, 1\}$ and a random choice of coins in the following experiment:

Experiment $\mathbf{Exp}^{\mathrm{IND-P}x\mathrm{-C}y-b}_{\Pi,A}(k)$
- $(e, d) \stackrel{R}{\leftarrow} \mathcal{K}(1^k)$
- $(m_0, m_1, s) \leftarrow A_1^{\mathcal{DO}_1(\cdot), \mathcal{EO}_1(\cdot)}(x, y, 1^k)$ *s.t.* $|m_0| = |m_1|$
- $c^* \leftarrow E_e(m_b)$
- *Return* $A_2^{\mathcal{DO}_2(\cdot), \mathcal{EO}_2(\cdot)}(s, c^*)$

Where we implement the oracles $\mathcal{EO}_i, \mathcal{DO}_i$ as follows, using the state variables of the experiment x, y, e, d, c^*:

$$\mathcal{EO}_1(m) = \begin{cases} \epsilon & \text{if } x = 0 \\ \mathcal{E}_e(m) & \text{if } x \in \{1, 2\} \\ e & \text{if } x = \mathrm{K} \end{cases}$$

$$\mathcal{EO}_2(m) = \begin{cases} \epsilon & \text{if } x \in \{0, 1\} \\ \mathcal{E}_e(m) & \text{if } x = 2 \\ e & \text{if } x = \mathrm{K} \end{cases}$$

$$\mathcal{DO}_1(c) = \begin{cases} \epsilon & \text{if } y = 0 \\ \mathcal{D}_d(c) & \text{otherwise} \end{cases}$$

$$\mathcal{DO}_2(c) = \begin{cases} \mathcal{D}_d(c) \text{ if } \begin{bmatrix} (y = 2 \wedge c \neq c^*) \\ \vee \ (y = \mathrm{R} \wedge \mathcal{D}_d(c) \notin \{m_0, m_1\}) \\ \vee \ (x \neq \mathrm{K} \wedge y = (\mathrm{G}, r) \wedge r(c, c*) \neq \text{TRUE}) \\ \vee \ (x = \mathrm{K} \wedge y = (\mathrm{G}, r) \wedge r_e(c, c*) \neq \text{TRUE}) \end{bmatrix} \\ \epsilon \qquad \text{Otherwise} \end{cases}$$

We say that Π is IND$-$P$x-$Cy*-Secure, if* $\mathbf{Adv}_{\Pi,A}^{\mathrm{IND-P}x-\mathrm{C}y}(\cdot)$ *is a negligible function.*

We can map the IND$-$P$x-$Cy specification, to the well-known IND-CPA, IND-CCA1, IND-CCA2, IND-rCCA and IND-gCCA specifications. Let $\phi =$ PUBLIC if $x =$K, otherwise let $\phi =$ PRIVATE. We say that Π is an IND-CPA (IND-CCA1, IND-CCA2, IND-rCCA) secure ϕ key cryptosystem, if it is IND$-$PK$-$Cy-Secure, for $y = 0$, $y = 1$, $y = 2$ or $y = \mathrm{R}$, respectively.

The IND-gCCA mapping is slightly more complex. We say that Π is IND$-$P$x-$CG-Secure, if for some $r \in \mathcal{R}(\Pi)$, the function $\mathbf{Adv}_{\Pi,A}^{\mathrm{IND-P}x-\mathrm{C(G},r)}(\cdot)$ is negligible, for any polynomial-time A. We say that Π is IND-gCCA-secure public key cryptosystem, if Π is IND$-$PK$-$CG-Secure.

3.2 Cascade Encryption is not Robust for CCA2 and gCCA

The following theorem shows that cascade of encryption schemes is *not* robust combiner, for the IND-gCCA and IND-CCA2 specifications. On the other hand, in the next subsection we show that cascade of encryption schemes *is* $(1,2)-$*robust combiner* for specifications IND-Px-Cy, for $x \in \{0, 1, 2, \mathrm{K}\}$ and $y \in \{0, 1, \mathrm{R}\}$, and in particular for IND-CPA, IND-CCA1, IND-rCCA. This is an interesting difference between gCCA and rCCA.

Theorem 1. *Cascade of encryption schemes is not a robust combiner, for specifications* IND-Px-Cy*, for* $x \in \{0, 1, 2, \mathrm{K}\}$ *and* $y \in \{2, \mathrm{G}\}$*, and specifically for* IND-CCA2 *and* IND-gCCA.

Proof: The example from [ADR02] presented above, shows that the claim holds for IND-Px-C2, for $x \in \{0, 1, 2, \mathrm{K}\}$, since the example did not require any chosen plaintext queries.

We now show a (slightly more complex) counterexample encryption scheme $\Pi'' = (\mathcal{K}'', \mathcal{E}'', \mathcal{D}'')$, s.t. for any encryption scheme $\Pi' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$, the cascade $\Pi = \Pi'' \circ \Pi'$ does not satisfy IND-Px-CG, for $x \in \{0, 1, 2, \mathrm{K}\}$. This proves the rest of the theorem.

To avoid clutter, we make the following (avoidable) simplifications:

1. $x = \mathrm{K}$, i.e. we prove for asymmetric (public-key) encryption.
2. $(\mathcal{K}', \mathcal{E}', \mathcal{D}')$ is an asymmetric encryption scheme that satisfies IND-gCCA.
3. For every $(e', d') \xleftarrow{R} \mathcal{K}'(1^k)$ holds $|d'| = k$.
4. The single bits 0 and 1 are legitimate plaintext messages.

The idea of the attack is that we design our counterexample $\Pi'' = (\mathcal{K}'', \mathcal{E}'', \mathcal{D}'')$ to allow A_1 to expose its key d'', by some queries; then we use this key to 'transform' the challenge ciphertext c^* (of the cascade) to another ciphertext $\tilde{c}$ s.t. with high probability $r_e(\tilde{c}, c^*) =$ FALSE, and in particular $\tilde{c} \neq c^*$.

An initial idea might be that $\mathcal{E}''$ will simply expose its key on some special input, e.g. ${\mathcal{E}''}_{e''}(m) =< 0, {\mathcal{E}'}_{e''}(m) >$ and ${\mathcal{D}''}_{d''}(< b, c >) = \{{\mathcal{D}'}_{d''}(c_0)$ if $b = 0$, d'' if $b = 1\}$. However, if the adversary gives $< 1, c >$ to the decryption oracle of the cascade, it will receive back ${\mathcal{D}'}_{d'}(c)$, which does not seem to help. We therefore designed a slightly more complex counterexample, where the adversary exposes d'' by a sequence of decryption oracle queries, one bit at a time.

Specifically, we define $\Pi'' = (\mathcal{K}'', \mathcal{E}'', \mathcal{D}'')$ as follows:

$$\mathcal{K}''(1^k) = \mathcal{K}'(1^k)$$
$${\mathcal{E}''}_{e''}(m) = < 0, {\mathcal{E}'}_{e''}(m), \perp, \perp >$$
$${\mathcal{D}''}_{d''}(< b, c_0, c_1, j >) = \begin{cases} {\mathcal{D}'}_{d''}(c_0) & \text{if } b = 0 \\ {\mathcal{D}'}_{d''}(c_{d''[j]}) & \text{if } b = 1 \end{cases}$$

We now present a poly-time adversary (A_1, A_2), achieving significant advantage in the $\text{IND} - \text{PK} - \text{C(G,}r\text{)}$ experiment, against the cascade $\Pi = \Pi'' \circ \Pi'$, for every $r \in \mathcal{R}(\Pi)$; this shows that Π is not IND-gCCA secure. In the first phase, A_1 finds the decryption key d''. In the second phase, A_2 receives 'challenge ciphertext' c^*, and uses d'' to modify it into ciphertext $\tilde{c} \neq c^*$, which decrypts to the same message as c^*; A_2 then uses the oracle to find the message. Since we focus (for simplicity) on $x =$ K, i.e. these are asymmetric encryption experiments, we assume that the public key is returned by the encryption oracle (i.e., is known to the adversary). The adversary just needs to encrypt two arbitrary (same length) messages; for simplicity we use the two single-bit messages 0 and 1. Namely:

Algorithm $A_1^{\mathcal{DO}_1(\cdot), \mathcal{EO}_1(\cdot)}(x, y, 1^k)$:
$(e'', e') \leftarrow \mathcal{EO}_1(0)$; (get public key (e'', e') from oracle)
For $b = 0, 1$ let $< 0, c_b, \perp, \perp > \leftarrow {\mathcal{E}''}_{e''}({\mathcal{E}'}_{e'}(b))$
For $j = 1$ to k do: $d''[j] \leftarrow \mathcal{DO}_1(< 1, c_0, c_1, j >)$
Return $(0, 1, ((e'', e'), d''))$

Algorithm $A_2^{\mathcal{DO}_2(\cdot), \mathcal{EO}_2(\cdot)}(((e'', e'), d''), c^*)$:
$\tilde{c} \leftarrow {\mathcal{E}''}_{e''}({\mathcal{D}''}_{d''}(c^*))$
Return $(\mathcal{DO}_2(\tilde{c}))$

Trivially from its design, Π'' is IND-CPA secure. Hence, for every efficient (PPT) keyed decryption-preserving relation $r \in \mathcal{R}(\Pi)$ for the cascade $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}) = (\mathcal{K}'', \mathcal{E}'', \mathcal{D}'') \circ (\mathcal{K}', \mathcal{E}', \mathcal{D}')$, with high probability, $r_e(\tilde{c}, c^*) =$ FALSE, and in particular $\tilde{c} \neq c^*$. (Otherwise, we can use the relation r to identify two random encryptions of the same message). Therefore, the $\mathcal{DO}_2$ oracle, will always output $\mathcal{D}_d(\cdot)$. This shows that the cascade Π is not an IND-gCCA secure encryption scheme. □

Note that this proof is essentially an extension of the example from [ADR02], mentioned before, showing that cascade is not robust for IND-CCA2.

3.3 Cascade Encryption is Robust for CPA, CCA1 and rCCA

The following Theorem shows, that cascade is a robust combiner of encryption schemes, for specifications IND-Px-Cy, where $x \in \{0, 1, 2, \text{K}\}$ and $y \in \{0, 1, \text{R}\}$, and in particular for

IND-CPA, IND-CCA1, IND-rCCA. This shows an interesting difference between gCCA and rCCA.

Theorem 2. *A cascade of length-uniform encryption schemes is a robust combiner for specifications* IND-Px-Cy*, for* $x \in \{0, 1, 2, \mathrm{K}\}$ *and* $y \in \{0, 1, \mathrm{R}\}$*, and in particular for IND-CPA, IND-CCA1 and IND-rCCA.*

Proof: The proof focuses on the cascade of two encryption schemes, extension to arbitrary number of schemes is trivial.

Let $\Pi = \Pi'' \circ \Pi'$ where $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and $\Pi'' = (\mathcal{K}'', \mathcal{E}'', \mathcal{D}'')$, $\Pi' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ are encryption schemes, and Π' is length-uniform. Consider any efficient adversary, i.e. pair of PPT algorithms $A = (A_1, A_2)$. The following two lemmas, use A to construct efficient (PPT) adversaries, $A'' = (A_1'', A_2'')$ and $A' = (A_1', A_2')$, against Π'' and Π' respectively, s.t.:

$$\mathbf{Adv}'^{\mathrm{IND-P}x\mathrm{-C}y}_{\Pi', A'}(k) = \mathbf{Adv}^{\mathrm{IND-P}x\mathrm{-C}y}_{\Pi, A}(k) \tag{1}$$

and

$$\mathbf{Adv}''^{\mathrm{IND-P}x\mathrm{-C}y}_{\Pi'', A''}(k) = \mathbf{Adv}^{\mathrm{IND-P}x\mathrm{-C}y}_{\Pi, A}(k) \tag{2}$$

Intuitively, to win its distinguishability game, A' (A'') generates keys for Π'' (respectively, Π'), and use the choices of A when playing against the cascade; see the proofs of the lemmas for details. The two lemmas show that if Π is not IND-Px-Cy secure, then both Π'' and Π' are not IND-Px-Cy secure, which completes the proof. □

We now present and prove the two lemmas. The first lemma shows a PPT adversary A'', s.t. equation 2 holds. This lemma holds also for the case $y = 2$ (i.e. CCA2 specification).

Lemma 1. *Let* $\Pi = \Pi'' \circ \Pi'$ *where* $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ *and* $\Pi'' = (\mathcal{K}'', \mathcal{E}'', \mathcal{D}'')$, $\Pi' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ *are encryption schemes, and* Π' *is length-uniform. For any pair of PPT algorithms* $A = (A_1, A_2)$*, there exists an efficient adversary* $A'' = (A_1'', A_2'')$*, s.t. equation 2 holds, for any* $x \in \{0, 1, 2, \mathrm{K}\}$ *and any* $y \in \{0, 1, 2, \mathrm{R}\}$*.*

Proof: Implement $A'' = (A_1'', A_2'')$ using A and Π'' as a 'black boxes', as follows:

Algorithm $A_1''(x, y, 1^k)$, with oracles $\mathcal{DO}''_1(\cdot), \mathcal{EO}''_1(\cdot)$:
$(e', d') \leftarrow \mathcal{K}'(1^k)$
$(m_0, m_1, s) \leftarrow A_1^{\mathcal{DP}_1[d', y](\cdot), \mathcal{EP}_1[e', x](\cdot)}(x, y, 1^k)$
Return $(\mathcal{E}'_{e'}(m_0), \mathcal{E}'_{e'}(m_1), (e', d', x, y, m_0, m_1, s))$

Algorithm $A_2''((e', d', x, y, m_0, m_1, s), c^*)$, with oracles $\mathcal{DO}''_2(\cdot), \mathcal{EO}''_2(\cdot)$:
Return $A_2^{\mathcal{DP}_2[d', y, m_0, m_1, c^*](\cdot), \mathcal{EP}_2[e', x](\cdot)}(s, c^*)$

The implementation also includes procedures $\mathcal{DP}_1, \mathcal{DP}_2$ and $\mathcal{EP}_i$ (for $i = 1, 2$), which are provided as oracles to A:

$$\text{Procedure } \mathcal{DP}_1[d', y](c) = \begin{cases} \epsilon & \text{if } (y = 0) \\ \mathcal{D}'_{d'}(\mathcal{DO}''_1(c)) & \text{otherwise} \end{cases}$$

$$\text{Procedure } \mathcal{DP}_2[d', y, m_0, m_1, c^*](c) = \begin{cases} \epsilon & \text{if } (y \in \{0, 1\}) \\ \epsilon & \text{if } (y = \mathrm{R}) \wedge (\mathcal{D}'_{d'}(\mathcal{DO}''_2(c)) \in \{m_0, m_1\}) \\ \epsilon & \text{if } (y = 2) \wedge (c = c^*) \\ \mathcal{D}'_{d'}(\mathcal{DO}''_2(c)) & \text{otherwise} \end{cases}$$

$$\text{Procedure } \mathcal{EP}_i[e',x](m;r''||r') = \begin{cases} \epsilon & \text{if } \ (x=0) \ \vee \ ((x=1)\wedge(i=2)) \\ \mathcal{EO}''_i(\mathcal{E}'_{e'}(m;r');r'') & \text{if } \ (x=1)\wedge(i=1) \\ (\mathcal{EO}''_i(m;r''),e') & \text{if } (x=\mathrm{K}) \end{cases}$$

Notice that for $x = \mathrm{K}$, i.e. public key, holds: $\mathcal{EP}_i(m) = (\mathcal{EO}''_i(m), e') = (e'', e')$. Also, recall that we assumed, for simplicity, that the random bits input to $\mathcal{EO}$, are split evenly between $\mathcal{EO}''$ and $\mathcal{EO}'$.

Let $\mathbf{Exp}^{\mathrm{IND-P}x-\mathrm{C}y-b}_{\Pi,A}(k;r)$ be the result of the experiment using random coins r. By the design of the experiment and of A'', and since Π' is length-uniform, we have:

$$\mathbf{Exp}^{\mathrm{IND-P}x-\mathrm{C}y-b}_{\Pi'',A''}(k;r) = \mathbf{Exp}^{\mathrm{IND-P}x-\mathrm{C}y-b}_{\Pi,A}(k;r)$$

The claim follows. □

To complete the proof of Theorem 2, the following lemma shows a PPT adversary A' s.t. equation 1 holds. The proof and constructions are similar to the previous lemma. Notice that here we do not require Π' (or Π'') to be length-uniform.

Lemma 2. *Let $\Pi = \Pi'' \circ \Pi'$ where $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ and $\Pi'' = (\mathcal{K}'', \mathcal{E}'', \mathcal{D}'')$, $\Pi' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ are encryption schemes. For any pair of PPT algorithms $A = (A_1, A_2)$, there exists another pair of PPT algorithms $A' = (A'_1, A'_2)$, s.t. equation 1 holds, for any $x \in \{0, 1, 2, \mathrm{K}\}$ and any $y \in \{0, 1, \mathrm{R}\}$.*

Proof: Implement $A' = (A'_1, A'_2)$ using A and Π' as a 'black boxes', as follows.

Algorithm $A'_1(x, y, 1^k)$, with oracles $\mathcal{DO}'_1(\cdot), \mathcal{EO}'_1(\cdot)$:
$(e'', d'') \leftarrow \mathcal{K}''(1^k)$
$(m_0, m_1, s) \leftarrow A_1^{\mathcal{DP}_1[d'',y](\cdot),\mathcal{EP}_1[e'',x](\cdot)}(x, y, 1^k)$
Return $(m_0, m_1, (e'', d'', x, y, m_0, m_1, s))$

Algorithm $A'_2((e'', d'', x, y, m_0, m_1, s), c^*)$, with oracles $\mathcal{DO}'_2(\cdot), \mathcal{EO}'_2(\cdot)$:
Return $A_2^{\mathcal{DP}_2[d'',y,m_0,m_1,c^*](\cdot),\mathcal{EP}_2[e'',x](\cdot)}(s, \mathcal{E}''_{e''}(c^*))$

The implementation also includes procedures $\mathcal{DP}_1, \mathcal{DP}_2$ and $\mathcal{EP}_i$ (for $i = 1, 2$), which are provided as oracles to A:

$$\text{Procedure } \mathcal{DP}_1[d'',y](c) = \begin{cases} \epsilon & \text{if } (y=0) \\ \mathcal{DO}'_1(\mathcal{D}''_{d''}(c)) & \text{otherwise} \end{cases}$$

$$\text{Procedure } \mathcal{DP}_2[d'',y,m_0,m_1,c^*](c) = \begin{cases} \mathcal{DO}'_2(\mathcal{D}''_{d''}(c)) & \text{if } \ (y=\mathrm{R})\wedge(\mathcal{DO}'_2(\mathcal{D}''_{d''}(c)) \notin \{m_0,m_1\}) \\ \epsilon & \text{otherwise} \end{cases}$$

$$\text{Procedure } \mathcal{EP}_i[e'',x](m;r''||r') = \begin{cases} \mathcal{E}''_{e''}(\mathcal{EO}'_i(m;r');r'') & \text{if } \ (x=1)\wedge(i=1) \\ (e'',\mathcal{EO}'_i(m;r')) & \text{if } (x=\mathrm{K}) \\ \epsilon & \text{otherwise} \end{cases}$$

Notice that for $x = \mathrm{K}$, i.e. public key, holds: $\mathcal{EP}_i(m) = (e'', \mathcal{EO}'_i(m)) = (e'', e')$.

For simplicity, we assume that $\mathcal{K}'$ and $\mathcal{K}''$ use the same number of random bits $\rho(k)$, which may depend only on the security parameter k. Let $r \in \{0,1\}^*$, and let $r = r''||r'||r^*$, where

$|r''| = |r'| = \rho(k)$. Since the first step of $\mathbf{Exp}^{\text{IND-P}x\text{-C}y\text{-}b}_{\Pi,A}(k;r)$ is to apply $\mathcal{K}$ to generate keys, then the keys generated at that step would be $(e'', d'') \leftarrow \mathcal{K}''(1^k; r'')$ and $(e', d') \leftarrow \mathcal{K}'(1^k; r')$.

Similarly, let $\hat{r} \in \{0,1\}^*$, and let $\hat{r} = \hat{r}'||\hat{r}''||\hat{r}^*$, where $|\hat{r}''| = |\hat{r}'| = \rho(k)$. The first step of $\mathbf{Exp}^{\text{IND-P}x\text{-C}y\text{-}b}_{\Pi',A'}(k;\hat{r})$ is to apply $\mathcal{K}'$ to generate keys for Π', and the next step is to invoke A'_1 which immediately applies $\mathcal{K}''$ to generate keys for Π''. Hence, the keys generated in this experiment would be $(e', d') \leftarrow \mathcal{K}'(1^k; \hat{r}')$ and $(e'', d'') \leftarrow \mathcal{K}''(1^k; \hat{r}'')$. Notice that here the first $\rho(k)$ bits are used to generate (e', d'), and second $\rho(k)$ bits used to generate (e'', d''), while the reverse order was used in $\mathbf{Exp}^{\text{IND-P}x\text{-C}y\text{-}b}_{\Pi,A}(k;r)$. This is one subtelty making this proof a bit harder (which is the reason we preferred to present the lemmas in this order).

Therefore, to ensure that the same keys are generated in both experiments, for a given $r = r''||r'||r^*$ we set $\hat{r} = r'||r''||r^*$ (i.e., $\hat{r}' = r'', \hat{r}'' = r'$ and $\hat{r}^* = r^*$). By the design of the experiment and of A', we have:

$$\mathbf{Exp}^{\text{IND-P}x\text{-C}y\text{-}b}_{\Pi',A'}(k;\hat{r}) = \mathbf{Exp}^{\text{IND-P}x\text{-C}y\text{-}b}_{\Pi,A}(k;r)$$

If r is uniformly distributed, than so is $\hat{r}$. The claim follows. □

4 Cascading Other Cryptographic Schemes

Cascading is a natural method to combine schemes. In this section, we study cascade combiners of other cryptographic schemes, beyond encryption.

We begin by discussing cascading of functions with a single input and output, such as one-way functions and (key-less) hash functions, namely $f \circ g(x) = f(g(x))$. In the following subsections we discuss cascading of keyed schemes.

4.1 Cascade of One Way Functions

Consider two functions $g : \mathcal{M}_g \to \mathcal{Y}_g$, $f : \mathcal{M}_f \to \mathcal{Y}_f$ s.t. $\mathcal{Y}_g \subseteq \mathcal{M}_f$. The *cascade* of f and g, denoted $f \circ g$, is defined as $f \circ g(x) \equiv f(g(x))$.

Unfortunately, cascade is not even a preserving combiner for one way functions. However, it is a robust combiner for one-way functions, if the component functions are length-preserving permutations (defined below). We first recap the relevant specifications.

Definition 2 (OWF). *Function $f : \mathcal{M} \to \mathcal{Y}$, s.t. for all sufficiently large l holds $\{0,1\}^l \subseteq \mathcal{M}$, is a one way function (*OWF*) if $\mathbf{Adv}^{\text{OWF}}_{f,A}(k) \equiv \mathbf{Pr}_{x \xleftarrow{R} \{0,1\}^k}\left[f(A(f(x), 1^k)) = f(x)\right]$ is a negligible function (in k), for every PPT adversary A.*

We first show that cascade is not even a preserving ((n,n)-robust) combiner, for one way functions.

Proposition 1. *Cascade is not a preserving combiner for one way functions.*

Proof: let h be a OWF. Let $g(x) = h(x)||0^{|h(x)|}$ and $f(x) = \begin{cases} 0 & \text{if } x = y0^{|y|} \\ h(x) & \text{otherwise} \end{cases}$

Trivially, both f and g are one-way functions, yet $f \circ g$ is not a OWF; in fact, $f \circ g(x) = f(g(x)) = 0$ for every x. □

We next show that, on the contrary, cascade is robust for the OWF specification, if the functions are guaranteed to be *length-preserving permutations*. A function $f : \mathcal{M} \to \mathcal{Y}$ is

length-preserving permutation, if for every sufficiently large k, the function f is a permutation over $\{0,1\}^k$. See also Levin [Lev87], for a notion of one-way functions that remain one-way when iterated.

Proposition 2. *Cascade of length-preserving permutations, is a robust combiner for the OWF specification.*

Proof: Let $h = f \circ g$, where f and g are length-preserving permutations; namely, $h(x) = f(g(x))$. Trivially, h is also a length-preserving permutation. Suppose that h is not a OWF, namely there is some PPT adversary A_h s.t. $\Pr_{x_h \in \{0,1\}^k}\left[h(A_h(h(x_h),1^k)) = h(x_h)\right]$ is not a negligible function. Let $A_g(y_g,1^k) = A_h(f(y_g),1^k)$ and $A_f(y_f,1^k) = g(A_h(y_f,1^k))$. Since both g and f are length-preserving permutations, it follows that for sufficiently large k:

$$\Pr_{x_g \in \{0,1\}^k}\left[g(A_g(g(x_g),1^k)) = g(x_g)\right] = \Pr_{x_h \in \{0,1\}^k}\left[h(A_h(h(x_h),1^k)) = h(x_h)\right]$$

and

$$\Pr_{x_f \in \{0,1\}^k}\left[f(A(f(x_f),1^k)) = f(x_f)\right] = \Pr_{x_h \in \{0,1\}^k}\left[h(A_h(h(x_h),1^k)) = h(x_h)\right]$$

where the probabilities are also over the coin tosses of the algorithms. □

4.2 Cascading of Cryptographic Hash Functions

We next study the tolerance of the cascade of cryptographic keyed hash functions. We use $\mathcal{K}(1^k)$ to denote the (finite) set of keys for (unary) security parameter 1^k. Namely, a *cryptographic hash function family* is a function $h : \mathcal{K}(1^k) \times \mathcal{M} \to \mathcal{Y}$. Usually, we write the first argument to h as subscript, i.e. as $h_K(m)$, where $K \in \mathcal{K}(1^k)$. For simplicity we insist that there is some polynomial $l(\cdot)$ (hash length) such that $\{h_K(m)|m \in \mathcal{M}\} = \{0,1\}^{l(k)}$ for $K \in \mathcal{K}(1^k)$. These assumptions are convenient for defining cascading, and hold for reasonable hash functions.

We define the cascade $h = f \circ g$ of cryptographic hash function families $f : \mathcal{K}_f \times \mathcal{M}_f \to \mathcal{Y}_f$ and $g : \mathcal{K}_g \times \mathcal{M}_g \to \mathcal{Y}_g$, where $\mathcal{Y}_g \subseteq \mathcal{M}_f$, as $h_{(K_f,K_g)}(m) = f_{K_f}(g_{K_g}(m))$. Clearly h is a cryptographic hash function family, $h : (\mathcal{K}_f \times \mathcal{K}_g) \times \mathcal{M}_g \to \mathcal{Y}_f$.

There are many different (and not equivalent) definitions of cryptographic hash function families, see e.g. [RS04]. We focus on four definitions: Coll ('strong collision resistance'), ePre (find pre-image with random key, to an adversary-chosen challenge), Pre[l] (find pre-image with random key and random challenge) and aSec[l] (find second preimage to a random challenge, using adversary-chosen key). Pre[l] is implied by all the other definitions in [RS04]. We show that cascade is not (even) preserving for Pre[l] and aSec[l], and is preserving (but not robust) for Coll and ePre.

The fact that cascade is not robust for Coll also follows as a trivial special case of the results of [BB06,Pie07], which show that any (black box) combiner for strong collision resistant hash functions (Coll) must, essentially, double the hash length.

We next present asymptotic versions of the specifications of cryptographic hash function families from [RS04]:

Definition 3 (Cryptographic hash functions). *Let $h : \mathcal{K} \times \mathcal{M} \to \mathcal{Y}$ be a cryptographic hash function family. Let $A = (A_1, A_2)$ be a pair of PPT algorithms. Define:*

$$\mathbf{Adv}^{\mathrm{Coll}}_{h,A}(k) = Pr\left[K \xleftarrow{R} \mathcal{K}(1^k); (m,m') \xleftarrow{R} A_1(K) : (m \neq m') \wedge \left(h_K(m) = h_K(m')\right)\right]$$

$$\mathbf{Adv}^{\mathrm{Pre}[l]}_{h,A}(k) = Pr\left[K \xleftarrow{R} \mathcal{K}(1^k); m \xleftarrow{R} \{0,1\}^l; y \leftarrow h_K(m); m' \xleftarrow{R} A_1(K,y) : h_K(m') = y\right]$$
$$\mathbf{Adv}^{\mathrm{ePre}}_{h,A}(k) = Pr\left[(y,s) \xleftarrow{R} A_1(1^k); K \xleftarrow{R} \mathcal{K}(1^k); m \xleftarrow{R} A_2(K,s) : h_K(m) = y\right]$$
$$\mathbf{Adv}^{\mathrm{aSec}[l]}_{h,A}(k) = Pr\left[(K,s) \xleftarrow{R} A_1(1^k); m \xleftarrow{R} \{0,1\}^l; m' \xleftarrow{R} A_2(m,s) : (m \neq m') \wedge \big(h_K(m) = h_K(m')\big)\right]$$

For $\phi \in \{\mathrm{Coll}, \mathrm{Pre}[l], \mathrm{ePre}, \mathrm{aSec}[l]\}$, let $\mathbf{Adv}^{\phi}_{h}(k) = \max_{A \in PPT} \mathbf{Adv}^{\phi}_{h,A}(k)$. We say that h (asymptotically) satisfies specifications ϕ, if $\mathbf{Adv}^{\phi}_{h}(k)$ is a negligible function.

Theorem 3 (Cascade of crypto-hash). *Cascade is a preserving, but not robust, combiner of cryptographic hash function families, for specifications Coll and ePre. Cascade is not preserving for specifications* Pre[l] *and* aSec[l].

Proof: We first demonstrate that cascade is not robust for each of the specifications. This is trivial. Consider the cascade $h \equiv f \circ g$. For Coll, Pre[l] and aSec[l], let $g_K(m) = 0$; for ePre, let $f_K(m) = 0$.

The fact that cascade is not (even) preserving for specifications Pre[l] and aSec[l], follows from a similar argument to the one in Proposition 1, as follows. For simplicity, assume that all hash functions in this argument has the domain of all strings, i.e. $\mathcal{M} = \{0,1\}^*$; also ignore the (technical) requirement that $\{h_K(m) | m \in \mathcal{M}\} = \{0,1\}^{l(k)}$ for $K \in \mathcal{K}(1^k)$, for some polynomial l. Given a cryptographic hash function family $u_K(m)$ that satisfies Pre[l] (or aSec[l]), it is trivial that the following f and g also satisfy Pre[l] (respectively aSec[l]), but their cascade $h = f \circ g$ does not satisfy Pre[l] or aSec[l], where:

$$g_K(m) \equiv u_K(m) || 0^{|u_K(m)|}$$
$$f_K(m) = \begin{cases} 0 & \text{if } m = y0^{|y|} \\ u_K(m) & \text{otherwise} \end{cases}$$

It remains to show that cascade is preserving for the Coll and ePre specifications. Consider first the Coll specification, and a collision for h, i.e. $f_{K_f}(g_{K_g}(m)) = f_{K_f}(g_{K_g}(m'))$ for $m \neq m'$. Let $y = g_{K_g}(m)$ and $y' = g_{K_g}(m')$. If $y = y'$, then this is a collision for g_{K_g}; otherwise, y and y' are a collision for f_{K_f}. Hence, if h fails to fulfill Coll, then either f or g (or both) also fails to fulfill Coll.

Finally, consider the ePre specification, and let $A_h = (A_{h,1}, A_{h,2})$ be an adversary which achieves significant advantage against $h = f \circ g$ in the ePre game, i.e. $\mathbf{Adv}^{\mathrm{ePre}}_{h,A_h}(k)$ is significant (not negligible). The following adversary, $A_f = (A_{f,1}, A_{f,2})$, trivially achieves significant advantage against f in the ePre game:

Algorithm $A_{f,1}(1^k)$:
  { $(y,s) \leftarrow A_{h,1}(1^k)$;
  $K_g \xleftarrow{R} \mathcal{K}_g(1^k)$;
  return $(y, (K_g, s))$ }
Algorithm $A_{f,2}(K_f, (K_g, s))$:
  { $m = A_{h,2}((K_f, K_g), s)$;
  return $g_{K_g}(m)$ }

We conclude that if f satisfies ePre, then h also satisfies ePre, i.e. cascade is preserving for ePre. This concludes the proof. □

4.3 Cascading Preserves Unforgeability

We next study cascading of signature and MAC (Message Authentication Code) schemes. Formally, we define both signature and MAC schemes as three efficient (polynomial-time) algorithms, $(\mathcal{K}, \mathcal{S}, \mathcal{V})$, for key-generation, signing/authenticating and verifying, respectively, s.t. for every $(s, v) \leftarrow \mathcal{K}(1^k)$ and every m in the domain of S, holds: $\mathcal{V}_v(m, \mathcal{S}_s(m)) = \text{TRUE}$. The key-generation and the signature/authentication algorithms are probabilistic; for simplicity, assume a deterministic verifying algorithm.

Let $\Pi'' = (\mathcal{K}'', \mathcal{S}'', \mathcal{V}'')$ and $\Pi' = (\mathcal{K}', \mathcal{S}', \mathcal{V}')$ be two signature/MAC schemes. Without loss of generality, assume that the two schemes use the same number of random bits $\rho(k) \in \mathbb{N}$ for given security parameter k. We define their cascade $\Pi = \Pi'' \circ \Pi'$, where $\Pi = (\mathcal{K}, \mathcal{S}, \mathcal{V})$, as follows:

- For any $k \in \mathbb{N}$ and any $r'', r' \in \{0,1\}^{\rho(k)}$, let $(v', s') \leftarrow \mathcal{K}'(1^k; r')$ and $(v'', s'') \leftarrow \mathcal{K}''(1^k; r'')$. Then let $\mathcal{K}(1^k; r''||r') = ((v'', v'), (s'', s'))$.
- $\mathcal{S}_{s'',s'}(m; r''||r') = S''_{s''}(S'_{s'}(m; r'); r'')$
- $\mathcal{V}_{v'',v'}(m, (\sigma'', \sigma')) = \mathcal{V}''_{v''}(\sigma', \sigma'') \wedge \mathcal{V}'_{v'}(m, \sigma')$

We define unforgeability of signature/authentication schemes as follows. For $\phi \in \{$ MAC, Signature $\}$, scheme $(\mathcal{K}, \mathcal{S}, \mathcal{V})$ is an unforgeable ϕ scheme if for every $A \in PPT$:

$$\Pr\left[\mathbf{Exp}^{\text{Forge},\phi}_{(\mathcal{K},\mathcal{S},\mathcal{V}),A}(1^k) = 1\right] \overset{\text{poly}}{\approx} 0$$

Where $\mathbf{Exp}^{\text{Forge},\phi}_{(\mathcal{K},\mathcal{S},\mathcal{V}),A}$ is the following experiment:

Experiment $\mathbf{Exp}^{\text{Forge},\phi}_{(\mathcal{K},\mathcal{S},\mathcal{V}),A}(k)$:
$(s, v) \overset{R}{\leftarrow} \mathcal{K}(1^k)$
If $\phi =$Signature then $\nu = v$ else $\nu = \perp$
$(m, \sigma) \overset{R}{\leftarrow} A^{\mathcal{S}_s(\cdot),\mathcal{V}_v(\cdot)}(1^n, \nu)$
Let $Q = \{x | A \text{ made query } \mathcal{S}_s(x)\}$
Return 1 if $m \notin Q$ and $\mathcal{V}_v(m, \sigma) =$'OK'

Proposition 3. *Cascade of MAC and signature schemes is a preserving combiner, but is not a robust combiner, for the unforgeable MAC scheme and unforgeable signature scheme specifications.*

Proof: Trivial. □

4.4 Cascading Commitment Schemes

We conclude our study of cascading, by investigating cascading of single-round commitment schemes. In a single-round commitment scheme, the recipient sends to the sender some (random) *binding key* K, which the the sender should use to compute commitments. Single-round commitment schemes are common in practice, often using heuristics (e.g., using the binding key as the key of HMAC [BCK96]).

We define single-round commitment schemes as three algorithms, $\Pi = (\mathcal{C}, \mathcal{D}, \mathcal{V})$, for Commit, Decommit and Validate, respectively[3]. The commit and decommit functions $\mathcal{C}, \mathcal{D}$ have three inputs each: a message $m \in \{0,1\}^*$, a public binding key $K \in \{0,1\}^k$ and randomness $r \in \{0,1\}^k$, and their respective outputs are: a commitment tag $c = \mathcal{C}_K(m;r)$ and a decommitment tag $d = \mathcal{D}_K(m;r)$. The validate function $\mathcal{V}$ has four inputs: the message m, the public binding key K, and the commitment and decommitment tags (c, d respectively). The validation function outputs 1 if only if c, d are a correct commitment and decommitment values for m. The definition follows.

Definition 4 (Single round commitment scheme). *A* single round commitment scheme *consists of three PPT algorithms, $\Pi = (\mathcal{C}, \mathcal{D}, \mathcal{V})$, s.t. for every $m \in \{0,1\}^*$, $K \in \{0,1\}^k$, $r \in \{0,1\}^k$ holds: $\mathcal{V}_K(m, C_K(m;r), D_K(m;r)) = 1$. We say that Π is* length-uniform*, if for every m_0, m_1, r_0, r_1, K s.t. $|m_0| = |m_1|$, holds: $|C_K(m_0; r_0)| = |C_K(m_1; r_1)|$.*

Commitment schemes should satisfy a confidentiality (indistinguishability) specification, called *hiding*, and an integrity specification, called *binding*. We present computational, asymptotic definitions for both specifications.

We first define the hiding specifications for commitment schemes, which is rather similar to the indistinguishability specifications presented earlier for encryption schemes. Intuitively, the (polytime) hiding specification is that no probabilistic polynomial time (PPT) adversary can distinguish between the commitments of any two messages of its choice. However notice that, we allow the adversary to choose the public commitment key (but not the public encryption key). In single round commitment schemes, confidentiality is protected by the choice of the random input bits at the commit phase; the binding key K cannot be relied upon, since it is received from the recipient (the binding key is crucial for the binding property).

Definition 5 (Hiding specification for single round commitment schemes). *Let $\Pi = (\mathcal{C}, \mathcal{D}, \mathcal{V})$ be a single round commitment scheme. Let $A = (A_1, A_2)$ be a pair of PPT algorithms, and:*

$$\boldsymbol{Adv}^{\text{Hide}}_{\Pi,A}(k) = \Pr\left[\begin{array}{l}(K, m_0, m_1, s) \xleftarrow{R} A_1(1^k); b \xleftarrow{R} \{0,1\}; b' \xleftarrow{R} A_2(C_K(m_b; r), s) : \\ (b' = b) \wedge (|m_0| = |m_1|)\end{array}\right]$$

We say that Π is (computationally) hiding*, if $\boldsymbol{Adv}^{\text{Hide}}_{\Pi,A}(k)$ is negligible for every PPT adversary A.*

We now define the binding (integrity) specification for single round commitment schemes. Here, we use the public binding key K. The binding key K is picked by the 'recipient' of the commitment, and therefore the adversary, acting now as a potentially malicious 'sender' of the commitment, has to use a random public binding key K, rather than choosing the worst possible key.

Intuitively, given (random) public binding key K, every (PPT) adversary A has negligible probability of finding a collision, i.e. values c, d, d', m, m' s.t. $\mathcal{V}_K(m, c, d) = \mathcal{V}_K(m', c, d') = 1$ (notice the commitment c is the same!). The definition follows.

[3] In most existing definitions of commitment schemes, e.g. in [Gol01], the decommit function returns the message (often limited to single bit), and there is no Validate function. However, this requires long commitment and/or decommitment strings (since they must contain the input). When combining such schemes, this may result in long commitment and/or decommitment strings. Our definition avoids this. Also notice that, for simplicity, our definition uses a random key, without explicit key generation function as in some other definitions.

Definition 6 (Binding specification for single round commitment schemes). *Let $\Pi = (\mathcal{C}, \mathcal{D}, \mathcal{V})$ be a single round commitment scheme. Let A be a PPT algorithm, and:*

$$\boldsymbol{Adv}^{\text{Binding}}_{\Pi,A}(k) = \Pr\left[\begin{array}{l} K \xleftarrow{R} \{0,1\}^k; (m_0, m_1, c, d_0, d_1, r) \leftarrow A(K) : \\ (m_0 \neq m_1) \wedge (\mathcal{V}_K(m_0, c, d_0) = 1) \wedge (\mathcal{V}_K(m_1, c, d_1) = 1) \end{array}\right]$$

We say that Π is (computationally) binding*, if $\boldsymbol{Adv}^{\text{Binding}}_{\Pi,A}(k)$ is negligible, for every PPT adversary A.*

Let $\Pi'' = (\mathcal{C}'', \mathcal{D}'', \mathcal{V}'')$ and $\Pi' = (\mathcal{C}', \mathcal{D}', \mathcal{V}')$ be two single round commitment schemes. The *cascade* of Π'' and Π', denoted $\Pi = \Pi'' \circ \Pi'$, is defined as $\Pi = (\mathcal{C}, \mathcal{D}, \mathcal{V})$ where:

$\mathcal{C}_K(m; r) = \mathcal{C}''_{K''}(\mathcal{C}'_{K'}(m; r')||\mathcal{D}'_{K'}(m; r'); r'')$ where $K = K''||K'$, $r = r''||r'$ and $|K''| = |K'| = |r''| = |r'| = \lfloor \frac{k}{2} \rfloor$.
$\mathcal{D}_K(m; r) = \left(\mathcal{D}''_{K''}(\mathcal{C}'_{K'}(m; r')||\mathcal{D}'_{K'}(m; r'); r''), \mathcal{D}'_{K'}(m; r'), \mathcal{C}'_{K'}(m'; r')\right)$
$\mathcal{V}_K(m, c, (d'', d', c')) = \left[\mathcal{V}''_{K''}(c'||d', c, d'')\right] \wedge \left[\mathcal{V}'_{K'}(m, c', d')\right]$

Lemma 3. *Cascade of single round, length-uniform commitment schemes, is a robust combiner for hiding, and a preserving combiner for binding.*

Proof: The robustness for hiding follows exactly as in Theorem 2. The fact the cascade preserves binding, is trivial. □

There exist commitment schemes which were proved to be binding, therefore such schemes can be combined using cascade to ensure that they are also (computationally) hiding. However, such schemes are rarely used in practice, due to their large overhead. Therefore, for practical applications, we need an efficient combiner for commitment schemes, which is robust for both hiding and binding. We present such combiner later on.

5 Simple Robust Combiners: Parallel, Copy and XOR-Input

In the previous sections, we found that cascade is a robust combiner for important confidentiality specifications (e.g. IND-CPA, IND-CCA1, IND-rCCA and Hiding), but is not robust for many other specifications. In this section, we discuss three additional simple, 'folklore' combiners, which are (often trivially) robust for important specifications. Specifically, we show such simple robust combiners for one-way functions, MAC and signature schemes, the binding and hiding specifications of commitment schemes, and encryption schemes.

5.1 Parallel Combiner: Robust for OWF

The parallel combiner 'splits' the input among several functions, and concatenates the result. In particular, let the parallel combiner for two keyless functions f'', f' be $f''||f'(x) =< f''(x''), f'(x') >$, where $x'' = x\left[1, \ldots, \left\lfloor \frac{|x|}{2} \right\rfloor\right]$, $x' = x\left[\left\lfloor \frac{|x|}{2} \right\rfloor + 1, \ldots, |x|\right]$. This trivial combiner is robust for the one-way functions specifications.

Lemma 4. *The parallel combiner $f \equiv f''||f'$ is robust for the OWF specifications.*

Proof: Let $A''(y'', 1^{k''}) = A(y''||f'(x'), 1^{2k''})[1 \ldots k'']$, where $x' \xleftarrow{R} \{0,1\}^{k''}$. If $f\left(A(y''||f'(x'), 1^{2k''})\right) = f(x''||x')$, then $y'' = f''(A(y''||f'(x'), 1^{2k''})[1 \ldots k''])$. Hence, $\mathbf{ADV}^{\text{OWF}}_{f,A}(2k'') \leq \mathbf{ADV}^{\text{OWF}}_{f'',A''}(k'')$.

Similarly, $\mathbf{ADV}^{\mathrm{OWF}}_{f,A}(2k''+1) \leq \mathbf{ADV}^{\mathrm{OWF}}_{f'',A''}(k'')$. We conclude that if $\mathbf{ADV}^{\mathrm{OWF}}_{f,A}(k)$ is significant (non-negligible), then $\mathbf{ADV}^{\mathrm{OWF}}_{f'',A''}(k'')$ is also significant. Hence, if f is not a OWF, then f'' is also not a OWF. The proof for f' follows by symmetry. □

This simple combiner is of limited practical value, since the parallel combiner doubles the input and output length of the candidate functions. Furthermore, one-way functions are rarely used directly in practice. The importance of the combiner (and lemma) is due to the fact that many primitives are equivalent to one way functions, for asymptotic, computational-security primitives; one example is commitment schemes. The lemma provides a (often wasteful) robust combiner for such primitives, in the following three steps. First, reduce (to OWF) each of the (say two) candidate schemes, resulting in (say two) candidate one-way functions. Second, use the parallel combiner on the two candidate one-way functions, resulting in a function which is OWF if at least one of the two candidates is a OWF. Finally, use the known construction of the primitive from a OWF. For example, this gives a combiner for single-round commitment schemes, which is robust for the (computational) hiding and binding specifications. Notice, however, that such combiners result in schemes with extremely high, often impractical, key and/or block lengths and complexities (see [Gol01,HL92]).

5.2 Copy Combiner: Robust for Unforgeablility, Collision-Resistance and Binding

Another folklore combiner is copy, which applies both candidates to the same input (and output the concatenated result, like the parallel combiner). The copy combiner for hash functions was first proposed by Preneel in [Pre93]. Preneel (and others) use the term 'cascade' for this combiner, but we will use here the term *copy combiner*, to avoid confusion with cascade combiners investigated earlier.

The copy combiner was intuitively expected to improve security compared to each of its component hash functions. This was recently contradicted by Joux [Jou04]. However, we observe that the copy combiner is, at least, robust, for the collision-resistant specifications of cryptographic hash functions, the unforgeability specifications of signature and MAC schemes, and the binding specifications of single round commitment schemes.

Formally, we define the copy combiner as follows:

- We define the copy combiner $h = h''||h'$ of two keyed (cryptographic hash) functions h'' and h' as: $h_{K''||K'}(x) = h''_{K''}(x)||h'_{K'}(x)$. For simplicity, assume $|K''| = |K'|$.
- Let $\Pi'' = (\mathcal{K}'', \mathcal{S}'', \mathcal{V}'')$ and $\Pi' = (\mathcal{K}', \mathcal{S}', \mathcal{V}')$ be two signature/MAC schemes. We define their copy combiner $\Pi = \Pi''||\Pi'$, where $\Pi = (\mathcal{K}, \mathcal{S}, \mathcal{V})$, as follows:
  - For any $r', r'' \in \{0,1\}^*$ and $k \in \mathbb{N}$, let $(v', s') \leftarrow \mathcal{K}'(1^k; r')$ and $(v'', s'') \leftarrow \mathcal{K}''(1^k; r'')$. Then let $\mathcal{K}(1^k; (r''||r')) = ((v'', v'), (s'', s'))$.
  - $\mathcal{S}_{s'',s'}(m; r''||r') = S''_{s''}(m; r'')||S'_{s'}(m; r')$, where $|r''| = |r'|$.
  - $\mathcal{V}_{v'',v'}(m, (\sigma'', \sigma')) = \mathcal{V}''_{v''}(m, \sigma'') \wedge \mathcal{V}'_{v'}(m, \sigma')$
- Let $\Pi'' = (\mathcal{C}'', \mathcal{D}'', \mathcal{V}'')$ and $\Pi' = (\mathcal{C}', \mathcal{D}', \mathcal{V}')$ be two single round commitment schemes. The *copy* of Π'' and Π', denoted $\Pi = \Pi''||\Pi'$, is defined as $\Pi = (\mathcal{C}, \mathcal{D}, \mathcal{V})$ where:
  $\mathcal{C}_{K''||K'}(m; r) = \left(\mathcal{C}''_{K''}(m; r''), \mathcal{C}'_{K'}(m; r')\right)$ where $K = K''||K'$, $r = r''||r'$ and $|K''| = |K'| = |r''| = |r'| = \left\lfloor \frac{k}{2} \right\rfloor$.
  $\mathcal{D}_{K''||K'}(m; r) = \left(\mathcal{D}''_{K''}(m; r''), \mathcal{D}'_{K'}(m; r')\right)$
  $\mathcal{V}_{K''||K'}(m, (c'', c'), (d'', d')) = \left[\mathcal{V}''_{K''}(m, c'', d'')\right] \wedge \left[\mathcal{V}'_{K'}(m, c', d')\right]$

We now state some properties of the copy combiner.

Proposition 4. *The copy combiner as defined above is:*

1. *Robust for the* Coll *and* aSec[l] *specifications of cryptographic hash functions.*
2. *Robust for the* unforgeability *specification of MAC/Signature schemes.*
3. *Robust for the* binding *specifications of single round commitment schemes.*
4. *Preserving, but* not *robust, for the* hiding *specifications of single round commitment schemes.*

Proof: Trivial. □

5.3 XOR-Input Combiner is Robust for Encryption

XOR-Input is our term for another classical robust combiner, originally proposed in [AB81] for (probabilistic) encryption schemes. We present the XOR-Input combiner for encryption schemes.

Let $\Pi'' = (\mathcal{K}'', \mathcal{E}'', \mathcal{D}'')$, $\Pi' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ be two encryption schemes. The *XOR-Input* of Π'' and Π', denoted $\Pi = (\mathcal{K}, \mathcal{E}, \mathcal{D}) = \Pi'' \oplus \Pi'$, is defined as:

- Let $(e', d') = \mathcal{K}'(1^k; r')$, $(e'', d'') = \mathcal{K}''(1^k; r'')$. Given $r'', r' \in \{0,1\}^\rho$, for some $\rho \in \mathbb{N}$, let $\mathcal{K}(1^k; (r''||r')) = ((e'', e'), (d'', d'))$.
- Let $e = (e'', e'), r = r''||r'||\hat{r}$; assume for simplicity known length of $r'', r', \hat{r}$. Then $\mathcal{E}_e(m; r) = \left(\mathcal{E}''_{e''}(m \oplus r; r''), \mathcal{E}'_{e'}(r; r')\right)$
- Let $c = (c'', c'), d = (d'', d')$. Then $\mathcal{D}_d(c) = \mathcal{D}'_{d'}(c') \oplus \mathcal{D}''_{d''}(c'')$

The XOR-Input combiner is robust for the (relatively weak) notions of CPA and CCA1, but is not even preserving for the (stronger) notions of CCA2, gCCA and rCCA.

Theorem 4. *The XOR-Input combiner of public-key encryption schemes is robust for specifications* IND-CPA *and* IND-CCA1*, but not preserving for specifications* IND-CCA2, IND-gCCA *and* IND-rCCA.

Proof: First, we show a simple counterexample showing that XOR-Input is not even preserving, for the IND-rCCA specification; a trivial modification to the counterexample applies for IND-gCCA and IND-CCA2. Without loss of generality, assume that the plaintext messages domain contains the set of two-bit messages $\{0,1\}^2$. The adversary first asks (at 'select phase') for encryption of the message 00; it receives the ciphertext (c''_0, c'_0), where $c''_0 = \mathcal{E}''_{e''}(00 \oplus r_0)$ and $c'_0 = \mathcal{E}'_{e'}(r_0)$, for some two-bit $r_0 \stackrel{R}{\leftarrow} \{0,1\}^2$. The adversary now selects the challenge plaintexts $m_0 = 10$ and $m_1 = 11$, and receives the challenge ciphertext (c''_c, c'_c). Here, $c''_c = \mathcal{E}''_{e''}(m_b \oplus r_c)$ and $c'_c = \mathcal{E}'_{e'}(r_c)$, and $b \stackrel{R}{\leftarrow} \{0,1\}, r_c \stackrel{R}{\leftarrow} \{0,1\}^2$. Now the adversary invokes the decryption oracle, over (c''_c, c'_0), and then over (c''_0, c'_c). For simplicity, assume neither oracle call returns ϵ (or trivially extend the proof to deal with or avoid this case, e.g. by using longer plaintexts). Hence, the oracle calls should return $y = m_b \oplus r_c \oplus r_0$ and $z = m_0 \oplus r_0 \oplus r_c = r_0 \oplus r_c$ (since $m_0 = 00$). Clearly, $m_b = y \oplus z$, showing that although both encryption systems could have satisfied the IND-rCCA specifications, their XOR-Input combination does not satisfy IND-rCCA.

It remains to show the positive claim. For simplicity, we prove only for IND-CCA1; the proof for IND-CPA is essentially a subset of this proof. Let $\Pi'' = (\mathcal{K}'', \mathcal{E}'', \mathcal{D}''), \Pi' = (\mathcal{K}', \mathcal{E}', \mathcal{D}')$ be two encryption schemes; without loss of generality, assume that the domain of

both is the set $\{0,1\}^*$ (all binary strings). Let $\Pi = \Pi'' \oplus \Pi'$ be the XOR-Input combiner applied to Π'', Π'. Assume that Π is not an IND-CCA1 secure public key cryptosystem; we will show that Π' is also not IND-CCA1 secure. The proof for Π'' is symmetric.

Let $A^{\oplus}$ be an adversary s.t. $\mathbf{Adv}^{\mathrm{IND-PK-C1}(k)}_{\Pi,A^{\oplus}}$ is significant (not negligible). We construct adversary A' s.t. $\mathbf{Adv}^{\mathrm{IND-PK-C1}}_{\Pi',A'}(k)$ is also significant. The design of A' is simple, since we deal with only CCA1 experiment; namely, decryption is only done *before* the messages $m_0,\ m_1$ are selected. Therefore, at the select phase, A' can easily answer all the queries of $A^{\oplus}$, by selecting keys for Π''.

When $A^{\oplus}$ selects the messages $m_0,\ m_1$, the adversary A' provides the same messages in its experiment. When A' receives ciphertext c^* to distinguish in its experiment, it gives to $A^{\oplus}$ the ciphertext $< \mathcal{E}''_{e''}(0^{|m_0|}), c^* >$. Next, A' simply outputs the same guess as it receives from $A^{\oplus}$. Trivially, A' has the same chance of winning as $A^{\oplus}$. □

6 Robust Combiners for Single Round Commitment Schemes

The combiners we presented so far for commitment schemes, were robust for either binding or hiding - but not for both properties. One exception is the highly-inefficient transformation to and from one way functions; this transformation requires at least one of the candidates to be both hiding and binding.

In this section, we present two $(2,3)-$robust combiners for single round commitment schemes, which ensure robustness for both binding and hiding separately (i.e., it is enough that two of the inputs are binding and two - possibly other - are hiding). Note that it is impossible to make a combiner, which is $(n,2n)$-robust for *both* binding and hiding separately; this observation is due to Unruh [Unr06].

The techniques of both subsections, seem likely to be applicable to design of combiners for other tasks.

6.1 Input-Sharing Combiner: Robust for Hiding and Binding

We now present the *input-sharing combiner* for single round commitment schemes. This combiner essentially generalizes - and merges - the copy combiner and the XOR-Input combiner. The Input-sharing combiner is based on a secret-sharing scheme, e.g. Shamir's [Sha79]. Secret sharing schemes consist of two functions: a *share generation* function G, and a *secret recovery* function R. The share generation algorithm G accepts input (secret), and outputs m *shares*; let $G(s)[i]$ denote the i^{th} share of secret s. The secret recovery algorithm R accepts $t \leq m$ shares, and outputs the secret. Secret-sharing schemes must further satisfy the secrecy condition, which essentially says that any subset of less than t shares, does not provide any information on the secret (even against computationally unbounded adversary). We call such schemes t-out-of-m secret sharing schemes; see definition, e.g. in [Gol04].

For simplicity, we focus on a 2-out-of-3 secret sharing scheme (G,R), and apply it to define the (2-out-of-3) Input-sharing combiner for single round commitment schemes. . In this combiner, the inputs to each of the three component commitment scheme, are *shares* of the input to the combined scheme.

For $i \in \{1,2,3\}$, let $\Pi_i = \left(\mathcal{C}^{(i)}, \mathcal{D}^{(i)}, \mathcal{V}^{(i)}\right)$ be a single round commitment scheme. The *2-out-of-3 secret sharing combiner* of $\{\Pi_i\}_{i\in\{1,2,3\}}$, denoted $\Pi = \bigotimes_{i\in\{1,2,3\}} \Pi_i$, is defined as $\Pi = (\mathcal{C}, \mathcal{D}, \mathcal{V})$ where:

$$\mathcal{C}_{\{K_i\}_{i\in\{1,2,3\}}}(m) = \left\langle \mathcal{C}^{(i)}_{K_i}\left(G(m)[i]\right)\right\rangle_{i=1,2,3}$$

$$\mathcal{D}_{\{K_i\}_{i\in\{1,2,3\}}}(m) = \left\langle \left(\mathcal{D}^{(i)}_{K_i}\left(s_i\right), s_i\right)\right\rangle_{i=1,2,3} \text{ where } s_i = G(m)[i]$$

$\mathcal{V}_{\{K_i\}_{i\in\{1,2,3\}}}(m, \{c_j, (d_j, s_j)\}_{j=1,2,3}) = \textsc{True}$ if and only if:
$((\forall i \neq j \in \{1,2,3\})m = R(\{s_i, s_j\})) \wedge \left((\forall i \in \{1,2,3\})\mathcal{V}^{(i)}_{K_i}(s_i, c_i, d_i) = 1\right)$

The robustness of the sharing combiner $\Pi = \bigotimes_{i\in\{1,2,3\}} \Pi_i$ follows easily from the properties of secret sharing schemes.

Proposition 5. *The input-sharing combiner ($\bigotimes$) of* 3 *single round commitment schemes, as defined above, is robust for both the* Binding specification *and the* Hiding *specification of single round commitment schemes.*

Proof: Trivial from the properties of secret-sharing schemes (see e.g. [Gol04]), and arguments as in Theorem 4.

Comment. In practical commitment schemes the additional decommitment strings d_i are often quite short, compared to the message (e.g. using HMAC [BCK96], the decommitment string is simply a short random string appended to the message before applying HMAC). However, the decommitment strings of the input-sharing combiner, as defined above, include the three shares. Most definitions and constructions of secret-sharing schemes were proved secure, without computational assumptions; for such schemes, the length of each share must be as long as the message. Krawczyk [Kra93] defined computationally-secure secret sharing, and presents a scheme where the length of *all* shares is as long as the message. For applications where the share-size can be a significant resource, e.g. as described in [Kra93], it is preferable to use such a computationally-secure, short-shares scheme. An alternative is to use the combiner in the next subsection, which requires short decommitment strings.

Comment. A similar input-sharing combiner was shown to be robust for CCA2-secure encryption in [DK05]. Other applications are likely.

6.2 Composing Combiners: the *E* and *D* robust combiners

Often, we may want to combine multiple combiners, e.g. to create a robust combiner for multiple specifications. We restrict our attention to simple compositions of two combiners, and focus on combiners for commitment schemes, although the ideas and techniques can be applied to other multiple-specifications primitives. Specifically, we show how to compose the cascade combiner (robust for *hiding*) and the copy combiner (robust for *binding*), resulting in efficient robust combiners for commitment schemes (ensuring both hiding and binding specifications).

The D combiner uses *four* candidate commitment schemes, $\mathcal{C}_{11}, \mathcal{C}_{12}, \mathcal{C}_{21}$, and $\mathcal{C}_{22}$, cascading $\mathcal{C}_{11}$ and $\mathcal{C}_{12}$ and connecting this in parallel to the cascade of $\mathcal{C}_{21}$ and $\mathcal{C}_{22}$. We call the result the D combiner, after its 'shape'. Namely:

$$D(\mathcal{C}_{11}, \mathcal{C}_{12}, \mathcal{C}_{21}, \mathcal{C}_{22}) = (\mathcal{C}_{11} \circ \mathcal{C}_{12})||(\mathcal{C}_{21} \circ \mathcal{C}_{22})$$

The D combiner is efficient in computation times (each operation requires one operation from each of the four candidate commitment schemes), and in the size of the commit and decommit strings (commit size is twice that of the candidate commitment schemes, and decommit size consist of four decommitments plus two commitments).

However, the D combiner has one significant drawback: it is $(3,4)$-robust, while the sharing combiner is $(2,3)$-robust. We can fix this by using only three commitment schemes, but using each of them *twice*, by connecting in parallel three cascades of two schemes each; we call this the E combiner.

The E combiner uses three candidate commitment schemes, $\mathcal{C}_0, \mathcal{C}_1$ and $\mathcal{C}_2$. Specifically, the E combiner is the copy combiner applied to three cascades of pairs of candidate schemes:

$$E(\mathcal{C}_0, \mathcal{C}_1, \mathcal{C}_2) = (\mathcal{C}_0 \circ \mathcal{C}_1)||(\mathcal{C}_1 \circ \mathcal{C}_2)||(\mathcal{C}_2 \circ \mathcal{C}_0)$$

Theorem 5. *The D and E combiners are robust for the* Binding *and* Hiding *specifications of single round, length-uniform commitment schemes.*

Proof: Cascade is a robust combiner of single round, length-uniform commitment schemes, for the hiding specification, and preserves binding (Lemma 3). The copy combiner is robust for binding and preserves hiding (Lemma 4). □

The E combiner extends to a $(k, 2k-1)$-robust combiner, for any $k \geq 2$, by applying the copy combiner over all cascades of k different schemes (out of the set of $2k-1$ schemes); however, the overhead may be excessive.

While we discussed the E combiner only for commitment schemes, it seems easy to define similar combiners for other schemes, e.g. for SignCryption [ADR02] and authenticated-encryption [RBB03] schemes. We do not include the details as they are essentially the same as presented above, and furthermore the resulting constructions do not seem to have any special advantages.

7 Conclusions

We presented simple and efficient robust combiners for several basic cryptographic schemes; some of these combiners were 'folklore', for many years. We believe that efficient robust combiners are an important requirement from practical cryptographic primitives; put differently, we should prefer specifications with an efficient robust combiner. Additional research may find robust combiners for other schemes, or prove that some schemes cannot be combined (in efficient and robust manner). Extensions to the robust combiner concept are also of interest, such as handling schemes with side channels (e.g. computational time and power requirements).

Acknowledgment

Many thanks to Haya Shulman, who helped me with careful scrutiny and gave me numerous and very useful comments. Thanks also to Mihir Bellare, Ran Canetti, Shai Halevi, Kath Knobe, Yehuda Lindell, Mark Manulis, Boaz Patt-Shamir, Dominique Unruh, Avi Wigderson and anonymous referees for helpful comments and discussions. This work was supported in part by Israeli Science Foundation grant ISF 298/03-10.5.

References

[AB81] C. A. Asmuth and G. R. Blakley. An efficient algorithm for constructing a cryptosystem which is harder to break than two other cryptosystems. *Comp. and Maths. with Appls.*, 7:447–450, 1981.

[ADR02] J.H. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. *Advances in Cryptology - Eurocrypt 2002*, 2332:83–107, 2002.

[BB06] Dan Boneh and Xavier Boyen. On the impossibility of efficiently combining collision resistant hash functions. In *Advances in Cryptology – CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 570–583. Berlin: Springer-Verlag, 2006. Available at http://www.cs.stanford.edu/ xb/crypto06b/.

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Message authentication using hash functions: the HMAC construction. *CryptoBytes*, 2(1):12–15, Spring 1996.

[BDPR98] Mihir Bellare, Anand Desai, David Pointcheval, and Phillip Rogaway. Relations among notions of security for public-key encryption schemes. In *Advances in cryptology – Crypto 1998*, pages 26–45, 1998.

[Ble98] D. Bleichenbacher. Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS #1. In *Advances in Cryptology – Crypto 1998*, volume 1462 of *Lecture Notes in Computer Science*, pages 1–12, 1998.

[BR06] Mihir Bellare and Phil Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. *Advances in Cryptology – Eurocrypt 2006*, pages 409–426, 2006.

[CKN03] Ran Canetti, Hugo Krawczyk, and Jesper Nielsen. Relaxing chosen-ciphertext security. *Advances in Cryptology – Crypto 2003*, 2729:565–582, 2003.

[DA99] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246, January 1999.

[DK94] I. Damgrd and L. Knudsen. Enhancing the strength of conventional cryptosystems, November 1994.

[DK05] Yevgeniy Dodis and Jonathan Katz. Chosen-ciphertext security of multiple encryption. In *Theory of Cryptography Conference (TCC)*, pages 188–209, 2005.

[EG85] S. Even and O. Goldreich. On the power of cascade ciphers. *ACM Tras. Computer Systems*, 3:108–116, May 1985.

[Gol01] Oded Goldreich. *The Foundations of Cryptography, Volume 1 (Basic Tools)*. Cambridge University Press, June 2001.

[Gol04] Oded Goldreich. *The Foundations of Cryptography, Volume 2 (Basic Applications)*. Cambridge University Press, 2004.

[Her02] Amir Herzberg. Tolerant combiners: Resilient cryptographic design. Report 2002/135 in Cryptology ePrint Archive, http://eprint.iacr.org/, August 2002.

[HILL99] Johan Hastad, Russell Impagliazzo, Leonid A. Levin, and Michael Luby. A pseudorandom generator from any one-way function. *SIAM J. Comput.*, 28(4):1364–1396, 1999.

[HKN+05] D. Harnik, J. Kilian, M. Naor, O. Reingold, and A. Rosen. On Robust Combiners for Oblivious Transfer and other Primitives. In *Advances in Cryptology – Eurocrypt 2005*, volume 5, pages 96–113. Springer, 2005.

[HL92] Amir Herzberg and Michael Luby. Pubic randomness in cryptography. In Ernest F. Brickell, editor, *Proc. CRYPTO 92*, pages 421–432. Springer-Verlag, 1992. Lecture Notes in Computer Science No. 740.

[Jou04] Antoine Joux. Multicollisions in iterated hash functions. application to cascaded constructions. In *Advances in Cryptology – CRYPTO 2004*, volume 3152, pages 306–316. Springer, 2004.

[Kra93] Hugo Krawczyk. Secret sharing made short. In *Advances in Cryptology – Crypto 1993*, volume 773, pages 136–146. Springer, August 1993.

[Kra01] Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is SSL?). In *Advances in Cryptology – Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331, 2001.

[KY06] Jonathan Katz and Moti Yung. Characterization of security notions for probabilistic private-key encryption. *J. Cryptology*, 19(1):67–95, 2006.

[Lev87] L.A. Levin. One way functions and pseudorandom generators. *Combinatorica*, 7(4):357–363, 1987.

[LV01] Arjen K. Lenstra and Eric R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology*, 14(4):255–293, 2001.

[MM93] U.M. Maurer and J.L. Massey. Cascade Ciphers: The Importance of Being First. *Journal of Cryptology*, 6(1):55–61, 1993.

[MP06] R. Meier and B. Przydatek. On robust combiners for private information retrieval and other primitives. In *Advances in Cryptology – CRYPTO 2006*, volume 6, pages 555–569, 2006.

[MPW07] Remo Meier, Bartosz Przydatek, and Jürg Wullschleger. Robuster combiners for oblivious transfer. In Salil P. Vadhan, editor, *proc. of 4th Theory of Cryptography Conference, TCC 2007*, volume 4392 of *Lecture Notes in Computer Science*, pages 404–418. Springer, February 2007.

[NY89] M. Naor and M. Yung. Universal one-way hash functions and their cryptographic applications. In *Proc. 21st ACM Symp. on Theory of Computing*, pages 33–43, Seattle, 1989. ACM.

[Pie07] Krzysztof Pietrzak. Non-trivial black-box combiners for collision-resistant hash-functions don't exist. In Moni Naor, editor, *Advances in Cryptology – Eurocrypt 2007, 26th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Barcelona, Spain, May 20-24, 2007, Proceedings*, volume 4515 of *Lecture Notes in Computer Science*, pages 23–33. Springer, 2007.

[Pre93] B. Preneel. *Analysis and design of cryptographic hash functions*. PhD thesis, Katholieke Universiteit Leuven, 1993.

[RBB03] P. Rogaway, M. Bellare, and J. Black. OCB: A block-cipher mode of operation for efficient authenticated encryption. *ACM Transactions on Information and System Security (TISSEC)*, 6(3):365–403, 2003.

[Rom90] J. Rompel. One-way functions are necessary and sufficient for secure signatures. *Proceedings of the twenty-second annual ACM symposium on Theory of computing*, pages 387–394, 1990.

[RS04] Rogaway and Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In *IWFSE: International Workshop on Fast Software Encryption*, volume 3017, pages 371–388. Springer, 2004.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Sho00] Victor Shoup. Using hash functions as a hedge against chosen ciphertext attack. In *Advances in Cryptology – Eurocrypt 2000*, pages 275–288, 2000.

[Sho01] Victor Shoup. A proposal for an ISO standard for public key encryption. Report 2001/112, Cryptology ePrint Archive, December 2001. Version 2.1.

[Som06] Christian Sommer. Robust combiners for cryptographic protocols. Master's thesis, Computer Science Department, ETH Zurich, 2006.

[Unr06] Dominique Unruh. E-mail communication, November 2006.

[ZHSI04] Rui Zhang, Goichiro Hanaoka, Jonji Shikata, and Hideki Imai. On the Security of Multiple Encryption or CCA-security+ CCA-security= CCA-security? *Proceedings of Public Key Cryptography (PKC'04)*, pages 360–374, March 2004.