On Tolerant Cryptographic Constructions

draft – comments appreciated

Amir Herzberg

Computer Science Department

Bar Ilan University

http://www.cs.biu.ac.il/~herzbea/

Abstract

We investigate how to construct secure cryptographic schemes, from few candidate schemes, some of which may be insecure. Namely, tolerant constructions tolerate the insecurity of some of the component schemes used in the construction. We define tolerant constructions, and investigate `folklore`, practical cascade and parallel constructions. We prove cascade of encryption schemes provide tolerance for indistinguishability under chosen ciphertext attacks, including a `weak adaptive` variant. Similarly, certain parallel constructions ensure tolerance for unforgeability of Signature/MAC schemes, OWF, ERF, AONT and certain collision-resistant hash functions. We present (new) tolerant constructions for (several variants of) commitment schemes. Our constructions are simple, efficient and practical. To ensure practicality, we use concrete security analysis (in addition to the simpler asymptotic analysis).

1 Introduction

Most cryptographic functions do not have an unconditional proof of security. The classical method to establish security is by cryptanalysis i.e. accumulated evidence of failure of experts to find weaknesses in the function. 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, it is impossible to predict the rate or impact of future cryptanalysis efforts; a mechanism which was attacked unsuccessfully for years may abruptly be broken by a new attack[1]. Hence, it is desirable to design systems to be *tolerant* of cryptanalysis and vulnerabilities (including known trapdoors). A tolerant cryptographic system remains secure following successful cryptanalysis of one or more cryptographic subsystems it contains. Tolerance does not imply unconditional-security; however, it would hopefully provide sufficient advanced-warning time to replace broken cryptographic components.

Many cryptographic systems and constructions use redundant components in the hope of achieving tolerance. The most familiar such construction is cascade. Cascading of cryptosystems is very natural; novices and experts alike believe that the cascade $E \circ E'$ of two cryptosystems E,E' is at least as secure as the more secure of the two, hopefully even more secure than both. Indeed, cascading of cryptosystems has been a common practice in cryptography for hundreds of years.

However, so far, there are few publications on tolerant cryptographic constructions. In [AB81], Asmuth and Blakely present a simple construction of a randomized cryptosystem from two component ciphers, with the hope of achieving tolerance; proof of security was given only in [GM84]; see variant for block ciphers in [HP86]. The highly related problem of cascading of block ciphers received some attention. Even and Goldreich showed that keyed cascade ensures tolerance against message recover attacks on block ciphers [EG85, Theorem 5], and conjectured that the result holds for other specifications of ciphers. Damgard and Knudsen [DK94] proved that it holds for security against key-recovery under chosen-plaintext attacks. Maurer and Massey [MM93] claimed that the proof in [EG85] "holds only under the uninterestingly restrictive assumption that the enemy cannot exploit information about the plaintext statistics", but we disagree. We extend the proof of [EG85] and show that, as expected intuitively and in

[1] In practice, we try to use conservative estimates of progress in cryptanalysis, based on past progress and other factors; see e.g. [LV01].

[EG85], keyed cascading provides tolerance to many confidentiality specifications, not only of block ciphers but also of other schemes such as public key and shared key cryptosystems. Our proof uses a strong notion of security under indistinguishability test – under plaintext only and non-adaptive chosen ciphertext attack (CCA1), as well as weak version of adaptive chosen ciphertext attack (wCCA2). On the other hand, we note that cascading does *not* provide tolerance for *adaptive* chosen ciphertext attack (CCA2), or if the length of the output is not a fixed function of the length of the input. This shows the importance of backing the intuition with analysis and proof.

Tolerance is relevant to any cryptographic scheme, not just for confidentiality. In particular, it is widely accepted that the parallel construction $g(x)||f(x)$, using the same input x to both functions, ensures tolerance for several *integrity* properties, such as (several variants of) collision-resistant hashing as well as Message Authentication Codes (MAC) and digital signatures. We prove that the parallel construction indeed provide tolerance for such integrity specifications. The parallel construction is used, for tolerance, in practical designs and standards, e.g. in the W3C XML-DSIG specifications and in the TLS protocol [RFC2246].

Once we realized the importance of using tolerant cryptographic designs, we began looking for tolerant constructions to different cryptographic goals (specifications). We present several results in this work, but this work is far from complete, and we speculate that some cryptographic specifications may simply not allow (efficient) tolerant constructions; in such cases, one may look for alternate specifications that provide the necessary functionality for most practical scenarios and applications, yet allow efficient tolerant constructions.

We wish to stress that efficiency is very critical for tolerant constructions; implementers will rarely be willing to tolerate performance loss, `just` in order to tolerate potential vulnerabilities in a cryptographic function. In fact, if would ignore efficiency, then it may be possible to ensure tolerance by using provable constructions of cryptographic mechanisms from few `basic` cryptographic mechanisms, which have simple tolerant designs. For example, many cryptographic mechanisms can be constructed from one-way functions; and we observe that it is sufficient that <u>one</u> of $\{g, f\}$ is a one-way function, to ensure that $g(x)||f(x')$ is also a one-way function. Provably-secure constructions based on one-way functions exist for many cryptographic mechanisms, e.g. pseudo-random generators [Go01,HILL99] and signature schemes [NY89]. Therefore, by using a tolerant construction of one way function (from multiple candidate one-way functions) as the basis of some cryptographic scheme, the scheme retains the proven security properties even if one of the candidate one-way functions is not secure. However, such constructions are often inefficient, and involve unacceptable degradation in security parameters (e.g., require absurd key and/or block sizes). To quantify loss in security and efficiency due to the constructions, we use concrete security measures, following [HL92, BKR94,BDJR97].

Our contributions. We consider our main contribution in the identification and formalization of *cryptographic tolerance* as a criteria for cryptographic specifications, and goal for constructions. Some additional contributions include:

- **Precise analysis of the security of several `folklore` constructions**. In particular, we show that cascade encryption indeed ensures tolerance – as long as each component encryption has fixed output length (for fixed length input), and for several variants of indistinguishability including a weak form of adaptive chosen ciphertext attack (weak CCA2), but *not* for the `regular` CCA2 specification. We note that the `multiple encryption` construction of [DK03] seems to ensure tolerant encryption for CCA2, but at significant overhead (ciphertext length more than doubles), which may be unacceptable for many applications.
- **Efficient, practical constructions for commitment schemes**. To our knowledge, these are the first provably-secure tolerant constructions of general cryptographic functions, beyond the folklore constructions, and few additional cryptographic constructions proven secure based on validity of either of two (specific) `hardness` assumptions, e.g. [Sh00, O92].
- **Compositions of constructions**. We define compositions of multiple constructions, to combine the benefits of different constructions. We also present a generic composition based on a simple

combinatorial object (composition structure). Finally, we use these to compose the cascade and parallel constructions, creating two efficient composite constructions for commitment schemes.

2 Notations and Definitions

We first fix some notations and recall the (standard) cryptographic definitions, for schemes to which we present tolerant constructions. We then define the concept of tolerant constructions.

2.1 Cryptographic Functions and Schemes

We find it convenient to define tolerant constructions for functions, taken from some general set F of (cryptographic) functions. However, much of our results deal with (*cryptographic*) *schemes*[2], which are finite sets of functions; if a function is `randomized`, we write the randomization bits as explicit input, for clarity and to facilitate concrete security analysis. We represent schemes by a single function, with an additional input parameter π choosing one of the functions in the scheme. We refer to a specific function in the scheme using *dot notation*, e.g. $\mathcal{S}.\pi$ refers to function of scheme $\mathcal{S}$. We next define some cryptographic functions and schemes, beginning with encryption.

Encryption. An *encryption scheme* $\mathcal{E}$ consists of three functions $<KG,E,D>$ (for key generation, encryption and decryption, respectively). The key generation function $\mathcal{E}.KG$ accepts as input a random string, and its output is a pair[3] of keys: e, d for encryption and decryption, respectively. We again use *dot notation* to refer to particular key, e.g. $e=\mathcal{E}.KG.e(r)$ returns the encryption key returned by $\mathcal{E}.KG$ on input r. We use subscripts to denote keys, and the random input to the encryption function. Encryption of message m, where $m \in M$ for some *message space* M, using key e and randomness r, is simply $\mathcal{E}.E_{e,r}(m) \in C$, where C is the *ciphertext space*. The decryption function $\mathcal{E}.D$ accepts as input ciphertext $c \in C$, and key d, and returns a message $m' \in M$ or a failure indicator $\perp$. The correctness requirement is $\mathcal{E}.D_d(\mathcal{E}.E_{e,r}(m))=m$, for any $m \in M$, $r \in \{0,1\}^*$ and r_{KG} such that $e=\mathcal{E}.KG.e(r_{KG})$, $d=\mathcal{E}.KG.d(r_{KG})$.

Security of Encryption. To define security for encryption schemes, we use the standard `indistinguishability experiment` approach of [BDJR97, GM84], but extend their definitions as follows:

- Since our results apply to both shared-key and public-key encryption, we use a flag *ISPUB* to signal when the encryption key is public.
- For quantitative security analysis, we bound the capabilities of the adversary (the `attack model`), including the total running time t, random bits ρ_A, the number of oracle queries (as an array *q[phase, oracle]*, e.g. *q[find, D]=0* disallows adaptive chosen-ciphertext queries) and the length of ciphertext (l) and plaintext (l-Δ) in queries. Previous definitions were asymptotic only or did not include all parameters.
- Cascade encryption is insecure for adaptive chosen ciphertext attack (CCA2). Recently, [DK03] presented constructions for `multiple encryption` schemes that appear to be tolerant for CCA2, but have significant overhead; we believe that in many applied scenarios, this overhead would not be acceptable. We found that cascade encryption is secure under a *weak-CCA2* attack model, where the attacker can chose ciphertext adaptively, but if the decrypted plaintext is one of the two chosen `test` plaintexts, then the oracle returns a special `bingo` signal, but does not identify the plaintext; we call this the `weak decrypt` oracle. Namely:
  $\mathcal{E}.wD_{d,p[0],p[1]}(c)=\{m:= \mathcal{E}.D_d(c)$; *if* $m=p[0]$ *or* $m=p[1]$ *return `bingo`; else return* $m;\}$
  Weak-CCA2 follows the criticism of [ADR02] on `regular` CCA2, but is even weaker than their

[2] Some definitions of cryptographic schemes are not as a collection of functions, but as a collection of probabilistic algorithms or machines. Often we can view them as functions with additional inputs for randomness and/or state.

[3] Notice public and shared secret key schemes share the same syntax, i.e. e can be either public or secret, e.g. $e=d$.

gCCA2 notion[4]; still, it may be sufficient in practice, in particular it allows the practical `feedback only CCA` attacks of [B98,K01].

Definition 2-1 [Indistinguishability Experiment] Let $\mathcal{E}$ be an encryption scheme and let $k, l, t, \rho, \Delta, \rho_A \in \mathcal{N}$, $ISPUB \in \{T/F\}$ and $q:\{select,find\}\times\{E,D,wD\} \rightarrow \mathcal{N}$. Let A^O be an (adversarial) algorithm with access to oracle O. Let $\mathbf{IndExp}_{A,\mathcal{E},ISPUB}(k,q,l,\Delta,t,\rho,\rho_A)$ be the following experiment:

(1) $r_{ed} \in_R \{0,1\}^k$; $e = \mathcal{S}.KG.e(r_{ed})$; $d = \mathcal{S}.KG.d(r_{ed})$

(2) Let O be an oracle to the functions: $\{\mathcal{E}.E_{e,r}, \mathcal{E}.D_d, \mathcal{E}.wD_{d,p[0],p[1]}\}$

(3) If $ISPUB=T$ then $e'=e$ else $e'=\lambda$

(4) $(p[0],p[1],state) \leftarrow A^O(\text{“select”}, e', 1^k)$; /* select phase */

(5) $b \in_R \{0,1\}$;

(6) $r \in_R \{0,1\}^\rho$; $c = \mathcal{E}.E_{e,r}(p[b])$;

(7) $\beta = A^O(\text{“find”}, c, state)$; /* find phase */

(8) Return “*win*” only if *all* of the following conditions hold, otherwise return “*loss*”:

a. $\beta = b$, and

b. $|p[1]| = |p[0]| \leq l-\Delta$, and

c. total running time of A^O is less than t, and

d. A^O makes at most $q[\varphi,f]$ calls to oracle $\mathcal{E}.f$ at phase $\varphi \in \{select, find\}$, and

e. A^O uses at most ρ_A random bits, and

f. A^O does <u>not</u> make oracle query $\mathcal{E}.D_d(c)$ during select phase, and

g. in its oracle queries, A^O uses m, c s.t. $|m| \leq l-\Delta$ and $|c| \leq l$.

The confidentiality specifications depend on the maximal *advantage a* for the adversary.

Definition 2-2 $\mathcal{E}$ satisfies **specification** $\boldsymbol{IND_{ISPUB}(a,k,q,l,\Delta,t,\rho,\rho_A)}$ if for every adversary A holds $\Pr[\mathbf{IndExp}_{A,\mathcal{E},ISPUB}(k,q,l,\Delta,t,\rho,\rho_A)=\text{“win”}] < \frac{1}{2}+a$.

We now also present asymptotic, polynomial-time complexities. Allowing polynomial number of each type of queries, possibly restricting queries to D for the `weaker` notions (cf. to CCA2), i.e. CPA, CCA1 and wCCA2.

Definition 2-3 $\mathcal{E}$ satisfies specification $CCA2\text{-}IND_{ISPUB}$ if $\mathcal{E} \in PPT$ and for any strictly positive polynomials $l, \Delta, t, \rho, \rho_A, a$ and positive polynomials $q[\varphi,f]$ for $\varphi \in \{select, find\}$ and $f \in \{E,D\}$, exists some integer k_0 such that for every $k \geq k_0$, holds: $\Pr[\text{IndExp}_{A,\mathcal{E},ISPUB}(k,q(k),l(k),\Delta(k),t(k),\rho(k),\rho_A(k))=\text{“win”}] < \frac{1}{2}+a(k)$. We say that $\mathcal{E}$ satisfies specifications $wCCA2\text{-}IND_{ISPUB}$, $CCA1\text{-}IND_{ISPUB}$, $CPA\text{-}IND_{ISPUB}$, respectively, if $\mathcal{E}$ satisfies $CCA2\text{-}IND_{ISPUB}$ restricted to $q[\text{“find”},D]=0$, $q[\text{“find”},D]=q[\text{“find”},wD]=0$, or also $q[\text{“select”},D]=0$)

Commitment. A (non-interactive) *commitment scheme* $\mathcal{C}$ consists of four functions $<KG,C,D,O>$ (for key generation, commit, decommit and open, respectively). The key generation function $\mathcal{C}.KG$ accepts as input a random string, and its output is a public commitment key ck. The commitment and decommit functions $\mathcal{C}.C$, $\mathcal{C}.D$ have both three inputs: a message $m \in M$, a public commitment key ck and randomness r, and their respective outputs are: a commitment $\mathcal{C}.C_{ck,r}(m)$ and a decommitment $\mathcal{C}.D_{ck,r}(m)$. The open

[4] Is cascade tolerant for gCCA? Is some comparably-efficient construction tolerant under CCA2 (or at least gCCA)? We think not, but have to leave these questions open at this time.

function $C.O$ has the same inputs[5], plus the commitment and decommitment values (c, d respectively), and its output is either a message $m' \in M$ or a failure indicator $\perp$. The correctness requirement is $C.O_{ck}(m,C.C_{ck,r}(m), C.D_{ck,r}(m))=m$, for any $m \in M$, r and r_{KG} such that $ck=C.KG(r_{KG})$, namely $C.O$ (open), when given all necessary inputs (message, commitment and decommitment), returns the original message.

Security of Commitment. Commitment schemes have a confidentiality property, called *hiding,* and an integrity property, called *binding.* We only sketch the asymptotic definitions; the final version of this manuscript will contain complete (and concrete) definitions.

Hiding. No PPT adversary can distinguish the commitments of any two messages of its choice.

Binding. Given (random) ck, every PPT adversary A has negligible probability of finding a *collision,* i.e. values c,d,d',m,m' s.t. $C.O_{ck}(m,c,d))=m$ and $C.O_{ck}(m',c,d'))=m'$ (notice the commitment c is the same!).

Following [ADR02], we also consider *relaxed binding,* where A has negligible probability of finding a message m s.t. when given $c=C.C_{ck,r}(m)$ and $d=C.D_{ck,r}(m)$, the PPT adversary A can find m', d' s.t. $C.O_{ck}(m',c,d'))=m'$. As motivated in [ADR02], known constructions for commitment schemes can use UOWHF for relaxed binding, but require the (strictly stronger) CRHF for (strict) binding. We show later a construction which is tolerant for both versions of binding (`strict` or relaxed).

Our construction is also tolerant for *trapdoor commitments* [BCC88], or *chameleon hash functions* [KR00]. In these schemes, the key generation produces also a secret *trapdoor key* $C.KG.t$. The schemes also define a *Switch* algorithm, which uses the trapdoor key, to transform any valid commitment to any message m to an indistinguishable commitment to any other message m' (adversary may chose both m and m').

Signature and MAC. A signature/MAC scheme[6] S consists of three functions $<KG,S,V>$ (for key generation, sign, and verify, respectively), where V returns a binary value (0 for false, 1 for Ok). The key generation function $E.KG$ accepts as input a random string, and its output is a pair of keys: s, v for signature and verification, respectively. The correctness requirement is $(S.V_v(S_{s,r}(m),m)=1)$, for any $m \in M$, r and r_{KG} such that $s=E.KG.s(r_{KG})$, $v=E.KG.v(r_{KG})$.

Security of Signature, MAC. As mentioned in the introduction, it is quite trivial, and known by `folklore`, that signature/MAC schemes are tolerant under parallel construction. More precisely, we prove that signature/MAC schemes are tolerant under `copy-concatenate` parallel construction, for the strong *existential unforgeability under adaptive chosen message attack* specification (but, the argument holds for most other notions of security for signature/MAC schemes).

A signature/MAC scheme S ensures *existential unforgeability under adaptive chosen message attack* if a PPT adversary given oracle to $S.S_{s,r}(\ )$ has negligible probability of producing (σ,m) s.t. $V_v(\sigma,m)=1$, if σ was never returned by $S.S_{s,r}(m)$. The probability is taken over the coin tosses of the adversary, and the random choice of r and r_{KG}, and with $s=E.KG.s(r_{KG})$, $v=E.KG.v(r_{KG})$. In MAC schemes, v is secret, while in signature schemes, the adversary is also given v.

Weakly Collision Resistant Hash Function. A PPT computable function $h:\{0,1\}^* \rightarrow \{0,1\}^k$, is said to be *weakly collision resistant hash function,* or to maintain specification *WCRHF*, if for PPT adversary A holds $Pr(A(x)=y \mid (y \neq x) \wedge (h(x)=h(y)) \approx 0$, where the probability is over the coin tosses of A and over random choice of $y \in_R \{0,1\}^l$, for some sufficiently large l.

Exposure-Resilient Function. A PPT computable function $f:\{0,1\}^n \rightarrow \{0,1\}^k$, is said to be *t-ERF* (Exposure-Resilient Function) if any PPT adversary cannot distinguish between the output of f applied to a random input, and a truly random k bit string, even if given any l bits from the input of f.

---

[5] Most definitions of open functions of commitment schemes do not include a message input, only commitment, decommitment and key. However, in practice, often the largest or only part of the decommitment is simply the message. By allowing the open funtion to depend also on the message, we can express the fact that practical commitment schemes usually output short commitments and empty or short decommitments; this is significant when designing tolerant constructions.

[6] Notice public and shared key schemes share the same syntax.

AONT. An All-Or-Nothing Transform (AONT) is a pair of PPT algorithms, T (the Transform) and I (the Inverter). The transform T has two inputs, a k-bit message m and a random string r. Its output consists of a pair *(secret,public),* called the *secret part* and *public part,* where the length of the secret part is usually fixed and denoted s. The inverter I has also two parameters, and its output is a message m or a special signal $\perp$ indicating failure to invert. The correctness requirement is simply that for every $m \in \{0,1\}^k$, and every random r, holds: $I(T(m,r))=r$.

We say that f satisfies the security specification $AONT(k,s,l)$ if there are no messages m, m' and some PPT adversary *distinguisher,* which can with significant probability distinguish between $f(m)$ and $f(m')$, given all *but* l bits of the secret part.

2.2 Performance specifications

For asymptotic security analysis, it is sufficient to require all algorithms to be probabilistic polynomial time. However, to allow concrete security analysis of constructions, we need concrete bounds on the complexities of the schemes. In this version of the work, we present such bounds (and concrete analysis) only for encryption schemes, as follows.

Definition 2-4 [Concrete complexity bounds for encryption schemes] Let $k, k', l, \Delta, \rho \in \mathcal{N}$, with $l>\Delta$, and let $\tau:\{KG,E,D\} \rightarrow \mathcal{N}$. Then for every encryption scheme $\mathcal{E}$ we define predicate $bounds[k, k', l, \Delta, \rho, \tau](\mathcal{E})$ as *True* if and only if:

1. For inputs of length up to k, $\mathcal{E}.KG(k)$ is computable in time $\tau[KG]$ and $|\mathcal{E}.KG(k)| \leq k'$.
2. There is a deterministic algorithm that computes $\mathcal{E}.E_{e,r}(m)$ in time $\tau[E]$, for every $e \in \{0,1\}^{k'}$ and every m s.t. $|m| \leq l-\Delta$, and reads up to the first ρ bits of r; also, $|\mathcal{E}.f_{e,r}(m)| \leq l$.
3. There is a deterministic algorithm that computes $\mathcal{E}.D_d(c)$ in time $\tau[D]$, for every $d \in \{0,1\}^{k'}$ and every c s.t. $|c| \leq l$; also, $|\mathcal{E}.D_{d,r}(c)| \leq l$.

2.3 Tolerant Constructions

We are interested in specifications (properties) of functions, including concrete security specifications and asymptotic security specifications. We define specifications simply as binary predicates over the set of functions F. Let $S(F)$ be the set of all specifications (predicates) over F. We say that $f \in F$ satisfies $s \in S(F)$ iff $s(f)=1$.

We say that a mapping c of p functions $f_1,\ldots f_p$ into a single function $c(f_1,\ldots,f_p)$, i.e. $c : F^p \rightarrow F$, is a *construction of plurality p over F.* Construction c is tolerant if $c(f_1,\ldots,f_p)$ satisfies some specification s' as long as a sufficient subset of $f_1,\ldots f_p$ satisfy specifications $s_1,\ldots s_p$, respectively (often, all specifications are identical, i.e. $s=s_1=\ldots=s_p$ and also often $s=s'$). To complete this definition, we need to identify the sufficient subset of $f_1,\ldots f_p$; following the works on secret-sharing, we define two variants of tolerance: based on threshold t ($0 \leq t<p$), and based on general access structure $\Gamma \subseteq P(\{1,\ldots p\})$ (Γ is a set of subsets of $\{1,\ldots p\}$). In addition, we often require that *all* of the candidate functions f_i satisfy some minimal specifications $b \in S(F)$, such as bounds on their complexities.

Definition 2-5 Consider some set of functions F, integer p, predicates $s', b, s_1,\ldots s_p \in S(F)$ and construction $c : F^p \rightarrow F$ of plurality p over F. Construction c is *t-tolerant for* $(s_1,\ldots s_p) \rightarrow s'$, *with threshold* $t<p$, if $s'(c(f_1, \ldots,f_p))$ holds provided $\sum_{i=1}^{p} s_i(f_i) \geq p-t$. Construction c is *Γ-tolerant for* $(s_1,\ldots s_p) \rightarrow s'$, *with access structure* $\Gamma \subseteq P(\{1,\ldots p\})$, if $s'(c(f_1, \ldots,f_p))$ holds provided for some $\lambda \in \Gamma$ holds $(i \in \lambda) \rightarrow s_i(f_i)=1$. Construction c is *t-tolerant* (*Γ-tolerant*) with *prerequisite b* if $s'(c(f_1, \ldots,f_p))$ holds provided $\left(\sum_{i=1}^{p} b(f_i) = p\right) \wedge \left(\sum_{i=1}^{p} s_i(f_i) \geq p-t\right)$ (respectively, $\left(\sum_{i=1}^{p} b(f_i) = p\right) \wedge \left(\exists \lambda \in \Gamma\right)\left(\sum_{i \in \lambda} s_i(f_i) = |\lambda|\right)$). If c is *0-tolerant for* $(s_1,\ldots s_p) \rightarrow s'$ then we say that c preserves $(s_1,\ldots s_p) \rightarrow s'$ (with or without prerequisites). If c is t-tolerant for $(s,\ldots s) \rightarrow s'$, then we say that c is t-tolerant for $s \rightarrow s'$; if $t=0$ then we say that c *preserves* $s \rightarrow s'$. If $s=s'$ then we say that c is t-tolerant for (or preserves) s. Finally, if c is *(p-1)-tolerant,* then we simply say that c is *tolerant*. ■

3 Cascade Constructions and their Tolerance

The most basic tolerant construction of cryptographic functions is the cascade construction $c_\circ$. We begin by discussing `simple cascading', which is cascading of functions with a single input and output, such as hash functions, namely $c_\circ(f,g)[x]=f\circ g(x)=f(g(x))$. In the following subsections we discuss cascading of keyed schemes.

3.1 Simple Cascade of Keyless Cryptographic Functions

Consider any two functions $g:D_g \rightarrow R_g$, $f:D_f \rightarrow R_f$ s.t. $R_g \subseteq D_f$. The simple cascade of f and g, denoted $f\circ g$ or $c_\circ(f,g)$, is a construction of plurality 2 defined as $c_\circ(f,g)=f\circ g(x)=f(g(x))$. Unfortunately, simple cascade rarely ensures tolerance, and often does not even preserve cryptographic specifications. So far, we found simple cascade ensures tolerance only to the one-way property, and that with a prerequisite requirement *perm(f),* which is true only if f is a permutation when restricted to input domains $\{0,1\}^l$ for some length l.

Lemma 3-1 Keyless cascade of two functions is…

1. 1-tolerant for specifications *OWF* with prerequisite *perm.*
2. 1-tolerant with prerequisite *perm* for specifications:
   $concrete\text{-}OWF^f(a,k,\rho_A, \tau_A) \wedge [Time(f,k) \le \tau_F] \rightarrow concrete\text{-}OWF^f(a,k,\rho_A, \tau_{A+}\tau_F) \wedge [Time(f,k) \le 2\tau_F]$
3. Preserves (0-tolerant), but <u>not</u> 1-tolerant, for specifications ERF (exposure resilient function) and AONT (all or nothing transform).
4. Not (even) *0*-tolerant for specifications *OWF* and *WCRHF*.

Proof: The negative claims (4 and part of 3) follow by simple exmples, e.g. to prove claim 4, let h be a OWF and/or WCRHF. Let $g(x)=h(x)||0^{|h(x)|}$ and $f(x)=\begin{cases} 0 & if\ x = y0^{|x|/2} \\ h(x) & else \end{cases}$. Trivially, both f and g are OWF and/or WCRHF, respectively, yet $f\circ g$ is neither OWF not WCRHF; in fact, $f\circ g(x)=0$ for every x.

Claims 1 and 2 follow from a simple reduction argument; the proof of claim 2 is in the appendix (and claim 1 immediately follows from claim 2). ■

We believe that we can generalize claims 1 and 2 for an appropriate family of regular functions. It would be interesting to find additional cryptographic specifications of keyless functions for which cascade provides tolerance.

3.2 Cascade Encryption is Tolerant

The cascade encryption, i.e. cascade of two[7] encryption schemes $\mathcal{E}$, $\mathcal{E}'$, is denoted $c_E(\mathcal{E},\mathcal{E}')$ or $\mathcal{E}\circ\mathcal{E}'$ and defined as follows. **Notation:** For convenience we explicitly write the inputs and outputs to the cascade (or any composition) as a tuple of inputs or outputs when appropriate, e.g. $<r,r'>$ to denote the pair of two random inputs (r to $\mathcal{E}$ and r' to $\mathcal{E}'$).

- $\mathcal{E}\circ\mathcal{E}'.KG.e(<r,r'>)=<\mathcal{E}.KG.e(r),\mathcal{E}'.KG.e(r')>$, $\mathcal{E}\circ\mathcal{E}'.KG.d(<r,r'>)=<\mathcal{E}.KG.d(r),\mathcal{E}'.KG.d(r')>$.
- $\mathcal{E}\circ\mathcal{E}'.E_{<e,e'>,<r,r'>}(m)=\mathcal{E}.E_{e,r}(\mathcal{E}'.E_{e',r'}(m))$
- $\mathcal{E}\circ\mathcal{E}'.D_{<d,d'>}(c)=\mathcal{E}'.D_{d'}(\mathcal{E}.D_d(c))$

Cascade encryption is a construction of plurality 2; the following lemma bounds the complexities:

Lemma 3-2 Let $\mathcal{E}, \mathcal{E}'$ be a pair of encryption schemes such that for $s \in \{\mathcal{E},\mathcal{E}'\}$ holds $bounds[k, k', l, \Delta, \rho, \tau](s) = True$ with $l > 2\Delta$. Then $\mathcal{E}\circ\mathcal{E}'$ is an encryption scheme with $bounds[2k, 2k', l, 2\Delta, 2\rho, 2\tau](\mathcal{E}\circ\mathcal{E}') = True$. ■

[7] The definitions and proofs extend trivially to cascade of arbitrary number of schemes.

We now investigate the security and tolerance of cascade encryption. As noted in the introduction, cascade encryption is an ancient, widely-deployed technique, usually in the hope of improving security – e.g., providing tolerance to weaknesses of one of the two cascaded encryption schemes. Is this secure? This depends on the adversary capabilities (`attack model`). Cascade encryption is *not* tolerant for adaptive chosen ciphertext attack (CCA2); simply consider $\mathcal{E}'$ which ignores the least significant bit of the ciphertext, allowing adversary to decrypt the challenge ciphertext (by flipping the LSb and invoking the decryption oracle). Our answer is yes, but with some important restrictions. However, as [ADR02] argued, this `attack` is so contrived, that it may indicate that CCA2 is overly restrictive, rather than a problem with cascade encryption. In [ADR02], the authors present a slightly weaker definition, gCCA, but we do not think cascade is tolerant under that definition, either; on the other hand, the following lemma shows that cascade encryption *is* tolerant under a slightly more related definition, weak CCA (wCCA), as presented above.

Also, note that the indistinguishability experiment restricted the adversary to select plaintexts of the same length. Obviously, the length of the ciphertext should be indistinguishable between any two plaintexts (of the same length). For simplicity, we define a predicate *FixedExtra* over encryption schemes, such that *FixedExtra*($\mathcal{E}'$) hold if the length of the ciphertext depends only on the length of the plaintext and on the security parameter; this holds for all practical cryptosystems. Clearly, if the length of the output of $\mathcal{E}'$ differs for two plaintexts of the same length, then cascading it with a secure $\mathcal{E}$ may not be sufficient to ensure indistinguishability. We therefore require *FixedExtra*($\mathcal{E}'$) to hold.

Lemma 3-3 Cascade encryption is 1-tolerant with prerequisite *FixedExtra*($\mathcal{E}'$), for specifications ***wCCA2-IND$_{ISPUB}$, CCA1-IND$_{ISPUB}$, CPA-IND$_{ISPUB}$***. Furthermore, let $k, k', l, \Delta, \rho, t^o, \rho^o_A \in \mathcal{N}$ s.t. $l>2\Delta$, $\tau:\{KG,E,D\} \rightarrow \mathcal{N}$, $q:\{select,find\}\times\{E,D,wD\} \rightarrow \mathcal{N}$ s.t. $q[find,D]=0$. Then, c_E is also 1-tolerant with the additional prerequisite *bounds*$[k, k', l, \Delta, \rho, \tau]$, for specifications $IND_{ISPUB}(a,k,q,l,\Delta,t, \rho, \rho_A) \rightarrow IND_{ISPUB}(a,k,q,l,2\Delta,t^0, \rho, \rho^o_A)$, where $\rho_A=\rho^o_A+2k+2\rho$ and $t=t^o+\tau[KG]+2\tau[E]+\sum_{j\in\{find,select\}}\sum_{f\in\{E,D,wD\}} q[j,f]\tau[f]$.

Proof: The proof is by contradiction; namely assume that for some $\mathcal{E}$, $\mathcal{E}'$ holds $s^o(\mathcal{E}\circ\mathcal{E}')=False$, and we show that $s(\mathcal{E})=s(\mathcal{E}')=False$.

Since $s^o(\mathcal{E}\circ\mathcal{E}')=False$, then there is some adversary A^o such that $p^o \equiv \Pr[\mathbf{IndExp}_{A,\mathcal{E},ISPUB}(k,q,l,t^o,\rho,\rho^o_A)="win"] \geq \frac{1}{2}+a$. We next show that given such adversary A^o as a black box, we can construct adversaries A, A' such that $p \equiv \Pr[\mathbf{IndExp}_{A,\mathcal{E},ISPUB}(k,q,l_0,t,\rho,\rho_A)="win"] \geq p^o \geq \frac{1}{2}+a$ and $p' \equiv \Pr[\mathbf{IndExp}_{A',\mathcal{E}',ISPUB}(k,q,l_0,t,\rho,\rho_A)="win"] \geq p^o \geq \frac{1}{2}+a$, where $l_0=l+\Delta$.

Namely, we prove (in the appendix) the following claims A, A':

Claim A (A'): given adversary A^o such that $p^o \geq \frac{1}{2}+a$ as a black box, we can construct adversary A (respectively A') such that $p \geq p^o \geq \frac{1}{2}+a$ (respectively $p' \geq p^o \geq \frac{1}{2}+a$).

This completes the proof, by showing that indeed $s(\mathcal{E})=s(\mathcal{E}')=False$. ∎

Cascading is a natural candidate construction for many cryptographic mechanisms; we now define and investigate tolerance of cascade of commitment and MAC/Signature schemes.

3.3 Cascade Commitment

We define cascade commitment $c_c(C,C')$ (or $C\circ C'$), i.e. cascade of two commitment schemes C, C', as follows. (The final version will also contain the simple extension to trapdoor commitment schemes.) We again wrote inputs and outputs as tuples.

- $C\circ C'.KG(<r,r'>)=<C.KG(r),C'.KG(r')>$
- $C\circ C'.C_{<ck,ck'>,<r,r'>}(m)=C.C_{ck,r}(C'.C_{ck',r'}(m))$
- $C\circ C'.D_{<ck,ck'>,<r,r'>}(m)=<C.D_{ck,r}(C'.C_{ck,r}(m)) , C'.D_{ck',r'}(m), C'.C_{ck',r'}(m)>$

- $C \circ C'.O_{<ck,ck'>}(m,c,<d,d',c'>)=C'.O_{ck'}(m,c',d')$ *if* $C.O_{ck}(c',c,d)=c' \neq \perp$, *else* $\perp$

As the following lemma shows, cascade ensures the privacy (hiding) property of commitment schemes, but only preserves the integrity (binding) property.

Lemma 3-4 Cascade commitment is tolerant for the hiding specification, and preserves (but is *not* tolerant for) the binding specification.■

We next show that cascade also preserves, but does not ensure tolerance, for other integrity properties, specifically of MAC/Signature schemes.

3.4 Cascading preserves, but is not tolerant for, MAC/Signature Schemes

We define cascade MAC/Signature $c_{MAC/Sign}(S,S')$ (or $S \circ S'$), i.e. cascade of two MAC/Signature schemes S, S', as follows. We again write inputs and outputs as tuples.

- $S \circ S'.KG.e(<r,r'>)=<S.KG.e(r),S'.KG.e(r')>$; $S \circ S'.KG.e(<r,r'>)=<S.KG.e(r),S'.KG.e(r')>$
- $S \circ S'.S_{<s,s'>,<r,r'>}(m)=<S.S_{s,r}(m),S'.S_{s',r'}(m)>$
- $S \circ S'.V_{<v,v'>}(<\sigma,\sigma'>,m)=S.V_v(\sigma,m) \wedge S'.V_{v'}(\sigma',m)$

Lemma 3-5 Cascade MAC/Signature is 0-tolerant for (i.e. preserves) the *existential unforgeability under adaptive chosen message attack* specification.

4 Parallel Constructions and their Tolerance

We now consider another important family of constructions, which are parallel applications of two or more cryptographic functions or schemes. Parallel constructions may use the same input to all functions, use different parts of the input to each function, or use some combination of the inputs to create the input to each function, often involving XOR or secret-sharing. Similarly, the output of some parallel constructions is simply the concatenation of the outputs of each function, while others `merge` the outputs, by XOR or secret–sharing.

4.1 Split-Parallel-Concat Construction for OWF

Possibly the simplest parallel construction `splits` the input among several functions, and concatenates the result. In particular, the Split-parallel-Concat (*sc)* construction for two keyless functions *f, f'* is defined as $sc(f,f')[<x,x'>]=f||_{SC}f'(<x,x'>)=<f(x),f(x')>$. This trivial construction is tolerant for One-Way Functions specifications, using two or more functions.

Lemma 4-1 The Split-Parallel-Concet (*sc*) construction is tolerant for OWF specifications.

Proof: use argument for transforming a weak OWF into a strong OWF (see e.g. [Go01]). ■

4.2 Copy-Parallel-Concat Construction for Integrity Specifications

The Copy-parallel-Concat (*cc*) construction is also trivial and well-known, but it is very practical and widely deployed. Here, the input to the construction is `copied` and used as input to each of the components; and the output is simply the concatenation of the output of all components. This simple, folklore construction provides tolerance for the integrity properties of collision-resistant hash functions, signature/MAC schemes and commitment schemes.

Let us first define the *cc* construction for keyless functions, e.g. (weakly collision resistant) hash functions. The *Copy-parallel-Concat (cc) parallel construction* of single-input (keyless) functions *f,g* is denoted as $f||g$ or $c_{||}(f,g)$, and defined as $c_{||}(f,g)=f||g(x)=f(x)||g(x)$. When the functions have inputs for random bits and/or keys, these are selected independently for the two functions, and the parallel construction is $f_{k,r}||g_{k',r'}(x)=f_{k,r}(x)||g_{k',r'}(x)$.

The *Copy-parallel-Concat (cc) parallel construction* of two Signature/MAC schemes S, S', denoted $c_{||}(S,S')=S||S'$, is defined as follows. The definitions and proofs extend trivially to arbitrary number of schemes.

- $\mathcal{S}||\mathcal{S}'.\mathcal{KG}(<r,r'>)=<\mathcal{S}.\mathcal{KG}(r), \mathcal{S}'.\mathcal{KG}(r')>$
- $\mathcal{S}||\mathcal{S}'.\mathcal{S}<s,s'>,<r,r'>(m)=<\mathcal{S}.\mathcal{S}s,r(m), \mathcal{S}'.\mathcal{S}s',r'(m)>$
- $\mathcal{S}||\mathcal{S}'.\mathcal{V}v,v'(m,<\sigma,\sigma'>)= \mathcal{S}.\mathcal{V}v(m, \sigma) \wedge \mathcal{S}'.\mathcal{V}v'(m, \sigma')$

Similarly, the *Copy-parallel-Concat (cc) parallel construction* of two commitment schemes $\mathcal{C}, \mathcal{C}'$, denoted $c_{||}(\mathcal{C},\mathcal{C}')=\mathcal{C}||\mathcal{C}'$, is defined as follows. The definition extends trivially to trapdoor commitment.

- $\mathcal{C}||\mathcal{C}'.KG(<r,r'>)=<\mathcal{C}.KG(r),\mathcal{C}'.KG(r')>$
- $\mathcal{C}||\mathcal{C}'.C_{<ck,ck'>,<r,r'>}(m)= <\mathcal{C}.C_{ck,r}(m), \mathcal{C}'.C_{ck',r'}(m)>$
- $\mathcal{C}||\mathcal{C}'.D_{<ck,ck'>,<r,r'>}(m)=<\mathcal{C}.D_{ck,r}(m) , \mathcal{C}'.D_{ck',r'}(m)>$
- $\mathcal{C}||\mathcal{C}'.O_{<ck,ck'>}(m,<c,c'>,<d,d'>)=\mathcal{C}.O_{ck}(m,c,d)$ *if* $\mathcal{C}.O_{ck}(m,c,d)=\mathcal{C}'.O_{ck'}(m,c',d')\neq\perp$, *else* $\perp$

As the following lemma shows, the parallel construction ensures tolerance for many integrity properties / specifications, but clearly is quite bad for privacy.

Lemma 4-2 The *Copy-parallel-Concat (cc) construction is…*

1. Tolerant for the `integrity ` specifications WCRHF of keyless functions.
2. Tolerant for the *existential unforgeability under adaptive chosen message attack* specification of Signature/MAC schemes.
3. Tolerant for the Binding and Relaxed-Binding specifications of commitment schemes.
4. Preserving, but *NOT* tolerant, for the `confidentiality` specifications Hiding of commitment schemes, *CCA1-IND , CPA-IND*, *CCA2-IND* and *wCCA-IND,* of encryption schemes, and *ERF*.
5. *NOT* tolerant (preserving??), for the `confidentiality` specifications OWF of keyless functions.
6. *NOT* even preserving for the `confidentiality` specifications AONT of keyless functions.

The quantitative versions of the claims and the (simple) proofs will be included in the final version.

4.3 XOR-Parallel-Concat Construction for Encryption and AONT

Another classical tolerant construction, originally proposed in [AB81] for encryption schemes, takes two inputs: a message (plaintext) and a random bit string of the same length, and applies one function to the random string, and the other function to the exclusive-OR of the message with the random string. Namely, the simple XOR-parallel-Concat (*xc*) construction for two keyless functions *f, f'* is defined as $xc(f,f')[<m,x>]=f||_{XC}f'(<m,x>)=<f(m\oplus x),f'(x)>$; generalization to more than two functions is trivial.

The definition for *xc* construction for encryption schemes $\mathcal{E}, \mathcal{E}'$, is similar:

- $\mathcal{E}||_{XC}\mathcal{E}'.KG.e(<r,r'>)=<\mathcal{E}.KG.e(r),\mathcal{E}'.KG.e(r')>$, $\mathcal{E}\circ\mathcal{E}'.KG.d(<r,r'>)=<\mathcal{E}.KG.d(r),\mathcal{E}'.KG.d(r')>$.
- $\mathcal{E}||_{XC}\mathcal{E}'.E_{<e,e'>,<r,r',x>}(m)=<\mathcal{E}.E_{e,r}(x) , \mathcal{E}'.E_{e',r'}(x\oplus m)>$
- $\mathcal{E}||_{XC}\mathcal{E}'.D_{<d,d'>}(<c,c'>)=\mathcal{E}'.D_{d'}(c') \oplus \mathcal{E}.D_d(c)$

Lemma 4-3 The *xc* construction of encryption schemes is tolerant for specifications $CCA1\text{-}IND_{ISPUB}$ and $CPA\text{-}IND_{ISPUB}$, but does not even preserve $wCCA2\text{-}IND_{ISPUB}$ (or $CCA2\text{-}IND_{ISPUB}$). The simple *xc* construction is tolerant for specifications $AONT(k,s,l)\rightarrow AONT(k,2s,s+l)$. ■

Comment. The *xc* construction seems unacceptable for AONT, as the number of bits in the secret part doubles, and the number of bits which the adversary can expose does not increase (remain *s-l*); however we didn't find a better tolerant construction for AONT.

4.4 Share-Parallel-Concat Construction for Tolerant Commitment

In the Share-Parallel-Concat construction, the inputs to each component commitment scheme are shares of the input to the construction. A secret sharing scheme is a pair of algorithms <*Share,*

Reconstruct>. *Share* accepts a message *m* as input, and outputs *n* secret values $s_1, \ldots, s_n$ which we call *shares*; it is randomized, i.e. also accepts some random input *r*. For convenience, let $Share_{i,r}(m)$ denote the i^{th} output of *Share* on input *m* with randomness *r*. *Reconstruct* is a deterministic algorithm which takes *n* shares, $s_1', \ldots, s_n'$, some of which may have the special value $\perp$ (for a missing share), and outputs a message *m'*. The correctness property is that for every message *m* holds *m=Reconstruct(Share(m))*.

Furthermore, secret sharing schemes support different *thresholds*, for tolerating exposure or corruption of shares. In particular, in our case, we are interested in the following two thresholds. First, secret sharing schemes have a *privacy threshold,* t_p, which determines the maximum number of shares which reveal `no information` about the message *m*. Second, they have a *soundness threshold* t_s, which determines the minimum number of correct shares which ensures it is impossible to recover an *incorrect* message $m' \neq m$ (and $m' \neq \perp$).

For simplicity, we present the *share-parallel-concat (sc)* construction for ensuring tolerance from three candidate commitment schemes, C_1, C_2 and C_3, and using an arbitrary unconditionally secure secret sharing scheme <*Share, Reconstruct*> with $n=3$, $t_p=1$, $t_s=2$, e.g. Shamir's scheme [S79]. Generalizations allowing threshold to $t>1$ insecure components (by using $2t+1$ components and shares) are straightforward.

- $sc(C_1, C_2, C_3).KG(<r_1,r_2,r_3>) = < C_1.KG(r_1), C_2.KG(r_2), C_3.KG(r_3) >$
- $sc(C_1, C_2, C_3).C_{<ck1,ck2,ck3>,<r,r1,r2,r3>}(m) = < C_1.C_{ck1,r1}(Share_{1,r}(m)), C_2.C_{ck2,r2}(Share_{2,r}(m)), C_3.C_{ck3,r3}(Share_{3,r}(m)) >$
- $sc(C_1, C_2, C_3).D_{<ck1,ck2,ck3>,<r,r1,r2,r3>}(m) = < C_1.D_{ck1,r1}(Share_{1,r}(m)), C_2.D_{ck2,r2}(Share_{2,r}(m)), C_3.D_{ck3,r3}(Share_{3,r}(m)), Share_{1,r}(m), Share_{2,r}(m), Share_{3,r}(m) >$
- $sc(C_1, C_2, C_3).O_{<ck1,ck2,ck3>}(m,<c_1,c_2,c_3>,<d_1,d_2,d_3,s_1,s_2,s_3>) = Reconstruct(C_1.O_{ck}(s_1,c_1,d_1), C_2.O_{ck}(s_2,c_2,d_2), C_3.O_{ck}(s_3,c_3,d_3))$

The tolerance of the share-parallel-concat scheme follows easily from the properties of the secret sharing scheme. Essentially, the shared-parallel-concat is a hybrid or generalization of the copy-parallel-concat and the XOR-parallel-concat constructions. The construction and lemma extend trivially to trapdoor commitment schemes.

Lemma 4-4 The Share-parallel-Concat (*sc*) construction of $2t+1$ commitment schemes is *t*-tolerant for Binding, relaxed-Binding and Hiding specifications, for every $t \geq 1$. ■

Comment. In most practical commitment schemes, decommitment requires mainly the original message, and the additional decommitment strings d_i are quite short. However, the Share-parallel-Concat construction uses long decommitment string; specifically, the decommitment includes $<d_1,d_2,d_3,s_1,s_2,s_3>$. The shares s_1,s_2,s_3 are at least as long as the message; using [S79], *each* share is as long as the message; namely the decommit string is three times as long as the message. This may be substantial overhead for some applications. The scheme we present in the next section avoids this overhead.

Comment. By using robust secret sharing and other tools, [DK03] achieve tolerant construction for the CCA2-IND specification of encryption schemes. However, their construction is very wasteful in the length of the ciphertext, which may rule it unacceptable in most applications; we expect cascade would remain the preferred tolerant construction for encryption (although it `only` ensures wCCA2-IND).

5 Composing Constructions, and Tolerant Commitment

Often, we may want to combine multiple constructions, e.g. to ensure tolerance to multiple specifications. We restrict our attention to compositions of two constructions. In the first subsection we present two ways to compose the cascade construction (tolerant for *hiding*) and the copy-parallel-concat (*cc*) construction (tolerant for *binding*), resulting in efficient tolerant constructions for commitment schemes (ensuring both *hiding* and *binding* specifications). In the second subsection, we generalize these results, by defining a composition as a mapping of (two) constructions, presenting a generic composition based on a combinatorial `composition structure` variable, and showing that the compositions for commitment schemes are a special case. In particular, we use the general lemmas of the second subsection, to prove the tolerance of the constructions for commitment in the first subsection.

5.1 `Composite` Tolerant Constructions for Commitment Scheme

The Share-parallel-Concat (*sc*) construction provides tolerant design for commitment schemes, but results in a long decommitment string (more than twice the original message), which may be problematic for many applications. Can we construct efficient tolerant commitment schemes, with short decommitment (and commitment) strings? In this sub-section we show two such constructions, with different tradeoffs, both of which are compositions of the cascade and copy-parallel-concat (*cc*) constructions. This builds on the fact that cascade is tolerant for the hiding specification, and copy-parallel-concat (*cc*) is tolerant for the binding specifications.

It therefore makes sense to combine them, e.g. use *four* candidate commitment schemes, C_{11}, C_{12}, C_{21}, and C_{22}, cascading C_{11} and C_{12} and connecting this in parallel to the cascade of C_{21} and C_{22}. We call the result the *D construction,* after its `shape`, as follows. We use the notation $C_{ij}(m)=C_{ij}.C_{kij,rij}(m)$, $D_{ij}(m)=C_{ij}.D_{kij,rij}(m)$, $O_{ij}(m)=C_{ij}.O_{kij}(m,c_{ij},d_{ij})$, $R=<r_{11},r_{12},r_{21},r_{22}>$, $K=<k11,k12,k21,k22>$.

- $D.KG(R)=< C_1.KG(r_{11}), C_2.KG(r_{12}), C_3.KG(r_{21}), C_3.KG(r_{22})>$
- $D.C_{K,R}(m)= < C_{12}(Co_{11}(m)), C_{22}(C_{21}((m))>$
- $D.D_{K,R}(m)=<D_{11}(m), C_{11}(m), D_{12}(Co_{11}(m)), D_{21}(m), C_{21}(m), D_{22}(C_{21}(m))>$
- $D.O_K(m,<c_{12},c_{22}>,<d_{11},c_{11},d_{12},d_{21},c_{21},d_{22}>)=$
  $= \{m \text{ if } (m=O_{11}(m)=O_{11}(m)) \wedge (c_{11}=O_{12}(c_{11})) \wedge (c_{21}=O_{22}(c_{21})), \text{otherwise } \perp\}$

The *D* construction is quite 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). In particular, in the size of the commit and decommit strings, it substantially improves upon the *sc* construction; this may be important for many applications.

However, the *D* construction has one significant drawback: it uses four component commitment schemes for 1-tolerance, while *sc* requires only three candidate schemes for 1-tolerance. However, 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 construction*. The definition is omitted for lack of space (and since it should be obvious – especially after reading the *next* subsection).

We next state the tolerance of the *D* and *E* constructions. The proof is given in the next section.

Lemma 5-1 Both *D* (*E*) construction of *2t+2* (respectively, *2t+1*) commitment schemes is *t*-tolerant for Binding, relaxed-Binding and Hiding specifications. ■

5.2 Composing Arbitrary Constructions

We now generalize the idea of combining multiple constructions, as in the previous subsection, to arbitrary constructions and specifications. We still limit our attention to compositions of two constructions. Such compositions accept as input two constructions *c* and *c'* and produce a composite construction denoted $c'\circ_I c$, where *I* is a mapping of the `candidate functions` to the constructions. We present few simple, and useful, compositions. First, we need to define the relevant mappings *I* and the composition for given *I*.

Let *c* be a construction of plurality *p* over *F* which is *t*-tolerant for $s \rightarrow s'$, and let *c'* be a construction of plurality *p'* over *F* which is *t'*-tolerant for $s' \rightarrow s''$. Let p° denote the plurality of the composition of *c* and *c'*; namely the input to the composite construction is an ordered set *f* of p° functions, $f[i] \in F$. The composite construction $c'\circ_I c$ first applies *c* to *p'* sets of *p* functions each, and then applies *c'* to the *p'* resulting functions. The composition is defined by the selection of the *p* functions input to each of the *p'* applications of the *c* construction, namely by a mapping $I:\{1,\ldots,p\}\times\{1,\ldots,p'\} \rightarrow \{1,\ldots,p^\circ\}$, where $I_i[j]$ identifies the j^{th} function input to the i^{th} *c* construction. Given *I,* the ***I*-composition** of *c'* and *c*, denoted $c'\circ_I c$, is

$$c'\circ_I c\left(f[1],\ldots,f[p^\circ]\right)=c'\left(c\left(f\left[I_1(1)\right],\ldots,f\left[I_1(p)\right]\right),\ldots,c\left(f\left[I_{p'}(1)\right],\ldots,f\left[I_{p'}(p)\right]\right)\right.$$

Consider cascade compositions of threshold-tolerant constructions. The following lemma shows that the security of the *I*-composition of two threshold-tolerant constructions, depends on a simple combinatorial property of mappings *I*. Consider mapping $I:\{1,\ldots,p\}\times\{1,\ldots,p'\}\rightarrow\{1,\ldots,p^\circ\}$ and some set $T\subseteq\{1,\ldots,p^\circ\}$ (of `weak input functions`). Let $G_i(I,T)=\{ I_i[j] \mid j=1,\ldots,p \} - T$, i.e. values $I_i[j]$, for some j, which are *not* in T; think of $G_i(I,T)$ as the `good selections` of I_i. Let $G(I,T)[t]=\{i \text{ s.t. } |G_i(I,T)|\geq p-t\}$. We say that I is a (good) *(t,t',t°)-threshold-composition-structure* if for every $T\subseteq\{1,\ldots,p^\circ\}$ s.t. $|T|\leq t^\circ$ holds: $|G(I,T)[t]|\geq p'-t'$.

Lemma 5-2 Let $I:\{1,\ldots,p\}\times\{1,\ldots,p'\}\rightarrow\{1,\ldots,p^\circ\}$ be a (good) (t,t',t°)-threshold-composition-structure. Let c be a construction of plurality p over F which is t-tolerant for $s\rightarrow s'$. Let c' be a construction of plurality p' over F which is t'-tolerant for $s'\rightarrow s''$. Then $c'\circ_I c$, is a construction of plurality p° over F which is t°-tolerant for $s\rightarrow s''$.

Proof: Consider any set f of p° functions, $f[i]\in F$, and assume that $p^\circ-t$ of them satisfy specification s. Namely, for some set $\{i_j\}$ of $p^\circ-t$ indexes holds $s(f[i_j])=1$. We need to prove that for every choice $T\subseteq\{1,\ldots,p^\circ\}$ of up to t° functions in f which do not satisfy s, the function resulting from applying composed construction $c\circ_I c'$ to $\{f[1],\ldots,f[p^\circ]\}$ satisfies s'' . Namely, we need to prove that $s''(c'\circ_I c(f[1],\ldots,f[p^\circ]))=1$. Let $f'[1],\ldots,f'[p']$ denote the p' intermediate functions, i.e. $f'[i]=c(f[I_i(1)],\ldots,f[I_i(p)])$; hence $c'\circ_I c(f[1],\ldots,f[p^\circ])=c'(f'[1],\ldots,f'[p'])$.

If $i\in G(I,T)[t]$, namely $|G_i(I,T)|\geq p-t$, then for at least $p-t$ of the functions $f[I_i(1)],\ldots,f[I_i(p)]$ holds $s(f[I_i(j)])=True$. Since c is *t-tolerant* for $s\rightarrow s'$ it follows that $s'(f'[i])=True$, for every $i\in G(I,T)$. Since c' is *t'-tolerant* for $s'\rightarrow s''$, it follows that: $s''(c'\circ_I c(f[1],\ldots,f[p^\circ]))=s''(c'(f'[1],\ldots,f'[p']))=1$. ■

We now present two simple threshold cascade compositions derived from the above lemma, by presenting two simple composition structures:

- Composition structure $D:\{0,1\}\times\{0,1\}\rightarrow\{0,1,2,3\}$ defined as $D_i[j]=2i+j$ for $i,j\in\{0,1\}$
- Composition structure $E:\{0,1\}\times\{0,1,2\}\rightarrow\{0,1,2\}$ defined as $E_i[j]=i+j \bmod 3$ for $i\in\{0,1\}$, $j\in\{0,1,2\}$

By simply checking the combinatorial definition of *(t,t',t°)-threshold-composition-structure* we get:

Lemma 5-3 *D* and *E* are both (good) *(0,1,1)* and *(1,0,1)* threshold-composition-structures.

From the two Lemmas, we get:

Lemma 5-4 Let c, c_D', c_E' be constructions of plurality 2, 2 and 3 respectively. If c is t-tolerant for $s\rightarrow s'$ where $t\in\{0,1\}$, and c_D' , c_E' are *(1-t)*-tolerant for $s'\rightarrow s''$, then $c_D'\circ_D c$ and $c_E'\circ_E c$ are both *1*-tolerant for $s\rightarrow s''$.

We can now prove Lemma 5-1, for $t=1$ (proof for $t>1$ is similar).

Proof of Lemma 5-1: Let c be c_c, i.e. c is the cascade construction; and let c_D', c_E' be $c_{||}$, i.e. the copy-parallel-concat construction, both for commitment schemes. Notice that $D= c_D'\circ_D c$, $E= c_E'\circ_E c$. The claim therefore follows immediately from Lemma 4-2, Lemma 3-4 and Lemma 5-4. ■

6 Conclusions and Open Questions

In this work we presented simple, efficient and practical tolerant constructions for some of the most important and practical cryptographic mechanisms, including encryption, signature/MAC and commitment schemes. For encryption and MAC/signature schemes, we simply proved the security of the (very efficient) `folklore` constructions; for commitment schemes, we present new constructions which are compositions of the folklore cascade and parallel (specifically, copy-parallel-concat) constructions. We also present definitions for tolerant constructions and compositions, and some simple yet useful results regarding compositions of constructions.

We believe that efficient tolerant constructions are an important requirement from practical cryptographic primitives; put differently, we should prefer specifications with an efficient tolerant construction. We presented efficient tolerant constructions for several of the important primitives (and specifications) of modern cryptography. However, for others, we did not find (yet?) a (reasonably efficient)

tolerant construction. This calls for additional research, to distinguish between specifications with efficient tolerant design, vs. specifications that do not have an efficient tolerant design (and possibly, to find alternate specifications which are sufficient for most applications/scenarios). For example, XOR-parallel-concat is tolerant for AONT, but has substantial loss in parameters (instead of s secret bits out of which l must remain secret, we need $2s$ secret bits out of which $s+l$ must remain secret). We hope follow-up works will investigate efficient tolerant constructions for AONT, and for other mechanisms not covered by our results.

Acknowledgements

I wish to thank Mihir Bellare, Ran Canetti, Shai Halevi, Kath Knobe, Boaz Patt-Shamir, Avi Wigderson and anonymous referees for helpful comments and discussions. This work is supported by ISF grant.

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.

[ABCV98] B. Aiello, M. Bellare, G. Di Crescenzo, and R. Venkatesan, Security amplification by construction: the case of doubly-iterated, ideal ciphers, Proc. of CRYPTO 98.

[ADR02] Jee Hea An, Yevgeniy Dodis and Tal Rabin, On the Security of Joint Signature and Encryption, in Theory and Application of Cryptographic Techniques, pp. 83-107, 2002. Also in Advances in Cryptology - EUROCRYPT 2002, volume 2332 of Lecture Notes in Computer Science, pages 83-107. Springer-Verlag, 2002.

[AN95] Ross Anderson, Roger Needham. *Robustness Principles for Public Key Protocols*. In Proceedings of Int'l. Conference on Advances in Cryptology (CRYPTO 95), Vol. 963 of Lecture Notes in Computer Science, pp. 236-247, Springer-Verlag, 1995.

[AN96] Martin Abadi, Roger Needham. *Prudent Engineering Practice for Cryptographic Protocols*. IEEE Transactions on Software Engineering, 22, 1 (Jan.), 1996, pp. 6-15.

[B98] Daniel Bleichenbacher. *Chosen ciphertext attacks against protocols based on the RSA encryption standard PKCS*#1. In Advances in Cryptology - CRYPTO '98, LNCS 1462, pages 1-12. Springer, 1998.

[BDJR97] M.Bellare, A.Desai, E.Jokipii, P.Rogaway: A Concrete Security Treatment of Symmetric Encryption, Proceedings of the 38th IEEE Symposium on Foundations of Computer Science (FOCS), pp. 394-403, 1997. Revised version at http://www-cse.ucsd.edu/users/mihir/papers/sym-enc.html.

[BKR94] Mihir Bellare, Joe Kilian and Phil Rogaway, "The security of cipher block chaining", Journal of Computer and System Sciences, Vol. 61, No. 3, Dec 2000, pp. 362-399. Extended abstract in Advances in Cryptology - Crypto 94 Proceedings, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed, Springer-Verlag, 1994.

[BN00] Mihir Bellare and Chanathip Namprempre. *Authenticated encryption: Relations among notions and analysis of the generic construction paradigm*. In T. Okamoto, editor, Asiacrypt 2000, volume 1976 of LNCS, pages 531-545. Springer-Verlag, Berlin Germany, Dec. 2000.

[BR97] Mihir Bellare and Phillip Rogaway, Collision-Resistant Hashing: Towards Making UOWHFs Practical, Extended abstract was in Advances in Cryptology- Crypto 97 Proceedings, Lecture Notes in Computer Science Vol. 1294, B. Kaliski ed, Springer-Verlag, 1997. Full paper available at http://www.cs.ucsd.edu/users/mihir/papers/tcr-hash.html.

[BSZ02] Joonsang Baek, Ron Steinfeld, and Yuliang Zheng. *Formal proofs for the security of signcryption*. In David Naccache and Pascal Pailler, editors, 5th International Workshop on Practice and Theory in Public Key Cryptosystems - PKC 2002, pp. 80-98, LNCS Vol. 2274, 2002.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography. In Proceedings of the 23rd Symposium on Theory of Computing, ACM STOC, 1991.

[DK94] Ivan B. Damgård, Lars Ramkilde Knudsen. Enhancing the Strength of Conventional Cryptosystems, BRICS report RS-94-38, November 1994.

[DK03] Yevgeniy Dodis and Jonathan Katz, Chosen Ciphertext Security of Multiple Encryption, Manuscript, December 2003.

[DPP94] Ivan B. Damgård, Torben P. Pedersen, Birgit Pfitzmann: On the Existence of Statistically Hiding Bit Commitment Schemes and Fail-Stop Signatures; Crypto '93, LNCS 773, Springer-Verlag, Berlin 1994, 250-265.

[DPP98] Ivan B. Damgård, Torben P. Pedersen, Birgit Pfitzmann: Statistical Secrecy and Multi-Bit Commitments; IEEE Transactions on Information Theory 44/3 (1998) 1143-1151.

[EG85] S. Even and O. Goldreich, On the Power of Cascade Ciphers, ACM Transactions on Computer Systems, Vol. 3, 1985, pp. 108-116.

[FIP180] National Institute of Standards and Technology, Federal Information Processing Standards Publication, FIPS Pub 180-1: Secure Hash Standard (SHA-1), April 17, (1995), 14 pages.

[Go01] Oded Goldreich, The Foundations of Cryptography, Volume 1 (Basic Tools), ISBN 0-521-79172-3, Cambridge University Press, June 2001.

[Go02] Oded Goldreich, Fragments of a Chapter on Encryptions Schemes, Extracts from working drafts of Volume 2, The Foundations of Cryptography.

[GGM84] Oded Goldreich and Shafi Goldwasser and Silvio Micali "How to Construct Random Functions" Journal of the ACM, 33(4), 1984, 792-807.

[GIL*90] Oded Goldreich, R. Impagliazzo, L. Levin, R. Venkatesen, D. Zuckerman. "Security preserving amplification of randomness", 31st Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, (1990), 318-326.

[GM84] Shafi Goldwasser and Silvio Micali. "*Probabilistic Encryption*," JCSS (28), 1984, 270-299.

[HILL99] Johan Hastad, Rudich Impagliazzo, Leonid A. Levin, and Mike Luby, Construction of a Pseudorandom Generator from any One-Way Function. SIAM Journal on Computing, Vol. 28, No. 4, pp. 1364-1396, 1999.

[HL92] Amir Herzberg and Mike Luby, "Public Randomness in Cryptography", proceedings of CRYPTO 1992, ICSI technical report TR-92-068, October, 1992.

[HM96] Shai Halevi and Silvio Micali, "Practical and Provably-Secure Commitment Schemes from Collision Free Hashing", in Advances in Cryptology - CRYPTO96, Lecture Notes in Computer Science 1109, Springer-Verlag, 1996, pp. 201-215.

[HP86] Amir Herzberg and Shlomit Pinter, “Composite Ciphers”, EE Pub. no. 576, Dept of Electrical Engineering, Technion, Haifa, Israel, Feb. 1986.

[K01] Hugo Krawczyk, "*The Order of Encryption and Authentication for Protecting Communications (or: How Secure Is SSL?)*," In Crypto '01, pp. 310-331, LNCS Vol. 2139, J. Kilian ed., Springer-Verlag, 2001.

[LV01] Arjen K. Lenstra and Eric R. Verheul. *Selecting Cryptographic Key Sizes*. Journal of Cryptology: The Journal of the International Association for Cryptologic Research, 14(4):255--293, September 2001.

[MM93] U.M. Maurer and J.L. Massey, Cascade ciphers: the importance of being first, Journal of Cryptology, Vol. 6, No. 1, pp. 55-61, 1993.

[MOV96] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, Handbook of Applied Cryptography, Section 9.2.6, CRC Press, ISBN 0-8493-8523-7, October 1996. Available online at http://www.cacr.math.uwaterloo.ca/hac/.

[RFC2246] T. Dierks, C. Allen, The TLS Protocol: Version 1.0, Network Working Group, Internet Engineering Task Force (IETF). Available online at http://www.ietf.org/rfc/rfc2246.txt.

[R00] Eric Rescorla. SSL and TLS: Designing and Building Secure Systems. Addison-Wesley, 2000.

[S79] Adi Shamir, How to share a secret, Comm. of the ACM, 22(11):612-613, 1979.

[Sc96] Bruce Schneier, Applied Cryptography, John Wiley & Sons, 1996.

[Sh00] Victor Shoup, *Using hash functions as a hedge against chosen ciphertext attacks*, Adv. in Cryptology -- Proc. of Eurocrypt '2000, LNCS 1807, pp. 275-288.

[Z97] Yuliang Zheng, *Digital signcryption or how to achieve cost(signature+encryption) << cost(signature)+cost(encryption*), in Advances in Cryptology - CRYPTO'97, Berlin, New York, Tokyo, 1997, vol. 1294 of Lecture Notes in Computer Science, pp. 165--179, Springer-Verlag.

Appendix

Proof of claim 2 of Lemma 3-1: Trivially, if $Time(f,k)\leq \tau_F$ and $Time(g,k)\leq \tau_F$ then $Time(f\circ g,k)\leq 2\tau_F$. Let $s(f)= concrete\text{-}OWF^f(a,k,\rho_A, \tau_A) \wedge [Time(f,k)\leq \tau_F]$, $s'(f)= concrete\text{-}OWF^f(a,k,\rho_A, \tau_{A+}\tau_F)$. It remains to prove that $s(f)\rightarrow s'(f\circ g)$ and that $s(g)\rightarrow s'(f\circ g)$.

Assume $f,\ g\in P_k$ and $Time(f,k)\leq \tau_F$, $Time(g,k)\leq \tau_F$. Trivially, $f\circ g\in P_k$. Assume that $s'(f\circ g)=False$; we prove that both $s(f)=False$ and $s(g)=False$.

Since $s'(f\circ g)=False$, there is some (possibly probabilistic) algorithm A s.t. $(\forall x\in\{0,1\}^k)time(A(f\circ g(x)))\leq \tau_A+\tau_F$ and $\Pr_{m\in_R\{0,1\}^k}\left[f\circ g(A(f\circ g(m)))=f\circ g(m)\right]\geq a$.

Define algorithms A_f, A_g as follows:

$A_f(y)=g(A(y))$, $A_g(y)=A(f(y))$

We first show that the running time of A_f and A_g over inputs of length k is bounded by $t+\tau$. Suppose A_f is given input $f(m)$ where $|f(m)|=|m|=k$ (remember that f is a permutation for inputs of any length k). Therefore, A_f gives input of the same length k to A. WLOG, we can assume that the output of A is also of length k (since otherwise clearly A loses). Therefore, the running time of A_f on input of length k is at most $t+\tau$; a similar argument holds for A_g.

It remains to show that $\Pr_{m\in_R\{0,1\}^k}\left[f(A_f(f(m)))=f(m)\right]\geq a$ and $\Pr_{m\in_R\{0,1\}^k}\left[g(A_g(g(m)))=g(m)\right]\geq a$.

Let X denote the k-bit strings x for whom A succeeds in inverting $f\circ g$, i.e. for every $x\in X$ holds $f\circ g(A(f\circ g(x)))= f\circ g(x)$. Since $\Pr_{x\in_R\{0,1\}^k}\left[f\circ g(A(f\circ g(x)))=f\circ g(x)\right]\geq a$, we know that $|X|\geq a\cdot 2^k$. Similarly let $X_f\equiv\left\{x\in\{0,1\}^k\middle|f\left(A_f\left(f(x)\right)\right)=f\left(x\right)\right\}$, $X_g\equiv\left\{x\in\{0,1\}^k\middle|g\left(A_g\left(g(x)\right)\right)=g\left(x\right)\right\}$. We show that $\left|X_f\right|\geq\left|X\right|$, $\left|X_g\right|\geq\left|X\right|$ and since $|X|\geq a\cdot 2^k$, the claim follows.

Let $x\in X$. Hence $f\circ g(A(f\circ g(x)))= f\circ g(x)$, namely $f(g(A_g(g(x)))=f(g(x))$. Since f is a permutation, it follows that $g(A_g(g(x)))=g(x)$, namely $x\in X_g$.

Similarly, let $x_f=g(x)$. Since $x\in X$, then $f\circ g(A(f\circ g(x)))= f\circ g(x)$, namely $f(A_f(f(x_f)))=f(x_f)$, i.e. $x_f\in X_f$. Since g is a permutation, it follows that $|X_f|\geq|X|$. ∎

Proof of Claim A, Lemma 3-3: We construct adversary A as follows. In the "select" phase, A selects randomly $r'_{ed}\in\{0,1\}^k$, and then uses $\mathcal{E}'.KG$ to compute the keys $e'=\mathcal{E}'.KG.e(r'_{ed})$, $d'=\mathcal{E}'.KG.d(r'_{ed})$.

Next, A invokes the "select" phase of A^o which returns plaintexts $p^o[0]$, $p^o[1]$ and state s^o. In its operation, A^o may invoke the oracles for functions $\{E,D,wD\}$ of $\mathcal{E}^o$; trivially, A can answer these queries by using the corresponding oracle for $\mathcal{E}$, and computing the corresponding function of $\mathcal{E}'$ (using keys e', d'). For example, to answer query of $\mathcal{E}^o.D_{d,d'}(c)$, we first invoke the oracle $\mathcal{E}.D_d$ on input c; denote the result as x. We now compute $\mathcal{E}^o.D_{<d,d'>}(c)=\mathcal{E}'.D_{d'}(\mathcal{E}.D_d(c))=\mathcal{E}'.D_{d'}(x)$, which is possible since A knows d'.

To complete the "select" phase, A computes $p[j]=\mathcal{E}'.E_{e',,r'[j]}(p^o[j])$ for $j\in\{0,1\}$ and $r'[j]\in_R\{0,1\}^*$ (for public key encryption, i.e. if $ISPUB=F$, then concatenate e' to $p[0]$ and $p[1]$). It then returns $p[0]$, $p[1]$ and $s=<s^o,e,e',d'>$. We later show that using $r'[j]\in_R\{0,1\}^\rho$ suffices.

In the "find" phase, A receives ciphertext c and state $s=<s^o,e,e',d'>$. It simply invokes the "find" phase of A^o on c and s^o, and returns the bit x returned by A^o.

We now show that $p^o\leq p$. The probabilities are taken over the coin tosses by A, by the `black-box` algorithm A^o, and by the experiment (for key-generation, encryption and b). Denote the coin tosses as follows:

- $r^o{}_A$: coins used by the `black-box` adversary A^o (provided by and known to A)
- Coins tossed by the experiment (unknown to A): for key generation (r_{ed}), to select the challenge plaintext $p[b]$ (coin b), and for encrypting $p[b]$ (bits r).
- Coins tossed by A for its own use, namely: for key generation (r'_{ed}) and to compute the encryptions of the plaintexts $E'.E_{e',,r'[j]}(p^o[j])$ (bits $r'[0]$, $r'[1]$).

Let $r_A=r^o{}_A||r'_{ed}||r'[0]||r'[1]$ denote all the random bits tossed by algorithm A. Let $w(r_A, r_{ed}, r, b)=true$ if and only if $\mathbf{IndExp}_{A,\mathcal{E},ISPUB}(k,q,l,t,\rho,\rho_A)=$"win" with the corresponding coin tosses.

Let $r^o{}_{ed}=r_{ed}||r'_{ed}$, $r^o=r||r'$. Let $w^o(r^o{}_A, r^o{}_{ed}, r^o, b)=true$ if and only if $\mathbf{IndExp}_{A^o,\mathcal{E}^o,ISPUB}(k,q,l,t^o,\rho,\rho^o{}_A)=$"win" with the corresponding coin tosses.

The claim follows by showing that $w^o(r^o{}_A, r^o{}_{ed}, r^o, b)\rightarrow w(r_A, r_{ed}, r, b)$. We show this holds, by showing that all conditions of step 8 of experiment $E=\mathbf{IndExp}_{A,\mathcal{E},ISPUB}(k,q,l,t,\rho,\rho_A)[r_A, r_{ed}, r, b]$ hold if they (conditions of step 8) hold in experiment $E^o=\mathbf{IndExp}_{A^o,\mathcal{E}^o,ISPUB}(k,q,l,t^o,\rho,\rho^o{}_A)[r^o{}_A, r^o{}_{ed}, r^o, b]$.

We use the following notation: let $x@E$ ($x@E^o$) denote the value of variable x during experiment E (respectively E^o). We omit the @ notation when the value is clearly identical in the two experiments. Also, let $c8\varphi@E$ ($c8\varphi@E^o$), where $\varphi\in\{a,b,\ldots,f\}$, be *true* if claim φ of step 8 holds during experiment E (respectively E^o).

Algorithm A returns the same bit β as returned by A^o, namely $\beta@E=\beta@E^o$. Hence if $c8a@E^o$ is *true*, i.e. $\beta@E^o=b$, then also $\beta@E=b$ and $c8a@E=true$.

By design of A above, for $j=\{0,1\}$ holds $p[j]@E=\mathcal{E}'.E_{e',r'[j]}(p^o[j]@E)$. If $c8b@E^o=true$, then $|p^o[1]@E|=|p^o[0]@E|\leq l-2\Delta$. Since $bounds[k, k', l, \Delta, \rho, \tau](\mathcal{E}')=True$, we have $|p[1]@E|=|p[0]@E'|\leq l-\Delta$. Hence, $c8b@E^o\rightarrow c8b@E$.

For $c8c$, we note that the running time of A consists of the running time of A^o in the corresponding experiment, plus the additional work by A. This extra work consists essentially of invoking the key generation algorithm once, doing two encryptions (to compute $p[0]$ and $p[1]$), and answering the oracle queries of A^o. Each oracle query $\mathcal{S}^o.f$ requires A to compute $\mathcal{E}'.f$; it follows that if the running time of A^o at E^o is bounded by t^o, then the running time of A at E is bounded by $t=t^o+\tau[KG]+2\tau[E]+\sum_{j\in\{find,select\}}\sum_{f\in\{E,D,wD\}}q[j,f]\tau[f]$. Hence, $c8c@E^o\rightarrow c8c@E$.

We note that A involves oracle $\mathcal{E}.f$ only to answer oracle call $\mathcal{E}^o.f$ of A^o. Hence, $c8d@E^o\rightarrow c8d@E$, $c8g@E^o\rightarrow c8g@E$ and $c8f$ holds since $q[find,D]=0$.

It remains to show that $c8e@E^o\rightarrow c8e@E$. Adversary A users random bits $r_A=r^o{}_A||r'_{ed}||r'[0]||r'[1]$ (including the random bits $r^o{}_A$ for running A^o internally). r'_{ed} is k bit long. Bits $r'[0]$, $r'[1]$ are used by A (only) to compute $p[0],p[1]$. Assume that $c8b$ holds at E^o, i.e. $|p[1]@E^o|=|p[0]@E^o|\leq l-2\Delta$. Since $bounds[k, k', l, \Delta, \rho, \tau](\mathcal{E}')=True$, we use at most ρ bits from $r'[j]$, for $j=\{0,1\}$, in computing $p[j]@E=\mathcal{E}'.E_{e'r'[j]}(p^o[j]@E)$. It follows that the total number of random bits used by A is at most $|r^o{}_A|+2k+2\rho$. If $c8e@E^o$ holds, i.e. $|r^o{}_A|\leq\rho^o{}_A$, it follows that A uses at most $|r_A|\leq\rho^o{}_A+2k+2\rho=\rho_A$ random bits. Hence $c8e@E^o\wedge c8b@E^o\rightarrow c8e@E$. ∎

Claim A', Lemma 3-3**:** given adversary A^o such that $p^o\geq\frac{1}{2}+a$ as a black box, we can construct adversary A' such that $p'\geq p^o\geq\frac{1}{2}+a$.

Proof of Claim A', Lemma 3-3: We construct adversary *A'* as follows. In the "select" phase, *A'* selects randomly $r_{ed} \in \{0,1\}^k$, and then uses $\mathcal{E}.KG$ to compute the keys $e=\mathcal{E}'.KG.e(r_{ed})$, $d=\mathcal{E}.KG.d(r_{ed})$, $s=\mathcal{E}.KG.s(r_{sv})$, $v=\mathcal{E}.KG.v(r_{sv})$.

Next, *A'* invokes the "select" phase of A^o which returns plaintexts *p[0], p[1]* and state s^o. In its operation, A^o may invoke the oracles for functions *{ES,D,V}* of $\mathcal{S}^o$; trivially, *A'* can answer these queries by using the corresponding oracle for $\mathcal{E}'$, and computing the corresponding function of $\mathcal{E}$. Finally, *A'* returns *p[0], p[1]* and $s=\langle s^o,e',e,d,s,v\rangle$. We later show that using $r[j] \in_R \{0,1\}^p$ suffices.

In the "find" phase, *A'* receives ciphertext *c* and state $s=\langle s^o,e',e,d\rangle$. It computes $c'=\mathcal{E}.E_{e,r}(c)$ and invokes the "find" phase of A^o on *c'* and s^o, and returns the bit *x* returned by A^o.

The rest of the proof follows exactly like in claim A. ∎