The Key Control Security of KDF Combiners*

Ritam Bhaumik

TII, Abu Dhabi, UAE
bhaumik.ritam@gmail.com

Abstract

At CRYPTO 2025, Bhaumik et al. formalised the notion of Key Control (KC) security of Key Derivation Functions (KDFs). A KC adversary, on seeing the root key of a KDF, attempts to manipulate its auxiliary inputs (the 'Context' string) to obtain a derived key from a pre-selected set of keys. In this paper we extend the notion of KC security to Key Combining Functions, which are KDFs that convert two root keys to a single derived key; we name the new notion Combining Key Control (CKC) security. We then investigate the CKC security of KDF Combiners and show that (up to certain limitations) it follows from the KC security of either of the component KDFs.

Keywords: Key derivation functions ⋆ Key control security ⋆ KDF combiners ⋆ Reduction proofs

1 Introduction

Key Derivation Functions. Modern cryptography relies heavily on generating and managing 'strong' key material, i.e., keys with high entropy content. Long strings with guarantees of high entropy can be expensive to generate, store, and transmit without compromising their security. On the other hand, a demand for long keys is not uncommon in modern applications, where it may be necessary to deploy several different cryptographic constructions at once. As such, a critical part of any cryptographic architecture today is an efficient algorithm which accepts a relatively short string with high entropy content and expands it into usable key material; such an algorithm is called a *Key Derivation Function* or KDF.

The high entropy input to a KDF is called a *root key*, and the expanded output is called a *derived key*. The desired length of the derived key is specified as part of the KDF's input, and is typically larger than the length of the root key. While it is possible to define a KDF interface where the only inputs are a root key and a desired output length, in practice a KDF takes two additional inputs: a *label* which contains information on the intended use of the derived key material, and a *context* which includes additional contextual information like session IDs, user IDs, or salts. The label is usually a short string fixed by the application, but the user has some degree of freedom regarding what to put in the context.

Key Control Security. Traditionally, the strength of key material is evaluated using the notion of pseudorandomness, which stipulates that the key bits should be computationally hard to distinguish from uniformly random bits. As such, conventional wisdom tells us that a KDF is secure as long as it is a good pseudorandom function (PRF), as is observed in much of the earlier research on the security of KDFs [AKMZ04, KKJY14, Kra10, SWG25]. However, in modern cryptographic systems, a KDF is usually composed with other parts of a protocol, e.g., a handshake to establish the root key; in some applications,

*This is the full version of the paper by the same title that will appear in the proceedings of SCN 2026.

this can expose KDFs to different threat scenarios which are not entirely captured by pseudorandomness. An interesting research direction in this area is to identify more nuanced security models which can capture such threat scenarios. One such security notion is that of *Key Control Security*.

Key Control Security aims to address the scenario where a user, on seeing the root key, adjusts the context with the intention of manipulating the derived key to their advantage. Informally, if a KDF provides Key Control Security, the user should fail to obtain a pre-selected derived key even with knowledge of the root key. Investigations into Key Control Security originated in a public comment [Nat22] by the Amazon team to the NIST publication SP 800-108 [Che22] on recommended KDF modes; it was subsequently incorporated into a revision of the same document as an informal definition.

A formal cryptographic treatment of the topic was initiated by Bhaumik et al. [BDI+25] at CRYPTO 2025. They defined Key Control Security through a two-stage random experiment, where the informal definition from [Che22] was generalised in several ways. In the first stage (called the *Commit* stage), the adversary plays around with the KDF, and eventually commits to an input label, a derived key length, and a target predicate for the derived key to satisfy. The root key is sampled with high entropy conditioned on the committed values. In the second stage (called the *Find* stage), the adversary attempts to find a suitable context so that the derived key satisfies the target predicate.

Cryptographic Combiners. Complex cryptographic protocols and applications often contain pipelines where different cryptographic primitives are chained together. The security of such pipelines may then depend on the security of several critical points, where compromising the security of a primitive at any such point ends up compromising the security of the whole pipeline. One way of mitigating this risk of single-point failure is to introduce redundancy at such points through the use of *combiners*.

A combiner is typically a node in a pipeline that runs in parallel several primitives with the same security goal, in the hope that as long as at least one of them is secure, it can protect this node of the pipeline from a security breach. The combiner splits the node input into the individual inputs of each parallel branch, and, once all the executions are complete, combines the branch outputs into the node output. Combiners have been studied for different cryptographic primitives like hash functions [Her05], signatures [BHMS17], KEMs [GHP18], and AEADs [PR20].

Our Contributions. In this paper we study how Key Control Security interacts with combiners. First we adapt the Key Control (KC) security game from [BDI+25] with minor embellishments into what we call the $\mathtt{KCS}$ experiment. We also formalise the notion of a *Key Combining Function* (KCF), which can be seen as a KDF that takes two root keys (possibly with different sources of entropy) and outputs a combined derived key. We then introduce the notion of *Combining Key Control (CKC) Security*—a generalisation of KC security for KCFs—through what we call the $\mathtt{CKCS}$ experiment.

Finally, we look at a special class of KCFs called *KDF Combiners*. A KDF Combiner $\mathcal{H}^{\mathcal{F}_1,\mathcal{F}_2}$ runs two KDFs $\mathcal{F}_1$ and $\mathcal{F}_2$ in parallel to get two derived keys T_1 and T_2 from the two root keys; it then joins T_1 and T_2 using a splicer function to get the final derived key. As the main technical contribution, we show that a KDF Combiner $\mathcal{H}^{\mathcal{F}_1,\mathcal{F}_2}$ provides (some amount of) CKC security as long as one of $\mathcal{F}_1$ and $\mathcal{F}_2$ is KC secure.[1]

Outline. In Section 2 we introduce notational conventions, some background material on information theory, and various formalisms related to oracle algorithms, KDFs, and KCFs. In Section 3 we define the games $\mathtt{KCS}$ and $\mathtt{CKCS}$ and all the associated notation. In Section 4 we introduce KDF combiners and

[1] Some papers study a stronger notion of combiner security, where the combiner provides some security guarantee even when one of the underlying primitives is replaced by an adversarial implementation that can query the other primitive(s). However, here we consider a weaker notion, where the insecure primitive is still assumed to be independent of the other primitive(s).

associated notation. In Section 5 we formally state and prove our main technical result on the CKC security of KDF combiners. In Section 6 we discuss fine points and caveats about our definitions and results.

2 Preliminaries

$\mathbb{N}$ will denote the set of natural numbers. For $a, b \in \mathbb{N}$ with $a \leq b$, $[a..b]$ will denote the set $\{c \in \mathbb{N} \mid a \leq c \leq b\}$. 1^n will denote the n-bit string with every bit set to 1. $(0, 1]$ will denote the set of all real numbers x satisfying $0 < x \leq 1$, and $[0, 1]$ will denote $(0, 1] \cup \{0\}$. The operator $\wedge$ will denote logical conjunction.

Style Conventions. Since parts of this paper are a little heavy on notation, we tried to follow some style conventions to make it easier to parse. Script-style block letters (e.g., $\mathcal{U}$) denote sets and spaces. Calligraphic block letters (e.g., $\mathcal{F}$) denote cryptographic constructions, oracles, and adversaries. Sans-serif block letters (e.g., T) denote inputs, outputs, and intermediate variables in experiments. Strings in typewriter font (e.g., KCS) denote random experiments. Boldface letters (e.g., $\mathbf{H}$) denote operators from probability and information theory. Greek small letters denote non-cryptographic functions and assorted parameters. Roman small letters denote non-random variables and assorted parameters.

ρ-injectivity. A function $f : \mathcal{U} \to \mathcal{V}$ is called *ρ-injective* if for any $v \in \mathcal{V}$,

$$|\{u \in \mathcal{U} \mid f(u) = v\}| \leq \rho. \tag{1}$$

ρ is called the *injectivity coefficient* of f if f is ρ-injective, and equality in Ineq. (1) is achieved for some $v \in \mathcal{V}$. f is an injective function if its injectivity coefficient $\rho = 1$, and a constant function if $\rho = |\mathcal{U}|$.

Sampling Notation. We use notation $\leftarrow_\$$ to denote sampling, with two different uses. For a finite set $\mathcal{S}$, we write $\mathsf{X} \leftarrow_\$ \mathcal{S}$ to indicate that X is a random variable sampled uniformly at random from $\mathcal{S}$ (i.e., each element of $\mathcal{S}$ is equally likely to be picked). For a randomized algorithm $\mathcal{A}$, we write $\mathsf{X} \leftarrow_\$ \mathcal{A}(\mathsf{Y}, 1^n)$ to denote that $\mathcal{A}$ on input a random variable Y and 1^n for a parameter n outputs X; we still view X as being 'sampled' because $\mathcal{A}$ is randomized.

Expectation. For a random variable X with support $\mathcal{S}$ and a function φ, expectation of $\varphi(\mathsf{X})$ is defined as

$$\mathbf{Ex}[\varphi(\mathsf{X})] := \sum_{x \in \mathcal{S}} \varphi(x) \cdot \mathbf{Pr}[\mathsf{X} = x].$$

This notation may be extended to make the sampling of X explicit, e.g.,

$$\mathop{\mathbf{Ex}}_{\mathsf{X} \leftarrow_\$ \mathcal{A}(\mathsf{Y}, 1^n)}[\varphi(\mathsf{X})] := \sum_{x \in \mathcal{S}} \varphi(x) \cdot \mathbf{Pr}[\mathcal{A}(\mathsf{Y}, 1^n) = x].$$

For a random variable Y and a value y in the support of Y, conditional expectation of $\varphi(\mathsf{X})$ given $\mathsf{Y} = y$ is defined as

$$\mathbf{Ex}[\varphi(\mathsf{X}) \mid \mathsf{Y} = y] := \sum_{x \in \mathcal{S}} \varphi(x) \cdot \mathbf{Pr}[\mathsf{X} = x \mid \mathsf{Y} = y].$$

Min-Entropy. The conditional min-entropy of X given $\mathsf{Y} = y$ is defined as

$$\mathbf{H}_\infty(\mathsf{X} \mid \mathsf{Y} = y) := -\log\left(\max_{x \in \mathcal{S}} \mathbf{Pr}[\mathsf{X} = x \mid \mathsf{Y} = y]\right).$$

Let $\mathbf{H}_\infty(\mathsf{X} \mid \mathsf{Y})$ denote the function $y \mapsto \mathbf{H}_\infty(\mathsf{X} \mid \mathsf{Y} = y)$ over the support of Y. Then the average conditional min-entropy of X given Y is defined as

$$\tilde{\mathbf{H}}_\infty(\mathsf{X} \mid \mathsf{Y}) := -\log\left(\mathbf{Ex}\left[2^{-\mathbf{H}_\infty(\mathsf{X}|\mathsf{Y})}\right]\right) = -\log\left(\mathbf{Ex}\left[\max_{x \in \mathcal{S}} \mathbf{Pr}[\mathsf{X} = x \mid \mathsf{Y}]\right]\right).$$

Oracles and Oracle Algorithms. For some domain $\mathscr{D}$ and range $\mathscr{R}$, an *oracle* is a black-box implementation of some function $O : \mathscr{D} \to \mathscr{R}$. The oracle itself is a function in the mathematical sense, because it maps every point in $\mathscr{D}$ to some point in $\mathscr{R}$, but is not an algorithm, because its internal circuit is unspecified. As such, it is an abstract theoretical construction, used in proofs to model idealised behaviour of certain components. We will not make a distinction between an oracle and the function it implements, and treat them interchangeably.

An *oracle algorithm* $\mathcal{A}$ is an abstract algorithm which makes black-box queries to one or more oracles; we write $\mathcal{A}^O$ to denote that $\mathcal{A}$ has oracle access to O. It is used to model the behaviour of higher-level constructions when some or all of the underlying components are assumed to behave ideally. An oracle algorithm can be turned into a (concrete) algorithm by 'instantiating' each of its oracles with a concrete primitive.

A proof using an oracle algorithm $\mathcal{A}^O$ usually assumes that $\mathcal{A}$ can only find information on O by making queries to it. To ensure this, O is usually sampled uniformly at random from a family of oracles $\mathscr{O}$, each with the same domain $\mathscr{D}$ and the same range $\mathscr{R}$,[2] using a random coin that is hidden from $\mathcal{A}$. We use the shorthand $\mathscr{O} : \mathscr{D} \to \mathscr{R}$ to indicate that every oracle in $\mathscr{O}$ has domain $\mathscr{D}$ and range $\mathscr{R}$.

Random Coins Formalism. Randomization steps in experiments and algorithms will be abstracted using random coins. Thus, a randomized algorithm is modelled as deterministic algorithm under the hood that takes some random coins as an additional input. Similarly, a random sampling step inside an experiment is modelled as invoking a deterministic choice function under the hood with some random coins as input. For brevity of notation, these coins will for the most part be kept implicit, and will only be explicitly notated when required in some step of the proof.

Key Derivation Functions. Let $\mathscr{K}$ denote a *root key space*, $\mathscr{L}$ denote a *label space*, $\mathscr{C}$ denote a *context space*, and $\mathscr{T}$ denote a *derived key space*. A *key derivation function* (KDF) $\mathcal{F} : \mathscr{K} \times \mathscr{L} \times \mathscr{C} \times \mathbb{N} \to \mathscr{T}$ takes as input a root key $\mathsf{K} \in \mathscr{K}$, a label $\mathsf{L} \in \mathscr{L}$ of length $|\mathsf{L}| = \lambda$, a context $\mathsf{C} \in \mathscr{C}$, and a desired output length $\ell \in \mathbb{N}$, and produces a derived key $\mathsf{T} \in \mathscr{T}$ of length $|\mathsf{T}| = \ell$. Here, $\mathcal{I} := (\mathscr{K}, \mathscr{L}, \mathscr{C})$ is called the *interface* of $\mathcal{F}$.[3]

The root key K is typically obtained from some key establishment protocol, and can be expected to have high entropy. The label L is usually fixed by the application, and λ is typically small. The context C contains additional information, and may be partially or fully controlled by the user.

Key Combining Functions. Let $\mathscr{K}_1$ and $\mathscr{K}_2$ denote two *root key spaces*, $\mathscr{L}$ denote a *label space*, $\mathscr{C}$ denote a *context space*, and $\mathscr{T}$ denote a *derived key space*. A *key combining function* (KCF) $\mathcal{G} : \mathscr{K}_1 \times \mathscr{K}_2 \times \mathscr{L} \times \mathscr{C} \times \mathbb{N} \to \mathscr{T}$

[2] We use the word 'range' in the looser sense of co-domain, not the stricter sense of domain image.

[3] We do not include $\mathbb{N}$ because it does not vary across different KDF's.

takes as input two root keys $\mathsf{K}_1 \in \mathscr{K}_1$ and $\mathsf{K}_2 \in \mathscr{K}_2$, a label $\mathsf{L} \in \mathscr{L}$ of length $|\mathsf{L}| = \lambda$, a context $\mathsf{C} \in \mathscr{C}$, and a desired output length $\ell \in \mathbb{N}$, and produces a derived key $\mathsf{T} \in \mathscr{T}$ of length $|\mathsf{T}| = \ell$.

A KCF is similar to a KDF, except it takes as input two root keys, potentially originating in two different key establishment protocols. Here, $\mathscr{J} := (\mathscr{K}_1, \mathscr{K}_2, \mathscr{L}, \mathscr{C})$ is called the interface of $\mathcal{G}$.

2.1 A Resampling Lemma

The following lemma shows that an oracle algorithm cannot detect whether an oracle is resampled at some point during a game, provided the resampled oracle and the original oracle agree on all points queried before the resampling.

Lemma 1. *Suppose an oracle algorithm $\mathcal{A}$ makes q queries to an oracle O sampled uniformly (and independently of $\mathcal{A}$'s random coins) from an oracle family $\mathscr{O}$. Let τ be the transcript recording these queries. Define*

$$\mathscr{O}' := \{O' \in \mathscr{O} \mid (\forall (x,u) \in \tau)(O'(x) = u)\},$$

and sample $O^\dagger \leftarrow_\$ \mathscr{O}'$. From this point, consider two experiments[4]: in the first, $\mathcal{A}$ keeps running with oracle access to O till it terminates; in the second, $\mathcal{A}$'s oracle is changed to $O^\dagger$ for the remainder of its execution. Suppose $\mathcal{A}$ outputs Z in the first experiment and $Z^\dagger$ in the second. Then Z and $Z^\dagger$ are identically distributed.

Proof. We treat the initial transcript τ as a random variable, and denote

$$\mathscr{O}'(\tau) := \{O' \in \mathscr{O} \mid (\forall (x,u) \in \tau)(O'(x) = u)\},$$

Fix a transcript $\tau_\star$ and an oracle $O_\star \in \mathscr{O}'(\tau_\star)$. Then by Bayes' rule we have

$$\mathbf{Pr}[O = O_\star \mid \tau = \tau_\star] = \frac{\mathbf{Pr}[\tau = \tau_\star \mid O = O_\star] \cdot \mathbf{Pr}[O = O_\star]}{\mathbf{Pr}[\tau = \tau_\star]} = \frac{\mathbf{Pr}[\tau = \tau_\star \mid O = O_\star]}{\mathbf{Pr}[\tau = \tau_\star] \cdot |\mathscr{O}|}.$$

Now we observe that, as long as the condition $O_\star \in \mathscr{O}'(\tau_\star)$ holds, $\mathbf{Pr}[\tau = \tau_\star \mid O = O_\star]$ *does not depend on the specific choice of* $O_\star$. To see this, let τ_i be the random variable denoting the i-th entry in τ; let (x,u) be the corresponding entry in $\tau_\star$, and let $\tau_\star^\circ$ be the snapshot of the transcript just before (x,u) was added to it. (We assume without loss of generality that $(x,u) \notin \tau_\star^\circ$, since repeating queries does not give $\mathcal{A}$ any extra information.) Then we have

$$\begin{aligned}\mathbf{Pr}[\tau_i = (x,u) \mid \tau_\star^\circ \wedge O = O_\star] &= \mathbf{Pr}[\mathcal{A} \text{ queries } x \wedge O(x) = u \mid \tau_\star^\circ \wedge O = O_\star] \\ &= \mathbf{Pr}[\mathcal{A} \text{ queries } x \mid \tau_\star^\circ] \cdot \mathbf{Pr}[O(x) = u \mid \tau_\star^\circ \wedge O = O_\star],\end{aligned}$$

because the random coins of $\mathcal{A}$ are independent from those used for sampling O. Now, since $O_\star \in \mathscr{O}'(\tau_\star)$, $O_\star(x) = u$, so $\mathbf{Pr}[O(x) = u \mid \tau_\star^\circ \wedge O = O_\star] = 1$. Thus, $\mathbf{Pr}[\tau_i = (x,u) \mid \tau_\star^\circ \wedge O = O_\star]$ does not depend on $O_\star$ as long as $O_\star \in \mathscr{O}$, and since this holds for all i, $\mathbf{Pr}[\tau = \tau_\star \mid O = O_\star]$ also does not depend on $O_\star$.

Trivially, for $O_\star \notin \mathscr{O}'(\tau_\star)$, $\mathbf{Pr}[\tau = \tau_\star \mid O = O_\star] = 0$, so $\mathbf{Pr}[O = O_\star \mid \tau = \tau_\star] = 0$. Thus, $\mathbf{Pr}[O = O_\star \mid \tau = \tau_\star]$ is a non-zero constant for each $O_\star \in \mathscr{O}'(\tau_\star)$, and is 0 for each $O_\star \notin \mathscr{O}'(\tau_\star)$. This is exactly the uniform distribution over $\mathscr{O}'(\tau_\star)$. Thus, conditioned on the event $\tau = \tau_\star$, O is distributed uniformly over $\mathscr{O}'(\tau_\star)$, which coincides with the distribution of $O^\dagger$ conditioned on the same event, giving the equality

$$\mathbf{Pr}[O = O_\star \mid \tau = \tau_\star] = \mathbf{Pr}[O^\dagger = O_\star \mid \tau = \tau_\star].$$

[4]This can be realised, for example, by running two parallel copies of the same experiment with the same random coins up to the branching point (i.e., first q queries), and then branching as described.

This equality carries over to the marginal distributions of O and $O^\dagger$, since

$$\begin{aligned}\mathbf{Pr}[O = O_\star] &= \sum_{\tau_\star} \mathbf{Pr}[O = O_\star \wedge \tau = \tau_\star] \\ &= \sum_{\tau_\star} \mathbf{Pr}[O = O_\star \mid \tau = \tau_\star] \cdot \mathbf{Pr}[\tau = \tau_\star] \\ &= \sum_{\tau_\star} \mathbf{Pr}[O^\dagger = O_\star \mid \tau = \tau_\star] \cdot \mathbf{Pr}[\tau = \tau_\star] \\ &= \sum_{\tau_\star} \mathbf{Pr}[O^\dagger = O_\star \wedge \tau = \tau_\star] = \mathbf{Pr}[O^\dagger = O_\star].\end{aligned}$$

Let $\tilde{\tau}$ be the random variable denoting the final transcript in the first experiment, $\tilde{\tau}^\dagger$ be the random variable denoting the final transcript in the second experiment, and $\tilde{\tau}_\star$ be a fixed final transcript. Obviously, when $O = O^\dagger = O_\star$, the two experiments are identical, which in particular implies that $\tilde{\tau}$ and $\tilde{\tau}^\dagger$ are identically distributed, so

$$\mathbf{Pr}[\tilde{\tau} = \tilde{\tau}_\star \mid O = O_\star] = \mathbf{Pr}[\tilde{\tau}^\dagger = \tilde{\tau}_\star \mid O^\dagger = O_\star].$$

Again, this carries over to the marginal distributions of $\tilde{\tau}$ and $\tilde{\tau}^\dagger$ as

$$\begin{aligned}\mathbf{Pr}[\tilde{\tau} = \tilde{\tau}_\star] &= \sum_{O_\star} \mathbf{Pr}[\tilde{\tau} = \tilde{\tau}_\star \wedge O = O_\star] \\ &= \sum_{O_\star} \mathbf{Pr}[\tilde{\tau} = \tilde{\tau}_\star \mid O = O_\star] \cdot \mathbf{Pr}[O = O_\star] \\ &= \sum_{O_\star} \mathbf{Pr}[\tilde{\tau}^\dagger = \tilde{\tau}_\star \mid O^\dagger = O_\star] \cdot \mathbf{Pr}[O^\dagger = O_\star] \\ &= \sum_{O_\star} \mathbf{Pr}[\tilde{\tau}^\dagger = \tilde{\tau}_\star \wedge O^\dagger = O_\star] = \mathbf{Pr}[\tilde{\tau}^\dagger = \tilde{\tau}_\star].\end{aligned}$$

Finally, we observe that the final output of $\mathcal{A}$ is only a function of the final transcript and $\mathcal{A}$'s own random coins, so when $\mathcal{A}$ obtains the same final transcript in the two experiments, Z and $\mathsf{Z}^\dagger$ are also identically distributed. In other words, for any output value z,

$$\mathbf{Pr}[\mathsf{Z} = z \mid \tilde{\tau} = \tilde{\tau}_\star] = \mathbf{Pr}[\mathsf{Z}^\dagger = z \mid \tilde{\tau}^\dagger = \tilde{\tau}_\star].$$

As before, we can carry this to the marginal distributions of Z and $\mathsf{Z}^\dagger$ as

$$\begin{aligned}\mathbf{Pr}[\mathsf{Z} = z] &= \sum_{\tilde{\tau}_\star} \mathbf{Pr}[\mathsf{Z} = z \wedge \tilde{\tau} = \tilde{\tau}_\star] \\ &= \sum_{\tilde{\tau}_\star} \mathbf{Pr}[\mathsf{Z} = z \mid \tilde{\tau} = \tilde{\tau}_\star] \cdot \mathbf{Pr}[\tilde{\tau} = \tilde{\tau}_\star] \\ &= \sum_{\tilde{\tau}_\star} \mathbf{Pr}[\mathsf{Z}^\dagger = z \mid \tilde{\tau}^\dagger = \tilde{\tau}_\star] \cdot \mathbf{Pr}[\tilde{\tau}^\dagger = \tilde{\tau}_\star] \\ &= \sum_{\tilde{\tau}_\star} \mathbf{Pr}[\mathsf{Z}^\dagger = z \wedge \tilde{\tau}^\dagger = \tilde{\tau}_\star] = \mathbf{Pr}[\mathsf{Z}^\dagger = z],\end{aligned}$$

thus completing the proof. □

Game $\mathtt{KCS}(\mathcal{O}, \mathcal{F}, \mathcal{A} := (\mathcal{A}_c, \mathcal{A}_f), n)$

1 : $O \leftarrow_\$ \mathcal{O}$

2 : $(\mathsf{K}, \mathsf{S} := (\mathsf{L}, \ell, \chi, \tau, \mathsf{A})) \leftarrow_\$ \mathcal{A}_c^{O}(1^n)$ // Commit Stage

3 : $\mathsf{C} \leftarrow_\$ \mathcal{A}_f^{O}(\mathsf{K}, \mathsf{S}, 1^n)$ // Find Stage

4 : $\mathsf{T} \leftarrow \mathcal{F}^{O}(\mathsf{K}, \mathsf{L}, \mathsf{C}, \ell)$

5 : **return** $\chi(\mathsf{T})$

Figure 1: The random experiment defining the KCS security notion.

3 Key Control Security

The following definitions are adapted from [BDI+25, Sec. 3], with some notational changes and the introduction of an explicit security parameter n. We first define a Key Control (or KC) Adversary.

Consider a KDF interface $\mathcal{I} := (\mathcal{K}, \mathcal{L}, \mathcal{C})$. Let n be a security parameter, drawn from a space $\mathcal{N}$. Consider constraint functions $q_c : \mathcal{N} \to \mathbb{N}$, $q_f : \mathcal{N} \to \mathbb{N}$, $\lambda^- : \mathcal{N} \to \mathbb{N}$, $\lambda^+ : \mathcal{N} \to \mathbb{N}$, $\ell^- : \mathcal{N} \to \mathbb{N}$, $\ell^+ : \mathcal{N} \to \mathbb{N}$, $\kappa : \mathcal{N} \to \mathbb{N}$, $\delta : \mathcal{N} \to (0, 1]$, and let α denote the tuple $(q_c, q_f, \lambda^-, \lambda^+, \ell^-, \ell^+, \kappa, \delta)$.

Let $\alpha(n)$ be the shorthand denoting $(q_c(n), q_f(n), \lambda^-(n), \lambda^+(n), \ell^-(n), \ell^+(n), \kappa(n), \delta(n))$, so α can be viewed as a function $\mathcal{N} \to \mathbb{N}^7 \times (0, 1]$. We call $\alpha(n)$ a *KC constraint set* if $\ell^-(n) \le \ell^+(n)$ and $\lambda^-(n) \le \lambda^+(n)$, and we call α a KC constraint-set function if $\alpha(n)$ is a KC constraint set for any security parameter n. Denote by $\mathscr{P}$ a space of KC constraint-set functions, and define $\mathscr{P}(n) := \{\alpha(n) \mid \alpha \in \mathscr{P}\}$. Note that $\mathscr{P}(n) \subseteq \mathbb{N}^7 \times (0, 1]$.

Finally, let $\mathfrak{X}$ be a *predicate-set function* such that for any $\ell \in \mathbb{N}$, $\mathfrak{X}(\ell)$ is a set of predicates over $\{0,1\}^\ell$. For any $\ell \in \mathbb{N}$ and any $\chi \in \mathfrak{X}_\ell$, denote $\delta_{\chi,\ell} := \mathbf{Pr}[\chi(\mathsf{U}_\ell)]$ where $\mathsf{U}_\ell \leftarrow_\$ \{0,1\}^\ell$.

Definition 1 (Key Control (KC) Adversary)**.** *An $(\alpha, \mathfrak{X}, \mathcal{I})$-KC adversary $\mathcal{A}$ is defined as a pair of oracle algorithms $(\mathcal{A}_c, \mathcal{A}_f)$, which behave as follows:*

- *$\mathcal{A}_c$ takes as input 1^n, makes up to $q_c(n)$ queries to its oracle, and outputs $(K, S := (L, \ell, \chi, \tau, A))$, where*
  - *$K \in \mathcal{K}$ is a root key;*
  - *$L \in \mathcal{L}$ denotes the label with $|L| \in [\lambda^-(n)..\lambda^+(n)]$;*
  - *$\ell \in [\ell^-(n)..\ell^+(n)]$ denotes the bit-length of the desired $\mathcal{F}$ output;*
  - *$\chi \in \mathfrak{X}(\ell)$ denotes a target predicate;*
  - *τ denotes the transcript of interaction between $\mathcal{A}_c$ and its oracle;*
  - *A denotes auxiliary information output by $\mathcal{A}_c$.*

  The outputs of $\mathcal{A}_c$ are subject to the following two restrictions:

$$\tilde{\mathbf{H}}_\infty(K \mid S) \ge \kappa(n), \tag{2}$$

$$\mathop{\mathbf{Ex}}_{S \leftarrow_\$ \mathcal{A}_c^{O}} \left[\delta_{\chi,\ell}\right] \le \delta(n). \tag{3}$$

- *$\mathcal{A}_f$ takes $(K, S, 1^n)$ as input, makes up to $q_f(n)$ queries to its oracle, and outputs a context $C \in \mathcal{C}$.*

The Key Control Security (KCS) game is the two-stage boolean random experiment defined in Fig. 1, parametrised by an oracle family $\mathcal{O}$, a KDF $\mathcal{F}$, a KC adversary $\mathcal{A}$, and a security parameter n. The

experiment is initialised by sampling an oracle O from $\mathcal{O}$. In the Commit stage, $\mathcal{A}_c$ receives 1^n as input, and gets oracle access to O; it makes $q_c(n)$ queries to O and outputs (K, S) (subject to the restrictions described above). In the Find stage, $\mathcal{A}_f$ receives $(\mathsf{K}, \mathsf{S}, 1^n)$ as input, and gets oracle access to O; it makes $q_f(n)$ queries to O, and outputs C. In the last step of the experiment, $\mathsf{T} := \mathcal{F}^O(\mathsf{K}, \mathsf{L}, \mathsf{C}, \ell)$ is computed, and the experiment succeeds if $\chi(\mathsf{T})$ is true.

Key Control Security Advantage. Let $\mathcal{X}$ denote the set of all possible predicate-set functions. Let $\varepsilon : \mathbb{N}^7 \times (0,1] \times \mathcal{X} \to [0,1]$ be a *control function* which maps a KC constraint set $\alpha(n)$ and a predicate-set function $\mathfrak{X}$ to a probability indicating the maximum degree of control associated with the pair $(\alpha(n), \mathfrak{X})$.

Definition 2 (Key Control Security (KCS) Advantage)**.** *For a KDF $\mathcal{F}$ with interface $\mathcal{I}$, let $\mathcal{F} := \{\mathcal{F}^O \mid O \in \mathcal{O}\}$ be a family of $\mathcal{F}$-instances indexed by $\mathcal{O}$. We say that $\mathcal{F}$ is ε-KC secure if for any KC constraint-set function α, any predicate-set function $\mathfrak{X}$, any $(\alpha, \mathfrak{X}, \mathcal{I})$-KC adversary $\mathcal{A}$, and any security parameter $n \in \mathcal{N}$, we have*

$$\mathbf{Pr}[\mathit{KCS}(\mathcal{O}, \mathcal{F}, \mathcal{A}, n)] \leq \varepsilon(\alpha(n), \mathfrak{X}).$$

3.1 Key Control Security for KCFs

Game $\mathtt{CKCS}(\mathcal{O}, \mathcal{G}, \mathcal{B} := (\mathcal{B}_c, \mathcal{B}_f), n)$

1 : $O \leftarrow_\$ \mathcal{O}$
2 : $(\mathsf{K}_1, \mathsf{K}_2, \mathsf{S} := (\mathsf{L}, \ell, \chi, \tau, \mathsf{A})) \leftarrow_\$ \mathcal{B}_c^O(1^n)$ // Commit Stage
3 : $\mathsf{C} \leftarrow_\$ \mathcal{B}_f^O(\mathsf{K}_1, \mathsf{K}_2, \mathsf{S}, 1^n)$ // Find Stage
4 : $\mathsf{T} \leftarrow \mathcal{G}^O(\mathsf{K}_1, \mathsf{K}_2, \mathsf{L}, \mathsf{C}, \ell)$
5 : **return** $\chi(\mathsf{T})$

Figure 2: The random experiment defining the CKCS security notion.

We next adapt the definition of KC security for KCFs, to define what we call *Combining Key Control* (CKC) security. Consider a KCF interface $\mathcal{J} := (\mathcal{K}_1, \mathcal{K}_2, \mathcal{L}, \mathcal{C})$. Let $q_c, q_f, \lambda^-, \lambda^+, \ell^-, \ell^+, \delta, \mathfrak{X}$ be as in Definition 1, and consider two constraint functions $\kappa_1 : \mathcal{N} \to \mathbb{N}, \kappa_2 : \mathcal{N} \to \mathbb{N}$. Let β denote the tuple $(q_c, q_f, \lambda^-, \lambda^+, \ell^-, \ell^+, \kappa_1, \kappa_2, \delta)$.

Let $\beta(n)$ be the shorthand denoting $(q_c(n), q_f(n), \lambda^-(n), \lambda^+(n), \ell^-(n), \ell^+(n), \kappa_1(n), \kappa_2(n), \delta(n))$, so β can be viewed as a function $\mathcal{N} \to \mathbb{N}^8 \times (0,1]$. We call $\beta(n)$ a *CKC constraint set* if $\ell^-(n) \leq \ell^+(n)$ and $\lambda^-(n) \leq \lambda^+(n)$, and we call β a CKC constraint-set function if $\beta(n)$ is a CKC constraint set for any security parameter n. Denote by $\mathcal{Q}$ a space of CKC constraint-set functions, and define $\mathcal{Q}(n) := \{\beta(n) \mid \beta \in \mathcal{Q}\}$. Note that $\mathcal{Q}(n) \subseteq \mathbb{N}^8 \times (0,1]$.

Definition 3 (Combining Key Control (CKC) Adversary)**.** *A $(\beta, \mathfrak{X}, \mathcal{J})$-CKC adversary $\mathcal{B}$ is defined as a pair of oracle algorithms $(\mathcal{B}_c, \mathcal{B}_f)$, which behave as follows:*

- *$\mathcal{B}_c$ takes as input 1^n, makes up to $q_c(n)$ queries to its oracle, and outputs $(\mathsf{K}_1, \mathsf{K}_2, \mathsf{S} := (\mathsf{L}, \ell, \chi, \tau, \mathsf{A}))$, where*
  - *$\mathsf{K}_1 \in \mathcal{K}_1$ and $\mathsf{K}_2 \in \mathcal{K}_2$ are root keys;*
  - *$\mathsf{L} \in \mathcal{L}$ denotes the label with $|\mathsf{L}| \in [\lambda^-(n)..\lambda^+(n)]$;*
  - *$\ell \in [\ell^-(n)..\ell^+(n)]$ denotes the bit-length of the desired $\mathcal{G}$ output;*
  - *$\chi \in \mathfrak{X}(\ell)$ denotes a target predicate;*

– τ *denotes the transcript of interaction between* $\mathcal{B}_c$ *and its oracle;*

– A *denotes auxiliary information output by* $\mathcal{B}_c$.

$\mathcal{B}_c$ *is subject to the following three restrictions:*

$$\tilde{\mathbf{H}}_\infty(K_1 \mid K_2, S) \geq \kappa_1(n), \tag{4}$$

$$\tilde{\mathbf{H}}_\infty(K_2 \mid K_1, S) \geq \kappa_2(n), \tag{5}$$

$$\mathop{\mathbf{Ex}}_{S \leftarrow\$ \mathcal{B}_c^{\mathcal{O}}} \left[\delta_{\chi,\ell}\right] \leq \delta(n). \tag{6}$$

- $\mathcal{B}_f$ *takes* $(K_1, K_2, S, 1^n)$ *as input, makes up to* $q_f(n)$ *queries to its oracle, and outputs a context* $C \in \mathscr{C}$.

The Combining Key Control Security (CKCS) game is the two-stage boolean random experiment defined in Fig. 2, parametrised by an oracle family $\mathcal{O}$, a KCF $\mathcal{G}$, a CKC adversary $\mathcal{B}$, and a security parameter n. The experiment is initialised by sampling an oracle O from $\mathcal{O}$. In the Commit stage, $\mathcal{B}_c$ receives 1^n as input, and gets oracle access to O; it makes $q_c(n)$ queries to O and outputs $(\mathsf{K}_1, \mathsf{K}_2, \mathsf{S})$ (subject to the restrictions described above). In the Find stage, $\mathcal{B}_f$ receives $(\mathsf{K}_1, \mathsf{K}_2, \mathsf{S}, 1^n)$ as input, and gets oracle access to O; it makes $q_f(n)$ queries to O, and outputs C. In the last step of the experiment, $\mathsf{T} := \mathcal{G}^O(\mathsf{K}_1, \mathsf{K}_2, \mathsf{L}, \mathsf{C}, \ell)$ is computed, and the experiment succeeds if $\chi(\mathsf{T})$ is true.

Combining Key Control Security Advantage. Let $\gamma : \mathbb{N}^8 \times (0,1] \times \mathscr{X} \to [0,1]$ be a control function which maps a CKC constraint set $\beta(n)$ and a predicate-set function $\mathfrak{X}$ to a probability indicating the maximum degree of control associated with $\beta(n)$.

Definition 4 (Combining Key Control Security Advantage)**.** *For a KCF* $\mathcal{G}$ *with interface* $\mathcal{J}$*, let* $\mathcal{G} := \{\mathcal{G}^O \mid O \in \mathcal{O}\}$ *be a family of* $\mathcal{G}$*-instances indexed by* $\mathcal{O}$*. We say that* $\mathcal{G}$ *is* γ*-CKC secure if for any CKC constraint-set function* $\beta(n)$*, any predicate-set function* $\mathfrak{X}$*, any* $(\beta, \mathfrak{X}, \mathcal{J})$*-CKC adversary* $\mathcal{B}$*, and any security parameter* n*, we have*

$$\mathbf{Pr}[\mathit{CKCS}(\mathcal{O}, \mathcal{G}, \mathcal{B}, n)] \leq \gamma(\beta(n), \mathfrak{X}).$$

4 KDF Combiners

Consider two KDF's $\mathcal{F}_1 : \mathscr{K}_1 \times \mathscr{L}_1 \times \mathscr{C}_1 \times \mathbb{N} \to \mathscr{T}_1$ and $\mathcal{F}_2 : \mathscr{K}_2 \times \mathscr{L}_2 \times \mathscr{C}_2 \times \mathbb{N} \to \mathscr{T}_2$, making oracle calls respectively to two independent oracles O_1 and O_2. A *KDF combiner* $\mathcal{H}^{\mathcal{F}_1,\mathcal{F}_2}$ makes oracle calls to $\mathcal{F}_1$ and $\mathcal{F}_2$ and provides the functionality of a KCF $\mathcal{G} : \mathscr{K}_1 \times \mathscr{K}_2 \times \mathscr{L} \times \mathscr{C} \times \mathbb{N} \to \mathscr{T}$. It consists of the following functions:

- a label splitter $\eta : \mathscr{L} \to \mathscr{L}_1 \times \mathscr{L}_2$, notationally decomposed as $\eta_1 : \mathscr{L} \to \mathscr{L}_1$ and $\eta_2 : \mathscr{L} \to \mathscr{L}_2$ (so that $\eta(\mathsf{L}) := (\eta_1(\mathsf{L}), \eta_2(\mathsf{L}))$ for any $\mathsf{L} \in \mathscr{L}$);
- a context splitter $\mu : \mathscr{C} \to \mathscr{C}_1 \times \mathscr{C}_2$, notationally decomposed as $\mu_1 : \mathscr{C} \to \mathscr{C}_1$ and $\mu_2 : \mathscr{C} \to \mathscr{C}_2$ (so that $\mu(\mathsf{C}) := (\mu_1(\mathsf{C}), \mu_2(\mathsf{C}))$ for any $\mathsf{C} \in \mathscr{C}$);
- an injective length splitter $\nu : \mathbb{N} \to \mathbb{N} \times \mathbb{N}$, notationally decomposed as $\nu_1 : \mathbb{N} \to \mathbb{N}$ and $\nu_2 : \mathbb{N} \to \mathbb{N}$ (so that $\nu(\ell) := (\nu_1(\ell), \nu_2(\ell))$ for any $\ell \in \mathbb{N}$);
- a key splicer $\vartheta : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^*$ satisfying the following:
  - for any $\mathsf{T}_1 \in \mathscr{T}_1$ and any $\mathsf{T}_2 \in \mathscr{T}_2$, $\vartheta(\mathsf{T}_1, \mathsf{T}_2) \in \mathscr{T}$;

– for any $\ell \in \mathbb{N}$, any $\mathsf{T}_1 \in \mathscr{T}_1$ with $|\mathscr{T}_1| = \nu_1(\ell)$, and any $\mathsf{T}_2 \in \mathscr{T}_2$ with $|\mathscr{T}_2| = \nu_2(\ell)$, $|\vartheta(\mathsf{T}_1, \mathsf{T}_2)| = \ell$;
– let ϑ_ℓ denote the restriction of ϑ to $\{0,1\}^{\nu_1(\ell)} \times \{0,1\}^{\nu_2(\ell)}$; then for any $\ell \in \mathbb{N}$, the injectivity coefficient of ϑ_ℓ is ρ.

These are assumed to be 'simple' functions, with no cryptographic strength. The final KCF is defined as

$$\mathcal{G}(\mathsf{K}_1, \mathsf{K}_2, \mathsf{L}, \mathsf{C}, \ell) := \vartheta_\ell(\mathcal{F}_1\left(\mathsf{K}_1, \eta_1(\mathsf{L}), \mu_1(\mathsf{C}), \nu_1(\ell)\right), \\ \mathcal{F}_2\left(\mathsf{K}_2, \eta_2(\mathsf{L}), \mu_2(\mathsf{C}), \nu_2(\ell)\right)).$$

$\mathcal{H}^{\mathcal{F}_1,\mathcal{F}_2}$ is thus essentially a KCF with a specific structure: it accepts two root keys K_1 and K_2, a label L, a context C, and a desired output length ℓ; it splits L into $(\mathsf{L}_1, \mathsf{L}_2)$, C into $(\mathsf{C}_1, \mathsf{C}_2)$, ℓ into (ℓ_1, ℓ_2), and makes parallel calls to $\mathcal{F}_1$ and $\mathcal{F}_2$ with $(\mathsf{K}_1, \mathsf{L}_1, \mathsf{C}_1, \ell_1)$ and $(\mathsf{K}_2, \mathsf{L}_2, \mathsf{C}_2, \ell_2)$ respectively; finally it combines the two derived keys T_1 and T_2 to produce the final derived key T.

5 KC Security of KDF Combiners

Consider two KDF's $\mathcal{F}_1 : \mathscr{K}_1 \times \mathscr{L}_1 \times \mathscr{C}_1 \times \mathbb{N} \to \mathscr{T}_1$ and $\mathcal{F}_2 : \mathscr{K}_2 \times \mathscr{L}_2 \times \mathscr{C}_2 \times \mathbb{N} \to \mathscr{T}_2$, making oracle calls respectively to two independent oracles $\mathcal{O}_1 : \mathscr{D}_1 \to \mathscr{R}_1$ and $\mathcal{O}_2 : \mathscr{D}_2 \to \mathscr{R}_2$, and a KDF combiner $\mathcal{H}^{\mathcal{F}_1,\mathcal{F}_2}$ with label splitter η, context splitter μ, length splitter ν, key splicer ϑ with injectivity coefficient ρ, and final KCF $\mathcal{G} : \mathscr{K}_1 \times \mathscr{K}_2 \times \mathscr{L} \times \mathscr{C} \times \mathbb{N} \to \mathscr{T}$.

We will associate with $\mathcal{H}$ a *predicate splitter* σ, which for any $\ell \in \mathbb{N}$ and any χ over $\{0,1\}^\ell$, takes χ as input and outputs a pair of predicates $\chi_1 := \sigma_1(\chi)$ over $\{0,1\}^{\ell_1}$ and $\chi_2 := \sigma_2(\chi)$ over $\{0,1\}^{\ell_2}$ (where $(\ell_1, \ell_2) := \nu(\ell)$) defined as follows:

- for any $\mathsf{T}_1 \in \{0,1\}^{\ell_1}$, $\chi_1(\mathsf{T}_1) := (\exists \mathsf{T}_2 \in \{0,1\}^{\ell_2})(\chi(\vartheta_\ell(\mathsf{T}_1, \mathsf{T}_2)))$;
- for any $\mathsf{T}_2 \in \{0,1\}^{\ell_2}$, $\chi_2(\mathsf{T}_2) := (\exists \mathsf{T}_1 \in \{0,1\}^{\ell_1})(\chi(\vartheta_\ell(\mathsf{T}_1, \mathsf{T}_2)))$.

Further consider two oracle families $\mathscr{O}_1 : \mathscr{D}_1 \to \mathscr{R}_1$ and $\mathscr{O}_2 : \mathscr{D}_2 \to \mathscr{R}_2$. Let $\mathscr{D} := (\{1\} \times \mathscr{D}_1) \cup (\{2\} \times \mathscr{D}_2)$, and $\mathscr{R} := \mathscr{R}_1 \cup \mathscr{R}_2$. For any $\mathcal{O}_1 \in \mathscr{O}_1, \mathcal{O}_2 \in \mathscr{O}_2$, define as $\mathcal{O}_1 \times \mathcal{O}_2$ the oracle $\mathcal{O} : \mathscr{D} \to \mathscr{R}$ with the following behaviour:

$$\mathcal{O}(1, x) := \mathcal{O}_1(x), \quad \forall x \in \mathscr{D}_1, \\ \mathcal{O}(2, y) := \mathcal{O}_2(y), \quad \forall y \in \mathscr{D}_2,$$

and define the oracle family $\mathscr{O} := \{\mathcal{O}_1 \times \mathcal{O}_2 \mid \mathcal{O}_1 \in \mathscr{O}_1, \mathcal{O}_2 \in \mathscr{O}_2\}$. Let $\mathscr{F}_1 := \{\mathcal{F}_1^{\mathcal{O}_1} \mid \mathcal{O}_1 \in \mathscr{O}_1\}$, $\mathscr{F}_2 := \{\mathcal{F}_2^{\mathcal{O}_2} \mid \mathcal{O}_2 \in \mathscr{O}_2\}$, and $\mathscr{G} := \{\mathcal{G}^{\mathcal{O}} \mid \mathcal{O} \in \mathscr{O}\} = \{\mathcal{G}^{\mathcal{O}_1 \times \mathcal{O}_2} \mid \mathcal{O}_1 \in \mathscr{O}_1, \mathcal{O}_2 \in \mathscr{O}_2\}$. Our core technical result is the following.

Theorem 1. *Suppose, for control functions $\varepsilon_1 : \mathbb{N}^7 \times (0,1] \times \mathscr{X} \to [0,1]$, $\varepsilon_2 : \mathbb{N}^7 \times (0,1] \times \mathscr{X} \to [0,1]$, $\mathscr{F}_1$ is ε_1-KC secure and $\mathscr{F}_2$ is ε_2-KC secure. Consider a CKC constraint-set function $\beta := (q_c, q_f, \lambda^-, \lambda^+, \ell^-, \ell^+, \kappa_1, \kappa_2, \delta) \in \mathcal{Q}$, and define the constraint functions $\lambda_1^-, \lambda_1^+, \ell_1^-, \ell_1^+, \delta_1, \lambda_2^-, \lambda_2^+, \ell_2^-, \ell_2^+, \delta_2$ as follows: for any $n \in \mathcal{N}$,*

- $\lambda_1^-(n) := \min\{|\eta_1(\mathsf{L})| \mid \mathsf{L} \in \mathscr{L}, |\mathsf{L}| \in [\lambda^-(n)..\lambda^+(n)]\}$;
- $\lambda_1^+(n) := \max\{|\eta_1(\mathsf{L})| \mid \mathsf{L} \in \mathscr{L}, |\mathsf{L}| \in [\lambda^-(n)..\lambda^+(n)]\}$;
- $\ell_1^-(n) := \min\{\nu_1(\ell) \mid \ell \in [\ell^-(n)..\ell^+(n)]\}$;
- $\ell_1^+(n) := \max\{\nu_1(\ell) \mid \ell \in [\ell^-(n)..\ell^+(n)]\}$;
- $\delta_1(n) := 2^{\max\{\ell - \nu_1(\ell) \mid \ell \in [\ell^-(n)..\ell^+(n)]\}} \rho\delta(n)$;

- $\lambda_2^-(n) := \min\{|\eta_2(L)| \mid L \in \mathscr{L}, |L| \in [\lambda^-(n)..\lambda^+(n)]\}$;
- $\lambda_2^+(n) := \max\{|\eta_2(L)| \mid L \in \mathscr{L}, |L| \in [\lambda^-(n)..\lambda^+(n)]\}$;
- $\ell_2^-(n) := \min\{\nu_2(\ell) \mid \ell \in [\ell^-(n)..\ell^+(n)]\}$;
- $\ell_2^+(n) := \max\{\nu_2(\ell) \mid \ell \in [\ell^-(n)..\ell^+(n)]\}$;
- $\delta_2(n) := 2^{\max\{\ell - \nu_2(\ell) \mid \ell \in [\ell^-(n)..\ell^+(n)]\}} \rho\delta(n)$.

Consider further a predicate-set function $\mathfrak{X} \in \mathscr{X}$, and define predicate-set functions $\mathfrak{X}_1$ and $\mathfrak{X}_2$ as follows:

- *for any $\ell_1 \in \mathbb{N}$ and any predicate χ_1 over $\{0,1\}^{\ell_1}$,*

$$\chi_1 \in \mathfrak{X}_1(\ell_1) \iff (\exists \ell \in \mathbb{N})\,((\ell_1 = \nu_1(\ell)) \wedge (\exists \chi \in \mathfrak{X}(\ell))(\chi_1 = \sigma_1(\chi)))\,;$$

- *for any $\ell_2 \in \mathbb{N}$ and any predicate χ_2 over $\{0,1\}^{\ell_2}$,*

$$\chi_2 \in \mathfrak{X}_2(\ell_2) \iff (\exists \ell \in \mathbb{N})\,((\ell_2 = \nu_2(\ell)) \wedge (\exists \chi \in \mathfrak{X}(\ell))(\chi_2 = \sigma_2(\chi)))\,.$$

Then, for the control function γ defined as

$$\gamma(\beta(n), \mathfrak{X}) := \min(\varepsilon_1(\alpha_1(n), \mathfrak{X}_1), \varepsilon_2(\alpha_2(n), \mathfrak{X}_2)) \tag{7}$$

for any $n \in \mathscr{N}$, $\mathscr{G}$ is γ-CKC secure, where $\alpha_1 := (q_c, q_f, \lambda_1^-, \lambda_1^+, \ell_1^-, \ell_1^+, \kappa_1, \delta_1)$ and $\alpha_2 := (q_c, q_f, \lambda_2^-, \lambda_2^+, \ell_2^-, \ell_2^+, \kappa_2, \delta_2)$.

Proof. Denote $\mathscr{J} := (\mathscr{K}_1, \mathscr{K}_2, \mathscr{L}, \mathscr{C})$, $\mathscr{I}_1 := (\mathscr{K}_1, \mathscr{L}_1, \mathscr{C}_1)$, and $\mathscr{I}_2 := (\mathscr{K}_2, \mathscr{L}_2, \mathscr{C}_2)$. For any $(\beta, \mathfrak{X}, \mathscr{J})$-CKC adversary $\mathcal{B} := (\mathcal{B}_c, \mathcal{B}_f)$, we will construct an $(\alpha_1, \mathfrak{X}_1, \mathscr{I}_1)$-KC adversary $\mathcal{A}_1 := (\mathcal{A}_{c,1}, \mathcal{A}_{f,1})$ and an $(\alpha_2, \mathfrak{X}_2, \mathscr{I}_2)$-KC adversary $\mathcal{A}_2 := (\mathcal{A}_{c,2}, \mathcal{A}_{f,2})$, such that for any $n \in \mathscr{N}$,

$$\mathbf{Pr}[\mathtt{CKCS}(\mathcal{O}, \mathcal{G}, \mathcal{B}, n)] \leq \min(\mathbf{Pr}[\mathtt{KCS}(\mathcal{O}_1, \mathcal{F}_1, \mathcal{A}_1, n)], \mathbf{Pr}[\mathtt{KCS}(\mathcal{O}_2, \mathcal{F}_2, \mathcal{A}_2, n)]). \tag{8}$$

Since $\mathcal{F}_1$ is ε_1-KC secure and $\mathcal{F}_2$ is ε_2-KC secure, we have

$$\begin{aligned} &\mathbf{Pr}[\mathtt{KCS}(\mathcal{O}_1, \mathcal{F}_1, \mathcal{A}_1, n)] \leq \varepsilon_1(\alpha_1(n), \mathfrak{X}_1), \\ &\mathbf{Pr}[\mathtt{KCS}(\mathcal{O}_2, \mathcal{F}_2, \mathcal{A}_2, n)] \leq \varepsilon_2(\alpha_2(n), \mathfrak{X}_2). \end{aligned} \tag{9}$$

Plugging Ineqs. (9) in Ineq. (8) gives

$$\mathbf{Pr}[\mathtt{CKCS}(\mathcal{O}, \mathcal{G}, \mathcal{B}, n)] \leq \min(\varepsilon_1(\alpha_1(n), \mathfrak{X}_1), \varepsilon_2(\alpha_2(n), \mathfrak{X}_2)) = \gamma(\beta(n), \mathfrak{X}). \tag{10}$$

Since Ineq. (10) holds for any $(\beta, \mathfrak{X}, \mathscr{J})$-CKC adversary $\mathcal{B}$ and any $n \in \mathscr{N}$, this will establish that $\mathcal{G}$ is γ-CKC secure, as promised in the theorem statement.

Description of $\mathcal{A}_1$. When adversary $\mathcal{A}_1$ plays a KCS game against $\mathcal{F}_1$, it simulates a CKCS game against $\mathcal{G}$ for $\mathcal{A}$. We describe the two stages of this game below. Fig. 3 illustrates $\mathcal{A}_{c,1}$, and Fig. 4 illustrates $\mathcal{A}_{f,1}$.

In the Commit stage of the outer game, $\mathcal{A}_{c,1}$ samples $O_2 \leftarrow_\$ \mathcal{O}_2$, and initiates the Commit stage of the inner game against $\mathcal{B}_c$. The queries of $\mathcal{B}_c$ are answered as follows:

- $(1, x)$: $\mathcal{A}_{c,1}$ queries its own oracle O_1 with x, and returns $O_1(x)$ to $\mathcal{A}_c$;

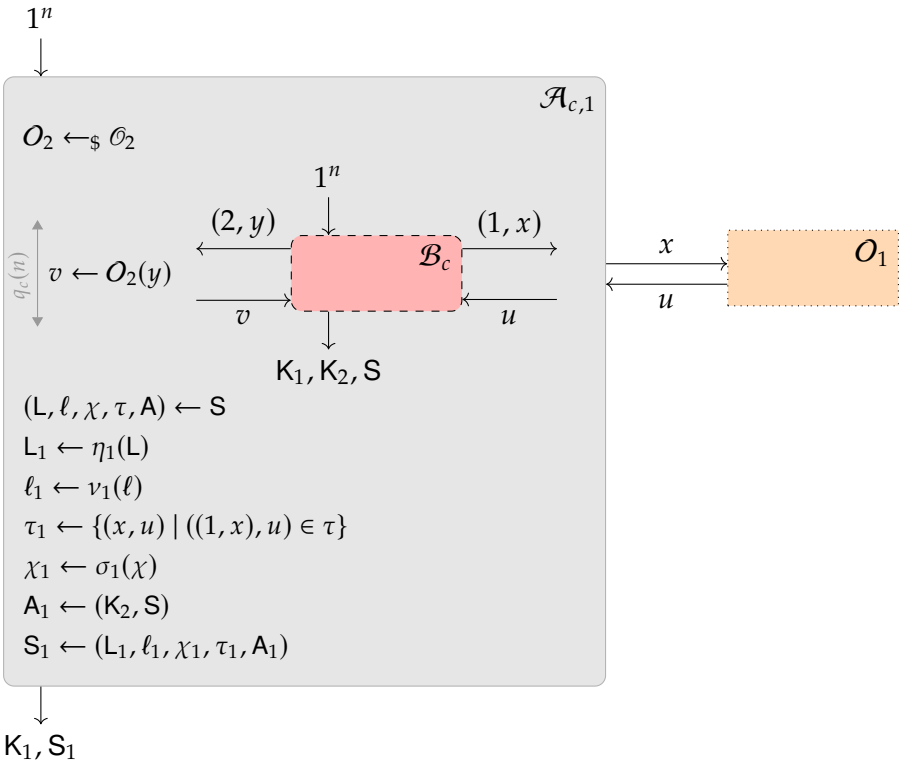


Figure 3: The oracle algorithm $\mathcal{A}_{c,1}^{O_1}$.

- $(2, y)$: $\mathcal{A}_{c,1}$ computes and returns $O_2(y)$ to $\mathcal{A}_c$.[5]

Eventually, $\mathcal{B}_c$ terminates, and outputs $(\mathsf{K}_1, \mathsf{K}_2, \mathsf{S} := (\mathsf{L}, \ell, \chi, \tau, \mathsf{A}))$. Upon receiving this, $\mathcal{A}_{c,1}$ computes $\mathsf{L}_1 \leftarrow \eta_1(\mathsf{L})$, $\ell_1 \leftarrow \nu_1(\ell)$, $\tau_1 \leftarrow \{(x, u) \mid ((1, x), u) \in \tau\}$, and $\chi_1 \leftarrow \sigma_1(\chi)$. Finally, it sets $\mathsf{A}_1 \leftarrow (\mathsf{K}_2, \mathsf{S})$, terminates, and outputs $(\mathsf{K}_1, \mathsf{S}_1 := (\mathsf{L}_1, \ell_1, \chi_1, \tau_1, \mathsf{A}_1))$.

In the Find stage of the outer game, $\mathcal{A}_{f,1}$ defines the family

$$\mathcal{O}_2' := \{O_2' \in \mathcal{O}_2 \mid (\forall((2, y), v) \in \tau)(O_2'(y) = v)\}$$

of oracles which agree with τ on the Commit queries of the form $(2, y)$ in τ, samples $O_2^\dagger \leftarrow_\$ \mathcal{O}_2'$, and initiates the Find stage of the inner game against $\mathcal{B}_f$. The queries of $\mathcal{B}_f$ are answered as follows:

- $(1, x)$: $\mathcal{A}_{f,1}$ queries its own oracle O_1 with x, and returns $O_1(x)$ to $\mathcal{A}_f$;
- $(2, y)$: $\mathcal{A}_{f,1}$ computes and returns $O_2^\dagger(y)$ to $\mathcal{A}_f$.

Eventually, $\mathcal{B}_f$ terminates, and outputs $\mathsf{C} \in \mathscr{C}$. On receiving this, $\mathcal{A}_{f,1}$ computes $\mathsf{C}_1 := \mu_1(\mathsf{C})$, terminates, and outputs C_1.

Validating $\mathcal{A}_1$. We need to show that as long as $\mathcal{B}$ is a valid $(\beta, \mathfrak{X}, \mathcal{J})$-CKC adversary, $\mathcal{A}_1$ is a valid $(\alpha_1, \mathfrak{X}_1, \mathcal{I}_1)$-KC adversary. The following are simple to observe from properties of $\mathcal{B}$ and the functions η, μ, σ:

- $\mathsf{K}_1 \in \mathscr{K}_1$;

[5] Since $\mathcal{A}_{c,1}$ samples O_2, we can assume it knows its circuit and can compute it directly without oracle queries.

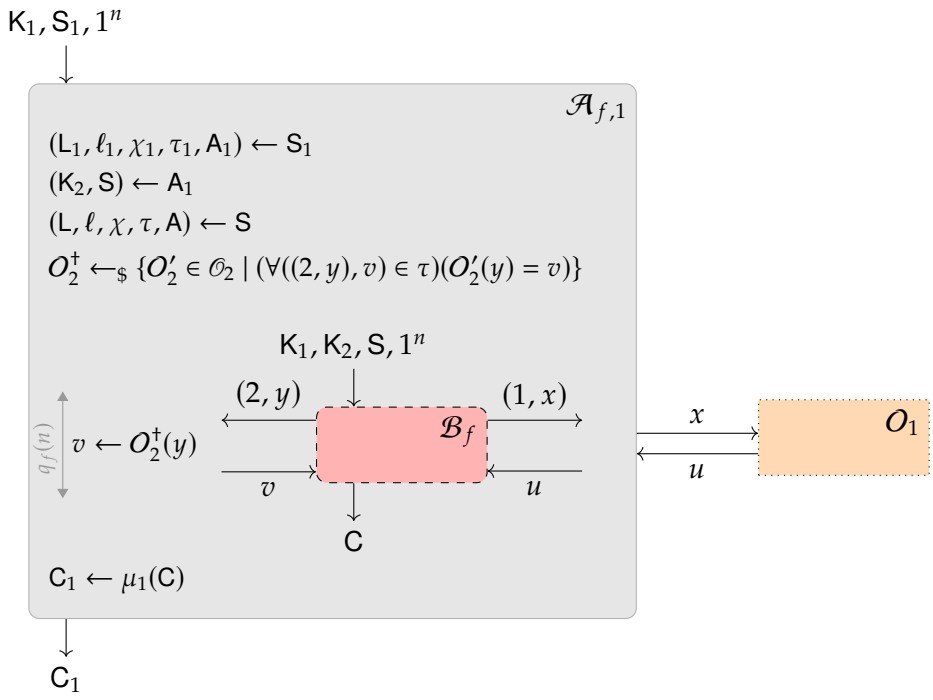


Figure 4: The oracle algorithm $\mathcal{A}_{f,1}^{O_1}$.

- $\mathsf{L}_1 = \eta_1(\mathsf{L}) \in \mathscr{L}_1$;
- $\mathsf{C}_1 = \mu_1(\mathsf{C}) \in \mathscr{C}_1$;
- $\chi_1 = \sigma_1(\chi) \in \mathfrak{X}_1(\ell_1)$.

Further, for all $n \in \mathcal{N}$, the following follow from the definition of α_1:

- $\mathcal{A}_{c,1}$ makes at most $q_c(n) = q_{c,1}(n)$ oracle queries, since each oracle query by $\mathcal{B}_c$ leads to at most one oracle query by $\mathcal{A}_{c,1}$;
- similarly, $\mathcal{A}_{f,1}$ makes at most $q_f(n) = q_{f,1}(n)$ oracle queries;
- when $|\mathsf{L}| \in [\lambda^-(n)..\lambda^+(n)]$, $|\mathsf{L}_1| = |\eta_1(\mathsf{L})| \in [\lambda_1^-(n)..\lambda_1^+(n)]$;
- when $\ell \in [\ell^-(n)..\ell^+(n)]$, $\ell_1 = \nu_1(\ell) \in [\ell_1^-(n)..\ell_1^+(n)]$.

This leaves the restrictions in Ineqs. (2) and (3). Since S_1 is fully determined by A_1, we have

$$\tilde{\mathbf{H}}_\infty(\mathsf{K}_1 \mid \mathsf{S}_1) = \tilde{\mathbf{H}}_\infty(\mathsf{K}_1 \mid \mathsf{A}_1) = \tilde{\mathbf{H}}_\infty(\mathsf{K}_1 \mid \mathsf{K}_2, \mathsf{S}) \geq \kappa_1(n),$$

the last inequality following from Ineq. (4), which $\mathcal{B}_c$ must obey since $\mathcal{B}$ is a valid $(\beta, \mathfrak{X}, \mathscr{J})$-CKC adversary. Thus, $\mathcal{A}_{c,1}$ obeys Ineq. (2). Next, observe that

$$\begin{aligned}
\delta_{\chi_1,\ell_1} &= \mathbf{Pr}[\chi_1(\mathsf{U}_{\ell_1})] \\
&= \mathbf{Pr}[(\exists \mathsf{T}_2 \in \{0,1\}^{\ell_2})(\chi(\vartheta_\ell(\mathsf{U}_{\ell_1}, \mathsf{T}_2)))] \\
&= \frac{|\{\mathsf{T}_1 \in \{0,1\}^{\ell_1} \mid (\exists \mathsf{T}_2 \in \{0,1\}^{\ell_2})(\chi(\vartheta_\ell(\mathsf{T}_1, \mathsf{T}_2)))\}|}{2^{\ell_1}} \\
&\leq \frac{\rho \cdot |\{\mathsf{T} \in \{0,1\}^{\ell} \mid \chi(\mathsf{T})\}|}{2^{\ell_1}}
\end{aligned} \tag{11}$$

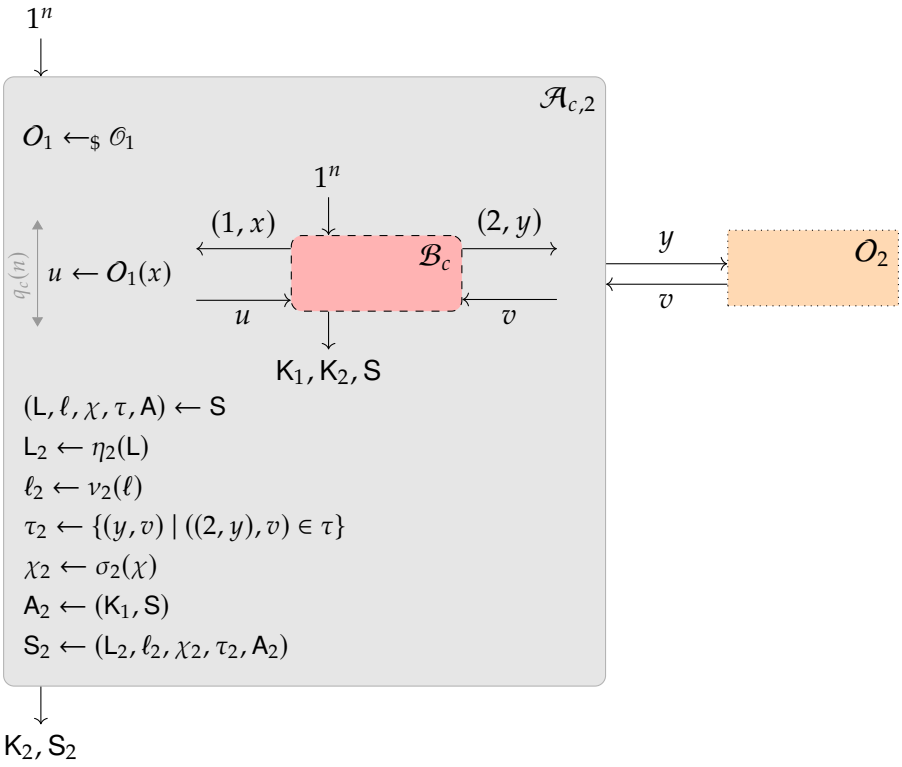


Figure 5: The oracle algorithm $\mathcal{A}_{c,2}^{O_2}$.

$$
\begin{aligned}
&= 2^{\ell-\ell_1}\rho \cdot \frac{|\{\mathsf{T} \in \{0,1\}^\ell \mid \chi(\mathsf{T})\}|}{2^\ell} \\
&= 2^{\ell-\nu_1(\ell)}\rho \cdot \mathbf{Pr}[\chi(\mathsf{U}_\ell)] \\
&= 2^{\ell-\nu_1(\ell)}\rho \cdot \delta_{\chi,\ell} \\
&\le 2^{\max\{\ell-\nu_1(\ell) \mid \ell \in [\ell^-(n)..\ell^+(n)]\}}\rho \cdot \delta_{\chi,\ell} = \frac{\delta_1(n)}{\delta(n)} \cdot \delta_{\chi,\ell}, \qquad (12)
\end{aligned}
$$

where Ineq. (11) follows from the ρ-injectivity of ϑ_ℓ. Taking expectations on both sides of Ineq. (12) yields

$$
\begin{aligned}
\mathop{\mathbf{Ex}}_{\mathsf{S}_1 \leftarrow_\$ \mathcal{A}_{c,1}^{O_1}} \left[\delta_{\chi_1,\ell_1}\right] &= \mathop{\mathbf{Ex}}_{\mathsf{S}_1 \leftarrow_\$ \mathcal{A}_{c,1}^{O_1}} \left[\mathbf{Pr}[\chi_1(\mathsf{U}_{\ell_1})]\right] \\
&\le \mathop{\mathbf{Ex}}_{\mathsf{S} \leftarrow_\$ \mathcal{A}_c^{O}} \left[\frac{\delta_1(n)}{\delta(n)} \cdot \delta_{\chi,\ell}\right] \\
&\le \frac{\delta_1(n)}{\delta(n)} \cdot \mathop{\mathbf{Ex}}_{\mathsf{S} \leftarrow_\$ \mathcal{A}_c^{O}} \left[\delta_{\chi,\ell}\right] \le \frac{\delta_1(n)}{\delta(n)} \cdot \delta(n) = \delta_1(n). \qquad (13)
\end{aligned}
$$

Thus, $\mathcal{A}_{c,1}$ also obeys Ineq. (3), and $\mathcal{A}_c$ is a valid $(\alpha_1, \mathfrak{X}_1, \mathcal{I}_1)$-KC adversary.

Description of $\mathcal{A}_2$. $\mathcal{A}_2$ is defined analogously to $\mathcal{A}_1$. When adversary $\mathcal{A}_2$ plays a KCS game against $\mathcal{F}_2$, it simulates a CKCS game against $\mathcal{G}$ for $\mathcal{A}$. We describe the two stages of this game below. Fig. 5 illustrates $\mathcal{A}_{c,2}$, and Fig. 6 illustrates $\mathcal{A}_{f,2}$.

In the Commit stage of the outer game, $\mathcal{A}_{c,2}$ samples $O_1 \leftarrow_\$ \mathcal{O}_1$, and initiates the Commit stage of the inner game against $\mathcal{B}_c$. The queries of $\mathcal{B}_c$ are answered as follows:

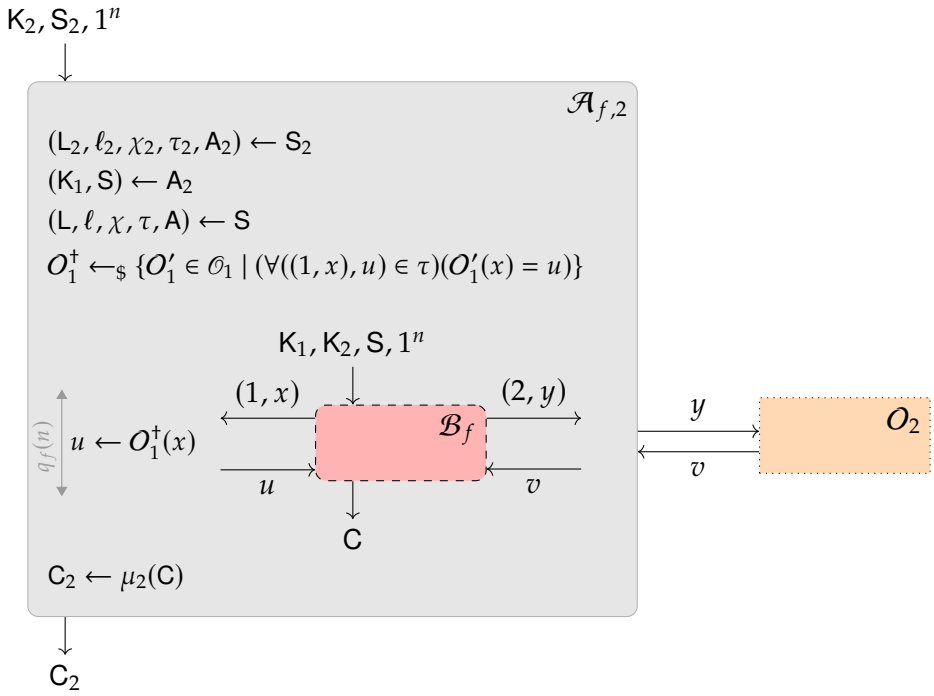


Figure 6: The oracle algorithm $\mathcal{A}_{f,2}^{O_2}$.

- $(1, x)$: $\mathcal{A}_{c,2}$ computes and returns $O_1(x)$ to $\mathcal{A}_c$;
- $(2, y)$: $\mathcal{A}_{c,2}$ queries its own oracle O_2 with y, and returns $O_2(y)$ to $\mathcal{A}_c$.

Eventually, $\mathcal{B}_c$ terminates, and outputs $(\mathsf{K}_1, \mathsf{K}_2, \mathsf{S} := (\mathsf{L}, \ell, \chi, \tau, \mathsf{A}))$. Upon receiving this, $\mathcal{A}_{c,2}$ computes $\mathsf{L}_2 \leftarrow \eta_2(\mathsf{L})$, $\ell_2 \leftarrow \nu_2(\ell)$, $\tau_2 \leftarrow \{(y, v) \mid ((2, y), v) \in \tau\}$, and $\chi_2 \leftarrow \sigma_2(\chi)$. Finally, it sets $\mathsf{A}_2 \leftarrow (\mathsf{K}_1, \mathsf{S})$, terminates, and outputs $(\mathsf{K}_2, \mathsf{S}_2 := (\mathsf{L}_2, \ell_2, \chi_2, \tau_2, \mathsf{A}_2))$.

In the Find stage of the outer game, $\mathcal{A}_{f,2}$ defines the family

$$\mathcal{O}_1' := \{O_1' \in \mathcal{O}_1 \mid (\forall((1,x),u) \in \tau)(O_1'(x) = u)\}$$

of oracles which agree with τ on the Commit queries of the form $(1, x)$ in τ, samples $O_1^\dagger \leftarrow_\$ \mathcal{O}_1'$, and initiates the Find stage of the inner game against $\mathcal{B}_f$. The queries of $\mathcal{B}_f$ are answered as follows:

- $(1, x)$: $\mathcal{A}_{f,2}$ computes and returns $O_1^\dagger(x)$ to $\mathcal{A}_f$;
- $(2, y)$: $\mathcal{A}_{f,2}$ queries its own oracle O_2 with y, and returns $O_2(y)$ to $\mathcal{A}_f$.

Eventually, $\mathcal{B}_f$ terminates, and outputs $\mathsf{C} \in \mathscr{C}$. On receiving this, $\mathcal{A}_{f,2}$ computes $\mathsf{C}_2 := \mu_2(\mathsf{C})$, terminates, and outputs C_2.

Validating $\mathcal{A}_2$. We need to show that as long as $\mathcal{B}$ is a valid $(\beta, \mathfrak{X}, \mathcal{J})$-CKC adversary, $\mathcal{A}_2$ is a valid $(\alpha_2, \mathfrak{X}_2, \mathcal{J}_2)$-KC adversary. The following are simple to observe from properties of $\mathcal{B}$ and the functions η, μ, σ:

- $\mathsf{K}_2 \in \mathscr{K}_2$;
- $\mathsf{L}_2 = \eta_2(\mathsf{L}) \in \mathscr{L}_2$;
- $\mathsf{C}_2 = \mu_2(\mathsf{C}) \in \mathscr{C}_2$;

- $\chi_2 = \sigma_2(\chi) \in \mathfrak{X}_2(\ell_2)$.

Further, for all $n \in \mathcal{N}$, the following follow from the definition of α_2:

- $\mathcal{A}_{c,2}$ makes at most $q_c(n) = q_{c,2}(n)$ oracle queries, since each oracle query by $\mathcal{B}_c$ leads to at most one oracle query by $\mathcal{A}_{c,2}$;
- similarly, $\mathcal{A}_{f,2}$ makes at most $q_f(n) = q_{f,2}(n)$ oracle queries;
- when $|\mathsf{L}| \in [\lambda^-(n)..\lambda^+(n)]$, $|\mathsf{L}_2| = |\eta_2(\mathsf{L})| \in [\lambda_2^-(n)..\lambda_2^+(n)]$;
- when $\ell \in [\ell^-(n)..\ell^+(n)]$, $\ell_2 = \nu_2(\ell) \in [\ell_2^-(n)..\ell_2^+(n)]$.

This leaves the restrictions in Ineqs. (2) and (3). Since S_2 is fully determined by A_2, we have

$$\tilde{\mathbf{H}}_\infty(\mathsf{K}_2 \mid \mathsf{S}_2) = \tilde{\mathbf{H}}_\infty(\mathsf{K}_2 \mid \mathsf{A}_2) = \tilde{\mathbf{H}}_\infty(\mathsf{K}_2 \mid \mathsf{K}_1, \mathsf{S}) \geq \kappa_2(n),$$

the last inequality following from Ineq. (4), which $\mathcal{B}_c$ must obey since $\mathcal{B}$ is a valid $(\beta, \mathfrak{X}, \mathcal{J})$-CKC adversary. Thus, $\mathcal{A}_{c,2}$ obeys Ineq. (2). Next, observe that

$$\begin{aligned}
\delta_{\chi_2,\ell_2} &= \mathbf{Pr}[\chi_2(\mathsf{U}_{\ell_1})] \\
&= \mathbf{Pr}[(\exists \mathsf{T}_1 \in \{0,1\}^{\ell_1})(\chi(\vartheta_\ell(\mathsf{U}_{\ell_1}, \mathsf{T}_1)))] \\
&= \frac{|\{\mathsf{T}_2 \in \{0,1\}^{\ell_2} \mid (\exists \mathsf{T}_1 \in \{0,1\}^{\ell_1})(\chi(\vartheta_\ell(\mathsf{T}_2, \mathsf{T}_1)))\}|}{2^{\ell_2}} \\
&\leq \frac{\rho \cdot |\{\mathsf{T} \in \{0,1\}^{\ell} \mid \chi(\mathsf{T})\}|}{2^{\ell_2}} \qquad (14) \\
&= 2^{\ell-\ell_2}\rho \cdot \frac{|\{\mathsf{T} \in \{0,1\}^{\ell} \mid \chi(\mathsf{T})\}|}{2^{\ell}} \\
&= 2^{\ell-\nu_2(\ell)}\rho \cdot \mathbf{Pr}[\chi(\mathsf{U}_\ell)] \\
&= 2^{\ell-\nu_2(\ell)}\rho \cdot \delta_{\chi,\ell} \\
&\leq 2^{\max\{\ell-\nu_2(\ell) \mid \ell \in [\ell^-(n)..\ell^+(n)]\}}\rho \cdot \delta_{\chi,\ell} = \frac{\delta_2(n)}{\delta(n)} \cdot \delta_{\chi,\ell}, \qquad (15)
\end{aligned}$$

where Ineq. (14) follows from the ρ-injectivity of ϑ_ℓ. Taking expectations on both sides of Ineq. (15) yields

$$\begin{aligned}
\mathop{\mathbf{Ex}}_{\mathsf{S}_2 \leftarrow\$ \mathcal{A}_{c,2}^{\mathcal{O}_2}}\left[\delta_{\chi_2,\ell_2}\right] &= \mathop{\mathbf{Ex}}_{\mathsf{S}_2 \leftarrow\$ \mathcal{A}_{c,2}^{\mathcal{O}_2}}\left[\mathbf{Pr}[\chi_2(\mathsf{U}_{\ell_1})]\right] \\
&\leq \mathop{\mathbf{Ex}}_{\mathsf{S} \leftarrow\$ \mathcal{A}_c^{\mathcal{O}}}\left[\frac{\delta_2(n)}{\delta(n)} \cdot \delta_{\chi,\ell}\right] \\
&\leq \frac{\delta_2(n)}{\delta(n)} \cdot \mathop{\mathbf{Ex}}_{\mathsf{S} \leftarrow\$ \mathcal{A}_c^{\mathcal{O}}}\left[\delta_{\chi,\ell}\right] \leq \frac{\delta_2(n)}{\delta(n)} \cdot \delta(n) = \delta_2(n). \qquad (16)
\end{aligned}$$

Thus, $\mathcal{A}_{c,2}$ also obeys Ineq. (3), and $\mathcal{A}_c$ is a valid $(\alpha_2, \mathfrak{X}_2, \mathcal{I}_2)$-KC adversary.

Derivation of Ineq. (8). To complete the proof, we need to establish Ineq. (8). To this end, we first identify the sources of randomness in the three games involved in Ineq. (8):

- $\mathtt{CKCS}(\mathcal{O}, \mathcal{G}, \mathcal{B}, n)$: from Fig. 2, the sources of randomness are the game's random coins used in the sampling of $\mathcal{O}$ from $\mathcal{O}$, $\mathcal{B}_c$'s own random coins, and $\mathcal{B}_f$'s own random coins;

- $\texttt{KCS}(\mathcal{O}_1, \mathcal{F}_1, \mathcal{A}_1, n)$: from Figs. 1, 3, and 4, the sources of randomness are the game's random coins used in the sampling of O_1 from $\mathcal{O}_1$, $\mathcal{A}_{c,1}$'s own random coins used in the sampling of O_2 from $\mathcal{O}_2$, $\mathcal{B}_c$'s own random coins, $\mathcal{A}_{f,1}$'s own random coins used in the sampling of O_2 from $\mathcal{O}'_2$, and $\mathcal{B}_f$'s own random coins;
- $\texttt{KCS}(\mathcal{O}_2, \mathcal{F}_2, \mathcal{A}_2, n)$: from Figs. 1, 5, and 6, the sources of randomness are the game's random coins used in the sampling of O_2 from $\mathcal{O}_2$, $\mathcal{A}_{c,2}$'s own random coins used in the sampling of O_1 from $\mathcal{O}_1$, $\mathcal{B}_c$'s own random coins, $\mathcal{A}_{f,2}$'s own random coins used in the sampling of O_1 from $\mathcal{O}'_1$, and $\mathcal{B}_f$'s own random coins.

Game $\texttt{CKCSR1}(\mathcal{O}, \mathcal{G}, \mathcal{B} := (\mathcal{B}_c, \mathcal{B}_f), n)$

1 : $O_1 \leftarrow_\$ \mathcal{O}_1$
2 : $O_2 \leftarrow_\$ \mathcal{O}_2$
3 : $O \leftarrow O_1 \times O_2$
4 : $(\mathsf{K}_1, \mathsf{K}_2, \mathsf{S} := (\mathsf{L}, \ell, \chi, \tau, \mathsf{A})) \leftarrow_\$ \mathcal{B}_c^{O}(1^n)$ // Commit Stage
5 : $O_2^\dagger \leftarrow_\$ \{O'_2 \in \mathcal{O}_2 \mid (\forall((2, y), v) \in \tau)(O'_2(y) = v)\}$ // Resampling
6 : $O^\dagger \leftarrow O_1 \times O_2^\dagger$
7 : $\mathsf{C} \leftarrow_\$ \mathcal{B}_f^{O^\dagger}(\mathsf{K}_1, \mathsf{K}_2, \mathsf{S}, 1^n)$ // Find Stage
8 : $\mathsf{T} \leftarrow \mathcal{G}^{O^\dagger}(\mathsf{K}_1, \mathsf{K}_2, \mathsf{L}, \mathsf{C}, \ell)$
9 : **return** $\chi(\mathsf{T})$

Figure 7: The random experiment defining the CKCSR1 security notion.

We observe that $\mathcal{O}$ is defined as $\{O_1 \times O_2 \mid O_1 \in \mathcal{O}_1, O_2 \in \mathcal{O}_2\}$, so sampling $O \leftarrow_\$ \mathcal{O}$ is distributionally equivalent to independently sampling $O_1 \leftarrow_\$ \mathcal{O}_1$ and $O_2 \leftarrow_\$ \mathcal{O}_2$ and setting $O := O_1 \times O_2$. Without loss of generality, we will assume henceforth that in $\texttt{CKCS}(\mathcal{O}, \mathcal{G}, \mathcal{B}, n)$ the sampling of O is carried out as above.

We consider two further modifications of the CKCS experiment:

- In CKCSR1 (Fig. 7), O_2 is resampled after the Commit stage, following the restriction that the new oracle $O_2^\dagger$ agrees with O_2 on all points in τ;
- In CKCSR2 (Fig. 8), O_1 is resampled after the Commit stage, following the restriction that the new oracle $O_1^\dagger$ agrees with O_1 on all points in τ.

We note that an oracle algorithm calling two oracles O_1 and O_2 can be seen as a single oracle algorithm with oracle access to O_2 by fixing O_1 (and vice versa). Thus Lemma 1 is applicable to both CKCSR1 and CKCSR2, and we can conclude that

$$\mathbf{Pr}[\texttt{CKCSR1}(\mathcal{O}, \mathcal{G}, \mathcal{B}, n)] = \mathbf{Pr}[\texttt{CKCSR2}(\mathcal{O}, \mathcal{G}, \mathcal{B}, n)] = \mathbf{Pr}[\texttt{CKCS}(\mathcal{O}, \mathcal{G}, \mathcal{B}, n)]. \tag{17}$$

Now consider the instance of $\texttt{CKCSR1}(\mathcal{O}, \mathcal{G}, \mathcal{B}, n)$ where the random coins used to sample O_1, O_2, and $O_2^\dagger$ are r_1, r_2, and $r_2^\dagger$ respectively, while the internal random coins used by $\mathcal{B}_c$ and $\mathcal{B}_f$ are r_c and r_f respectively. Denote the final output of this instance as $\texttt{CKCSR1}(\mathcal{O}, \mathcal{G}, \mathcal{B}, n)[r_1, r_2, r_c, r_2^\dagger, r_f]$ (reflecting the order in which they come into play).

Using the same coins, we can construct an instance of $\texttt{KCS}(\mathcal{O}_1, \mathcal{F}_1, \mathcal{A}_1, n)$ as follows: the game uses r_1 to sample O_1, $\mathcal{A}_{c,1}$ uses r_2 to sample O_2, $\mathcal{B}_c$ uses r_c as internal coins, $\mathcal{A}_{f,1}$ uses $r_2^\dagger$ to sample $O_2^\dagger$, and $\mathcal{B}_f$ uses r_f as internal coins. Denote the final output of this instance as $\texttt{KCS}(\mathcal{O}_1, \mathcal{F}_1, \mathcal{A}_1, n)[r_1, r_2, r_c, r_2^\dagger, r_f]$.

Game $\texttt{CKCSR2}(\mathcal{O}, \mathcal{G}, \mathcal{B} := (\mathcal{B}_c, \mathcal{B}_f), n)$

$O_1 \leftarrow_\$ \mathcal{O}_1$
$O_2 \leftarrow_\$ \mathcal{O}_2$
$O \leftarrow O_1 \times O_2$
$(\mathsf{K}_1, \mathsf{K}_2, \mathsf{S} := (\mathsf{L}, \ell, \chi, \tau, \mathsf{A})) \leftarrow_\$ \mathcal{B}_c^{O}(1^n)$ // Commit Stage
$O_1^\dagger \leftarrow_\$ \{O_1' \in \mathcal{O}_1 \mid (\forall((1, x), u) \in \tau)(O_1'(x) = u)\}$ // Resampling
$O^\dagger \leftarrow O_1^\dagger \times O_2$
$\mathsf{C} \leftarrow_\$ \mathcal{B}_f^{O^\dagger}(\mathsf{K}_1, \mathsf{K}_2, \mathsf{S}, 1^n)$ // Find Stage
$\mathsf{T} \leftarrow \mathcal{G}^{O^\dagger}(\mathsf{K}_1, \mathsf{K}_2, \mathsf{L}, \mathsf{C}, \ell)$
return $\chi(\mathsf{T})$

Figure 8: The random experiment defining the CKCSR2 security notion.

The two instances begin with the same O_1 and O_2. The executions of $\mathcal{B}_c$ are also identical, so the outputs $\mathsf{K}_1, \mathsf{K}_2$ and $\mathsf{S} := (\mathsf{L}, \ell, \chi, \tau, \mathsf{A})$ of $\mathcal{B}_c$ are the same. In the KCS instance, this is followed by some deterministic 'packing' steps in $\mathcal{A}_{c,1}$, followed by some deterministic 'unpacking' steps in $\mathcal{A}_{f,1}$, to eventually retrieve $(\mathsf{K}_1, \mathsf{K}_2, \mathsf{S})$; in the CKCSR1 instance, these steps are skipped and $\mathcal{B}_f$ directly receives $(\mathsf{K}_1, \mathsf{K}_2, \mathsf{S})$. $O_2^\dagger$ is again the same in the two instances (since τ is the same). The executions of $\mathcal{B}_f$ are also identical, to output the same C.

At this point, the two instances finally diverge. Denote $(\mathsf{L}_1, \mathsf{L}_2) := \eta(\mathsf{L})$, $(\mathsf{C}_1, \mathsf{C}_2) := \mu(\mathsf{C})$, and $(\ell_1, \ell_2) := \nu(\ell)$. In the KCS instance, $\mathcal{A}_{f,1}$ outputs C_1 and the final output of the game is $\chi_1(\mathcal{F}_1^{O_1}(\mathsf{K}_1, \mathsf{L}_1, \mathsf{C}_1, \ell_1))$; in the CKCSR1 instance, the final output of the game is $\chi(\mathcal{G}^{O^\dagger}(\mathsf{K}_1, \mathsf{K}_2, \mathsf{L}, \mathsf{C}, \ell))$. Since $\mathcal{G} = \mathcal{H}^{\mathcal{F}_1, \mathcal{F}_2}$, we have

$$\mathcal{G}^{O^\dagger}(\mathsf{K}_1, \mathsf{K}_2, \mathsf{L}, \mathsf{C}, \ell) = \vartheta_\ell\left(\mathcal{F}_1^{O_1}(\mathsf{K}_1, \mathsf{L}_1, \mathsf{C}_1, \ell_1), \mathcal{F}_2^{O_2^\dagger}(\mathsf{K}_2, \mathsf{L}_2, \mathsf{C}_2, \ell_2)\right). \tag{18}$$

Writing $\mathsf{T}_1 := \mathcal{F}_1^{O_1}(\mathsf{K}_1, \mathsf{L}_1, \mathsf{C}_1, \ell_1)$, and $\mathsf{T}_2^\dagger := \mathcal{F}_2^{O_2^\dagger}(\mathsf{K}_2, \mathsf{L}_2, \mathsf{C}_2, \ell_2)$, we can rewrite Eq. (18) as

$$\mathcal{G}^{O^\dagger}(\mathsf{K}_1, \mathsf{K}_2, \mathsf{L}, \mathsf{C}, \ell) = \vartheta_\ell(\mathsf{T}_1, \mathsf{T}_2^\dagger).$$

Thus we have

$$\begin{aligned} \texttt{CKCSR1}(\mathcal{O}, \mathcal{G}, \mathcal{B}, n)[r_1, r_2, r_c, r_2^\dagger, r_f] &= \chi(\vartheta_\ell(\mathsf{T}_1, \mathsf{T}_2^\dagger)), \\ \texttt{KCS}(\mathcal{O}_1, \mathcal{F}_1, \mathcal{A}_1, n)[r_1, r_2, r_c, r_2^\dagger, r_f] &= \chi_1(\mathsf{T}_1). \end{aligned} \tag{19}$$

We recall that

$$\chi_1(\mathsf{T}_1) = (\exists \mathsf{T}_2 \in \{0,1\}^{\ell_2})(\chi(\vartheta_\ell(\mathsf{T}_1, \mathsf{T}_2))). \tag{20}$$

By definition of $\mathsf{T}_2^\dagger$, we have $|\mathsf{T}_2^\dagger| = \ell_2$, so $\mathsf{T}_2^\dagger \in \{0,1\}^{\ell_2}$. Thus, if $\chi(\vartheta_\ell(\mathsf{T}_1, \mathsf{T}_2^\dagger))$ is true, $\mathsf{T}_2^\dagger$ serves as a witness for the right hand side of Eq. (20), so we have

$$\chi(\vartheta_\ell(\mathsf{T}_1, \mathsf{T}_2^\dagger)) \implies \chi_1(\mathsf{T}_1). \tag{21}$$

From Eqs. (19) and (21), we get

$$\texttt{CKCSR1}(\mathcal{O}, \mathcal{G}, \mathcal{B}, n)[r_1, r_2, r_c, r_2^\dagger, r_f] \implies \texttt{KCS}(\mathcal{O}_1, \mathcal{F}_1, \mathcal{A}_1, n)[r_1, r_2, r_c, r_2^\dagger, r_f]. \tag{22}$$

Taking probabilities on both sides over the choice of random coins in (22) gives

$$\mathbf{Pr}[\mathtt{CKCSR1}(\mathcal{O},\mathcal{G},\mathcal{B},n)] \leq \mathbf{Pr}[\mathtt{KCS}(\mathcal{O}_1,\mathcal{F}_1,\mathcal{A}_1,n)]. \tag{23}$$

By a symmetric argument, we can show that

$$\mathbf{Pr}[\mathtt{CKCSR2}(\mathcal{O},\mathcal{G},\mathcal{B},n)] \leq \mathbf{Pr}[\mathtt{KCS}(\mathcal{O}_2,\mathcal{F}_2,\mathcal{A}_2,n)]. \tag{24}$$

Ineq. (8) follows on plugging in Eq. (17) in Ineqs. (23) and (24), thus completing the proof. □ □

6 Discussion

Our Definition of KC Security. The way we define KC security in Sec. 3 differs in some ways from the way it is defined in [BDI+25]. For one, we make the dependence of certain parameters on the security parameter n explicit. While here we stick to an information theoretic setting and do not get into running times and complexities, in a concrete instantiation of the game the running times and tape sizes of different adversaries would be functions of n. Another difference is that we let $\mathfrak{X}$ be a parameter of the game, to capture predicates 'of interest' to the application in question. The control function ε also takes $\mathfrak{X}$ as an input, which allows for fine-tuning based on predicates of interest; for instance, if $\mathfrak{X}$ contains predicates which can break KC security but do not lead to meaningful vulnerabilities, $\varepsilon(\cdot,\mathfrak{X},\cdot)$ can be set to 1, so the construction still remains ε-KC secure.[6]

Interpretation of Theorem 1. While by and large this theorem tells us that a KDF combiner guarantees CKC security as long as one of its component KDFs has KC security, there are nuances to consider. It is important to note how we split predicates: $\chi_1(\mathsf{T}_1)$ is true as long as T_1 can be paired with any T_2 such that $\chi(\vartheta(\mathsf{T}_1,\mathsf{T}_2))$ holds. This does not capture well predicates where T_1 and T_2 are compared. For instance, if $\nu_1(\ell) = \lfloor \ell/2 \rfloor$, $\nu_2(\ell) = \lceil \ell/2 \rceil$, $\vartheta(\mathsf{T}_1,\mathsf{T}_2) = \mathsf{T}_1 \| \mathsf{T}_2$, and $\chi(\vartheta(\mathsf{T}_1,\mathsf{T}_2)) = [\mathsf{T}_1 = \mathsf{T}_2]$, both χ_1 and χ_2 will always return 1. This does not mean that such a KDF combiner is necessarily broken; it just means that our generic reduction fails to give a meaningful bound here, and a more dedicated analysis is needed.[7]

Decay Factors: ρ and $2^{\ell-\ell_1}$. It is also important to note that the CKC security of the combiner has two decay factors. First, it decays with the injectivity coefficient ρ of ϑ_ℓ, so the reduction does not work very well with splicers that have a high ρ. One example of such a 'bad' splicer is $\vartheta(\mathsf{T}_1,\mathsf{T}_2) = \mathsf{T}_1 \oplus \mathsf{T}_2$ (corresponding to the XOR combiner). On the other hand, it works well with the fully injective splicer of a concatenation combiner (from the previous paragraph's example). The other decay factor is $\max(2^{\ell-\ell_1})$. This can make the bound trivial (greater than 1) when $\delta(n)$ is already large to begin with, while it works well for small $\delta(n)$. Since the original use case of KC security considered a single target key (i.e., $\delta(n) = 2^{-\ell}$), it is safe to say a bound that works for small $\delta(n)$ is still useful.[8]

References

[AKMZ04] Carlisle Adams, Guenther Kramer, Serge Mister, and Robert Zuccherato. On The Security of Key Derivation Functions. In Kan Zhang and Yuliang Zheng, editors, *Information Security*, pages 134–145, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. *(Cited on p. 1.)*

[6] Both of these modifications also apply to CKC security.

[7] The bound in Theorem 1 is only meaningful if at least one of $\delta_1(n)$ and $\delta_2(n)$ is small.

[8] Again, in both examples, the CKC security of the combiners is not necessarily bad; we can only say that our reduction does not give good bounds for them.

[BDI+25] Ritam Bhaumik, Avijit Dutta, Akiko Inoue, Tetsu Iwata, Ashwin Jha, Kazuhiko Minematsu, Mridul Nandi, Yu Sasaki, Meltem Sönmez Turan, and Stefano Tessaro. Cryptographic Treatment of Key Control Security: In Light of NIST SP 800-108. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025*, pages 371–403, Cham, 2025. Springer Nature Switzerland. *(Cited on pp. 2, 7, and 19.)*

[BHMS17] Nina Bindel, Udyani Herath, Matthew McKague, and Douglas Stebila. Transitioning to a Quantum-Resistant Public Key Infrastructure. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography*, pages 384–405, Cham, 2017. Springer International Publishing. *(Cited on p. 2.)*

[Che22] Lily Chen. Recommendation for Key Derivation Using Pseudorandom Functions. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-108 Rev. 1, 8 2022. *(Cited on p. 2.)*

[GHP18] Federico Giacon, Felix Heuer, and Bertram Poettering. KEM Combiners. In Michel Abdalla and Ricardo Dahab, editors, *Public-Key Cryptography – PKC 2018*, pages 190–218, Cham, 2018. Springer International Publishing. *(Cited on p. 2.)*

[Her05] Amir Herzberg. On Tolerant Cryptographic Constructions. In Alfred Menezes, editor, *Topics in Cryptology – CT-RSA 2005*, pages 172–190, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg. *(Cited on p. 2.)*

[KKJY14] Ju-Sung Kang, Nayoung Kim, Wangho Ju, and Ok-Yeon Yi. A Security Analysis of Key Expansion Functions Using Pseudorandom Permutations. In David Naccache and Damien Sauveron, editors, *Information Security Theory and Practice. Securing the Internet of Things*, pages 10–23, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. *(Cited on p. 1.)*

[Kra10] Hugo Krawczyk. Cryptographic Extraction and Key Derivation: The HKDF Scheme. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, pages 631–648, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. *(Cited on p. 1.)*

[Nat22] National Institute of Standards and Technology. Public Comments and Resolutions on Draft NIST SP 800-108 Revision 1, Recommendation for Key Derivation Using Pseudorandom Functions. Available at https://csrc.nist.gov/pubs/sp/800/108/r1/upd1/final, 1 2022. *(Cited on p. 2.)*

[PR20] Bertram Poettering and Paul Rösler. Combiners for AEAD. *IACR Transactions on Symmetric Cryptology*, 2020(1):121–143, May 2020. *(Cited on p. 2.)*

[SWG25] Yaobin Shen, Lei Wang, and Dawu Gu. Security Analysis of NIST Key Derivation Using Pseudorandom Functions. Cryptology ePrint Archive, Paper 2025/815, 2025. *(Cited on p. 1.)*