Simulation-based Security Notion of Correlation Robust Hashing with Applications to MPC *

Hongrui Cui † Chun Guo ‡ Xiaojie Guo § Xiao Wang ¶ Kang Yang ‖
Yu Yu **

February 14, 2026

Abstract

This work studies the security and constructions of correlation robust (CR) hash functions in secure multi-party computation (MPC). Existing definitions of CR hashing are all game-based (i.e., no simulator to achieve programmability or extractability), but MPC protocols are proven secure in the simulation-based models including both stand-alone and universal composability models. We found that for some MPC protocols, e.g., TinyOT-like authenticated-triple generation protocols and correlated oblivious transfer (COT) extension protocols, such a mismatch could lead to a gap in security proofs, even for the semi-honest adversary and stand-alone model.

To bridge the gap, we introduce a simulation-based security notion for CR hash functions to allow secure composition. Instead of building from scratch, we introduce such a simulator to a wide class of existing ideal-cipher-based CR hashing constructions, and derive the security bound from their original game-based CR security. This enables us to obtain an efficient CR hashing construction making just one call to a blockcipher, and is much more efficient than the construction from a random oracle used in previous TinyOT-like protocols. We showcase the utility of the new CR notion in easing security proofs and mitigating the risk of errors on two classes of protocols: (1) authenticated-triple generation protocols in the TinyOT family with a countermeasure; (2) COT extension protocols with bootstrapped iterations.

1 Introduction

Correlation robust (CR) hash functions [IKNP03] and their extensions [CKKZ12, GKWY20, GKW+20, RR21] are important cryptographic tools for designing modern cryptographic protocols. They are useful in extracting the best efficiency in garbled circuits [BHKR13, ZRE15, GKWY20, GKW+20, RR21], oblivious transfer (OT) extension [IKNP03, ALSZ13, KOS15, BCG+19a, YWL+20, GKWY20, CT21, Roy22, GYW+23, GWYY25], and distributed point function [ZGY+24], which are fundamental primitives of multi-party computation, zero-knowledge proofs and many other protocols.

*For convenience, a table of contents is appended at the end of this submission.

†Shanghai Jiao Tong University. Email: rickfreeman@sjtu.edu.cn

‡School of Cyber Science and Technology, Shandong University, Key Laboratory of Cryptologic Technology and Information Security of Ministry of Education, Shandong University, Shandong Research Institute of Industrial Technology. Email: chun.guo.sc@gmail.com

§Shanghai Qi Zhi Institute, Shanghai Jiao Tong University. Email: guoxiaojie@sqz.ac.cn

¶Northwestern University. Email: wangxiao1254@gmail.com

‖State Key Laboratory of Cryptology. Email: yangk@sklc.org

**Shanghai Jiao Tong University, Shanghai Qi Zhi Institute. Email: yuyu@yuyu.hk

Roughly speaking, a hash function $\mathsf{H} : \{0,1\}^\lambda \to \{0,1\}^\lambda$ is correlation robust if the keyed function $F_\Delta(x) = \mathsf{H}(\Delta \oplus x)$ is a secure PRF. By incorporating the circular property and/or adding tweaks into inputs, other related notions, e.g., tweakable circular correlation robustness (TCCR), were also studied [Roy22, GKWY20]. CR hash functions are popularly used in the context of high-performance protocols due to the fact that they can often be constructed with one blockcipher call (e.g., hardware accelerated AES, which could be 50× faster than hardware accelerated SHA256 [GKWY20]). Therefore, correlation robustness is an elegant balance between security and efficiency.

Although widely used in practical cryptographic implementations, we find that there are inherent limitations in existing CR definitions. All previous definitions for CR are game-based, meaning that they do not provide a simulator to achieve programmability or extractability, two crucial properties for simulation-based security of MPC protocols. On the other hand, MPC protocols are proven secure in simulation-based models, including the stand-alone model [Can00] and the universal composability (UC) model [Can01]. Such a mismatch leads to a gap when proving the security of some MPC protocols that use CR hash function, even for semi-honest adversaries and the stand-alone model. As a result, many protocols have to use an RO for provable security, although a CR hash could have been sufficient, due to the inability to simulate; others use CR hash functions but the proof has a gap, making provable security of the MPC protocols unclear. Such MPC protocols include, for example,

- Correlated oblivious transfer (COT) extension with bootstrapped iterations such as [BCG+19a, YWL+20, Roy22, GYW+23], where a large number of fresh COTs are extended from a small number of base COTs sharing the same global key Δ. The sender with Δ obtains M_0 and the receiver obtains (b, M_b) such that $M_b = M_0 \oplus b \cdot \Delta$ for bit b. The extension procedure involves hashing the base COTs, either to convert them into regular oblivious transfer correlations or to use them as one-time pad masks.

- TinyOT-like authenticated AND-triple generation [NNOB12, WRK17a, WRK17b, HSS17, HSS20, ZCSH18, YWZ20], where Δ is one party's key of information-theoretic message authentication codes (IT-MACs) to authenticate the other party's value x so that one party holds a random K_x and the other party holds $M_x = K_x + x \cdot \Delta$. The parties generate shares of bits x, y, z such that $z = x \cdot y$ and each share value is IT-MAC-authenticated. Here, the Δ-related K_x values are hashed to compute the authenticated multiplication between two shared bit values.

- The authenticated multiplication triple generation in MASCOT [KOS16]. Here the triples are like AND-triples but with two differences: they are over an arbitrary finite field $\mathbb{F}$ and the values are SPDZ-authenticated [DPSZ12]. The MASCOT protocol builds on the random oblivious transfer functionality, which can be obtained by hashing existing COT correlations.

- Authenticated OT protocols [NNOB12, BLN+21], where the OT sender has (m_0, m_1) and the OT receiver has a choice bit b and m_b. Moreover, the OT sender has an IT-MAC key Δ_S to authenticate b and m_b, while the OT receiver uses its IT-MAC key Δ_R to authenticate m_0 and m_1. The parties first generate the authenticated choice bit b and then hash the authentication tags to securely select m_b from (m_0, m_1).

- Preprocessing garbled circuits [HSS17, HSS20]. With the free-XOR optimization [KS08], each party will output its free-XOR key Δ_i in the preprocessing phase. Generating the garbled circuit with row-reduction techniques [ZRE15, RR21] requires hashing Δ_i-related wire labels.

The limitation of existing CR notions is inherent. As mentioned before, CR notion is defined based on the pseudorandomness of the corresponding PRF. PRF security in turn requires the key, in this case the correlation Δ, to be private from the distinguisher. However, in many MPC protocols, this key is in the output and hence available to the distinguisher/environment, invalidating the PRF (thus CR) security. Below, we dive into more details using an example. A typical CR-related message is in the form of $U = \mathsf{H}(M) \oplus \mathsf{H}(M \oplus \Delta) \oplus X$ (generated by a sender) for a CR hash function H, two strings $M, X \in \{0,1\}^\lambda$, where the sender holds two secret values Δ, X and the receiver knows M. In proofs of security for these MPC protocols under the simulation-based model, a simulator $\mathcal{S}$ simulates a CR-related message U by sampling it uniformly at random. At first glance, the simulation is computationally indistinguishable from the real message due to the CR property. However, this claim is not true when Δ is output to a party. In particular, the party's output Δ would be obtained at the end of protocol execution by a distinguisher (also referred to as an environment in the UC model) between the ideal-world execution and the real-world execution. Then, the distinguisher has access to H, and can use Δ to trivially distinguish the simulated message U from a real message, which fails the proof. Some prior works solve this problem by modeling H as a programmable random oracle (RO), but its instantiation has a much worse throughput than instantiations of an ideal cipher; e.g., SHA256 is up to 50× slower than AES128 when hashing 128-bit strings.

An endeavor to overcome the limitation of existing CR notions is to change the ideal functionality so that Δ does not appear in the output. We argue that this approach is not favorable as the these ideal functionalities typically serve as basis for higher-level protocols. Therefore, changing the output behavior of these ideal functionalities would cause further complication downstream. An ideal solution is to argue security using the efficient blockcipher-based CR hash constructions while keeping Δ in the ideal functionality output.

1.1 Our Contributions

Considering the simulation requirements of MPC protocols that existing game-based CR definitions fail to deliver, we formalize simulation-based CR notions for the first time. In particular, we develop a family of new notions of correlation robustness, which are built upon an ideal model and involve an implicit programming in their security experiments. With a particular interest in instantiating the ideal model with the ideal cipher model (ICM), we also describe concretely efficient blockcipher-based instantiations for these notions. Meanwhile, the required programming can be captured by the phase separation in the experiment of our CR notions. Here, the tedious probability analysis of a failed programming is already accomplished by the theorems of our CR notions, making the proofs more modular. Our new CR notions can be directly applied in the security proofs of several kinds of MPC protocols, which enables one to achieve simpler proofs and better efficiency. We apply them to fix the provable security of two state-of-the-art protocols to demonstrate their effectiveness. Specifically, we have the following results.

1.1.1 New CR Notions for MPC.

Motivated by the above, we seek simulation-based CR notions that simultaneously fulfill: (i) enabling modular constructions of simulators and arguments in MPC security proofs, and (ii) being weaker than indifferentiability, so that it can be achieved by efficient constructions using just a single blockcipher call.

Since simulation requires the usage of programmable ideal primitives, we work in the ideal model in which a public ideal primitive **Prim** is accessible by all parties, including the adversary. The

hash construction $\mathsf{H}^{\mathbf{Prim}}$ is built upon **Prim** as well. This increases generality: by setting **Prim** to an ideal cipher or a random oracle, concrete definitions in the ICM or the ROM are derived.

In this ideal model, the classical CR requires that *when the global key* Δ *is secret*, the real-world oracles $(\mathsf{K}_{\Delta}^{\mathsf{H}^{\mathbf{Prim}}}(\cdot) = \mathsf{H}^{\mathbf{Prim}}(\Delta \oplus \cdot), \mathbf{Prim})$ are indistinguishable [1] from the ideal-world oracles $(\mathbf{F}, \mathbf{Prim})$, where $\mathbf{F}$ is a random function that has the same output length as $\mathsf{K}_{\Delta}^{\mathsf{H}^{\mathbf{Prim}}}$.

In several types of MPC protocols as mentioned in the introduction, the key Δ is revealed to the distinguisher at the end of protocol execution, due to which no hash functions can fulfill the classical CR notions. To exactly address this revealing, in our ideal world, the primitive **Prim** is replaced by a *simulator* $\mathcal{S}^{\mathsf{GlobalK}_{\Delta}}$ that has access to a *key checking oracle* $\mathsf{GlobalK}_{\Delta}$: every time $\mathcal{S}^{\mathsf{GlobalK}_{\Delta}}$ comes up with a guess Δ^* of the true key Δ, it can query $\mathsf{GlobalK}_{\Delta}(\Delta^*)$ to check if $\Delta^* = \Delta$. $\mathcal{S}^{\mathsf{GlobalK}_{\Delta}}$ mimics **Prim** by querying $\mathsf{GlobalK}_{\Delta}$, such that:

(i) In the first phase, Δ is secret, and the real-world oracles $(\mathsf{K}_{\Delta}^{\mathsf{H}^{\mathbf{Prim}}}, \mathbf{Prim})$ are indistinguishable from the ideal-world oracles $(\mathbf{F}, \mathcal{S}^{\mathsf{GlobalK}_{\Delta}})$;

(ii) In the second phase, Δ is revealed to the distinguisher, and the real-world oracle **Prim** remains indistinguishable from the ideal-world oracle $\mathcal{S}^{\mathsf{GlobalK}_{\Delta}}$.

In this way, a construction $\mathsf{H}^{\mathbf{Prim}}$ satisfying our new definitions would be equipped with a simulator, which enables straightforward constructions of simulators in the MPC security proofs for higher-level protocols. We stress that this notion is *weaker* than indifferentiability, since the key Δ remains secret in the first phase. We refer to Sect. 2.2 for rigorous elaborations.

1.1.2 Provably Secure Constructions.

We focus on the strongest CR variant mTCCRL [GWYY25], which includes all added features: multi-user security (introduced in [GKW+20]), tweakable (introduced in [GKWY20]), circularity (introduced in [CKKZ12]), and key leakage (introduced in [GWYY25]). We formalize its simulation-based variant S-mTCCRL. We provide two S-mTCCRL secure constructions of $\mathsf{H}^{\mathbf{Prim}}$, one in the ICM and the other in the ROM.

S-mTCCRL in the ICM. When **Prim** $= \mathbf{E}$ is an ideal cipher, using a linear orthomorphism σ[2], we prove that any hash construction with classical mTCCRL security already satisfies the new S-mTCCRL definition. To give such a general proof, we consider a unified model $\mathsf{GH}^{\mathbf{E}}$ of hash functions built upon a single call to $\mathbf{E}$, which is both general enough to cover all existing 1-call hash functions [GKWY20, GKW+20, CT21] and rich enough to support designing a simulator. We then prove the S-mTCCRL security of $\mathsf{GH}^{\mathbf{E}}$ based on its mTCCRL security, with a non-influential reduction loss. This indicates that for *a wide class* of constructions using a *single ideal-cipher call* (i.e., instances of $\mathsf{GH}^{\mathbf{E}}$), mTCCRL security implies S-mTCCRL security with comparable security bounds. From the relationship between mTCCRL and S-mTCCRL, we obtain that the state-of-the-art hash construction [GKW+20] $\widehat{\mathsf{MMO}}^{\mathbf{E}}(x, i) := \mathbf{E}(i, \sigma(x)) \oplus \sigma(x)$, which was proven in [GWYY25] to be classical mTCCRL secure, has already satisfied the S-mTCCRL security. Besides $\widehat{\mathsf{MMO}}^{\mathbf{E}}$, this relation may enable utilizing more existing results in the ICM. We refer the reader to Sect. 2.3 for more details.

[1] In the sense of indistinguishability of oracles or random systems [Mau02].

[2] σ is *linear* if $\sigma(x \oplus y) = \sigma(x) \oplus \sigma(y)$ for all $x, y \in \{0,1\}^n$; σ is an *orthomorphism* [CLL+18] if both σ and $\sigma'(x) := \sigma(x) \oplus x$ are permutations. One can define $\sigma(x_L \| x_R) := x_R \oplus x_L \| x_L$, where x_L and x_R are the left and right halves [CLL+18, GKW+20].

S-mTCCRL in the ROM. When $\mathbf{Prim} = \mathbf{RO}$ is a random oracle, we prove that $\mathsf{H}^{\mathbf{Prim}}(w, i) := \mathbf{RO}(w\|i)$ is S-mTCCRL secure. While this result might not seem surprising, it serves as a sanity check of the S-mTCCRL definition. In addition, its simulator abstracts the random oracle simulation procedure and probability analysis previously appeared in the proofs of MPC protocols [NNOB12, WRK17a, WRK17b, HSS17, HSS20, ZCSH18, YWZ20] into the simulator of the hash function, thereby easing security proofs and mitigating the risk of errors.

1.1.3 Applications of S-mTCCRL.

The new S-mTCCRL notion can be applied to fix issues regarding simulation in all works that we discussed before and enable many protocols to enjoy the efficiency of CR hash functions with provable security. Furthermore, we can directly use our S-mTCCRL notion in the security reduction, without diving into the low-level random oracle/ideal cipher model and manually dealing with its programming in proofs. In other words, our S-mTCCRL notion can save the work in security proofs and make them more modular and readable.

As a demonstration, we utilize this notion and prove the UC security of the state-of-the-art authenticated AND triple protocol of Yang et al. [YWZ20]. As mentioned above, we identified a common flaw in the security proof of protocols in the TinyOT family. We show how to fix it by tweaking the ideal functionality and adding a randomization step in the protocol. Additionally, using our new notion, we prove the UC security of the silent COT extension protocol in [YWL+20] (which works in bootstrapped iterations in its original form). The second proof does not require changing the original protocol or functionality. We believe that our fix and the techniques used in the new proof may inspire future work in MPC.

1.1.4 Organizations.

We provide a more detailed overview of our results in Sect. 2 and the necessary preliminaries in Sect. 3. Then, in Sect. 4, we provide our new security definitions; in Sect. 5, we provide an efficient, provably secure hash construction built upon a single blockcipher call. As an application, we prove the UC security of the authenticated AND protocol of [YWZ20] using our new security definitions in Sect. 6. We defer the additional definitions, a random oracle-based provably secure hash construction, detailed proofs, additional constructions in the random permutation model, and an additional application in COT extension [YWL+20] to the Appendices.

2 Technical Overview

This section provides an overview of the background, motivation, and main technical contributions of this work.

2.1 Limitations of Game-based Correlation Robustness

The Notions of Correlation Robustness. Hash functions with different kinds of correlation robustness have been widely used since the efficient OT extension of Ishai et al. [IKNP03]. Existing notions state the pseudorandomness property of hash functions when the inputs are padded with a global key Δ. We summarize them as follows:

- *Correlation robustness* (CR) [IKNP03]. A hash function $\mathsf{H} : \mathcal{W} \to \mathcal{W}$ is correlation robust, if the key-whitened function $\mathsf{K}^{\mathsf{H}}_{\Delta}(x) := \mathsf{H}(x \oplus \Delta)$ keyed by Δ is indistinguishable from a random function $\mathbf{F} : \mathcal{W} \to \mathcal{W}$.

- *Circular correlation robustness* (CCR) [CKKZ12] combines circularity with correlation robustness. A hash function $\mathsf{H} : \mathcal{W} \times \{0,1\} \to \mathcal{W}$ is circular correlation robust, if the circular key-whitened function $\mathsf{CK}^{\mathsf{H}}_{\Delta}(x,b) := \mathsf{H}(x \oplus \Delta) \oplus b \cdot \Delta$ is indistinguishable from a random function $\mathbf{F} : \mathcal{W} \times \{0,1\} \to \mathcal{W}$ as long as the distinguisher cannot query both $(x,0)$ and $(x,1)$. We implicitly impose this restriction for the following notions.

- *Tweakable circular correlation robustness* (TCCR) [GKWY20, CT21]. A hash function $\mathsf{H} : \mathcal{W} \times \mathcal{I} \to \mathcal{W}$ satisfies TCCR, if the tweakable circular key-whitened function $\mathsf{TCK}^{\mathsf{H}}_{\Delta}(w,i,b) := \mathsf{H}(w \oplus \Delta, i) \oplus b \cdot \Delta$ is indistinguishable from a random function $\mathbf{F} : \mathcal{W} \times \mathcal{I} \times \{0,1\} \to \mathcal{W}$.

- *Multi-user TCCR* (mTCCR) [GKW+20] extends TCCR notion to the multi-user setting [BBM00, BBT16, BT16, HT16, GKW+20], where there are u global keys $\boldsymbol{\Delta} = (\Delta_1, ..., \Delta_u)$. A hash function $\mathsf{H} : \mathcal{W} \times \mathcal{I} \to \mathcal{W}$ meets mTCCR, if the multi-user tweakable circular key-whitened function $\mathsf{mTCK}^{\mathsf{H}}_{\boldsymbol{\Delta}}(\mathsf{id}, w, i, b) := \mathsf{H}(w \oplus \Delta_{\mathsf{id}}, i) \oplus b \cdot \Delta_{\mathsf{id}}$ is indistinguishable from a random function $\mathbf{F} : \{1, ..., u\} \times \mathcal{W} \times \mathcal{I} \times \{0,1\} \to \mathcal{W}$.

- *Multi-user TCCR with key leakage* (mTCCRL) [Roy22, GWYY25]. In several MPC protocols [KOS15, OOS17, Sch18, BCG+19a, YWL+20, WYKW21, Roy22, Dia22], the distinguisher is allowed to make selective-failure attacks on the global key added to hash inputs. More specifically, the distinguisher can choose an identifier id and a predicate P to check whether $P(\Delta_{\mathsf{id}}) = 0$ or not from a key leaking oracle $\mathsf{mL}_{\boldsymbol{\Delta}}(\mathsf{id}, P)$. The mTCCRL security ensures that $\mathsf{mTCK}^{\mathsf{H}}_{\boldsymbol{\Delta}}(\mathsf{id}, w, i, b)$ and a random function $\mathbf{F} : \{1, ..., u\} \times \mathcal{W} \times \mathcal{I} \times \{0,1\} \to \mathcal{W}$ are indistinguishable when the distinguisher is given the key leaking oracle $\mathsf{mL}_{\boldsymbol{\Delta}}$.

Insufficiency of Existing CR Notions. The above variants of correlation robustness offer more efficient and conservative alternatives to random oracles and are used in many MPC applications, such as the reduction from standard OT correlations to COT correlations and circuit garbling. Nevertheless, for protocols that require CR-style indistinguishability but include the global key as a part of their outputs, their security cannot be based on the existing CR notions.

The reason for this insufficiency is as follows. In these MPC protocols, the simulation-based proof (whether in the standalone or the UC model) requires that given the outputs of honest parties, the views of the corrupted parties controlled by an adversary in the real world are indistinguishable from the simulated views in the ideal world. Crucially, in the real world some protocol messages are one-time pads (OTPs) with the keys output by some key-whitened function while in the ideal world these messages are simulated by random values. While existing CR notions could guarantee the indistinguishability between the two worlds when the global key is hidden from the distinguisher, in some important MPC applications including authenticated triple protocols in the TinyOT family and COT extension protocols, the distinguisher gets the global key as part of the honest party's output. Now the distinguisher can use the known global keys of honest parties to run the key-whitened function to compute the OTP keys. These keys can be distinguished from random values, making the two worlds trivially distinguishable.

To be provably secure, the protocols can adopt the programming-based approach, which is used in the TinyOT-style protocols [NNOB12, HSS17, YWZ20]. They model the hash function H (implemented by SHA-256 or SHA-3) as a random oracle and, after the global key Δ of each honest party being revealed to the distinguisher, handle the simulation by programming the random oracle to maintain the consistency required by the key-whitened function *as soon as the distinguisher uses* Δ *to query the random oracle to run this function.* The Half-Tree COT protocol [GYW+23] adopts a similar approach, but for concrete efficiency, it constructs the hash function from a blockcipher, which can be captured in the random permutation model (RPM) or the ideal cipher model (ICM).

The simulation in the Half-Tree protocol will program the RPM/ICM after the distinguisher learns Δ so that the consistency holds for the RPM/ICM-based hash construction.

A major drawback of the approach is that the proof should analyze the programming explicitly by bounding the probability of a failed programming. Such an analysis is often non-trivial and makes the proof quite complicated, especially in the case of the concretely efficient RPM/ICM-based hash where the permutation allows both forward and backward queries. Some works (detailed in Sect. 2.4) even claim their security without considering the programming and have flawed proofs. Directly fixing these proofs is unsatisfactory: the fix to one proof cannot be reused by other proofs since the consistency to be programmed is protocol-specific, although the underlying probability analysis does use some common techniques.

In the above (mis)use case of CR hash in MPC, due to the aforementioned distinguishing strategy, we postulate that a programmable ideal primitive (e.g., random oracle or ideal cipher) is inevitable. Nevertheless, a black-box separation result for the standard model is beyond the scope of this paper.

2.2 S-mTCCRL: Simulatable mTCCRL

We propose simulatable mTCCRL (abbreviated as S-mTCCRL), a new variant of correlation robustness to modularize the programming-based proofs for the pseudorandomness of key-whitened function even if the global key is revealed to the distinguisher in protocol output. The S-mTCCRL notion states the indistinguishability between the real world (running the mTCCRL key-whitened function) and the ideal world (with a simulator to simulate the interfaces of some ideal primitive). The experiment of S-mTCCRL contains the following two phases:

- In phase 1, the distinguisher $\mathcal{D}$ has access to three oracles $(O_1, O_2, \mathsf{mL}_{\boldsymbol{\Delta}})$. In the real world, the first two oracles are an ideal primitive **Prim** (random oracle or ideal cipher) and the tweakable circular key-whitened function $\mathsf{mTCK}^{\mathsf{H}}_{\boldsymbol{\Delta}}(\mathsf{id}, w, i, b) := \mathsf{H}(w \oplus \Delta_{\mathsf{id}}, i) \oplus b \cdot \Delta_{\mathsf{id}}$ constructed from the calls to the ideal primitive **Prim**, and the key leaking oracle $\mathsf{mL}_{\boldsymbol{\Delta}}$ is defined on the u real keys $\boldsymbol{\Delta} = (\Delta_1, ..., \Delta_u)$. In the ideal world, the first two oracles are the interfaces provided by a simulator $\mathcal{S}$ and a random function $\mathbf{F}$ respectively, and $\mathsf{mL}_{\boldsymbol{\Delta}}$ is defined on a random $\boldsymbol{\Delta} = (\Delta_1, ..., \Delta_u)$.

- At the end of phase 1, the key vector $\boldsymbol{\Delta}$ (underlying $\mathsf{mTCK}^{\mathsf{H}}_{\boldsymbol{\Delta}}$ and $\mathsf{mL}_{\boldsymbol{\Delta}}$ in the real world, or $\mathsf{mL}_{\boldsymbol{\Delta}}$ in the ideal world) is given to the distinguisher $\mathcal{D}$. Meanwhile, in the ideal world, the query-answer pairs of the oracle O_2 in phase 1 are given to the simulator for the subsequent simulation.

- In phase 2, the distinguisher $\mathcal{D}$ can only query the oracle O_1, which is either the real **Prim** (in the real world) or the simulated one (in the ideal world). Note that querying $\mathsf{mL}_{\boldsymbol{\Delta}}$ in this phase is meaningless for $\mathcal{D}$ since it knows $\boldsymbol{\Delta}$. $\mathcal{D}$ is finally asked to decide which world it is in.

Since $\mathcal{D}$ has "known" the keys $\boldsymbol{\Delta} = (\Delta_1, ..., \Delta_u)$ in phase 2, the simulator has to simulate **Prim** that are consistent with the random $\boldsymbol{\Delta}$ in the ideal world. Without any information about $\boldsymbol{\Delta}$, this is clearly infeasible. Therefore, we give the simulator access to a *multi-key checking oracle* $\mathsf{GlobalK}_{\boldsymbol{\Delta}}$, which accepts an input (id, Δ') and outputs if $\Delta' = \Delta_{\mathsf{id}}$ or not. For the MPC protocols using this security notion, this simulator can serve as a subroutine of the protocol simulator to simulate the ideal primitive **Prim**. Now, the protocol simulator only needs to prepare a list of the query-answer pairs of the oracle O_2 in phase 1 and implement $\mathsf{GlobalK}_{\boldsymbol{\Delta}}$ for the S-mTCCRL simulator (e.g., relying on the global-key query allowed by the target functionality, see Sect. 6.2 for example), instead of diving into the programming details processed by the S-mTCCRL simulator.

Table 1: Our new security notions for simulatable CR and their "base" CR notions (in the multi-user setting and with the selective-failure leakage of global keys).

"Base" CR notion	Our security notions	Tweak	Circularity	Construction
mTCCRL [GWYY25]	S-mTCCRL (Sect. 4)	✓	✓	$\widehat{\mathsf{MMO}}^{\mathbf{E}}$ (Sect. 5.3)
mTCRL [GKW+20]	S-mTCRL (Sect. A.2)	✓	✗	$\widehat{\mathsf{MMO}}^{\mathbf{E}}$ (Sect. 5.3)
mCCRL [GKW+20]	S-mCCRL (Sect. A.2)	✗	✓	$\widehat{\mathsf{MMO}}^{\pi}$ (Sect. E.1)
mCRL [GKW+20]	S-mCRL (Sect. A.2)	✗	✗	$\widehat{\mathsf{MMO}}^{\pi}$ (Sect. E.1)

This programming is implicit in phase 2 since, as noted, the simulator will simulate **Prim** to ensure the consistency with the random $\mathbf{\Delta}$.

Weaker Variants. In the main text, we mainly focus on S-mTCCRL, which is the strongest variant including all four features: multi-user security (m) [GKW+20], tweak (T) [GKWY20], circularity (C) [CKKZ12], and selective-failure leakage on global keys (L) [GWYY25]. By removing the feature of circularity or tweak, we can obtain weaker variants of S-mTCCRL for different grades of security requirements. Furthermore, we always consider the selective-failure leakage since the variants without this leakage can be seen as special cases of the considered variants. We refer to Table 1 for a summary and Appendix A for details.

To ease understanding, none of the above four features are present in the notion sketched in Sect. 1.1. Therefore, that notion is the "most basic" S-CR, i.e., the simulator-based variant of the plain correlation robust notion CR. For clarity, we include rigorous definitions of CR and S-CR in Appendix A.1.

Usefulness of S-mTCCRL. Using our new notion S-mTCCRL as an assumption substantially eases security proofs for the protocols. Concretely, as shown in Sect. 6, the reduction of the protocols only needs to keep track of the construction queries during protocol execution (phase 1), while the complicated and error-prone simulation of the ideal primitive (which is typically a random oracle in existing protocols) can be offloaded in a black-box manner to the S-mTCCRL simulator, and the relevant reduction loss is given by the S-mTCCRL bound.

Comparison with Similar Notions. Our simulator-based definition may be regarded as a weaker variant of indifferentiability tailored to a certain class of protocols. The indifferentiability game of $\mathsf{H}^{\mathbf{Prim}}$ requires the real-world oracles $\mathsf{H}^{\mathbf{Prim}}$ and **Prim** to be indistinguishable from the ideal-world oracles $\mathbf{F}$ and a simulator $\mathcal{S}^{\mathbf{F}}$. Differences include: (i) usually, there is no secret key; (ii) the indifferentiability simulator $\mathcal{S}^{\mathbf{F}}$ queries $\mathbf{F}$ to simulate consistent answers; (iii) the construction oracle $\mathsf{H}^{\mathbf{Prim}}/\mathbf{F}$ is always available. Our notion S-mTCCRL is weaker, but it yields constructions with *much better efficiency and security bounds* (to be discussed later in Sect. 2.3.1).

The "two phases with key revealing" nature of the S-mTCCRL experiment may remind the reader of the UCE framework in the ideal model [BHK13, BHK14]. However, in the UCE experiments the faithful public ideal function is always available, and there is *no simulator*. This difference stems from different motivations: UCE aims to provide a well-defined assumption for random oracle-like functions, while S-mTCCRL aims to guarantee compositions in simulation-based MPC proofs.

Unlike UCE [BHK14], our current treatment does not fit into multi-stage settings [RSS11]: the simulator used in our proofs has to keep all internal states across the two phases. It remains an open question whether S-mTCCRL could be a choice for multi-stage settings.

2.3 S-mTCCRL Secure Hash Constructions

We first present a blockcipher-based construction, which is our main solution. We then show random oracle-based solutions for protocols using the SHA standard.

2.3.1 Security of Blockcipher-based Constructions via a Unified Approach.

We prove that S-mTCCRL is achievable by the state-of-the-art hash functions built upon a single blockcipher call [GKW+20, GKWY20, GWYY25]. Instead of proving from scratch, we establish a relation between mTCCRL and S-mTCCRL in the ICM, and then derive the results by gathering this relation and existing mTCCRL results [GWYY25].

Relation between mTCCRL and S-mTCCRL in the ICM. It is natural to seek the relation between our new notion S-mTCCRL and the existing mTCCRL of [GWYY25]. Due to the requirement of programmable primitives, our new notion S-mTCCRL seems incomparable with mTCCRL [GWYY25]. Still, one would ask if mTCCRL constructions *in the ideal model* are S-mTCCRL as well.

We give a positive (although limited) answer in the ideal cipher model. Concretely, let $\mathbf{E} : \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ be an n-bit ideal cipher with κ-bit keys. We consider a general model $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}} : \{0,1\}^n \times \mathcal{I} \to \{0,1\}^n$ of hash constructions that makes a single call to $\mathbf{E}$. The model is defined as

$$\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}(x,i) := f_{\mathsf{out}}\Big(x, i, \mathbf{E}(g(i), f_{\mathsf{in}}(x,i))\Big),$$

where $f_{\mathsf{in}} : \{0,1\}^n \times \mathcal{I} \to \{0,1\}^n$, $g : \mathcal{I} \to \{0,1\}^\kappa$ and $f_{\mathsf{out}} : \{0,1\}^n \times \mathcal{I} \times \{0,1\}^n \to \{0,1\}^n$ can be any functions. This model is adapted from [BRSS10] and is both *general* enough to unify many existing 1-call hash constructions [GKWY20, GKW+20, CT21] and *syntactically rich* enough to enable the design of S-mTCCRL simulators. We refer to Sect. 5.1 for elaborations.

We prove that for any instantiations of f_{in}, g and f_{out} subject to a few injectivity restrictions, if $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$ is mTCCRL secure, then it is also S-mTCCRL secure with comparable bounds. In other words, for *a wide class* of constructions using a *single ideal cipher-call*, mTCCRL implies S-mTCCRL security.

Our proof proceeds with first designing a simulator and then reasoning about the indistinguishability of the simulated responses. As a technical novelty, in the latter step we use the assumed mTCCRL security of $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$ to bound the probability of certain bad events in *real-world experiments*. This deviates from most existing simulation security proofs [CHK+16, ABD+13] that analyzed bad events in *ideal world experiments*. We refer to Sect. D.2 and Sect. D.3 for details.

Concrete results in the ICM. Since the construction $\widehat{\mathsf{MMO}}^{\mathbf{E}}(x,i) := \mathbf{E}(i, \sigma(x)) \oplus \sigma(x)$ is an instance of $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$, the above result on $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$ enables deriving concrete S-mTCCRL bounds for $\widehat{\mathsf{MMO}}^{\mathbf{E}}$ from its mTCCRL bounds [GWYY25]. The derived bounds are comparable with [GWYY25]. In detail, assume that: (i) the u user keys $\Delta_1, ..., \Delta_u$ are independently and uniformly sampled from u oracle-independent sets $\mathcal{R}_1, ..., \mathcal{R}_u$ respectively, such that for some parameter C, $\min\{|\mathcal{R}_1|, ..., |\mathcal{R}_u|\} \geq C$, (ii) the distinguisher asks key leaking queries to at most M distinct users, and for each of them, the distinguisher asks at most $q_{L,max}$ leaking queries, and (iii) the distinguisher asks at most μ construction queries per tweak (across multiple users), then security of $\widehat{\mathsf{MMO}}^{\mathbf{E}}$ is ensured up to roughly $C/(\mu M q_{L,max})$ ideal cipher queries and $C/(\mu M q_{L,max})$

construction queries. Note that we follow [GWYY25] and allow the u keys sampled from different sets to widen applications.

Since $|\Delta_1|, ..., |\Delta_u| \leq |x| = n$, the optimal choice is to have $C \approx 2^n$. One could limit μ to $O(1)$ by carefully using distinct tweaks in the protocols (see applications in Sect. 6), and the $\log_2 \mu$ bits security loss is small. The parameter $q_{L,max}$ could be limited to $O(n)$ as well. By these, $\widehat{\mathsf{MMO}}^{\mathbf{E}}$ has nearly $n - \log_2 n - \log_2 M$ bits of security. On the other hand, M may be as large as u, yielding a notable loss. Unfortunately, the bound is tight in both S-mTCCRL and mTCCRL settings: we refer to [GWYY25, Appendix B] for a matching attack. By this, to have sufficient security, one needs to carefully choose the various parameters.

Prior to this work, the only known approach to securely instantiating protocols with simulation-based security in the ICM is to use constructions indifferentiable from random oracles [CDMP05], such as the PGV variants [GBK16], the keyed XORP construction $\mathsf{XORP}^*(x, i) := \mathbf{E}(i\|0, x) \oplus \mathbf{E}(i\|1, x)$ [BN18, GBJ+23], or letting $H^{\mathbf{E}}(x, i)$ be the $(\lambda - O(\lambda))$-bit truncation of $\mathbf{E}(i, 0^{O(\lambda)}\|x)$ [CLL19] (where we view λ as a security parameter). However, PGV variants and keyed XORP consume 2 blockcipher calls and are less efficient, while truncation results in impractically poor security bounds if applied to AES with a block length of 128 bits. Our results thus provide the first "random oracle alternative" with both optimal efficiency (1 ideal cipher-call) and practical security bounds for simulation-based secure protocols.

Constructions for weaker notions. Our S-mTCCRL result implies that $\widehat{\mathsf{MMO}}^{\mathbf{E}}$ satisfies the *non-circular variant* of S-mTCCRL as well. For the non-circular notions, the linear orthomorphism σ can be securely removed. However, the performance benefit is negligible.

Finally, by fixing the tweak input, we also derive bounds for the permutation-based variant $\widehat{\mathsf{MMO}}^{\pi}(x) := \pi(\sigma(x)) \oplus \sigma(x)$ w.r.t. the non-tweakable variants S-mCCRL and S-mCRL (albeit with loose bounds). These provide secure constructions for different grades of requirements. We refer to Appendix E for details.

2.3.2 Security of Random Oracle-based Constructions.

We prove that the construction $\mathsf{ROH}^{\mathbf{RO}}(w, i) := \mathbf{RO}(f_{\mathsf{in}}(w, i))$ is S-mTCCRL secure, where $f_{\mathsf{in}} : \mathcal{W} \times \mathcal{I} \to \mathcal{X}$ can be any efficiently invertible bijection. We refer to Appendix B for details. Therefore, S-mTCCRL is also achievable by the folklore random oracle-based constructions such as $\mathbf{RO}(w\|i)$. More importantly, it enables a modular approach to proving security for MPC protocols employing such folklore random oracle-based constructions.

2.4 Applications of S-mTCCRL in MPC

We demonstrate the power of S-mTCCRL by using it in MPC protocols to improve performance and simplify proofs. Our first application is the malicious protocol for (multi-party) leaky authenticated AND (LaAND) triples, which is an intermediate step towards authenticated AND triples and serves as an important building block for malicious secret-sharing-based MPC on Boolean circuits [NNOB12, HSS17] and authenticated garbling [WRK17a, WRK17b, KRRW18, YWZ20]. We focus on LaAND since only this step requires CR-style hashing and the leakage can be removed by a standard cut-and-bucketing step [NNOB12, ABF+17]. In this application, we replace the random oracle in the state-of-the-art protocol [YWZ20] with an S-mTCCRL-secure hash function, leading to a more efficient instantiation and eliminating the non-black-box probability analysis of programming. Additionally, we identify some proof flaws in previous protocols and provide a

patched protocol, which relies on the ideal model pinned to the instantiation of an S-mTCCRL-secure hash to extract the randomness used by the simulation. This extraction is novel and may be of independent interest. Our second application is COT extension protocols with bootstrapped iterations [ALSZ13, YWL+20]. When a COT extension protocol works in bootstrapped iterations, COT triples generated by previous iterations serve as raw material for the current iteration. Therefore, the global key underlying the COT triples appears in the view of the adversary and our new S-mTCCRL notion can be applied. We present a new and corrected UC security proof of the malicious Single-Point COT (SPCOT) sub-protocol in the Ferret COT extension protocol [YWL+20]. As in the previous application, the focus on SPCOT is due to the scope of CR-style hashing. The detailed proof of this protocol is postponed to Appendix G. Below, we explain how to apply our S-mTCCRL notion to these applications.

Leaky Authenticated AND Triples. Generalized from the two-party TinyOT protocol [NNOB12], an n-party LaAND triple consists of three authenticated shares $(\langle\langle a\rangle\rangle, \langle\langle b\rangle\rangle, \langle\langle a\cdot b\rangle\rangle)$, where $a, b \xleftarrow{\$} \mathbb{F}_2$. The authenticity means that for each party P_i and its share $\langle\langle x\rangle\rangle_i = (x^i, \{\mathsf{M}_j[x^i], \mathsf{K}_i[x^j]\}_{j\neq i})$ for $x \in \{a, b, a\cdot b\}$, it holds a global key $\Delta_i \in \mathbb{F}_{2^\lambda}$ and maintains $\sum_{i\in[n]} x^i = x$ along with the following information-theoretic message authentication codes (IT-MACs):

$$\forall j \neq i : \mathsf{M}_i[x^j] = \mathsf{K}_i[x^j] \oplus x^j \cdot \Delta_i \,.$$

Meanwhile, the leakage indicates that the adversary can launch selective-failure attacks against the honest parties' shares $\{a^i\}_i$. This leakage can be removed through a local cut-and-bucketing step [NNOB12, ABF+17], effectively reducing the problem of generating authenticated AND triples to generating LaAND triples.

Many multi-party LaAND protocols merely use minicrypt assumptions and the random oracle to achieve malicious security. However, we have identified a significant caveat overlooked by these proofs, leading to flawed security claims:

- *Caveat: On the randomness of random-oracle-dependent messages.* In these protocols, an honest party P_i sends messages to corrupted parties that are masked by the random oracle outputs, which depend partially on its global key Δ_i and bit shares (typically, b^i) included in its protocol outputs. These messages are used to compute cross-terms $\{a^j \cdot b^i\}_j$ and are simulated by random values during the simulation. However, the arguments supporting this randomness are flawed.

  In [WRK17a, WRK17b, ZCSH18, YWZ20], the proofs argue that the distinguisher cannot distinguish these messages from random values due to the randomness of Δ_i and the polynomially many queries to the random oracle. However, as noted in Sect. 2.1, in the MPC security definition, the distinguisher can access the random oracle, Δ_i, and the output bit shares at the end of execution to check the consistency of these messages. Since this consistency holds only with negligible probability for the uniformly simulated messages, the simulation is trivially distinguishable from the real execution, thus failing the proofs.

  The proofs from [HSS17, HSS20] further address this consistency issue by adding global-key queries to the ideal functionality and programming the random oracle as in the original TinyOT. The intuition from TinyOT is that, to check consistency, the distinguisher must query the random oracle with Δ_i-dependent queries. From these queries, the simulator can extract every possible Δ_i' and use the global-key query of the ideal functionality to determine whether $\Delta_i' = \Delta_i$. If true, the simulator can use this Δ_i' to program the random oracle to artificially maintain consistency. However, this technique still fails for the constructions in [HSS17, HSS20] since their consistency equations also involve the output bit shares of

an honest party. These bit shares are hidden from the simulator, making it infeasible to determine the equation that needs to be programmed. We also note that a similar issue exists in the existing TinyOT protocols for leaky authenticated OT [NNOB12, BLN+21]. A quick fix for this family of issues is to use "augmented global-key queries": if the simulator can guess Δ_i, the functionality additionally reveals the outputs of honest parties to fulfill the programming.

With this caveat in mind, we improve the state-of-the-art minicrypt multi-party LaAND protocol [YWZ20] by replacing the random oracle with an S-mTCCRL hash function and enhancing it to achieve UC security. Due to the S-mTCCRL abstraction, our improved protocol benefits from a more modular security proof without the need to consider any programming detail.

Our improved LaAND protocol is as follows. Suppose that we can generate random authenticated shares. We generate random $(\langle\langle a\rangle\rangle, \langle\langle b\rangle\rangle, \langle\langle r\rangle\rangle)$ and correct $\langle\langle r\rangle\rangle$ to $\langle\langle a \cdot b\rangle\rangle$ via the XOR-homomorphism of authenticated shares and a difference $d = a \cdot b \oplus r$. For malicious security, it is necessary to ensure that the difference d is honestly computed. The n parties can guarantee this correctness by jointly computing their shares of $d \cdot \oplus_{i\in[n]}\Delta_i$ and using their global keys to perform a SPDZ-style check [DPSZ12]. The most important step is to compute the shares of $d \cdot \oplus_{i\in[n]}\Delta_i$. Note that the authenticated shares $\langle\langle b\rangle\rangle$ and $\langle\langle r\rangle\rangle$ already allow two distinct parties P_i and P_j to locally share $b^j \cdot \Delta_i$ and $r^j \cdot \Delta_i$. Hence, each party P_i can locally compute its share Φ^i of $b \cdot \oplus_{i\in[n]}\Delta_i$, as well as its share of $r \cdot \oplus_{i\in[n]}\Delta_i$. It remains for the two distinct parties P_i and P_j to share their cross-term $a^j \cdot \Phi^i$ to locally compute their shares of $a \cdot b \cdot \oplus_{i\in[n]}\Delta_i$.

In our protocol, the two parties compute this cross-term via the COT-based multiplication between a bit and a field element. Given an S-mTCCRL-secure hash function H (implemented in an ideal model), P_i sets $s^i := \mathsf{H}(\mathsf{K}_i[a^j])$[3] and

$$u^{i,j} := \mathsf{H}(\mathsf{K}_i[a^j]) \oplus \mathsf{H}(\mathsf{K}_i[a^j] \oplus \Delta_i) \oplus \Phi^i, \tag{1}$$

which is sent to P_j to compute $s^j := \mathsf{H}(\mathsf{M}_i[a^j]) \oplus a^j \cdot u^{i,j}$. One can check that $s^i \oplus s^j = a^j \cdot \Phi^i$, i.e., s^i and s^j are the shares of the cross-term. In contrast to the previous protocol [YWZ20] which requires H as a random oracle in Equation 1 and explicit programming as in the caveat, we reduce the pseudorandomness of $u^{i,j}$ to the S-mTCCRL security. Thus, the protocol simulator can invoke the two-phase S-mTCCRL simulator (see Sect. 2.2) as its subroutine to emulate the underlying ideal model. In essence, the phase switch of the S-mTCCRL simulator is an implicit programming. In other words, the complex probability analysis is done in the S-mTCCRL theorems, making the proof modular.

However, to switch the S-mTCCRL simulator into phase 2, the protocol simulator should provide it with a list of all Δ_i-circular query-response pairs of H in $\{u^{i,j}\}_j$, i.e., the entries to be programmed. If the underlying ideal model is the random oracle model (i.e., H is a random oracle), the simulation follows the idea recalled in the aforementioned caveat: the protocol simulator observes the distinguisher's queries to the random oracle, extracts the global key Δ_i and the output bits via "augmented global-key queries", and uses them to compute Φ^i and the required list. Nevertheless, for a hash function using a *generic* ideal primitive, especially the efficient construction in the ICM, we seek a more modularized approach to extract the global key and utilize the S-mTCCRL security.

Modularized Global Key Extraction. Firstly, we observe that the ideal primitive simulator $\mathcal{S}_{\mathbf{Prim}}$ utilizes a global key guessing oracle $\mathsf{GlobalK}_{\boldsymbol{\Delta}}$ to identify valid global keys when processing the ideal primitive queries relevant to the programmed query-response pairs of H. Moreover, when

[3]For simplicity, we omit some details, e.g., distinct tweaks.

processing the non-relevant ideal primitive queries, $\mathcal{S}_{\mathbf{Prim}}$ follows the same lazy simulation strategy regardless of the phase switch, i.e., the $\mathcal{S}_{\mathbf{Prim}}$ randomly samples and records **Prim** query-response pairs only when needed. This motivates us to utilize auxiliary $\mathcal{S}_{\mathbf{Prim}}$ instances and extract candidate Δ_i values, which are tested via the augmented global-key queries of the ideal LaAND functionality. Once a valid global key is found, the MPC simulator could construct the valid query-response pairs of H and perform the phase switch. The lazy simulation property of $\mathcal{S}_{\mathbf{Prim}}$ ensures that the delayed phase switch only brings negligible difference.

Nevertheless, the above simulation strategy requires preparing the query-response pairs of H, as in Equation 1, without knowing the honest parties' outputs. As Φ^i contains randomness related to the honest parties' outputs, this leads to a dilemma as the Δ_i extraction requires honest parties's outputs but acquiring such information relies on an extracted Δ_i value.

We address this dilemma by adding a random zero sharing to the Φ^i values. Let $\mathcal{M} \subseteq [n]$ be the set of corrupted parties, we have:

$$\begin{aligned}
\Phi^i &= b^i\Delta_i \oplus \bigoplus_{j\neq i}(\mathsf{M}_j[b^i] \oplus \mathsf{K}_i[b^j])\,,\\
&= b^i\Delta_i \oplus \bigoplus_{j\in\mathcal{M}}(\underbrace{\mathsf{K}_j[b^i]} \oplus b^i\underbrace{\Delta_j} \oplus \underbrace{\mathsf{M}_i[b^j]} \oplus \underbrace{b^j}\Delta_i) \oplus \bigoplus_{j\notin\mathcal{M},j\neq i}(\mathsf{M}_j[b^i] \oplus \mathsf{M}_i[b^j] \oplus b^j\Delta_i)\,,\\
&= b\Delta_i \oplus \bigoplus_{j\in\mathcal{M}}(\underbrace{\mathsf{K}_j[b^i]} \oplus \underbrace{\mathsf{M}_i[b^j]} \oplus b^i\underbrace{\Delta_j}) \oplus \bigoplus_{j\notin\mathcal{M},j\neq i}(\mathsf{M}_j[b^i] \oplus \mathsf{M}_i[b^j])\,,
\end{aligned}$$

where the underlined terms are chosen by the adversary and thus known by the MPC simulator. Since $(b, b^i) \in \{0,1\}^2$, these two values can be enumerated. The only unknown value is $V_i := \oplus_{j\notin\mathcal{M},j\neq i}(\mathsf{M}_j[b^i] \oplus \mathsf{M}_i[b^j])$ and the V_i values of the honest parties collectively form an additive zero sharing. Therefore, by adding an additional random zero sharing, the marginal distributions of the honest V_i's are effectively randomized as they now depend on the randomness of the additional zero sharing, which is internal and does not appear in the environment's view.

Ferret COT Protocol. Ferret [YWL+20] is a maliciously secure protocol that generates many COT correlations from a limited number of base COT correlations. Similar to the line of work on "silent OT" [BCG+19b, CRR21, BCG+22, RRT23], the protocol relies on the Learning Parity with Noise assumption and has sublinear communication. A crucial step in Ferret is to generate Single-Point COT correlations (SPCOT) where their choice bits form a one-hot vector (i.e., only one bit is 1 and the rest bits are 0). The SPCOT protocol in [YWL+20] uses the following punctured GGM tree.

To generate a SPCOT correlation ($\boldsymbol{v} = \boldsymbol{w} \oplus \boldsymbol{u} \cdot \Delta$) of length $n = 2^h$, where the only non-zero position in $\boldsymbol{u} \in \mathbb{F}_2^n$ is indexed by $\alpha \in [0, n)$, the sender first generates a GGM tree of depth h and defines the leaf nodes as a vector $\boldsymbol{v} \in \mathbb{F}_{2^\lambda}^n$. The sender and receiver invoke the COT protocol for h times to get $(\Delta, \boldsymbol{q}) \in \mathbb{F}_{2^\lambda} \times \mathbb{F}_{2^\lambda}^h$ and $(\boldsymbol{r}, \boldsymbol{t}) \in \mathbb{F}_2^h \times \mathbb{F}_{2^\lambda}^h$, respectively, such that $\boldsymbol{q} = \boldsymbol{t} \oplus \boldsymbol{r} \cdot \Delta$. Let α have the binary decomposition $\alpha_1, ..., \alpha_h$ and K_0^i (resp., K_1^i) be the sum of even (resp., odd) nodes for the i^{th} level. The receiver specifies the input α by sending $b_i = r_i \oplus \overline{\alpha_i}$ for each $i \in [h]$. Then, the sender sends the sum indexed by $\overline{\alpha_i}$ for each $i \in [h]$ by sending $M_0^i = K_0^i \oplus \mathsf{H}(q_i \oplus b_i\Delta)$ and $M_1^i = K_1^i \oplus \mathsf{H}(q_i \oplus \overline{b_i}\Delta)$ (omitting tweaks for simplicity), which allows the receiver to recover all the leaf nodes in the GGM tree except the α-indexed leaf node, denoted as $\{w_i\}_{i\neq\alpha}$. To let the receiver recover w_α, the sender sends $c = \Delta \oplus \bigoplus_{i\in[0,n)} v_i$ to the receiver, who computes $w_\alpha = c \oplus \bigoplus_{i\neq\alpha} w_i = v_\alpha \oplus \Delta$. Finally, some consistency check is performed to ensure malicious security.

The security of H is required for the security against a corrupted receiver. In this case, the simulation strategy claimed in [YWL+20] is to (i) uniformly sample M_0^i, M_1^i due to the pseudoran-

domness of the GGM tree; (ii) uniformly sample c due to the (mT)CCR security of H as shown in Equation 2.

$$c = \bigoplus_{i\in[0,n)} v_i \oplus \Delta = M_0^h \oplus M_1^h \oplus \mathsf{H}(t_h) \oplus \underbrace{\mathsf{H}(t_h \oplus \Delta) \oplus \Delta}_{\text{Pseudorandom}}. \tag{2}$$

Even in the standalone model considered by [YWL+20], this simulation fails since the SPCOT outputs contain the same global key Δ as the base COTs, which serves as the underlying randomness in the CCR experiment. In other words, the distinguisher is given the SPCOT outputs and access to H so that it is trivial to use the global key Δ to distinguish the two worlds. In the UC model, this issue also exists since the environment has the same ability.

Nevertheless, we prove that with the S-mTCCRL notion, UC security can hold even for the environment with Δ. The high-level idea is that, to ensure that the environment cannot distinguish the two worlds after learning Δ, we use the implicit programming in the S-mTCCRL notion to keep the consistency of Equation 2. This programming is done by giving the S-mTCCRL simulator a list of the CR-style query-answer pairs in Equation 2; we need not dive into probability analysis, making the proof more modular. We defer the detailed protocol and proof to Appendix G due to page limits.

3 Preliminaries

For any positive integer n, denote by $[n]$ the set $\{1, ..., n\}$. For two non-negative integers m, n such that $m > n$, denote by $[n, m)$ the set $\{n, ..., m-1\}$. For any set or list $\mathcal{S}$, denote by $|\mathcal{S}|$ its size, i.e., the number of elements in $\mathcal{S}$. Denote by $\mathcal{F}_{\mathcal{X}_1\times\mathcal{X}_2\times...,\mathcal{W}}$ the set of all functions from $\mathcal{X}_1 \times \mathcal{X}_2 \times ...$ to $\mathcal{W}$, and $\mathcal{E}_{\mathcal{I},\mathcal{W}}$ the set of all blockciphers with keyspace $\mathcal{I}$ and message space $\mathcal{W}$. To simplify notations, let $\mathcal{KWC} := \mathcal{F}_{[u]\times\mathcal{W}\times\mathcal{I}\times\{0,1\},\mathcal{W}}$.

Our security definitions will be given in the ideal model, in which a public ideal primitive is accessible by all parties, including the adversary. For generality, we denote this primitive by **Prim** (as we did in the Introduction), and denote its appropriate space by $\mathcal{P}$. When $\mathbf{Prim} = \mathbf{RO} \xleftarrow{\$} \mathcal{P} = \mathcal{F}_{\mathcal{X},\mathcal{W}}$ is a random oracle, it provides an interface **RO** that maps $x \in \mathcal{X}$ to $\mathbf{RO}(x) \in \mathcal{W}$. When $\mathbf{Prim} = \mathbf{E} \xleftarrow{\$} \mathcal{P} = \mathcal{E}_{\mathcal{I},\mathcal{W}}$ is an ideal blockcipher, it provides an interface **E** that maps $(\delta, k, z) \in \{+, -\} \times \mathcal{I} \times \mathcal{W}$ to $\mathbf{E}(\delta, k, z) \in \mathcal{W}$, where $\delta = +$ indicates enciphering queries and $\delta = -$ indicates inverse.

Our security definition in Sect. 4 will need *transcripts of queries to a function* $\mathbf{F} \in \mathcal{KWC}$ *and the answers*. Formally, for any Turing machine $\mathrm{TM}^{\mathbf{F}}$ that has access to **F**, we will denote by $\mathcal{Q}_F = \{(\mathsf{id}_1, w_1, i_1, b_1, z_1), \ldots\}$ the transcripts of queries of $\mathrm{TM}^{\mathbf{F}}$ to **F** and the answers. A tuple $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$ indicates $\mathrm{TM}^{\mathbf{F}}$ queried $\mathbf{F}(\mathsf{id}, w, i, b)$ and obtained z as the response.

3.1 Multi-user tweakable circular correlation robust hashing

In this section, we recall the notion of *multi-user tweakable circular correlation robust (mTCCRL) hashing with key leakages* from [GWYY25], in the ideal model. We first formalize necessary oracles (which have been mentioned in Sect. 2.1).

Definition 1. *Given a hash function* $\mathsf{H}^{\mathbf{Prim}} : \mathcal{W} \times \mathcal{I} \to \mathcal{W}$ *(defined upon* **Prim***) and a vector of* u *secret keys* $\mathbf{\Delta} = (R_1, ..., R_u)$ *of* u *users, the multi-user tweakable circular key-whitened oracle is defined as*

$$\mathsf{mTCK}_{\mathbf{\Delta}}^{\mathsf{H}^{\mathbf{Prim}}}(id, w, i, b) := \mathsf{H}^{\mathbf{Prim}}(w \oplus R_{id}, i) \oplus b \cdot R_{id}, \tag{3}$$

Given a vector of u secret keys $\boldsymbol{\Delta} = (R_1, ..., R_u)$ of u users, the multi-user key leaking oracle $\mathsf{mL}_{\boldsymbol{\Delta}}(\mathsf{id}, P)$ takes a user index $\mathsf{id} \in [u]$ and a predicate $P \in \mathcal{P}$ (for a set $\mathcal{P}$ that is the parameter of the notion) as input and aborts the session of the id-th user if and only if $P(R_{\mathsf{id}}) \neq 1$. This means if the adversary keeps querying $\mathsf{mL}_{\boldsymbol{\Delta}}(\mathsf{id}, P)$ with $P(R_{\mathsf{id}}) = 1$ then nothing happens (by this, it gains information about R_{id} by knowing $P(R_{\mathsf{id}}) = 1$); once it queries $\mathsf{mL}_{\boldsymbol{\Delta}}(\mathsf{id}, P)$ with $P(R_{\mathsf{id}}) = 0$ then all the keyed oracles of the id-th user never reply queries any more—namely, $\mathsf{mL}_{\boldsymbol{\Delta}}$ never replies queries of the form $(\mathsf{id}, \star)$ and $\mathsf{mTCK}^{\mathsf{H}^{\mathbf{Prim}}}_{\boldsymbol{\Delta}}$ never replies queries of the form $(\mathsf{id}, \star, \star, \star)$.

mTCCRL. We now recall the mTCCRL notion [GWYY25], which roughly requires the aforementioned oracle $\mathsf{mTCK}^{\mathsf{H}^{\mathbf{Prim}}}_{\boldsymbol{\Delta}}$ to be indistinguishable from a random function sampled from $\mathcal{KWC} = \mathcal{F}_{[u]\times\mathcal{W}\times\mathcal{I}\times\{0,1\},\mathcal{W}}$, in the presence of both **Prim** and $\mathsf{mL}_{\boldsymbol{\Delta}}$.

Definition 2 (mTCCRL advantage)**.** *Given a (double-input) function $\mathsf{H}^{\mathbf{Prim}} : \mathcal{W} \times \mathcal{I} \to \mathcal{W}$, a tuple of key sets $\overrightarrow{\mathcal{R}} = \mathcal{R}_1 \times ... \times \mathcal{R}_u \subseteq \mathcal{W}^u$, a set $\mathcal{P}$ of "allowed" predicates and a distinguisher $\mathcal{D}$, define*

$$\mathbf{Adv}^{\mathrm{mTCCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{D}) := \Big|\Pr_{\mathbf{Prim}\xleftarrow{\$}\mathcal{P},\boldsymbol{\Delta}\xleftarrow{\$}\overrightarrow{\mathcal{R}}}\left[\mathcal{D}^{\mathbf{Prim},\mathsf{mTCK}^{\mathsf{H}^{\mathbf{Prim}}}_{\boldsymbol{\Delta}},\mathsf{mL}_{\boldsymbol{\Delta}}} = 1\right] - \Pr_{\mathbf{F}\xleftarrow{\$}\mathcal{KWC},\mathbf{Prim}\xleftarrow{\$}\mathcal{P},\boldsymbol{\Delta}\xleftarrow{\$}\overrightarrow{\mathcal{R}}}\left[\mathcal{D}^{\mathbf{Prim},\mathbf{F},\mathsf{mL}_{\boldsymbol{\Delta}}} = 1\right]\Big|,$$

where we require that:

(i) $\mathcal{D}$ never queries both $(\mathsf{id}, w, i, 0)$ and $(\mathsf{id}, w, i, 1)$ to its second oracle (for any id, w, i).

(ii) Every query (id, P) of $\mathcal{D}$ to its third oracle $\mathsf{mL}_{\boldsymbol{\Delta}}$ has $P \in \mathcal{P}$.

We call an mTCCRL distinguisher $\mathcal{D}$ $(q_C, q_P, M, q_{L,max}, \mu)$-*bounded*, if:

(i) $\mathcal{D}$ makes at most q_P queries to **Prim** and at most q_C queries to $\mathsf{mTCK}^{\mathsf{H}^{\mathbf{Prim}}}_{\boldsymbol{\Delta}}/\mathbf{F}$;

(ii) For all $i \in \mathcal{I}$, the number of queries (across all the u users) of the form $(\star, \star, i, \star)$ to $\mathsf{mTCK}^{\mathsf{H}^{\mathbf{Prim}}}_{\boldsymbol{\Delta}}/\mathbf{F}$ is at most μ;

(iii) $\mathcal{D}$ makes key leaking queries with at most M distinct user indices id, and for any of them, $\mathcal{D}$ makes at most $q_{L,max}$ queries to $\mathsf{mL}_{\boldsymbol{\Delta}}(\mathsf{id}, \cdot)$.

As shown in [GWYY25], the five parameters give a complicated though fine-grained characterization of adversarial power in the mTCCRL setting.

The definition of mTCCR, first formalized in [GKW+20], can be obtained by dropping the key leaking oracle $\mathsf{mL}_{\boldsymbol{\Delta}}$ from Definition 2. As discussed in Sect. 2.1, mTCCR may also be viewed as an extreme case of mTCCRL in which the distinguisher does not query $\mathsf{mL}_{\boldsymbol{\Delta}}$.

A zoo of weaker variants can be obtained by "dropping" the features (multi-user, tweakable, circularity, key leakage) from mTCCRL. While these notions may be useful in relevant protocols, we defer the formal elaboration to Appendix A.2 due to page limits.

4 S-mTCCRL: Simulatable TCCRL Hashing

Consider a function $\mathsf{H}^{\mathbf{Prim}} : \mathcal{W} \times \mathcal{I} \to \mathcal{W}$ defined upon a public ideal primitive **Prim**. Using the oracles $\mathsf{mTCK}^{\mathsf{H}}_{\boldsymbol{\Delta}}$ and $\mathsf{mL}_{\boldsymbol{\Delta}}$ formalized in Definition 1, the *simulation-based* variant of mTCCRL

The S-mTCCRL experiment $\mathbf{Exp}^{\text{S-mTCCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u,\mathcal{D},\mathcal{S},\beta}$:

1. **Initialization:**
   - If $\beta = 0$: sample an ideal primitive $\mathbf{Prim} \xleftarrow{\$} \mathcal{P}$ (from the appropriate space $\mathcal{P}$) and a vector of u keys $\boldsymbol{\Delta} \xleftarrow{\$} \overrightarrow{\mathcal{R}}$;
   - Else, i.e., $\beta = 1$: sample a random function $\mathbf{F} \xleftarrow{\$} \mathcal{KWC}$ and a vector of u keys $\boldsymbol{\Delta} \xleftarrow{\$} \overrightarrow{\mathcal{R}}$.
2. **Phase 1:**
   - If $\beta = 0$: run $end \leftarrow \mathcal{D}^{\mathbf{Prim},\mathsf{mTCK}^{\mathsf{H}^{\mathbf{Prim}}}_{\boldsymbol{\Delta}},\mathsf{mL}_{\boldsymbol{\Delta}}}$ with **Prim**, the multi-user tweakable circular key-whitened oracle $\mathsf{mTCK}^{\mathsf{H}^{\mathbf{Prim}}}_{\boldsymbol{\Delta}}$, and the key leaking oracle $\mathsf{mL}_{\boldsymbol{\Delta}}$, till $\mathcal{D}$ outputs *end* indicating the end of phase 1;
   - Else, i.e., $\beta = 1$: run $end \leftarrow \mathcal{D}^{\mathcal{S}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}}}.O_1,\mathbf{F},\mathsf{mL}_{\boldsymbol{\Delta}}}$ with the simulator's interface $\mathcal{S}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}}}.O_1$, the random function $\mathbf{F}$, and the key leaking oracle $\mathsf{mL}_{\boldsymbol{\Delta}}$, till $\mathcal{D}$ outputs *end* indicating the end of phase 1;
3. **Key revealing:** Give $\mathcal{D}$ the key vector $\boldsymbol{\Delta}$ used in phase 1. If $\beta = 1$ then invoke $\mathcal{S}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}}}.PhaseSwitch(\mathcal{Q}_F)$, where $\mathcal{Q}_F$ is the transcript of queries and answers of $\mathcal{D}$ to $\mathbf{F}$ in the first phase (transcript has been formalized in Sect. 3).
4. **Phase 2:**
   - If $\beta = 0$: run $\beta' \leftarrow \mathcal{D}^{\mathbf{Prim}}$ with **Prim** and obtain its output β';
   - Else, i.e., $\beta = 1$: run $\beta' \leftarrow \mathcal{D}^{\mathcal{S}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}}}.O_1}$ with the simulator's interface and obtain its output β'.
5. **Finalization:** Output β' (the output of $\mathcal{D}$).

Figure 1: Experiments in the S-mTCCRL setting. Note that in Phase 2, $\mathcal{D}$ has got $\boldsymbol{\Delta}$, and its access to $\mathsf{mL}_{\boldsymbol{\Delta}}$ is thus not needed.

(S-mTCCRL) requires $(\mathbf{Prim}, \mathsf{mTCK}^{\mathsf{H}}_{\boldsymbol{\Delta}}, \mathsf{mL}_{\boldsymbol{\Delta}})$ to be indistinguishable from $(\mathcal{S}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}}}, \mathbf{F}, \mathsf{mL}_{\boldsymbol{\Delta}})$ for some simulator $\mathcal{S}$ in a 2-phase game described in Fig. 1 (and as sketched in Sect. 2.2). The *multi-key guessing oracle* $\mathsf{GlobalK}_{\boldsymbol{\Delta}}$ is implemented by relevant protocol entities to help the simulator confirm its key guesses, and it accepts inputs (id, Δ') and replies if $\Delta' = \Delta_{\mathsf{id}}$ or not. The formal definition is as follows.

Definition 3 (S-mTCCRL advantage)**.** *Given a double-input function* $\mathsf{H}^{\mathbf{Prim}} : \mathcal{W} \times \mathcal{I} \to \mathcal{W}$*, a tuple of key sets* $\overrightarrow{\mathcal{R}} = \mathcal{R}_1 \times ... \times \mathcal{R}_u \subseteq \mathcal{W}^u$*, a set* $\mathcal{P}$ *of "allowed" predicates, a distinguisher* $\mathcal{D}$ *and a simulator* $\mathcal{S}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}}}$ *providing an interface* O_1*, define*

$$\begin{aligned}&\mathbf{Adv}^{\text{S-mTCCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{D},\mathcal{S})\\ &:= \left|\Pr[\mathbf{Exp}^{\text{S-mTCCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u,\mathcal{D},\mathcal{S},0} = 1] - \Pr[\mathbf{Exp}^{\text{S-mTCCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u,\mathcal{D},\mathcal{S},1} = 1]\right|.\end{aligned}$$

Furthermore:

(i) $\mathcal{D}$ *never queries both* $(\mathsf{id}, w, i, 0)$ *and* $(\mathsf{id}, w, i, 1)$ *to its third oracle (for any* id, w, i*).*

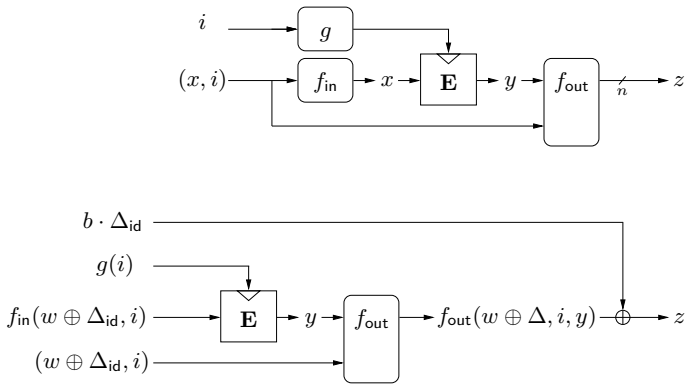


Figure 2: (Up) The general hash $\mathsf{GH}[f_{\mathsf{in}}, f_{\mathsf{out}}, g]^{\mathbf{E}}$ making a single call to a blockcipher $\mathbf{E}$. f_{in}, f_{out}, and g are arbitrary functions (with restrictions listed in Sect. 5.1). (Bottom) The oracle $\mathsf{mTCK}_{\boldsymbol{\Delta}}^{\mathsf{GH}[f_{\mathsf{in}}, f_{\mathsf{out}}, g]^{\mathbf{E}}}$ defined upon $\mathsf{GH}[f_{\mathsf{in}}, f_{\mathsf{out}}, g]^{\mathbf{E}}$ (we only illustrate the key Δ_{id}).

(ii) Every query (id, P) of $\mathcal{D}$ to its third oracle $\mathsf{mL}_{\boldsymbol{\Delta}}$ has $P \in \mathcal{P}$.

Similarly to Sect. 3.1, we also use the notion $(q_C, q_P, M, q_{L,max}, \mu)$-*bounded distinguishers* to quantify adversarial power. Asymptotically, if there exists a simulator $\mathcal{S}$ such that for any distinguisher $\mathcal{D}$, $\mathbf{Adv}^{\text{S-mTCCRL}}_{\mathsf{H}^{\mathbf{Prim}}, \overrightarrow{\mathcal{R}}, \mathcal{P}_{\mathsf{free}}, \mu, u}(\mathcal{D}, \mathcal{S})$ is negligibly small, then $\mathsf{H}^{\mathbf{Prim}}$ is S-mTCCRL-secure.

It remains open if S-mTCCRL is strictly stronger than mTCCRL of [GWYY25].

5 S-mTCCRL Hashing in the Ideal Cipher Model

In this section, we consider the case of $\mathbf{Prim} = \mathbf{E} \in \mathcal{E}_{\{0,1\}^\kappa, \{0,1\}^n}$ being an ideal blockcipher, and prove that the hash $\widehat{\mathsf{MMO}}^{\mathbf{E}}(x, i) := \mathbf{E}(+, i, \sigma(x)) \oplus \sigma(x)$ of [GKW+20] is S-mTCCRL secure in the ideal cipher model.

As mentioned in Sect. 2.3.1, instead of proving the claim from scratch, we establish and utilize a relation between S-mTCCRL and mTCCRL using the model $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$ (capturing a wide class of 1-call construction). Therefore, we provide a more detailed discussion of the model $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$ and of certain restrictions needed for a fine-grained security proof in Sect. 5.1. We then provide our S-mTCCRL result on $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$ and its proof idea in Sect. 5.2. We finally derive concrete results for $\widehat{\mathsf{MMO}}^{\mathbf{E}}$ in Sect. 5.3.

5.1 Unified Model of 1-call Blockcipher-based Hash

As mentioned in the Introduction, $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}} : \{0,1\}^n \times \mathcal{I} \to \{0,1\}^n$ is defined as

$$\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}(x, i) := f_{\mathsf{out}}\Big(x, i, \mathbf{E}(+, g(i), f_{\mathsf{in}}(x, i))\Big),$$

where $\mathbf{E} \in \mathcal{E}_{\{0,1\}^\kappa, \{0,1\}^n}$ is a blockcipher and $f_{\mathsf{in}} : \{0,1\}^n \times \mathcal{I} \to \{0,1\}^n$, $g : \mathcal{I} \to \{0,1\}^\kappa$ and $f_{\mathsf{out}} : \{0,1\}^n \times \mathcal{I} \times \{0,1\}^n \to \{0,1\}^n$ are arbitrary functions (modulo some injectivity restrictions elaborated below). To have better security bounds, we don't allow the key $g(i)$ to depend on the input w.

5.1.1 Restrictions.

We impose three restrictions on the functions f_{in}, f_{out}, and g. First, $g : \mathcal{I} \to \{0,1\}^\kappa$ is injective.

Second, the function $f_{\mathsf{in}}(\cdot, i)$ is bijective for all $i \in \{0,1\}^n \times \mathcal{I}$. Denote its inverse by $f_{\mathsf{in}}^{-1} : \mathcal{I} \times \{0,1\}^n \to \{0,1\}^n$, i.e., $f_{\mathsf{in}}^{-1}(i, f_{\mathsf{in}}(w,i)) = w$ for all $i \in \mathcal{I}$ and all $w \in \{0,1\}^n$. This also means that given w, i, x, the key Δ such that $x = f_{\mathsf{in}}(w \oplus \Delta, i)$ can be computed via $\Delta \leftarrow f_{\mathsf{in}}^{-1}(i,x) \oplus w$.

Third, the function $f_{\mathsf{out}}(w, i, \cdot)$ is bijective for all $(w,i) \in \{0,1\}^n \times \mathcal{I}$. Denote its inverse by $f_{\mathsf{out}}^{-1} : \{0,1\}^n \times \mathcal{I} \times \{0,1\}^n \to \{0,1\}^n$, i.e., $f_{\mathsf{out}}^{-1}(w, i, f_{\mathsf{out}}(w,i,y)) = y$ for all $(w,i) \in \{0,1\}^n \times \mathcal{I}$ and all $y \in \{0,1\}^n$.

Finally, consider the computation flow of the oracle $\mathsf{mTCK}_{\Delta}^{\mathsf{H}}$ in Eq. (3). We require that for any $(w,i,b,y) \in \{0,1\}^n \times \mathcal{I} \times \{0,1\} \times \{0,1\}^n$, the function $h_{w,i,b,y} : \{0,1\}^n \to \{0,1\}^n$ defined via $h_{w,i,b,y}(\Delta) := f_{\mathsf{out}}(w \oplus \Delta, i, y) \oplus b \cdot \Delta$ is bijective, and there exists an efficient function $kd_y : \{0,1\}^n \times \mathcal{I} \times \{0,1\} \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ computing its inverse, i.e., $kd_y(w,i,b,y,z) = \Delta$ (or equivalently, $f_{\mathsf{out}}(w \oplus kd_y(w,i,b,y,z), i, y) \oplus b \cdot kd_y(w,i,b,y,z) = z$). Note that this implies that for any $(w,i,b,z) \in \{0,1\}^n \times \mathcal{I} \times \{0,1\} \times \{0,1\}^n$, the function $h'_{w,i,b,z}(\Delta) := f_{\mathsf{out}}^{-1}(w \oplus \Delta, i, z \oplus b \cdot \Delta)$ is bijective as well: otherwise, the existence of Δ_1, Δ_2 with $y = h'_{w,i,b,z}(\Delta_1) = h'_{w,i,b,z}(\Delta_2)$ implies $h_{w,i,b,y}(\Delta_1) = h_{w,i,b,y}(\Delta_2) = z$, contradicting the bijectivity of $h_{w,i,b,y}$. The bijectivity will be crucial in our proof (in Appendix D.6).

The last restriction seems strong, since it means

$$\max_{(w,i,b,y) \in \{0,1\}^n \times \mathcal{I} \times \{0,1\} \times \{0,1\}^n,\ z^\circ \in \{0,1\}^n} \Pr[\Delta \xleftarrow{\$} \mathcal{R} : f_{\mathsf{out}}(w \oplus \Delta, i, y) \oplus b \cdot \Delta = z^\circ]$$

is small. But constructions severely violating this restriction may be insecure. For example, if $f_{\mathsf{out}}(w \oplus \Delta, i, y) := y$ (i.e., the input $w \oplus \Delta$ is not fed forward, and $\Pr[\Delta \xleftarrow{\$} \mathcal{R} : f_{\mathsf{out}}(w \oplus \Delta, i, z^\circ) \oplus b \cdot \Delta = z^\circ] = 1$), then given a construction query (id, w, i, b) and its response z, we recover $f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i) = \mathbf{E}(+, g(i), z)$ and further recover the key Δ_{id} to fully break the construction.

As elaborated below, the aforementioned restrictions are indeed fulfilled by virtually all practical designs.

Examples. The construction $\widehat{\mathsf{MMO}}^{\mathbf{E}}(x,i) := \mathbf{E}(+, i, \sigma(x)) \oplus \sigma(x)$ of [GKW+20] has

$$\begin{aligned}
g(i) &= i, \quad \text{(which is injective)} \\
f_{\mathsf{in}}(w,i) &= \sigma(w), & f_{\mathsf{in}}^{-1}(i,x) &= \sigma^{-1}(x), \\
f_{\mathsf{out}}(w,i,y) &= y \oplus \sigma(w), & f_{\mathsf{out}}^{-1}(w,i,z) &= z \oplus \sigma(w), \\
kd_y(w,i,b,y,z) &= \begin{cases} \sigma^{-1}(z \oplus y) \oplus w & \text{if } b = 0 \\ \varphi^{-1}(z \oplus y \oplus \sigma(w)) & \text{if } b = 1 \end{cases}
\end{aligned}$$

where $\varphi : \{0,1\}^n \to \{0,1\}^n$ is the permutation such that $\varphi(\Delta) = \sigma(\Delta) \oplus \Delta$. Other variants of $\widehat{\mathsf{MMO}}^{\mathbf{E}}$ are also instances of $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$, as long as their components satisfy the restrictions on $f_{\mathsf{in}}, f_{\mathsf{out}}$ and g.

5.2 S-mTCCRL Security of $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$

The S-mTCCRL security of the general model $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$ is given in Theorem 1 below.

Theorem 1. *Let $\mathcal{P}$ be any set of allowed predicates and $q_P \le 2^n/2$. If $\mathbf{E} \in \mathcal{E}_{\{0,1\}^\kappa, \{0,1\}^n}$ is modeled as an ideal cipher, then there exists a simulator $\mathcal{S}_G^{\mathsf{GlobalK}_\Delta}$ such that for any $(q_C, q_P, M, q_{L,max}, \mu)$-bounded S-mTCCRL distinguisher $\mathcal{D}$ against $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$ and any constant $T > 0$, there exists*

a $(q_C, q_P + \mu T q_C, M, q_{L,max}, \mu)$*-bounded mTCCRL distinguisher* $\mathcal{B}$ *such that*

$$\mathbf{Adv}^{\text{S-mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{D},\mathcal{S}_G) \le 2\mathbf{Adv}^{\text{mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{B}) + \frac{3\mu(q_P+q_C)}{2^n} + \frac{2}{T}.$$

In addition, $\mathcal{S}_G^{\mathsf{GlobalK}_{\mathbf{\Delta}}}$ *makes at most* μq_P *queries to its oracle* $\mathsf{GlobalK}_{\mathbf{\Delta}}$ *and runs in time* $O(\mu q_P + q_C)$*.*

The presence of the adjustable constant T in Theorem 1 corresponds to the bad event's (adjustable) threshold, and is typical of security proofs that involve a threshold-based "bad event" [CS14]. The term $2\mathbf{Adv}^{\text{mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{B})$ grows with T, while the term $2/T$ decreases with T. In concrete settings we can choose a good enough value for T. E.g., in our concrete example in in Sect. 5.3, it has $2\mathbf{Adv}^{\text{mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{B}) = O(Tq_Pq_C/2^n)$, and setting $T \approx \sqrt{\frac{2^n}{q_Pq_C}}$ yields $2\mathbf{Adv}^{\text{mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{B})+2/T = O(\sqrt{\frac{q_Pq_C}{2^n}})$, indicating security up to the same amount of queries as that indicated by $\mathbf{Adv}^{\text{mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{B})$ solely.

The full proof is deferred to Appendix D. Here we provide the core ideas. The crux is to construct the simulator $\mathcal{S}_G$ that provides an interface $O_1 : \{+,-\} \times \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ that mimics an ideal cipher consistent with $\mathbf{F}$ when necessary. It extends the simulation ideas of Sect. B. In phase 1, $\mathcal{S}_G$ simply replies $O_1(\delta,k,v)$ by (ideal cipher-style) lazy sampling. When *PhaseSwitch*$(\mathcal{Q}_F)$ is invoked, $\mathcal{S}_G$ makes a copy $\mathcal{Q}_{SF}$ of $\mathcal{Q}_F$.

Then in phase 2, for each new query $O_1(\delta,k,v)$ $\mathcal{S}_G$ checks if there exists a record $(\mathsf{id},i,w,b,z) \in \mathcal{Q}_{SF}$ that is "linked" to $O_1(\delta,k,v)$. Here "being linked" means Δ_{id} equals $f_{\mathsf{in}}^{-1}(i,v)\oplus w$ (when $\delta = +$) or $kd_y(w,i,b,y,v)$ (when $\delta = -$). (So the existence of kd_y helps.) Again, this checking is achieved by querying $\mathsf{GlobalK}_{\mathbf{\Delta}}$ for information about Δ_{id}. If so, $\mathcal{S}_G$ responds $O_1(\delta,k,v)$ consistently with (id,i,w,b,v); otherwise, it samples a response for $O_1(\delta,k,v)$. The formal description of $\mathcal{S}_G$ is in Appendix D.1.

The above simulation strategy fails in two cases:

(i) A query $O_1(\delta,k,v)$ replied in phase 1 turns out to be "linked" to some $(\mathsf{id},i,w,b,z) \in \mathcal{Q}_{SF}$.

(ii) In phase 2, some query $O_1(\delta,k,v)$ is "linked" to (at least) two distinct records (id,i,w,b,z), $(\mathsf{id}',i',w',b',z') \in \mathcal{Q}_{SF}$.

The two cases resemble the analogues in the proof of Theorem 3. Though, unlike Theorem 3, here we consider *real-world executions* and formalize a closely related bad event Chain. We then relate the occurrence of Chain to breaking the mTCCRL security of $\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}}$, and this bounds the probability of Chain in the real world. Conditioned on that Coll does not occur, the indistinguishability of the real and ideal world experiments are then bounded using the randomness mapping argument of [CHK+16, ABD+13]. We refer to Appendices D.2—D.5 for the formal elaborations.

5.3 Concrete Results: S-mTCCRL security of $\widehat{\mathsf{MMO}}^{\mathbf{E}}$

This section derives concrete S-mTCCRL bounds for the hash

$$\widehat{\mathsf{MMO}}^{\mathbf{E}}(x,i) := \mathbf{E}(+,i,\sigma(x)) \oplus \sigma(x),$$

where $\mathbf{E} \in \mathcal{E}_{\{0,1\}^n,\{0,1\}^n}$ is a blockcipher and σ is a linear orthomorphism.

Regarding S-mTCCRL, we follow Guo et al. [GWYY25] and focus on using *oracle-free* predicates, i.e., for every query $\mathsf{mL}_{\mathbf{\Delta}}(\mathsf{id}, P)$, the predicate evaluation $P(\Delta_{\mathsf{id}})$ does not query the ideal cipher $\mathbf{E}$. Let $\mathcal{P}_{\mathsf{free}}$ be the set of all such oracle-free predicates. With this, for any $(q_C, q_P, M, q_{L,max}, \mu)$-bounded distinguisher $\mathcal{B}$, Guo et al. [GWYY25, Theorem 1] proved a tight mTCCRL bound for $\widehat{\mathsf{MMO}}^{\mathbf{E}}$:

$$\begin{aligned}&\mathbf{Adv}^{\text{mTCCRL}}_{\widehat{\mathsf{MMO}}^{\mathbf{E}},\mathcal{R},\mathcal{P}_{\mathsf{free}},u}(\mathcal{B})\\&\leq \frac{2\mu q_P(M+1)(q_{L,max}+1)}{C}+\frac{(\mu-1)\cdot q_C(M+1)(q_{L,max}+1)}{C}.\end{aligned} \tag{4}$$

Gathering Eq. (4) with Theorem 1 and using $C \leq 2^n$ yields:

Corollary 1. *If $q_P \leq 2^n/2$, σ is a linear orthomorphism and $\mathbf{E} \in \mathcal{E}_{\{0,1\}^n,\{0,1\}^n}$ is modeled as an ideal cipher, then there exists a simulator $\mathcal{S}_1$ such that the S-mTCCRL advantage of $\widehat{\mathsf{MMO}}^{\mathbf{E}}$ for any $(q_C, q_P, M, q_{L,max}, \mu)$-bounded distinguisher $\mathcal{D}$ and any constant $T > 0$ has upper bound*

$$\begin{aligned}&\mathbf{Adv}^{\text{S-mTCCRL}}_{\widehat{\mathsf{MMO}}^{\mathbf{E}},\mathcal{R},\mathcal{P}_{\mathsf{free}},u}(\mathcal{D},\mathcal{S}_1) \leq \frac{3\mu(q_P+q_C)}{2^n}+\frac{2}{T}\\&+2\times\left(\frac{2\mu(q_P+\mu T q_C)(M+1)(q_{L,max}+1)}{C}+\frac{(\mu-1)q_C(M+1)(q_{L,max}+1)}{C}\right)\\&\leq \frac{7\mu q_P(M+1)(q_{L,max}+1)+4(\mu+1)^2 T q_C(M+1)(q_{L,max}+1)}{C}+\frac{2}{T},\end{aligned} \tag{5}$$

In addition, $\mathcal{S}_1^{\mathsf{GlobalK}_{\mathbf{\Delta}}}$ makes at most μq_P queries to its oracle $\mathsf{GlobalK}_{\mathbf{\Delta}}$ and runs in time $O(\mu q_P + q_C)$.

Since Eq. (5) holds for any $T > 0$, there exists an optimal T. But setting

$$T = \sqrt{\frac{C}{4(\mu+1)^2 q_C(M+1)(q_{L,max}+1)}},$$

suffices to yield a bound comparable with [GWYY25]:

$$\begin{aligned}&\mathbf{Adv}^{\text{S-mTCCRL}}_{\widehat{\mathsf{MMO}}^{\mathbf{E}},\mathcal{R},\mathcal{P}_{\mathsf{free}},u}(\mathcal{D},\mathcal{S}_1)\\&\leq \frac{7\mu q_P(M+1)(q_{L,max}+1)}{C}+6(\mu+1)\sqrt{\frac{q_C(M+1)(q_{L,max}+1)}{C}},\end{aligned} \tag{6}$$

Setting $M = 0$ and $q_{L,max} = 0$ in Eq. (6) yields the bound

$$\frac{7\mu q_P}{C}+6(\mu+1)\sqrt{\frac{q_C}{C}} \tag{7}$$

in the extreme case where $\mathcal{D}$ *does not query* $\mathsf{mL}_{\mathbf{\Delta}}$. This special case will be useful in some protocols (see Appendix G).

Asymptotically, the bounds in Eqs. (6) and (7) are comparable with Guo et al.'s mTCCRL [GWYY25] and mTCCR [GKW+20] bounds on $\widehat{\mathsf{MMO}}^{\mathbf{E}}$ respectively, and are tight: we refer to [GWYY25, Sect. 4.4 and Appendix B] for discussions.

6 UC-Secure Authenticated AND Triple Generation from S-mTCCRL

As recalled in Sect. 2.4, the previous multi-party TinyOT-style triple generation protocols with minicrypt assumptions have flawed proofs. Moreover, these protocols rely on the random oracle model, which requires somewhat heavy instantiation in practice. In this section, we show that the state-of-the-art protocol [YWZ20] can be adapted to fix the proof flaws and augmented to UC security using our S-mTCCRL hash functions, which can be efficiently instantiated with a block cipher. We will focus on the generation of leaky authenticated AND triples. It is known that this leakage can be removed using the bucketing technique [NNOB12, YWZ20] to obtain authenticated AND triples.

6.1 Basic Primitives

To build up our protocol, we require the following primitives.

Definition 4 (Linear universal hash functions)**.** *We say that a family of linear functions* $\mathcal{H} = \{\mathsf{H}' : \mathbb{F}_{2^\lambda}^n \to \mathbb{F}_{2^\lambda}^m\}$ *is* ε*-almost 1-universal, if it holds that for every non-zero* $x \in \mathbb{F}_{2^\lambda}^n$*,*

$$\Pr_{\mathsf{H}' \xleftarrow{\$} \mathcal{H}} [\mathsf{H}'(x) = 0] \leq \varepsilon\,.$$

IT-MACs and Authenticated Shares. We use the following notation for IT-MACs and authenticated shares in a multi-party setting. For any two distinct parties P_i and P_j among n parties (where each party P_i samples a global key $\Delta_i \in \mathbb{F}_{2^\lambda}$), P_i can use an IT-MAC to authenticate a bit vector $\boldsymbol{x}^j \in \mathbb{F}_2^n$ of P_j: it samples a local key $\mathsf{K}_i[\boldsymbol{x}^j] \xleftarrow{\$} \mathbb{F}_{2^\lambda}^n$ and gives P_j a tag $\mathsf{M}_i[\boldsymbol{x}^j] := \mathsf{K}_i[\boldsymbol{x}^j] \oplus \boldsymbol{x}^j \cdot \Delta_i$.

An authenticated share of $\boldsymbol{x} \in \mathbb{F}_{2^\lambda}^n$, denoted by $\langle\langle \boldsymbol{x} \rangle\rangle$, means that each party P_i has an additive share $\boldsymbol{x}^i \in \mathbb{F}_{2^\lambda}^n$ such that $\bigoplus_{i\in[n]} \boldsymbol{x}^i = \boldsymbol{x}$. Moreover, the share $\boldsymbol{x}^i$ of each party P_i is authenticated by the other $n-1$ parties using IT-MACs. As a result, each party P_i holds its share $\langle\langle \boldsymbol{x} \rangle\rangle_i := (\boldsymbol{x}^i, \{\mathsf{M}_j[\boldsymbol{x}^i]\}_{j\neq i}, \{\mathsf{K}_i[\boldsymbol{x}^j]\}_{j\neq i})$. Authenticated shares satisfy the following XOR-homomorphism:

- $\langle\langle \boldsymbol{c} \rangle\rangle := \langle\langle \boldsymbol{a} \rangle\rangle \oplus \langle\langle \boldsymbol{b} \rangle\rangle$: For each $i \in [n]$,
$$\langle\langle \boldsymbol{c} \rangle\rangle_i := (\boldsymbol{a}^i \oplus \boldsymbol{b}^i, \{\mathsf{M}_j[\boldsymbol{a}^i] \oplus \mathsf{M}_j[\boldsymbol{b}^i], \mathsf{K}_i[\boldsymbol{a}^j] \oplus \mathsf{K}_i[\boldsymbol{b}^j]\}_{j\neq i}).$$
- $\langle\langle \boldsymbol{y} \rangle\rangle := \langle\langle \boldsymbol{x} \rangle\rangle \oplus \boldsymbol{d}$: For each $i \neq 1 \in [n]$,
$$\langle\langle \boldsymbol{y} \rangle\rangle_i := (\boldsymbol{x}^i, \{\mathsf{M}_j[\boldsymbol{x}^i]\}_{j\neq i}, (\{\mathsf{K}_i[\boldsymbol{x}^j]\}_{j\neq i,1}, \mathsf{K}_i[\boldsymbol{x}^1] \oplus \boldsymbol{d} \cdot \Delta_i)),$$
and $\langle\langle \boldsymbol{y} \rangle\rangle_1 := (\boldsymbol{x}^1 \oplus \boldsymbol{d}, \{\mathsf{M}_j[\boldsymbol{x}^i]\}_{j\neq i}, \{\mathsf{K}_i[\boldsymbol{x}^j]\}_{j\neq i})$.

Basic ideal functionalities. Our protocol invokes the functionality $\mathcal{F}_{\mathsf{aShare}}$ for authenticated shares, the functionality $\mathcal{F}_{\mathsf{Zero}}$ for zero sharing, the functionality $\mathcal{F}_{\mathsf{Com}}$ for commitment, and the functionality $\mathcal{F}_{\mathsf{Rand}}$ for coin-tossing. These functionalities are formally defined in Appendix F.1.

6.2 Leaky Authenticated AND Triples

In Figs. 3 and 4, we present the functionality $\mathcal{F}_{\mathsf{LaAND}}$ for leaky authenticated AND triples, which invokes the following macro for authenticated shares:

$$\mathsf{AuthShare}(\mathtt{sid}, \mathtt{ssid}, \boldsymbol{x}, \{\Delta_i\}_{i\in[n]})\,.$$

This macro is invoked by functionalities $\mathcal{F}_{\mathsf{aShare}}$ and $\mathcal{F}_{\mathsf{LaAND}}$. Let $\mathcal{M} \subset [n]$ denote the set of corrupted parties. Given a session ID $\mathtt{sid}$, a sub-session ID $\mathtt{ssid}$, a vector $\boldsymbol{x} \in (\mathbb{F}_2)^\ell$, and n global keys $\{\Delta_i\}_{i\in[n]} \in (\mathbb{F}_{2^\lambda})^n$:

Functionality $\mathcal{F}_{\mathsf{LaAND}}$

This functionality runs with parties $P_1, \ldots, P_n$. Let $\mathcal{M} \subset [n]$ be the subscript set of corrupted parties and $\mathcal{L}$ be the set of allowed selective-failure queries. Let $\{\mathcal{R}_i\}_{i\in[n]}$ be the set of distributions on $\{0,1\}^\lambda$ such that $\mathcal{R}_i$ fixes the LSB of each sample to (n **mod** 2) for $i = 1$ and to 1 for each $i \neq 1$.

Initialization: Upon receiving $(\mathsf{Init}, \mathtt{sid}, i)$ from P_i,

1. If P_i is corrupted, receive $(\mathsf{GK}, \mathtt{sid}, i, \Delta_i)$ from the adversary, where $\Delta_i \in \{0,1\}^\lambda$. Otherwise, $\Delta_i \xleftarrow{\$} \mathcal{R}_i$. Store $(\mathtt{sid}, i, \Delta_i)$, initialize an empty list A_i of outputs, and ignore all subsequent $(\mathsf{Init}, \mathtt{sid}, i)$ calls with the same $(\mathtt{sid}, i)$.

Authenticated shares & Leaky authenticated AND triples: Given stored $(\mathtt{sid}, i, \Delta_i)$ for each $i \in [n]$, this functionality outputs such random tuples: Upon receiving $(\mathsf{Gen}, \mathtt{sid}, \mathtt{ssid}, \ell_1, \ell_2)$ from all parties, where $\ell_1, \ell_2 \in \mathbb{N}^+$ and $\ell_2 \geq \lambda$,

1. **Selective-failure queries:** For each $i \in [n] \setminus \mathcal{M}$ and $j \in \mathcal{M}$ in parallel:
   (a) Receive $(\mathsf{Leak}, \mathtt{sid}, \mathtt{ssid}, i, j, L_{i,j})$ from the adversary, where $L_{i,j} \in \mathcal{L}$.
   (b) If $\Delta_i \notin L_{i,j}$, send $(\mathsf{Fail}, \mathtt{sid}, \mathtt{ssid}, i, j)$ to the adversary and abort; otherwise, do nothing.
2. **Global-key extraction with abort:** Receive $(\mathsf{Quit}, \mathtt{sid}, \mathtt{ssid}, b)$ from the adversary. If $b = 1$, send $(\mathsf{Bye}, \mathtt{sid}, \mathtt{ssid}, \{\Delta_i\}_{i\in[n]\setminus\mathcal{M}})$ to the adversary and abort. Otherwise, do nothing.
3. Sample $\boldsymbol{x} \xleftarrow{\$} \{0,1\}^{\ell_1}$ and $\{\langle\!\langle \boldsymbol{x} \rangle\!\rangle_i\}_{i\in[n]} \xleftarrow{\$} \mathsf{AuthShare}(\mathtt{sid}, \mathtt{ssid}, \boldsymbol{x}, \{\Delta_i\}_{i\in[n]})$.
4. Sample $\boldsymbol{a}, \boldsymbol{b} \xleftarrow{\$} \{0,1\}^{\ell_2}$, compute component-wise AND $\boldsymbol{c} := \boldsymbol{a} \wedge \boldsymbol{b} \in \{0,1\}^{\ell_2}$, and $\{\langle\!\langle \boldsymbol{v} \rangle\!\rangle_i\}_{i\in[n]} \xleftarrow{\$} \mathsf{AuthShare}(\mathtt{sid}, \mathtt{ssid}, \boldsymbol{v}, \{\Delta_i\}_{i\in[n]})$ for each $\boldsymbol{v} \in \{\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{c}\}$.

Figure 3: The functionality of leaky authenticated AND triples (Part I).

1. Receive $(\mathsf{Shr}, \mathtt{sid}, \mathtt{ssid}, \{\boldsymbol{x}^i\}_{i\in\mathcal{M}})$ from the adversary, where $\boldsymbol{x}^i \in (\mathbb{F}_2)^\ell$ for each $i \in \mathcal{M}$, and sample $\boldsymbol{x}^i \xleftarrow{\$} (\mathbb{F}_2)^\ell$ for each $i \notin \mathcal{M}$ such that $\bigoplus_{i\in[n]} \boldsymbol{x}^i = \boldsymbol{x}$.
2. For each $i \in [n]$,
   (a) If P_i is corrupted, receive $(\mathsf{Tag}, \mathtt{sid}, \mathtt{ssid}, i, \{\mathsf{M}_j[\boldsymbol{x}^i]\}_{j\neq i})$ from the adversary, where $\mathsf{M}_j[\boldsymbol{x}^i] \in (\mathbb{F}_{2^\lambda})^\ell$ for each $j \neq i$, and compute $\mathsf{K}_j[\boldsymbol{x}^i] := \mathsf{M}_j[\boldsymbol{x}^i] \oplus \boldsymbol{x}^i \cdot \Delta_j \in (\mathbb{F}_{2^\lambda})^\ell$ for each $j \neq i$.
   (b) Otherwise, receive $(\mathsf{LocalKey}, \mathtt{sid}, \mathtt{ssid}, i, \{\mathsf{K}_j[\boldsymbol{x}^i]\}_{j\in\mathcal{M}})$ from the adversary, where $\mathsf{K}_j[\boldsymbol{x}^i] \in (\mathbb{F}_{2^\lambda})^\ell$ for each $j \in \mathcal{M}$, sample $\mathsf{K}_j[\boldsymbol{x}^i] \xleftarrow{\$} (\mathbb{F}_{2^\lambda})^\ell$ for each $j \notin \mathcal{M}$ and $j \neq i$, and compute $\mathsf{M}_j[\boldsymbol{x}^i] := \mathsf{K}_j[\boldsymbol{x}^i] \oplus \boldsymbol{x}^i \cdot \Delta_j \in (\mathbb{F}_{2^\lambda})^\ell$ for each $j \neq i$.
3. For each $i \in [n]$, compute $\langle\!\langle \boldsymbol{x} \rangle\!\rangle_i := (\boldsymbol{x}^i, \{\mathsf{M}_j[\boldsymbol{x}^i], \mathsf{K}_i[\boldsymbol{x}^j]\}_{j\neq i})$.
4. Return $\{\langle\!\langle \boldsymbol{x} \rangle\!\rangle_i\}_{i\in[n]}$.

Functionality $\mathcal{F}_{\mathsf{LaAND}}$

1. **Share queries with selective failure (Leakage):**

   (a) Receive $(\mathsf{LeakA}, \mathtt{sid}, \mathtt{ssid} \| 0, \{\boldsymbol{e}^j\}_{j\in[n]\setminus\mathcal{M}}, \boldsymbol{q})$ from the adversary, where $\boldsymbol{e}^j \in (\{0,1\}^\lambda)^{\ell_2}$ and $\boldsymbol{q} \in \{0,1\}^{\ell_2}$ for each $j \in [n] \setminus \mathcal{M}$.

   (b) Sample $\mathsf{H}' \xleftarrow{\$} \mathcal{H}$, where $\mathcal{H} = \{\mathsf{H}' : (\{0,1\}^\lambda)^{\ell_2} \to \{0,1\}^\lambda\}$ is a family of linear 1-universal hash functions.

   (c) Send $(\mathsf{Hash}, \mathtt{sid}, \mathtt{ssid}, \mathsf{H}')$ to the adversary, receive $(\mathsf{LeakA}, \mathtt{sid}, \mathtt{ssid} \| 1, \delta)$ from the adversary, where $\delta \in \{0,1\}^\lambda$, and check

   $$\mathsf{H}'\left(\bigoplus_{j\in[n]\setminus\mathcal{M}}(\boldsymbol{a}^j \odot \boldsymbol{e}^j) \oplus \left(\boldsymbol{q} \oplus \bigoplus_{j\in[n]\setminus\mathcal{M}} \boldsymbol{a}^j \wedge \mathsf{lsb}(\boldsymbol{e}^j)\right) \cdot \bigoplus_{j\in[n]\setminus\mathcal{M}} \Delta_j\right) = \delta.$$

   If this check fails or $\mathsf{lsb}(\bigoplus_{i\in[n]} \Delta_i) = 0$, abort; otherwise, do nothing.

2. For each $i \in [n]$, output $(\mathsf{Out}, \mathtt{sid}, \mathtt{ssid}, i, \Delta_i, \langle\!\langle \boldsymbol{x} \rangle\!\rangle_i, \langle\!\langle \boldsymbol{a} \rangle\!\rangle_i, \langle\!\langle \boldsymbol{b} \rangle\!\rangle_i, \langle\!\langle \boldsymbol{c} \rangle\!\rangle_i)$ to P_i.

3. For each $i \in [n]$, append $(\mathtt{sid}, \mathtt{ssid}, i, \langle\!\langle \boldsymbol{x} \rangle\!\rangle_i, \langle\!\langle \boldsymbol{a} \rangle\!\rangle_i, \langle\!\langle \boldsymbol{b} \rangle\!\rangle_i, \langle\!\langle \boldsymbol{c} \rangle\!\rangle_i)$ to A_i.

4. Ignore all subsequent $(\mathsf{Gen}, \mathtt{sid}, \mathtt{ssid}, \ldots)$ calls with the same $(\mathtt{sid}, \mathtt{ssid})$.

Global-key queries: Upon receiving $(\mathsf{Guess}, \mathtt{sid}, i, \Delta_i')$ from the adversary, where $\Delta_i' \in \{0,1\}^\lambda$, if P_i is honest and $\Delta_i' = \Delta_i$, send $(\mathsf{Success}, \mathtt{sid}, \{A_i\}_{i\in[n]\setminus\mathcal{M}})$ to the adversary; otherwise, do nothing.

Figure 4: The functionality of leaky authenticated AND triples (Part II).

This functionality generates both authenticated shares $\langle\!\langle \boldsymbol{x} \rangle\!\rangle$ and leaky authenticated AND triples $(\langle\!\langle \boldsymbol{a} \rangle\!\rangle, \langle\!\langle \boldsymbol{b} \rangle\!\rangle, \langle\!\langle \boldsymbol{c} \rangle\!\rangle)$, where there exists selective-failure leakage about the shares of $\langle\!\langle \boldsymbol{a} \rangle\!\rangle$ held by honest parties. Compared to the counterpart in [YWZ20], our functionality $\mathcal{F}_{\mathsf{LaAND}}$ has the following differences:

1. The functionality additionally allows the adversary to learn the global keys of all honest parties at the cost of aborting the functionality.

2. The LaAND triples (and authenticated shares) are generated in a batch, and the share queries with selective failure are applied to these triples.

3. The functionality additionally allows the adversary to extract the randomness of all honest parties if it can guess the global key of some honest party.

These differences do not affect the essential security promised by $\mathcal{F}_{\mathsf{LaAND}}$. This functionality is parameterized by some set $\mathcal{L}$ of allowed selective-failure queries. Following [Roy22], $\mathcal{L}$ is a subset of the power set of $\{0,1\}^\lambda$, is closed under intersection, and contains $\{0,1\}^\lambda$.

To realize this functionality, we present a protocol Π_{LaAND} in Figs. 5 and 6 based on a S-mTCCRL hash function and a linear universal hash function. This protocol runs in the ($\mathcal{F}_{\mathsf{aShare}}$, $\mathcal{F}_{\mathsf{Zero}}$, $\mathcal{F}_{\mathsf{Com}}$, $\mathcal{F}_{\mathsf{Rand}}$)-hybrid model. The security of Π_{LaAND} is proved in Theorem 2. As discussed in Sect. 2.4, our proof strategy utilizes the ideal primitive simulator itself to extract a valid adversarial guess of an honest party's global key, which returns the outputs of all honest parties from the $\mathcal{F}_{\mathsf{LaAND}}$

Protocol Π_{LaAND}

Initialization: All parties initialize their global keys in $\mathcal{F}_{\mathsf{aShare}}$.

1. For each $i \in [n]$, P_i sends $(\mathsf{Init}, \mathtt{sid}, i)$ to $\mathcal{F}_{\mathsf{aShare}}$ parameterized by $\{\mathcal{R}_i\}_{i\in[n]}$.

Authenticated shares & Leaky authenticated AND triples: All parties generate $\ell := \ell_1 + 3\ell_2$ random authenticated shares.

2. All parties send $(\mathsf{Gen}, \mathtt{sid}, \mathtt{ssid}, \ell)$ to $\mathcal{F}_{\mathsf{aShare}}$, which outputs $(\mathsf{Out}, \mathtt{sid}, \mathtt{ssid}, i, \Delta_i, \langle\langle \boldsymbol{x}\|\boldsymbol{a}\|\boldsymbol{b}\|\boldsymbol{r}\rangle\rangle_i)$ to P_i for each $i \in [n]$, where $\boldsymbol{x} \in \{0,1\}^{\ell_1}$ and $\boldsymbol{a}, \boldsymbol{b}, \boldsymbol{r} \in \{0,1\}^{\ell_2}$.

All parties share $\langle\langle \boldsymbol{c}\rangle\rangle = \langle\langle \boldsymbol{a} \wedge \boldsymbol{b}\rangle\rangle$. Let $\mathsf{H} : (\{0,1\}^\lambda)^2 \to \{0,1\}^\lambda$ be a S-mTCCRL-secure hash function and $\mathcal{H} = \{\mathsf{H}' : (\{0,1\}^\lambda)^{\ell_2} \to \{0,1\}^\lambda\}$ be a family of linear 1-universal hash functions.

3. All parties send $(\mathsf{Zero}, \mathtt{sid}, \mathtt{ssid}, \ell_2 + 1)$ to $\mathcal{F}_{\mathsf{Zero}}$, which outputs $(\mathsf{Val}, \mathtt{sid}, \mathtt{ssid}, \boldsymbol{z}^i \| u^i)$ to P_i for each $i \in [n]$, where $\boldsymbol{z}^i \in (\{0,1\}^\lambda)^{\ell_2}$ and $u^i \in \{0,1\}^\lambda$.

4. For each $i \in [n]$, P_i computes $\boldsymbol{\Phi}^i := \boldsymbol{b}^i \cdot \Delta_i \oplus \bigoplus_{j\neq i} (\mathsf{K}_i[\boldsymbol{b}^j] \oplus \mathsf{M}_j[\boldsymbol{b}^i]) \oplus \boldsymbol{z}^i$.

5. For each $k \in [\ell_2]$, proceed as follows in parallel:

   (a) For each $i, j \in [n]$ and $j \neq i$, P_i computes

$$\begin{aligned} T_{i,j,k} &:= \mathtt{sid}\|\mathtt{ssid}\|i\|j\|k \in \{0,1\}^\lambda, \\ \mathsf{K}_i[a_k^j]_{\Phi_k^i} &:= \mathsf{H}(\mathsf{K}_i[a_k^j], T_{i,j,k}) \in \{0,1\}^\lambda, \\ u_k^{i,j} &:= \mathsf{K}_i[a_k^j]_{\Phi_k^i} \oplus \mathsf{H}(\mathsf{K}_i[a_k^j] \oplus \Delta_i, T_{i,j,k}) \oplus \Phi_k^i \in \{0,1\}^\lambda, \end{aligned}$$

   sends $u_k^{i,j}$ to P_j, and computes $\mathsf{M}_j[a_k^i]_{\Phi_k^j} := a_k^i \cdot u_k^{j,i} \oplus \mathsf{H}(\mathsf{M}_j[a_k^i], T_{j,i,k})$.

   (b) For each $i \in [n]$, P_i computes

$$\begin{aligned} s_k^i := a_k^i \cdot \Phi_k^i \oplus \bigoplus_{j\neq i} \left(\mathsf{K}_i[a_k^j]_{\Phi_k^i} \oplus \mathsf{M}_j[a_k^i]_{\Phi_k^j}\right) \oplus r_k^i \cdot \Delta_i \\ \oplus \bigoplus_{j\neq i} \left(\mathsf{K}_i[r_k^j] \oplus \mathsf{M}_j[r_k^i]\right) \in \{0,1\}^\lambda, \end{aligned}$$

   and $d_k^i := \mathsf{lsb}(s_k^i) \in \{0,1\}$.

Figure 5: The protocol of leaky authenticated AND triples (Part I).

functionality. Therefore, under a natural lazy sampling property of the ideal primitive simulator (which is satisfied by our constructions), our proof strategy is agonistic to the exact ideal primitive underlying the hash construction. We defer the detailed proofs, along with the definition of lazy sampling, to Appendix F.3.

Theorem 2. *Given a hash function* $\mathsf{H}^{\mathbf{Prim}}$*, which invokes* **Prim** c_{H} *times, such that an* $(\frac{n^2\ell_2}{4}, q_P + c_{\mathsf{H}} \cdot \frac{n^2\ell_2}{4}, n, n-1, 1)$*-bounded distinguisher has advantage at most* ε_{H} *in the S-mTCCRL game (Definition 3) for at most* n *users and* $(\lambda-1)$*-bit min-entropy per key, and satisfies the* $\varepsilon_{\mathsf{lazy}}$*-lazy* **Prim** *simulation, and a family* $\mathcal{H} = \{\mathsf{H}' : (\{0,1\}^\lambda)^{\ell_2} \to \{0,1\}^\lambda\}$ *of linear* ε*-almost 1-universal*

Protocol Π_{LaAND}

6. For each $i \in [n]$, P_i sends $(\mathsf{Commit}, \mathtt{sid}, \mathtt{ssid} \| 0, i, \boldsymbol{d}^i)$ to $\mathcal{F}_{\mathsf{Com}}$.
7. For each $i \in [n]$, after receiving $(\mathsf{Committed}, \mathtt{sid}, \mathtt{ssid} \| 0, j)$ from $\mathcal{F}_{\mathsf{Com}}$ for all $j \in [n]$ and $j \neq i$, P_i sends $(\mathsf{Open}, \mathtt{sid}, \mathtt{ssid} \| 0, i)$ to $\mathcal{F}_{\mathsf{Com}}$, which outputs $(\mathsf{Opened}, \mathtt{sid}, \mathtt{ssid} \| 0, i, \boldsymbol{d}^i)$ to all parties, and computes $\boldsymbol{d} := \bigoplus_{i \in [n]} \boldsymbol{d}^i \in \{0,1\}^{\ell_2}$ and $\boldsymbol{t}^i := \boldsymbol{s}^i \oplus \boldsymbol{d} \cdot \Delta_i \in (\{0,1\}^\lambda)^{\ell_2}$.
8. All parties send $(\mathsf{Rand}, \mathtt{sid}, \mathtt{ssid}, \mathcal{H})$ to $\mathcal{F}_{\mathsf{Rand}}$, which outputs $(\mathsf{Res}, \mathtt{sid}, \mathtt{ssid}, \mathsf{H}')$ to all parties.
9. For each $i \in [n]$, P_i computes $\alpha^i := \mathsf{H}'(\boldsymbol{t}^i) + u^i$ and sends $(\mathsf{Commit}, \mathtt{sid}, \mathtt{ssid} \| 1, i, \alpha^i)$ to $\mathcal{F}_{\mathsf{Com}}$.
10. For each $i \in [n]$, after receiving $(\mathsf{Committed}, \mathtt{sid}, \mathtt{ssid} \| 1, j)$ from $\mathcal{F}_{\mathsf{Com}}$ for all $j \in [n]$ and $j \neq i$, P_i sends $(\mathsf{Open}, \mathtt{sid}, \mathtt{ssid} \| 1, i)$ to $\mathcal{F}_{\mathsf{Com}}$, which outputs $(\mathsf{Opened}, \mathtt{sid}, \mathtt{ssid} \| 1, i, \alpha^i)$ to all parties, and checks if $\bigoplus_{i \in [n]} \alpha^i = 0$. If this check fails, P_i aborts; otherwise, P_i does nothing.
11. All parties compute $\langle\langle \boldsymbol{c} \rangle\rangle := \langle\langle \boldsymbol{r} \rangle\rangle \oplus \boldsymbol{d}$.
12. For each $i \in [n]$, P_i outputs $(\mathsf{Out}, \mathtt{sid}, \mathtt{ssid}, i, \Delta_i, \langle\langle \boldsymbol{x} \rangle\rangle_i, \langle\langle \boldsymbol{a} \rangle\rangle_i, \langle\langle \boldsymbol{b} \rangle\rangle_i, \langle\langle \boldsymbol{c} \rangle\rangle_i)$.

Figure 6: The protocol of leaky authenticated AND triples (Part II).

hash functions, protocol Π_{LaAND} *(Figs. 5 and 6) UC-realizes functionality* $\mathcal{F}_{\mathsf{LaAND}}$ *(Figs. 3 and 4) in the* $(\mathcal{F}_{\mathsf{aShare}}, \mathcal{F}_{\mathsf{Zero}}, \mathcal{F}_{\mathsf{Com}}, \mathcal{F}_{\mathsf{Rand}})$*-hybrid model with an advantage of at most*

$$\varepsilon_{\mathsf{H}} + 2\varepsilon + \frac{5}{2^\lambda} + \varepsilon_{\mathsf{lazy}}$$

against any adversary making at most q_P **Prim***-queries, when the command* $(\mathsf{Gen}, \mathtt{sid}, \ldots)$ *is invoked only once per* `sid`*.*

References

[ABD+13] Elena Andreeva, Andrey Bogdanov, Yevgeniy Dodis, Bart Mennink, and John P. Steinberger. On the indifferentiability of key-alternating ciphers. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part I*, volume 8042 of *LNCS*, pages 531–550. Springer, Berlin, Heidelberg, August 2013.

[ABF+17] Toshinori Araki, Assi Barak, Jun Furukawa, Tamar Lichter, Yehuda Lindell, Ariel Nof, Kazuma Ohara, Adi Watzman, and Or Weinstein. Optimized honest-majority MPC for malicious adversaries - breaking the 1 billion-gate per second barrier. In *2017 IEEE Symposium on Security and Privacy*, pages 843–862. IEEE Computer Society Press, May 2017.

[ALSZ13] Gilad Asharov, Yehuda Lindell, Thomas Schneider, and Michael Zohner. More efficient oblivious transfer and extensions for faster secure computation. In Ahmad-Reza Sadeghi, Virgil D. Gligor, and Moti Yung, editors, *ACM CCS 2013*, pages 535–548. ACM Press, November 2013.

[BBM00] Mihir Bellare, Alexandra Boldyreva, and Silvio Micali. Public-key encryption in a multi-user setting: Security proofs and improvements. In Bart Preneel, editor, *EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 259–274. Springer, Berlin, Heidelberg, May 2000.

[BBT16] Mihir Bellare, Daniel J. Bernstein, and Stefano Tessaro. Hash-function based PRFs: AMAC and its multi-user security. In Marc Fischlin and Jean-Sébastien Coron, editors, *EUROCRYPT 2016, Part I*, volume 9665 of *LNCS*, pages 566–595. Springer, Berlin, Heidelberg, May 2016.

[BCG+19a] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 291–308. ACM Press, November 2019.

[BCG+19b] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, and Peter Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part III*, volume 11694 of *LNCS*, pages 489–518. Springer, Cham, August 2019.

[BCG+22] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Nicolas Resch, and Peter Scholl. Correlated pseudorandomness from expand-accumulate codes. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 603–633. Springer, Cham, August 2022.

[BHK13] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Instantiating random oracles via UCEs. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 398–415. Springer, Berlin, Heidelberg, August 2013.

[BHK14] Mihir Bellare, Viet Tung Hoang, and Sriram Keelveedhi. Cryptography from compression functions: The UCE bridge to the ROM. In Juan A. Garay and Rosario Gennaro, editors, *CRYPTO 2014, Part I*, volume 8616 of *LNCS*, pages 169–187. Springer, Berlin, Heidelberg, August 2014.

[BHKR13] Mihir Bellare, Viet Tung Hoang, Sriram Keelveedhi, and Phillip Rogaway. Efficient garbling from a fixed-key blockcipher. In *2013 IEEE Symposium on Security and Privacy*, pages 478–492. IEEE Computer Society Press, May 2013.

[BLN+21] Sai Sheshank Burra, Enrique Larraia, Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, Emmanuela Orsini, Peter Scholl, and Nigel P. Smart. High-performance multi-party computation for binary circuits based on oblivious transfer. *Journal of Cryptology*, 34(3):34, July 2021.

[BN18] Srimanta Bhattacharya and Mridul Nandi. Full indifferentiable security of the xor of two or more random permutations using the χ^2 method. In Jesper Buus Nielsen and Vincent Rijmen, editors, *EUROCRYPT 2018, Part I*, volume 10820 of *LNCS*, pages 387–412. Springer, Cham, April / May 2018.

[BRSS10] John Black, Phillip Rogaway, Thomas Shrimpton, and Martijn Stam. An analysis of the blockcipher-based hash functions from PGV. *Journal of Cryptology*, 23(4):519–545, October 2010.

[BT16] Mihir Bellare and Björn Tackmann. The multi-user security of authenticated encryption: AES-GCM in TLS 1.3. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 247–276. Springer, Berlin, Heidelberg, August 2016.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, January 2000.

[Can01] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *42nd FOCS*, pages 136–145. IEEE Computer Society Press, October 2001.

[CDMP05] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-Damgård revisited: How to construct a hash function. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 430–448. Springer, Berlin, Heidelberg, August 2005.

[CHK+16] Jean-Sébastien Coron, Thomas Holenstein, Robin Künzler, Jacques Patarin, Yannick Seurin, and Stefano Tessaro. How to build an ideal cipher: The indifferentiability of the Feistel construction. *Journal of Cryptology*, 29(1):61–114, January 2016.

[CKKZ12] Seung Geol Choi, Jonathan Katz, Ranjit Kumaresan, and Hong-Sheng Zhou. On the security of the "free-XOR" technique. In Ronald Cramer, editor, *TCC 2012*, volume 7194 of *LNCS*, pages 39–53. Springer, Berlin, Heidelberg, March 2012.

[CLL+18] Shan Chen, Rodolphe Lampe, Jooyoung Lee, Yannick Seurin, and John P. Steinberger. Minimizing the two-round Even-Mansour cipher. *Journal of Cryptology*, 31(4):1064–1119, October 2018.

[CLL19] Wonseok Choi, ByeongHak Lee, and Jooyoung Lee. Indifferentiability of truncated random permutations. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 175–195. Springer, Cham, December 2019.

[CRR21] Geoffroy Couteau, Peter Rindal, and Srinivasan Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part III*, volume 12827 of *LNCS*, pages 502–534, Virtual Event, August 2021. Springer, Cham.

[CS14] Shan Chen and John P. Steinberger. Tight security bounds for key-alternating ciphers. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 327–350. Springer, Berlin, Heidelberg, May 2014.

[CS15] Benoit Cogliati and Yannick Seurin. On the provable security of the iterated Even-Mansour cipher against related-key and chosen-key attacks. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part I*, volume 9056 of *LNCS*, pages 584–613. Springer, Berlin, Heidelberg, April 2015.

[CT21] Yu Long Chen and Stefano Tessaro. Better security-efficiency trade-offs in permutation-based two-party computation. In Mehdi Tibouchi and Huaxiong Wang, editors, *ASIACRYPT 2021, Part II*, volume 13091 of *LNCS*, pages 275–304. Springer, Cham, December 2021.

[Dia22] Benjamin E. Diamond. On the security of KOS. Cryptology ePrint Archive, Report 2022/1371, 2022.

[DPSZ12] Ivan Damgård, Valerio Pastro, Nigel P. Smart, and Sarah Zakarias. Multiparty computation from somewhat homomorphic encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 643–662. Springer, Berlin, Heidelberg, August 2012.

[GBJ+23] Aldo Gunsing, Ritam Bhaumik, Ashwin Jha, Bart Mennink, and Yaobin Shen. Revisiting the indifferentiability of the sum of permutations. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part III*, volume 14083 of *LNCS*, pages 628–660. Springer, Cham, August 2023.

[GBK16] Praveen Gauravaram, Nasour Bagheri, and Lars R. Knudsen. Building indifferentiable compression functions from the PGV compression functions. *Des. Codes Cryptogr.*, 78(2):547–581, 2016.

[GGM84] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984.

[GKW+20] Chun Guo, Jonathan Katz, Xiao Wang, Chenkai Weng, and Yu Yu. Better concrete security for half-gates garbling (in the multi-instance setting). In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 793–822. Springer, Cham, August 2020.

[GKWY20] Chun Guo, Jonathan Katz, Xiao Wang, and Yu Yu. Efficient and secure multiparty computation from fixed-key block ciphers. In *2020 IEEE Symposium on Security and Privacy*, pages 825–841. IEEE Computer Society Press, May 2020.

[GWYY25] Chun Guo, Xiao Wang, Kang Yang, and Yu Yu. On tweakable correlation robust hashing against key leakages. *Designs, Codes and Cryptography*, pages 1–38, 2025. Eprint version: https://eprint.iacr.org/2024/163.

[GYW+23] Xiaojie Guo, Kang Yang, Xiao Wang, Wenhao Zhang, Xiang Xie, Jiang Zhang, and Zheli Liu. Half-tree: Halving the cost of tree expansion in COT and DPF. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part I*, volume 14004 of *LNCS*, pages 330–362. Springer, Cham, April 2023.

[HSS17] Carmit Hazay, Peter Scholl, and Eduardo Soria-Vazquez. Low cost constant round MPC combining BMR and oblivious transfer. In Tsuyoshi Takagi and Thomas Peyrin, editors, *ASIACRYPT 2017, Part I*, volume 10624 of *LNCS*, pages 598–628. Springer, Cham, December 2017.

[HSS20] Carmit Hazay, Peter Scholl, and Eduardo Soria-Vazquez. Low cost constant round MPC combining BMR and oblivious transfer. *Journal of Cryptology*, 33(4):1732–1786, October 2020.

[HT16] Viet Tung Hoang and Stefano Tessaro. Key-alternating ciphers and key-length extension: Exact bounds and multi-user security. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 3–32. Springer, Berlin, Heidelberg, August 2016.

[IKNP03] Yuval Ishai, Joe Kilian, Kobbi Nissim, and Erez Petrank. Extending oblivious transfers efficiently. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 145–161. Springer, Berlin, Heidelberg, August 2003.

[KOS15] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 724–741. Springer, Berlin, Heidelberg, August 2015.

[KOS16] Marcel Keller, Emmanuela Orsini, and Peter Scholl. MASCOT: Faster malicious arithmetic secure computation with oblivious transfer. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 830–842. ACM Press, October 2016.

[KRRW18] Jonathan Katz, Samuel Ranellucci, Mike Rosulek, and Xiao Wang. Optimizing authenticated garbling for faster secure two-party computation. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 365–391. Springer, Cham, August 2018.

[KS08] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008, Part II*, volume 5126 of *LNCS*, pages 486–498. Springer, Berlin, Heidelberg, July 2008.

[LSS17] Jooyoung Lee, Martijn Stam, and John P. Steinberger. The security of Tandem-DM in the ideal cipher model. *Journal of Cryptology*, 30(2):495–518, April 2017.

[Mau02] Ueli M. Maurer. Indistinguishability of random systems. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 110–132. Springer, Berlin, Heidelberg, April / May 2002.

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 681–700. Springer, Berlin, Heidelberg, August 2012.

[OOS17] Michele Orrù, Emmanuela Orsini, and Peter Scholl. Actively secure 1-out-of-N OT extension with application to private set intersection. In Helena Handschuh, editor, *CT-RSA 2017*, volume 10159 of *LNCS*, pages 381–396. Springer, Cham, February 2017.

[Roy22] Lawrence Roy. SoftSpokenOT: Quieter OT extension from small-field silent VOLE in the minicrypt model. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 657–687. Springer, Cham, August 2022.

[RR21] Mike Rosulek and Lawrence Roy. Three halves make a whole? Beating the half-gates lower bound for garbled circuits. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 94–124, Virtual Event, August 2021. Springer, Cham.

[RRT23] Srinivasan Raghuraman, Peter Rindal, and Titouan Tanguy. Expand-convolute codes for pseudorandom correlation generators from LPN. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part IV*, volume 14084 of *LNCS*, pages 602–632. Springer, Cham, August 2023.

[RSS11] Thomas Ristenpart, Hovav Shacham, and Thomas Shrimpton. Careful with composition: Limitations of the indifferentiability framework. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 487–506. Springer, Berlin, Heidelberg, May 2011.

[Sch18] Peter Scholl. Extending oblivious transfer with low communication via key-homomorphic PRFs. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018, Part I*, volume 10769 of *LNCS*, pages 554–583. Springer, Cham, March 2018.

[WRK17a] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Authenticated garbling and efficient maliciously secure two-party computation. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 21–37. ACM Press, October / November 2017.

[WRK17b] Xiao Wang, Samuel Ranellucci, and Jonathan Katz. Global-scale secure multiparty computation. In Bhavani M. Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *ACM CCS 2017*, pages 39–56. ACM Press, October / November 2017.

[WYKW21] Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *2021 IEEE Symposium on Security and Privacy*, pages 1074–1091. IEEE Computer Society Press, May 2021.

[YWL+20] Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. Ferret: Fast extension for correlated OT with small communication. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1607–1626. ACM Press, November 2020.

[YWZ20] Kang Yang, Xiao Wang, and Jiang Zhang. More efficient MPC from improved triple generation and authenticated garbling. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1627–1646. ACM Press, November 2020.

[ZCSH18] Ruiyu Zhu, Darion Cassel, Amr Sabry, and Yan Huang. NANOPI: Extreme-scale actively-secure multi-party computation. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 862–879. ACM Press, October 2018.

[ZGY+24] Wenhao Zhang, Xiaojie Guo, Kang Yang, Ruiyu Zhu, Yu Yu, and Xiao Wang. Efficient actively secure DPF and RAM-based 2PC with one-bit leakage. In *2024 IEEE Symposium on Security and Privacy*, pages 561–577. IEEE Computer Society Press, May 2024.

[ZRE15] Samee Zahur, Mike Rosulek, and David Evans. Two halves make a whole - reducing data transfer in garbled circuits using half gates. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, volume 9057 of *LNCS*, pages 220–250. Springer, Berlin, Heidelberg, April 2015.

A Deferred Security Notions

A.1 Formal Definitions of the Basic Notions CR and S-CR

As mentioned in Sect. 2.2, the model sketched in the Introduction (Sect. 1.1) is the "most basic" S-CR, i.e., the simulator-based variant of the plain correlation robust model CR. Here we serve their formal definitions.

First, the CR advantage concerns with the (single-keyed) oracle $\mathsf{K}_{\Delta}^{\mathsf{H}^{\mathbf{Prim}}}(\cdot) := \mathsf{H}^{\mathbf{Prim}}(\Delta \oplus \cdot)$ defined on a single-input hash $\mathsf{H}^{\mathbf{Prim}}$, and is defined as the advantage of distinguishing $(\mathbf{Prim}, \mathsf{K}_{\Delta}^{\mathsf{H}^{\mathbf{Prim}}})$ from $(\mathbf{Prim}, \mathbf{F})$.

Definition 5 (CR advantage)**.** *Given a single-input function $\mathsf{H}^{\mathbf{Prim}} : \mathcal{W} \to \mathcal{W}$, a key-set $\mathcal{R} \subseteq \mathcal{W}$, and a distinguisher $\mathcal{D}$, define*

$$\mathbf{Adv}^{\mathrm{CR}}_{\mathsf{H}^{\mathbf{Prim}},\mathcal{R}}(\mathcal{D}) := \left| \Pr_{\mathbf{Prim} \xleftarrow{\$} \mathcal{P}, \Delta \xleftarrow{\$} \mathcal{R}} \left[\mathcal{D}^{\mathbf{Prim}, \mathsf{K}_{\Delta}^{\mathsf{H}^{\mathbf{Prim}}}} = 1\right] - \Pr_{\mathbf{Prim} \xleftarrow{\$} \mathcal{P}, \mathbf{F} \xleftarrow{\$} \mathcal{F}_{\mathcal{W},\mathcal{W}}} \left[\mathcal{D}^{\mathbf{Prim},\mathbf{F}} = 1\right] \right|.$$

The S-CR notion is the simulation-based extension of CR. Its experiments (see Fig. 7 for a formal description) are obtained by making the following modifications to Fig. 1 (S-mTCCRL experiments):

(i) replacing the key vector $\boldsymbol{\Delta}$ with a single key Δ;

(ii) replacing the oracle $\mathbf{F} \xleftarrow{\$} \mathcal{KWC}$ with $\mathbf{F} \xleftarrow{\$} \mathcal{F}_{\mathcal{X},\mathcal{W}}$;

(iii) replacing the oracle $\mathsf{mTCK}^{\mathsf{H}}_{\boldsymbol{\Delta}}$ with $\mathsf{K}_{\Delta}^{\mathsf{H}^{\mathbf{Prim}}}$;

(iv) dropping the key leaking oracle $\mathsf{mL}_{\boldsymbol{\Delta}}$;

(v) replacing the multi-key guessing oracle $\mathsf{GlobalK}_{\boldsymbol{\Delta}}$ of the simulator with the single-key guessing oracle $\mathsf{GlobalK}_{\Delta}$.

Definition 6 (S-CR advantage)**.** *Given a single-input function $\mathsf{H}^{\mathbf{Prim}} : \mathcal{W} \to \mathcal{W}$, a key-set $\mathcal{R} \subseteq \mathcal{W}$, a distinguisher $\mathcal{D}$ and a simulator $\mathcal{S}^{\mathsf{GlobalK}_{\Delta}}$ providing an interface O_1, define*

$$\mathbf{Adv}^{\text{S-CR}}_{\mathsf{H}^{\mathbf{Prim}},\mathcal{R}}(\mathcal{D},\mathcal{S}) := \left| \Pr\left[\mathbf{Exp}^{\text{S-CR}}_{\mathsf{H}^{\mathbf{Prim}},\mathcal{R},\mathcal{D},\mathcal{S},0} = 1\right] - \Pr\left[\mathbf{Exp}^{\text{S-CR}}_{\mathsf{H}^{\mathbf{Prim}},\mathcal{R},\mathcal{D},\mathcal{S},1} = 1\right] \right|.$$

Asymptotically, if there exists a simulator $\mathcal{S}$ such that for any distinguisher $\mathcal{D}$ $\mathbf{Adv}^{\text{S-CR}}_{\mathsf{H}^{\mathbf{Prim}},\mathcal{R}}(\mathcal{D},\mathcal{S})$ is negligibly small, then $\mathsf{H}^{\mathbf{Prim}}$ is S-CR-secure.

A.2 Zoo of weaker variants of mTCCRL and S-mTCCRL

As demonstrated in previous works [GKWY20, GKW+20, CT21], hash constructions aiming at weaker variants of mTCCRL may achieve better efficiency. E.g., "non-tweakable" (single-input) constructions may utilize the more efficient fixed-key blockcipher [GKWY20], while constructions without circular security may not need the linear orthomorphism σ [GKWY20]. Therefore, in this section we put forward weaker variants of mTCCRL and S-mTCCRL without "tweakable"

The S-CR experiment $\mathbf{Exp}^{\text{S-CR}}_{\mathsf{H}^{\mathbf{Prim}},\mathcal{R},\mathcal{D},\mathcal{S},\beta}$:

1. **Initialization:**
   - If $\beta = 0$: sample an ideal primitive $\mathbf{Prim} \xleftarrow{\$} \mathcal{P}$ and a (single) key $\Delta \xleftarrow{\$} \mathcal{R}$;
   - Else, i.e., $\beta = 1$: sample a random function $\mathbf{F} \xleftarrow{\$} \mathcal{F}_{\mathcal{W},\mathcal{W}}$ and a key $\Delta \xleftarrow{\$} \mathcal{R}$.
2. **Phase 1:**
   - If $\beta = 0$: run $end \leftarrow \mathcal{D}^{\mathbf{Prim},\mathsf{K}^{\mathsf{H}^{\mathbf{Prim}}}_{\Delta}}$ with $\mathbf{Prim}$ and the (single-user) key-whitened oracle $\mathsf{K}^{\mathsf{H}^{\mathbf{Prim}}}_{\Delta}$, till $\mathcal{D}$ outputs end and ends phase 1;
   - Else, i.e., $\beta = 1$: run $end \leftarrow \mathcal{D}^{\mathcal{S}^{\mathsf{GlobalK}_{\Delta}}.O_1,\mathbf{F}}$ with the simulator's interface $\mathcal{S}^{\mathsf{GlobalK}_{\Delta}}.O_1$ and $\mathbf{F}$, till $\mathcal{D}$ outputs end indicating the end of phase 1;
3. **Key revealing:** Give $\mathcal{D}$ the key Δ used in phase 1. If $\beta = 1$ then invoke $\mathcal{S}^{\mathsf{GlobalK}_{\Delta}}.PhaseSwitch(\mathcal{Q}_F)$, where $\mathcal{Q}_F$ is the transcript of queries and answers of $\mathcal{D}$ to $\mathbf{F}$ in the first phase.
4. **Phase 2:**
   - If $\beta = 0$: run $\beta' \leftarrow \mathcal{D}^{\mathbf{Prim}}$ with $\mathbf{Prim}$ and obtain its output β';
   - Else, i.e., $\beta = 1$: run $\beta' \leftarrow \mathcal{D}^{\mathcal{S}^{\mathsf{GlobalK}_{\Delta}}.O_1}$ with the simulator's interface and obtain its output β'.
5. **Finalization:** Output β' (the output of $\mathcal{D}$).

Figure 7: Experiments for the S-CR notion.

or/and circularity to complete the picture of different flavors of multi-user CR and S-CR security definitions. For this, we first introduce the non-tweakable and non-circular key-whitened oracles. We then formalize the weaker variants of mTCCRL and S-mTCCRL respectively. We finally give a brief discussion on their relations.

A.2.1 Key-whitened oracles without circularity or tweaks.

Recall that the multi-user tweakable circular key-whitened oracle $\mathsf{mTCK}^{\mathsf{H}^{\mathbf{Prim}}}_{\mathbf{\Delta}}(\mathsf{id}, w, i, b)$ introduced in Sect. 3.1 was defined on a *double-input* hash function $\mathsf{H}^{\mathbf{Prim}} : \mathcal{W} \times \mathcal{I} \to \mathcal{W}$. Still consider such a (double-input) hash and a vector of u secret keys $\mathbf{\Delta} = (R_1, ..., R_u)$ of u users, the *multi-user tweakable key-whitened oracle* is defined as

$$\mathsf{mTK}^{\mathsf{H}^{\mathbf{Prim}}}_{\mathbf{\Delta}}(\mathsf{id}, w, i) := \mathsf{H}^{\mathbf{Prim}}(w \oplus \Delta_{\mathsf{id}}, i). \tag{8}$$

The non-tweakable key-whitened oracles, on the other hand, are defined on a *single-input* hash function $\mathsf{H}^{\mathbf{Prim}} : \mathcal{W} \to \mathcal{W}$. Given such a hash and a vector of u secret keys $\mathbf{\Delta} = (R_1, ..., R_u)$ of u users, the *multi-user key-whitened oracle* is defined as

$$\mathsf{mK}^{\mathsf{H}^{\mathbf{Prim}}}_{\mathbf{\Delta}}(\mathsf{id}, w) := \mathsf{H}^{\mathbf{Prim}}(w \oplus \Delta_{\mathsf{id}}), \tag{9}$$

while the *multi-user circular key-whitened oracle* is defined as

$$\mathsf{mCK}^{\mathsf{H}^{\mathbf{Prim}}}_{\mathbf{\Delta}}(\mathsf{id}, w, b) := \mathsf{H}^{\mathbf{Prim}}(w \oplus \Delta_{\mathsf{id}}) \oplus b \cdot \Delta_{\mathsf{id}}. \tag{10}$$

A.2.2 Multi-user CR definition zoo.

The mTCCRL security notion is given in Definition 2, Sect. 3.1. Br replacing the multi-user tweakable circular key-whitened oracle $\mathsf{mTCK}^{\mathsf{H}^{\mathbf{Prim}}}_{\mathbf{\Delta}}$ with the above non-circular $\mathsf{mTK}^{\mathsf{H}^{\mathbf{Prim}}}_{\mathbf{\Delta}}$ or non-

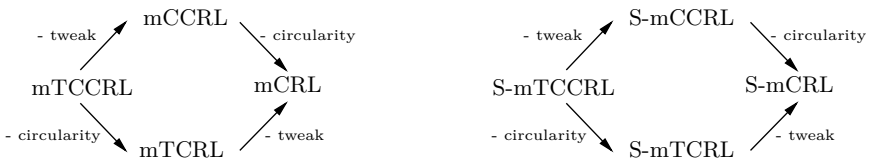


Figure 8: Multi-user correlation robust (CR) and S-CR security notions with key leakages and various combinations of T (tweakable) and C (circularity). (Left) Variants of classical CR notions; (Right) Variants of our new S-CR notions for UC security. In each rectangle, the notion at left is the strongest, while the notion at right is the weakest. Each arrow indicates that removing one feature (marked by the labels "$-$ X", X $\in$ {tweak, circularity}) from a stronger notion yields a corresponding weaker notion. Consequently, each arrow also indicates a (trivial) implication.

tweakable variants $\mathsf{mK}_{\boldsymbol{\Delta}}^{\mathsf{H}^{\mathbf{Prim}}}$ and $\mathsf{mCK}_{\boldsymbol{\Delta}}^{\mathsf{H}^{\mathbf{Prim}}}(\mathsf{id}, w, i)$, we naturally derive other multi-user CRL notions. This leads us to consider 4 different notions denoted as "mu$X$CRL" with $X \in \{\bot, \mathrm{C}, \mathrm{T}, \mathrm{TC}\}$: $\bot$ is the empty string, the "T" in X indicates *tweakable*, and the additional "C" in X indicates *circular*. The suffix L always exists, indicating that we always consider the presence of the key leaking oracle $\mathsf{mL}_{\boldsymbol{\Delta}}$.

The 4 notions are obtained by starting from mTCCRL and by removing one of the two features (tweakable, circularity) at a time, as shown in Fig. 8 (Left). It may be tempting to give a single unified definition for all of them. However, such a huge definition would be unclear. We thereby given them in turn in the subsequent three paragraphs.

mTCRL: removing circularity. We first remove circularity from mTCCRL, and this yields mTCRL.

Definition 7 (mTCRL advantage)**.** *Given a double-input function* $\mathsf{H}^{\mathbf{Prim}} : \mathcal{W} \times \mathcal{I} \to \mathcal{W}$*, a tuple of key sets* $\overrightarrow{\mathcal{R}} = \mathcal{R}_1 \times ... \times \mathcal{R}_u \subseteq \mathcal{W}^u$*, a set* $\mathcal{P}$ *of "allowed" predicates and a distinguisher* $\mathcal{D}$*, define*

$$\mathbf{Adv}^{\mathrm{mTCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{D}) := \Big|\mathrm{Pr}_{\mathbf{Prim}\xleftarrow{\$}\mathcal{P},\boldsymbol{\Delta}\xleftarrow{\$}\overrightarrow{\mathcal{R}}}\left[\mathcal{D}^{\mathbf{Prim},\mathsf{mTK}_{\boldsymbol{\Delta}}^{\mathsf{H}^{\mathbf{Prim}}},\mathsf{mL}_{\boldsymbol{\Delta}}} = 1\right] - \mathrm{Pr}_{\mathbf{F}\xleftarrow{\$}\mathcal{F}_{[u]\times\mathcal{W}\times\mathcal{I},\mathcal{W}},\mathbf{Prim}\xleftarrow{\$}\mathcal{P},\boldsymbol{\Delta}\xleftarrow{\$}\overrightarrow{\mathcal{R}}}\left[\mathcal{D}^{\mathbf{Prim},\mathbf{F},\mathsf{mL}_{\boldsymbol{\Delta}}} = 1\right]\Big|.$$

We require that every query (id, P) *of* $\mathcal{D}$ *to its third oracle* $\mathsf{mL}_{\boldsymbol{\Delta}}$ *has* $P \in \mathcal{P}$*.*

mCCRL: removing tweak. We then remove "tweakable" from mTCCRL, and this yields mCCRL. Note that these "non-tweakable" notions concern with single-input functions $\mathsf{H}^{\mathbf{Prim}} : \mathcal{W} \to \mathcal{W}$.

Definition 8 (mCCRL advantages)**.** *Given a single-input function* $\mathsf{H}^{\mathbf{Prim}} : \mathcal{W} \to \mathcal{W}$*, a tuple of key sets* $\overrightarrow{\mathcal{R}} = \mathcal{R}_1 \times ... \times \mathcal{R}_u \subseteq \mathcal{W}^u$*, a set* $\mathcal{P}$ *of "allowed" predicates and a distinguisher* $\mathcal{D}$*, define*

$$\mathbf{Adv}^{\mathrm{mCCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{D}) := \Big|\mathrm{Pr}_{\mathbf{Prim}\xleftarrow{\$}\mathcal{P},\boldsymbol{\Delta}\xleftarrow{\$}\overrightarrow{\mathcal{R}}}\left[\mathcal{D}^{\mathbf{Prim},\mathsf{mCK}_{\boldsymbol{\Delta}}^{\mathsf{H}^{\mathbf{Prim}}},\mathsf{mL}_{\boldsymbol{\Delta}}} = 1\right] - \mathrm{Pr}_{\mathbf{F}\xleftarrow{\$}\mathcal{F}_{[u]\times\mathcal{W}\times\{0,1\},\mathcal{W}},\mathbf{Prim}\xleftarrow{\$}\mathcal{P},\boldsymbol{\Delta}\xleftarrow{\$}\overrightarrow{\mathcal{R}}}\left[\mathcal{D}^{\mathbf{Prim},\mathbf{F},\mathsf{mL}_{\boldsymbol{\Delta}}} = 1\right]\Big|.$$

We require that: (i) $\mathcal{D}$ *never queries both* $(\mathsf{id}, w, 0)$ *and* $(\mathsf{id}, w, 1)$ *to its second oracle (for any* id, w*); (ii) Every query* (id, P) *of* $\mathcal{D}$ *to its third oracle* $\mathsf{mL}_{\boldsymbol{\Delta}}$ *has* $P \in \mathcal{P}$*.*

mCRL: most basic. Finally, we remove both circularity and "tweakable" from mTCCRL to yield mCRL.

Definition 9 (mCRL advantages). *Given a single-input function* $\mathsf{H}^{\mathbf{Prim}} : \mathcal{W} \to \mathcal{W}$*, a tuple of key sets* $\overrightarrow{\mathcal{R}} = \mathcal{R}_1 \times ... \times \mathcal{R}_u \subseteq \mathcal{W}^u$*, a set* $\mathcal{P}$ *of "allowed" predicates and a distinguisher* $\mathcal{D}$*, define*

$$\mathbf{Adv}^{\mathrm{mCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{D}) := \Big|\Pr_{\mathbf{Prim}\xleftarrow{\$}\mathcal{P},\mathbf{\Delta}\xleftarrow{\$}\overrightarrow{\mathcal{R}}}\left[\mathcal{D}^{\mathbf{Prim},\mathsf{mK}^{\mathsf{H}^{\mathbf{Prim}}}_{\mathbf{\Delta}},\mathsf{mL}_{\mathbf{\Delta}}}=1\right] - \Pr_{\mathbf{F}\xleftarrow{\$}\mathcal{F}_{[u]\times\mathcal{W},\mathcal{W}},\mathbf{Prim}\xleftarrow{\$}\mathcal{P},\mathbf{\Delta}\xleftarrow{\$}\overrightarrow{\mathcal{R}}}\left[\mathcal{D}^{\mathbf{Prim},\mathbf{F},\mathsf{mL}_{\mathbf{\Delta}}}=1\right]\Big|.$$

We require that every query (id, P) *of* $\mathcal{D}$ *to its third oracle* $\mathsf{mL}_{\mathbf{\Delta}}$ *has* $P \in \mathcal{P}$*.*

A.2.3 Multi-user S-CR definition zoo.

In a similar vein to the above discussions on multi-user CR definitions, we can derive a multi-user S-CR definition zoo. Concretely, the S-mTCCRL security notion is given in Definition 3, Sect. 4. By replacing the multi-user tweakable circular key-whitened oracle $\mathsf{mTCK}^{\mathsf{H}^{\mathbf{Prim}}}_{\mathbf{\Delta}}$ with the above non-circular $\mathsf{mTK}^{\mathsf{H}^{\mathbf{Prim}}}_{\mathbf{\Delta}}$ or non-tweakable variants $\mathsf{mK}^{\mathsf{H}^{\mathbf{Prim}}}_{\mathbf{\Delta}}$ and $\mathsf{mCK}^{\mathsf{H}^{\mathbf{Prim}}}_{\mathbf{\Delta}}(\mathsf{id}, w, i)$, we naturally derive other multi-user S-CRL notions. This leads us to consider 4 different notions denoted as "S-mXCRL" with $X \in \{\perp, \mathrm{C}, \mathrm{T}, \mathrm{TC}\}$.

The 4 notions are obtained by starting from S-mTCCRL and by removing one of the two features (tweakable, circularity) at a time, as shown in Fig. 8 (Right).

S-mTCRL: removing circularity. We first remove circularity from S-mTCCRL to yield S-mTCRL.

Definition 10 (S-mTCRL advantages). *Given a double-input function* $\mathsf{H}^{\mathbf{Prim}} : \mathcal{W} \times \mathcal{I} \to \mathcal{W}$*, a tuple of key sets* $\overrightarrow{\mathcal{R}} = \mathcal{R}_1 \times ... \times \mathcal{R}_u \subseteq \mathcal{W}^u$*, a set* $\mathcal{P}$ *of "allowed" predicates, a distinguisher* $\mathcal{D}$ *and a simulator* $\mathcal{S}^{\mathsf{GlobalK}_{\mathbf{\Delta}}}$ *providing an interface* O_1*, define*

$$\mathbf{Adv}^{\text{S-mTCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{D},\mathcal{S}) := \Big|\Pr\left[\mathbf{Exp}^{\text{S-mTCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u,\mathcal{D},\mathcal{S},0}=1\right] - \Pr\left[\mathbf{Exp}^{\text{S-mTCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u,\mathcal{D},\mathcal{S},1}=1\right]\Big|,$$

where experiment $\mathbf{Exp}^{\text{S-mTCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u,\mathcal{D},\mathcal{S},\beta}$ *is obtained by replacing* $\mathsf{mTCK}^{\mathsf{H}^{\mathbf{Prim}}}_{\mathbf{\Delta}}/\mathbf{F} \xleftarrow{\$} \mathcal{F}_{[u]\times\mathcal{W}\times\mathcal{I}\times\{0,1\},\mathcal{W}}$ *in* $\mathbf{Exp}^{\text{S-mTCCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u,\mathcal{D},\mathcal{S},\beta}$ *with the non-circular* $\mathsf{mTK}^{\mathsf{H}^{\mathbf{Prim}}}_{\mathbf{\Delta}}/\mathbf{F} \xleftarrow{\$} \mathcal{F}_{[u]\times\mathcal{W}\times\mathcal{I},\mathcal{W}}$*.*

We require that every query (id, P) *of* $\mathcal{D}$ *to its third oracle* $\mathsf{mL}_{\mathbf{\Delta}}$ *has* $P \in \mathcal{P}$*.*

S-mCCRL: removing tweak. We then remove "tweakable" from S-mTCCRL to yield S-mCCRL.

Definition 11 (S-mCCRL advantages). *Given a single-input function* $\mathsf{H}^{\mathbf{Prim}} : \mathcal{W} \to \mathcal{W}$*, a tuple of key sets* $\overrightarrow{\mathcal{R}} = \mathcal{R}_1 \times ... \times \mathcal{R}_u \subseteq \mathcal{W}^u$*, a set* $\mathcal{P}$ *of "allowed" predicates and a distinguisher* $\mathcal{D}$*, define*

$$\mathbf{Adv}^{\text{S-mCCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{D},\mathcal{S}) := \Big|\Pr\left[\mathbf{Exp}^{\text{S-mCCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u,\mathcal{D},\mathcal{S},0}=1\right] - \Pr\left[\mathbf{Exp}^{\text{S-mCCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u,\mathcal{D},\mathcal{S},1}=1\right]\Big|,$$

where experiment $\mathbf{Exp}^{\text{S-mCCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u,\mathcal{D},\mathcal{S},\beta}$ *is obtained by replacing* $\mathsf{mTCK}^{\mathsf{H}^{\mathbf{Prim}}}_{\mathbf{\Delta}}/\mathbf{F} \xleftarrow{\$} \mathcal{F}_{[u]\times\mathcal{W}\times\mathcal{I}\times\{0,1\},\mathcal{W}}$ *in experiment* $\mathbf{Exp}^{\text{S-mTCCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u,\mathcal{D},\mathcal{S},\beta}$ *with the "non-tweakable" oracles* $\mathsf{mCK}^{\mathsf{H}^{\mathbf{Prim}}}_{\mathbf{\Delta}}/\mathbf{F} \xleftarrow{\$} \mathcal{F}_{[u]\times\mathcal{W}\times\{0,1\},\mathcal{W}}$*.*

We require that: (i) $\mathcal{D}$ *never queries both* $(\mathsf{id}, w, 0)$ *and* $(\mathsf{id}, w, 1)$ *to its second oracle (for any* id, w*); (ii) Every query* (id, P) *of* $\mathcal{D}$ *to its third oracle* $\mathsf{mL}_{\mathbf{\Delta}}$ *has* $P \in \mathcal{P}$*.*

S-mCRL: most basic. Finally, we remove both circularity and "tweakable" from S-mTCCRL to yield S-mCRL.

Definition 12 (S-mCRL advantages). *Given a single-input function* $\mathsf{H}^{\mathbf{Prim}} : \mathcal{W} \to \mathcal{W}$*, a tuple of key sets* $\overrightarrow{\mathcal{R}} = \mathcal{R}_1 \times ... \times \mathcal{R}_u \subseteq \mathcal{W}^u$*, a set* $\mathcal{P}$ *of "allowed" predicates and a distinguisher* $\mathcal{D}$*, define*

$$\mathbf{Adv}^{\text{S-mCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{D},\mathcal{S}) := \Big|\Pr[\mathbf{Exp}^{\text{S-mCCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u,\mathcal{D},\mathcal{S},0} = 1] - \Pr[\mathbf{Exp}^{\text{S-mCCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u,\mathcal{D},\mathcal{S},1} = 1]\Big|,$$

where experiment $\mathbf{Exp}^{\text{S-mCCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u,\mathcal{D},\mathcal{S},\beta}$ *is obtained by replacing* $\mathsf{mTCK}^{\mathsf{H}^{\mathbf{Prim}}}_{\mathbf{\Delta}}/\mathbf{F} \xleftarrow{\$} \mathcal{F}_{[u]\times\mathcal{W}\times\mathcal{I}\times\{0,1\},\mathcal{W}}$ *in* $\mathbf{Exp}^{\text{S-mTCCRL}}_{\mathsf{H}^{\mathbf{Prim}},\overrightarrow{\mathcal{R}},\mathcal{P},u,\mathcal{D},\mathcal{S},\beta}$ *with* $\mathsf{mK}^{\mathsf{H}^{\mathbf{Prim}}}_{\mathbf{\Delta}}/\mathbf{F} \xleftarrow{\$} \mathcal{F}_{[u]\times\mathcal{W},\mathcal{W}}$.

We require that every query (id, P) *of* $\mathcal{D}$ *to its third oracle* $\mathsf{mL}_{\mathbf{\Delta}}$ *has* $P \in \mathcal{P}$.

A.2.4 Brief relations within the zoo.

Fig. 8 also presents a complete picture of the various variants of CR and S-CR notions mentioned in this paper. We refer to the caption of Fig. 8 for detailed elaborations. Non-trivial relations/separations between the notions, as well as whether the key leakage oracle $\mathsf{mL}_{\mathbf{\Delta}}$ does matter, are not among the main goal of this paper, and we leave them for future work.

B S-mTCCRL Hashing in the Random Oracle Model

As mentioned in Sect. 4, the public primitive **Prim** can be any type of random primitive oracle. In this section, we consider the case of $\mathbf{Prim} = \mathbf{RO} \in \mathcal{F}_{\mathcal{X},\mathcal{W}}$ being a random oracle and $\mathsf{H}^{\mathbf{RO}}(w,i) = \mathsf{ROH}^{\mathbf{RO}}(w,i) := \mathbf{RO}(f_{\mathsf{in}}(w,i))$, where $f_{\mathsf{in}} : \mathcal{W}\times\mathcal{I} \to \mathcal{X}$ can be any efficiently invertible bijection with $f^{-1}_{\mathsf{in}} : \mathcal{I}\times\mathcal{X} \to \mathcal{W}$ being its inverse. This includes the folklore constructions such as $f_{\mathsf{in}}(w,i) := w\|i$. We prove that $\mathsf{ROH}^{\mathbf{RO}}$ is S-mTCCRL secure and justifies the folklore constructions.

Concretely, we follow Guo et al. [GWYY25] and restrict key leaking oracle queries to *oracle-free* predicates. A predicate P is oracle-free, if the evaluation of $P(\Delta)$ does not query **RO** for all $\Delta \in \mathcal{W}$. Let $\mathcal{P}_{\mathsf{free}}$ be the set of all oracle-free predicates. With this, the S-mTCCRL security of $\mathsf{ROH}^{\mathbf{RO}}(w,i)$ is given in Theorem 3.

Theorem 3. *Let* $C := \min\{|\mathcal{R}_1|, ..., |\mathcal{R}_u|\}$*. If* $\mathbf{RO} \in \mathcal{F}_{\mathcal{X},\mathcal{W}}$ *is modeled as a random oracle, then there exists a simulator* $\mathcal{S}_{RO}$ *such that the S-mTCCRL advantage of* $\mathsf{ROH}^{\mathbf{RO}}(w,i) = \mathbf{RO}(f_{\mathsf{in}}(w,i))$ *for any* $(q_C, q_P, M, q_{L,max}, \mu)$*-bounded distinguisher* $\mathcal{D}$ *has upper bound*

$$\mathbf{Adv}^{\text{S-mTCCRL}}_{\mathsf{ROH}^{\mathbf{RO}},\overrightarrow{\mathcal{R}},\mathcal{P}_{\mathsf{free}},u}(\mathcal{D},\mathcal{S}_{RO}) \le \frac{(M+1)(q_{L,max}+1)}{C} \cdot \Big(\mu q_P + \frac{(\mu-1)\cdot q_C}{2}\Big). \tag{11}$$

In addition, $\mathcal{S}^{\mathsf{GlobalK}_{\mathbf{\Delta}}}_{RO}$ *makes at most* μq_P *queries to its oracle* $\mathsf{GlobalK}_{\mathbf{\Delta}}$ *and runs in time* $O(\mu q_P + q_C)$.

Since $\mathcal{R}_1, ..., \mathcal{R}_u \subseteq \mathcal{W}$, it holds $C \le |\mathcal{W}|$. By Theorem 3, the attack advantage is small as long as $q_P \ll C/(\mu M q_{L,max})$ (which roughly bounds time complexity) and $q_C \ll C/(\mu M q_{L,max})$ (which roughly bounds data complexity). This appears comparable with $\widehat{\mathsf{MMO}}^{\mathbf{E}}$ (see Sect. 2.3). However, random oracles enable adopting much larger set $\mathcal{W}$ and thus larger C, and the concrete security could be higher than the ideal cipher-based construction $\widehat{\mathsf{MMO}}^{\mathbf{E}}$.

The full proof is deferred to Appendix C, and we provide the core ideas below. The crux is to construct the simulator $\mathcal{S}_{RO}$. As mentioned in Sect. 4, $\mathcal{S}_{RO}$ shall provide an interface $O_1 : \mathcal{X} \to \mathcal{W}$ to mimic $\mathbf{RO} \in \mathcal{F}_{\mathcal{X},\mathcal{W}}$, and is consistent with $\mathbf{F}$ when necessary.

In phase 1, $\mathcal{S}_{RO}$ just replies $O_1(x)$ by lazy sampling. When *PhaseSwitch*$(\mathcal{Q}_F)$ is invoked, $\mathcal{S}_{RO}$ copies the records of $\mathcal{Q}_F$ into an internal set $\mathcal{Q}_{SF}$. Then in phase 2, for each new query $O_1(x)$ $\mathcal{S}_{RO}$ checks if there exists a record $(\mathsf{id}, i, w, b, z) \in \mathcal{Q}_{SF}$ that is "linked" to $O_1(x)$—formally, $x = f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i)$. If so, $\mathcal{S}_{RO}$ replies $O_1(x)$ with z to ensure consistency; otherwise, $\mathcal{S}_{RO}$ samples a response for $O_1(x)$. This checking requires information about the key Δ_{id}, to which the oracle $\mathsf{GlobalK}_{\mathbf{\Delta}}$ will help. Concretely, for each $(\mathsf{id}, i, w, b, z) \in \mathcal{Q}_{SF}$ $\mathcal{S}_{RO}$ computes the candidate key Δ'_{id} that satisfies $x = f_{\mathsf{in}}(w \oplus \Delta'_{\mathsf{id}}, i)$, and queries $\mathsf{GlobalK}_{\mathbf{\Delta}}(\mathsf{id}, \Delta'_{\mathsf{id}})$ to see if $\Delta'_{\mathsf{id}} = \Delta_{\mathsf{id}}$: if so, then $(\mathsf{id}, i, w, b, z)$ is the "right" record. The formal description of $\mathcal{S}_{RO}$ is in Appendix C.1.

The above simulation strategy fails in two cases:

(i) A query $O_1(x)$ replied in phase 1 turns out to "link" to some $(\mathsf{id}, i, w, b, z) \in \mathcal{Q}_{SF}$. Intuitively, this is unlikely, since $\mathcal{D}$ does not know keys $\mathbf{\Delta}$ in phase 1.

(ii) In phase 2, some query $O_1(x)$ "links" to (at least) two distinct records

$$(\mathsf{id}, i, w, b, z), (\mathsf{id}', i', w', b', z') \in \mathcal{Q}_{SF}\,,$$

(so that $\mathcal{S}_{RO}$ cannot ensure consistency with both). Due to the injectivity of f_{in}, this relates to a collision-type event among two independently sampled keys Δ_{id} and $\Delta_{\mathsf{id}'}$, which is unlikely.

The two cases are formalized as a *bad event* Coll in our proof. Conditioned on that Coll does not occur, the indistinguishability of the real and ideal world experiments are then bounded using a standard *randomness mapping argument* of [CHK+16, ABD+13]. We refer to Appendices C.2—C.6 for the formal elaborations.

C Proof of Theorem 3 (S-mTCCRL Security of $\mathsf{ROH}^{\mathbf{RO}}$)

Recall that $\mathsf{ROH}^{\mathbf{RO}}(w, i) = \mathbf{RO}(f_{\mathsf{in}}(w, i))$, where $f_{\mathsf{in}} : \mathcal{W} \times \mathcal{I} \to \mathcal{X}$ is an efficiently invertible bijection with $f_{\mathsf{in}}^{-1} : \mathcal{I} \times \mathcal{X} \to \mathcal{W}$ being its inverse.

To prove Theorem 3, we first describe the simulator in Sect. C.1. We then sketch our proof ideas in Sect. C.2. Our main proof is in Sect. C.3—C.6, and we refer to Sect. C.2 for a detailed roadmap.

C.1 Constructing the simulator

C.1.1 Variables of the simulator.

To handily describe lazy sampling during simulation, we follow [CHK+16, ABD+13, CS15] and make the randomness used by $\mathcal{S}_{RO}$ explicit through a random function $\mathbf{ro} \in \mathcal{F}_{\mathcal{X},\mathcal{W}}$. Namely, $\mathcal{S}_{RO}$ queries $\mathbf{ro}$ to have random values rather than straightforwardly sampling. We denote by $\mathcal{S}_{RO}^{\mathsf{GlobalK}_{\mathbf{\Delta}},\mathbf{ro}}$ the simulator that queries $(\mathsf{GlobalK}_{\mathbf{\Delta}}, \mathbf{ro})$ to emulate. As argued in [ABD+13], explicit randomness is an equivalent formalism of lazying sampling (because $\mathcal{S}_{RO}$ could sample a sufficient proportion of $\mathbf{ro}$ during its initialization phase).

$\mathcal{S}_{RO}$ maintains a list $\mathcal{Q}_{SRO}$ to keep track of already encountered random oracle queries. The list $\mathcal{Q}_{SRO}$ is initialized to empty. As the execution proceeds, $\mathcal{S}_{RO}$ adds new records of the form (x, y) to $\mathcal{Q}_{SRO}$, which means $\mathcal{S}_{RO}$ has answered $O(x)$ with y. $\mathcal{S}_{RO}$ maintains another list $\mathcal{Q}_{SF} = \{(\mathsf{id}_1, w_1, i_1, b_1, z_1), \ldots\}$ to keep the $\mathbf{F}$-query records it received via the procedure *PhaseSwitch*$(\mathcal{Q}_F)$.

C.1.2 Simulating strategy.

As mentioned in Sect. 4, the simulator shall provide an interface $O_1 : \mathcal{X} \to \mathcal{W}$ that mimics $\mathbf{RO} \in \mathcal{F}_{\mathcal{X},\mathcal{W}}$ and is consistent with $\mathbf{F}$ when necessary. To this end, upon $\mathcal{D}$ querying $O_1(x)$, $\mathcal{S}_{RO}$ first checks if $(x, y) \in \mathcal{Q}_{SRO}$ for some y, i.e., if its answer has been defined, and returns y if it is the case.

Otherwise, the (newly simulated) answer to $O_1(x)$ may have to be consistent with some previously appeared $\mathbf{F}$-query record $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_{SF}$ that has $x = f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i)$. This requires information about the key Δ_{id}, which can be obtained from $\mathsf{GlobalK}_{\boldsymbol{\Delta}}$. Concretely, $\mathcal{S}_{RO}$ computes $v \leftarrow f_{\mathsf{in}}^{-1}(i, x)$, checks every record $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_{SF}$ under this i, computes the candidate key $\Delta'_{\mathsf{id}} = w \oplus v$ and queries $\mathcal{S}_{RO}$ for its correctness. If Δ'_{id} is the correct id-th key, $\mathcal{S}_{RO}$ adds a record $(x, z \oplus b \cdot \Delta'_{\mathsf{id}})$ to $\mathcal{Q}_{SRO}$, which is consistent with $(\mathsf{id}, w, i, b, z)$. Otherwise, $O_1(x)$ can be defined "freely": $\mathcal{S}_{RO}$ acquires the answer $y \leftarrow \mathbf{ro}(x)$ from the random oracle and adds (x, y) to $\mathcal{Q}_{SRO}$. In either case, $\mathcal{S}_{RO}$ finally replies $\mathcal{D}$ with the newly defined answer y.

Finally, the procedure $\mathit{PhaseSwitch}(\mathcal{Q}_F)$, once called, copies the records from $\mathcal{Q}_F$ to its own list $\mathcal{Q}_{SF}$.

C.1.3 Pseudocode description.

We now provide pseudocode to formally describe the simulator $\mathcal{S}_{RO}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}},\mathbf{ro}}$ as explained above.

Simulator $\mathcal{S}_{RO}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}},\mathbf{ro}}$
Lists $\mathcal{Q}_{SRO}$ and $\mathcal{Q}_{SF}$, initialized to empty

public procedure $O_1(x)$
if $(x, y) \in \mathcal{Q}_{SRO}$ for some $y \in \mathcal{W}$ **then** // $O_1(x)$ has been defined
  return y
end if
// $O_1(x)$ is new and has to be defined
$v \leftarrow f_{\mathsf{in}}^{-1}(i, x)$
for all $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_{SF}$ **do**
// Since $\mathcal{Q}_{SF} = \emptyset$ initially, in phase 1 $\mathcal{S}_{RO}$ will skip this for loop and simply respond $O_1(x)$ with $y \leftarrow \mathbf{ro}(x)$.
  $\Delta'_{\mathsf{id}} \leftarrow v \oplus w$ // A candidate for the id-th key
  $b_R \leftarrow \mathsf{GlobalK}_{\boldsymbol{\Delta}}(\mathsf{id}, \Delta'_{\mathsf{id}})$
  if $b_R =$ **true then**
    $y \leftarrow z \oplus b \cdot \Delta'_{\mathsf{id}}$
    Add (x, y) to $\mathcal{Q}_{SRO}$
    return y
  end if
end for
$y \leftarrow \mathbf{ro}(x)$
Add (x, y) to $\mathcal{Q}_{SRO}$
return y

public procedure $\mathit{PhaseSwitch}(\mathcal{Q}_F)$
$\mathcal{Q}_{SF} \leftarrow \mathcal{Q}_F$

C.1.4 Complexity of the simulator.

The following lemma shows that the simulator runs in time polynomial in the number of queries it receives.

Lemma 1. *For any $(q_C, q_P, M, q_{L,max}, \mu)$-bounded S-mTCCRL distinguisher $\mathcal{D}$, the simulator $\mathcal{S}_{RO}^{\mathsf{GlobalK}_{\mathbf{\Delta}},\mathbf{ro}}$ makes at most q_P queries to its randomness* **ro** *and at most μq_P queries to $\mathsf{GlobalK}_{\mathbf{\Delta}}$.*

Proof. The claim on simulator queries to **ro** is clear from the design of $\mathcal{S}_{RO}^{\mathsf{GlobalK}_{\mathbf{\Delta}},\mathbf{ro}}$. Regarding the other claim, consider any query $O_1(x)$ made by $\mathcal{D}$, and let $v = f_{\mathsf{in}}^{-1}(i, x)$. By our assumption regarding the parameter μ, the number of records $(\mathsf{id}, i, w, b, z) \in \mathcal{Q}_{SF}$ is at most μ. For each of such records, $\mathcal{S}_{RO}$ makes at most 1 query to $\mathsf{GlobalK}_{\mathbf{\Delta}}$. Therefore, for each query $O_1(x)$ made by $\mathcal{D}$, $\mathcal{S}_{RO}$ makes at most μ queries to $\mathsf{GlobalK}_{\mathbf{\Delta}}$. Thus the claim. □

C.2 Proof idea

In all the following, we consider some fixed distinguisher $\mathcal{D}$, and assume that it is deterministic (this is wlog since we consider computationally unbounded distinguishers). To simplify notations and make randomness explicit, we will denote $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}$ the ideal world execution (which, according to Fig. 1, should have been $\mathbf{Exp}^{\text{S-mTCCRL}}_{\mathsf{ROH}^{\mathbf{RO}},\overrightarrow{\mathcal{R}},\mathcal{P}_{\mathsf{free}},u,\mathcal{D},\mathcal{S}_{RO},1}$) in which:

(i) The simulator $\mathcal{S}_{RO}$ uses $\mathbf{ro} \in \mathcal{F}_{\mathcal{X},\mathcal{W}}$ as its explicit randomness, and

(ii) The random function oracle is $\mathbf{F} \in \mathcal{KWC}$, and

(iii) The key vector sampled at the beginning equals $\mathbf{\Delta} = (\Delta_1, ..., \Delta_u) \in \overrightarrow{\mathcal{R}}$.

It is easy to see the tuple of random variables $(\mathbf{ro}, \mathbf{F}, \mathbf{R})$ has captured all the randomness of the ideal world execution.

Similarly, denote $\mathcal{D}^{\Sigma_{\mathrm{re}}(\mathbf{RO},\mathbf{\Delta})}$ the real execution (i.e., $\mathbf{Exp}^{\text{S-mTCCRL}}_{\mathsf{ROH}^{\mathbf{RO}},\overrightarrow{\mathcal{R}},\mathcal{P}_{\mathsf{free}},u,\mathcal{D},\mathcal{S}_{RO},0}$ in Fig. 1) in which the random oracle is **RO** and the key vector in use equals $\mathbf{\Delta} = (\Delta_1, ..., \Delta_u)$.

By construction, our simulator $\mathcal{S}_{RO}$ is expected to emulate a random oracle that is consistent with $\mathbf{F}$ when needed. The only exceptions are that when the key vector $\mathbf{\Delta} = (\Delta_1, ..., \Delta_u)$ is revealed at the end of phase 1:

(i) there exist two query records $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_{SF}$ and $(x, y) \in \mathcal{Q}_{SRO}$ such that $x = f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i)$. In this case, $\mathcal{S}_{RO}$ cannot modify y to ensure $z = y \oplus b \cdot \Delta_{\mathsf{id}}$ anymore, and such two records (once existed) enable distinguishing;

(ii) there exist two query records $(\mathsf{id}, w, i, b, z), (\mathsf{id}', w', i', b', z') \in \mathcal{Q}_{SF}$ such that $f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i) = f_{\mathsf{in}}(w' \oplus \Delta_{\mathsf{id}'}, i')$. In this case, if $\mathcal{D}$ queries $O_1(f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i))$ then the response is inconsistent with either z or z'.

Therefore, we define the existence of such records as a *bad event* Coll in the *ideal world executions*. (This is a typical approach [CHK+16, ABD+13]. We will take an "opposite" approach in Sect. C.4.) We then derive upper bound on the probability of Coll in Sect. C.4: this proof is motivated by [GWYY25]. Then, in Sect. C.6.3, we employ the randomness mapping argument (developed in [CHK+16, ABD+13]) to associate some real world executions to the "good" real world executions in which Coll does not occur, such that the associated executions yield the same outputs. Finally, in Sect. D.5 we prove that "good" real world executions and associated ideal world executions occur with sufficiently close probabilities, and this completes the S-mTCCRL proof.

The intermediate experiment Σ_{inter} for the S-mTCCRL proof of $\mathsf{ROH}^{\mathbf{RO}}$:

1. **Initialization:** Sample $\mathbf{F} \xleftarrow{\$} \mathcal{KWC}$, $\mathbf{ro} \xleftarrow{\$} \mathcal{F}_{\mathcal{X},\mathcal{W}}$, and u keys $\boldsymbol{\Delta}^{\circ} = (\Delta_1^{\circ}, ..., \Delta_u^{\circ}) \xleftarrow{\$} \overrightarrow{\mathcal{R}}$.
2. **Phase 1:** Run $end \leftarrow \mathcal{D}^{\mathcal{S}^{(\mathsf{GlobalK}_{\boldsymbol{\Delta}},\mathbf{ro})}.O_1,\mathbf{F},\mathsf{mL}_{\boldsymbol{\Delta}}}$ with the simulated interface $\mathcal{S}^{(\mathsf{GlobalK}_{\boldsymbol{\Delta}},\mathbf{ro})}.O_1$, $\mathbf{F}$ and $\mathsf{mL}_{\boldsymbol{\Delta}}$, till $\mathcal{D}$ outputs end indicating the end of phase 1;
3. **Key revealing:** Let $\mathcal{Q}_L = ((\mathsf{id}_1, P_1, r_1), ...)$ be the transcript of key leaking queries and answers of $\mathcal{D}$ in phase 1. Sample another key vector $\boldsymbol{\Delta} \xleftarrow{\$} \overrightarrow{\mathcal{R}}(\mathcal{Q}_L)$ and pass $\boldsymbol{\Delta}$ to $\mathcal{D}$. Invoke $\mathcal{S}^{(\mathsf{GlobalK}_{\boldsymbol{\Delta}},\mathbf{ro})}.PhaseSwitch(\mathcal{Q}_F)$, where $\mathcal{Q}_F$ is the transcript of queries and answers of $\mathcal{D}$ to $\mathbf{F}$ in phase 1.
4. **Phase 2:** Run $\beta' \leftarrow \mathcal{D}^{\mathcal{S}^{(\mathsf{GlobalK}_{\boldsymbol{\Delta}},\mathbf{ro})}.O_1}$ and obtain its output β'.
5. **Finalization:** Output β' (the output of $\mathcal{D}$).

Figure 9: Intermediate experiment Σ_{inter} for the S-mTCCRL proof of $\mathsf{ROH}^{\mathbf{RO}}$.

C.3 Intermediate world

To facilitate proof, we introduce an intermediate world experiment. For necessary notations, we will denote by $\mathcal{Q}_L = \{(\mathsf{id}_1, P_1, r_1), \ldots\}$ the transcript of $\mathcal{D}$'s queries to $\mathsf{mL}_{\boldsymbol{\Delta}}$, $\boldsymbol{\Delta} = (\Delta_1, ..., \Delta_u)$, and answers. A triple (id, P, r) in $\mathcal{Q}_L$ indicates $\mathcal{D}$ queried $\mathsf{mL}_{\boldsymbol{\Delta}}(\mathsf{id}, P)$ and: when $r = 0$, it indicates $P(\Delta_{\mathsf{id}}) = 0$ and the id-th user's session is then aborted; when $r = 1$, it indicates $P(\Delta_{\mathsf{id}}) = 1$.

With this, we further denote by

$$\begin{aligned} \mathcal{R}_{\mathsf{id}}(\mathcal{Q}_L) &:= \{\Delta^*_{\mathsf{id}} \in \mathcal{R}_{\mathsf{id}} : P(\Delta^*_{\mathsf{id}}) = r \text{ for all } (\mathsf{id}, P, r) \in \mathcal{Q}_L\}, \\ \overrightarrow{\mathcal{R}}(\mathcal{Q}_L) &:= \mathcal{R}_1(\mathcal{Q}_L) \times \mathcal{R}_2(\mathcal{Q}_L) \times ... \times \mathcal{R}_u(\mathcal{Q}_L). \end{aligned} \tag{12}$$

To understand, note that the key leaking oracle $\mathsf{mL}_{\mathbf{R}}$ instantiated with a key vector $\mathbf{R} \in \overrightarrow{\mathcal{R}}(\mathcal{Q}_L)$ would give responses that are consistent with all query records in $\mathcal{Q}_L$.

The intermediate experiment Σ_{inter} can be then described with the help of $\overrightarrow{\mathcal{R}}(\mathcal{Q}_L)$. Concretely, Σ_{inter} is the same as the ideal world experiment, except that: *at the end of phase 1,* Σ_{inter} *samples a new "dummy" key vector from* $\overrightarrow{\mathcal{R}}(\mathcal{Q}_L)$ *and reveal this "dummy" vector to* $\mathcal{D}$*, where* $\mathcal{Q}_L$ *is* $\mathcal{D}$*'s key leaking query transcript in phase 1.* This means the revealed keys may *not* be the same as the keys actually used in phase 1. For convenience, we denote by $\boldsymbol{\Delta}^{\circ}$ the keys used in phase 1, and by $\boldsymbol{\Delta}$ the "dummy" keys sampled and revealed to $\mathcal{D}$ at the end of phase 1: as indicated by the formal description of Σ_{inter} in Fig. 9.

In a similar vein to Sect. C.2, we will denote $\mathcal{D}^{\Sigma_{\text{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^{\circ},\mathbf{R})}$ the intermediate world execution in which:

(i) The simulator $\mathcal{S}_{RO}$ uses $\mathbf{ro}$ as its explicit randomness, and

(ii) The random function oracle is $\mathbf{F} \in \mathcal{KWC}$, and

(iii) The key vector sampled at the beginning equals $\boldsymbol{\Delta}^{\circ} = (\Delta_1^{\circ}, ..., \Delta_u^{\circ}) \in \overrightarrow{\mathcal{R}}$, and

(iv) The "dummy" vector sampled at the end of phase 1 equals $\boldsymbol{\Delta} = (\Delta_1, ..., \Delta_u) \in \overrightarrow{\mathcal{R}}(\mathcal{Q}_L)$.

C.4 Event Coll in intermediate world executions

We use $\mathcal{D}^{\Sigma_{\text{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^{\circ},\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}$, $\mathcal{Q} = (\mathcal{Q}_F, \mathcal{Q}_{SRO}, \mathcal{Q}_L)$ to denote the event that $\mathcal{D}$ obtains the query records (or transcripts) $\mathcal{Q}_F, \mathcal{Q}_{SRO}$ and $\mathcal{Q}_L$ in phase 1 of the intermediate world execution

$\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$. Note that although $\mathcal{Q}_{SRO}$ is an internal set of $\mathcal{S}_{RO}$, $\mathcal{D}$ knows all its contents by the design of $\mathcal{S}_{RO}$. Let $\mathcal{T}$ be the set of $\mathcal{Q}$ such that $\Pr[\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}] > 0$ if and only if $\mathcal{Q} \in \mathcal{T}$ (i.e., the set of transcripts that can be actually obtained in intermediate world executions).

Then, as discussed in Sect. C.2, during an intermediate world execution $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$, the event Coll occurs if either of the following is fulfilled:

(i) At the end of phase 1 of $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$, there exist $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$ and $(x, y) \in \mathcal{Q}_{SRO}$ such that $x = f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i)$.

(ii) At the end of phase 1 of $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$, there exist two distinct records $(\mathsf{id}, w, i, b, z)$ and $(\mathsf{id}', w', i', b', z')$ in $\mathcal{Q}_F$ such that $f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i) = f_{\mathsf{in}}(w' \oplus \Delta_{\mathsf{id}'}, i')$.

We follow [GWYY25] to bound the probability of Coll. For this, for every user index $\mathsf{id} \in [u]$, we define a subcondition:

- $\mathsf{Coll}_{\mathsf{id}}$: At the end of phase 1 of $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$, either of the following holds for the index id:
  (i) there exist (w, i, b, z) such that $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$ and $(x, y) \in \mathcal{Q}_{SRO}$ such that $x = f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i)$.
  (ii) there exist (w, i, b, z) and $(\mathsf{id}', w', i', b', z') \in \mathcal{Q}_F$ such that $f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i) = f_{\mathsf{in}}(w' \oplus \Delta_{\mathsf{id}'}, i')$.

It is easy to see $\mathsf{Coll} = \vee_{\mathsf{id}=1}^{u} \mathsf{Coll}_{\mathsf{id}}$.

We next focus on $\mathsf{Coll}_{\mathsf{id}}$. This subcondition is fulfilled only if the id-th key Δ_{id} in the "dummy" key vector $\mathbf{R}$ falls into certain sets of "bad keys". Concretely, for subevent (i), consider any $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$. Define

$$\mathcal{Q}_{SRO}[i] := \{(x, y) \in \mathcal{Q}_{SRO} : f_{\mathsf{in}}^{-1}(i, x) = v \text{ for some } v\}. \tag{13}$$

Then, for any candidate key Δ^* it holds $(f_{\mathsf{in}}(w \oplus \Delta^*, i), \star) \in \mathcal{Q}_{SRO}[i]$ (i.e., the tweak i limits its possibilities). With this in mind, define

$$\mathsf{BadK}_1(\mathsf{id}, w, i, b, z) := \{\Delta^* \in \mathcal{R}_{\mathsf{id}} : (f_{\mathsf{in}}(w \oplus \Delta^*, i), \star) \in \mathcal{Q}_{SRO}[i]\}. \tag{14}$$

Then, for each record $(x, y) \in \mathcal{Q}_{SRO}[i]$, the number of Δ^* such that $f_{\mathsf{in}}(\Delta^* \oplus w, i) = x$ is exactly one, since f_{in} is a bijection. This means

$$\left|\mathsf{BadK}_1(\mathsf{id}, w, i, b, z)\right| \le \left|\mathcal{Q}_{SRO}[i]\right|.$$

Define a set of "bad keys" for id as:

$$\mathsf{BadK}_1(\mathcal{Q}, \mathsf{id}) := \bigcup_{(\mathsf{id},w,i,b,z)\in\mathcal{Q}_F} \mathsf{BadK}_1(\mathsf{id}, w, i, b, z).$$

We will calculate the upper bound on its size later.

We follow similar ideas for subevent (ii). First, note that $(\mathsf{id}, w, i, b, z)$ and $(\mathsf{id}', w', i', b', z') \in \mathcal{Q}_F$ have $f_{\mathsf{in}}(\Delta_{\mathsf{id}} \oplus w, i) = f_{\mathsf{in}}(\Delta_{\mathsf{id}'} \oplus w', i')$ only if $\mathsf{id}' \neq \mathsf{id}$: otherwise, it implies $(w, i) = (w', i')$ which is not possible by the restriction (i) of Definition 2.

When $\mathsf{id}' \neq \mathsf{id}$, subevent (ii) occurs only if Δ_{id} falls into certain bad sets. For this, for each pair of distinct records $(\mathsf{id}, w, i, b, z)$, $(\mathsf{id}', w', i, b', z') \in \mathcal{Q}_F$ with $\mathsf{id}' \neq \mathsf{id}$, define

$$\mathsf{BadK}_2\big((\mathsf{id}, w, i, b, z), (\mathsf{id}', w', i, b', z')\big) := \big\{\Delta^* \in \mathcal{R}_{\mathsf{id}} : f_{\mathsf{in}}(\Delta^* \oplus w, i) = f_{\mathsf{in}}(\Delta_{\mathsf{id}'} \oplus w', i')\big\}.$$

The number of Δ^* satisfying $f_{\mathsf{in}}(\Delta^* \oplus w, i) = f_{\mathsf{in}}(\Delta_{\mathsf{id}'} \oplus w', i')$ is exactly 1. Therefore,

$$\big|\mathsf{BadK}_2\big((\mathsf{id}, w, i, b, z), (\mathsf{id}', w', i, b', z')\big)\big| = 1.$$

Define another set of "bad keys" for id as:

$$\begin{aligned}&\mathsf{BadK}_2(\mathcal{Q}, \mathsf{id})\\ &:= \bigcup_{(\mathsf{id},w,i,b,z),(\mathsf{id}',w',i',b',z') \in \mathcal{Q}_F} \mathsf{BadK}_2\big((\mathsf{id}, w, i, b, z), (\mathsf{id}', w', i', b', z')\big).\end{aligned}$$

Since the "dummy" key vector $\boldsymbol{\Delta} = (\Delta_1, ..., \Delta_u)$ is sampled from $\overrightarrow{\mathcal{R}}(\mathcal{Q}_L)$ at the end, the id-th key Δ_{id} in $\boldsymbol{\Delta}$ is sampled from $\mathcal{R}_{\mathsf{id}}(\mathcal{Q}_L)$, and it holds

$$\begin{aligned}\Pr\big[\Delta_{\mathsf{id}} \in \mathsf{BadK}_i(\mathcal{Q}, \mathsf{id}) \mid \mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}\big] &\leq \min\left\{\frac{\big|\mathsf{BadK}_i(\mathcal{Q}, \mathsf{id})\big|}{\big|\mathcal{R}_{\mathsf{id}}(\mathcal{Q}_L)\big|}, 1\right\}\\ &\leq \frac{\big|\mathsf{BadK}_i(\mathcal{Q}, \mathsf{id})\big|}{\big|\mathcal{R}_{\mathsf{id}}(\mathcal{Q}_L)\big|}\end{aligned}$$

for $i = 1, 2$. Since $\mathsf{Coll}_{\mathsf{id}}$ is fulfilled if and only if $\Delta_{\mathsf{id}} \in \big(\mathsf{BadK}_1(\mathcal{Q}, \mathsf{id}) \cup \mathsf{BadK}_2(\mathcal{Q}, \mathsf{id})\big)$, we have

$$\begin{aligned}&\Pr\big[\mathsf{Coll}_{\mathsf{id}}\big]\\ &\leq \sum_{\mathcal{Q} \in \mathcal{T}} \Bigg(\Pr\big[\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}\big]\\ &\qquad\qquad \times \Pr\big[\Delta_{\mathsf{id}} \in \big(\mathsf{BadK}_1(\mathcal{Q}, \mathsf{id}) \cup \mathsf{BadK}_2(\mathcal{Q}, \mathsf{id})\big) \mid \mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}\big]\Bigg)\\ &\leq \sum_{\mathcal{Q} \in \mathcal{T}} \left(\Pr\big[\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}\big] \times \frac{\big|\mathsf{BadK}_1(\mathcal{Q}, \mathsf{id})\big| + \big|\mathsf{BadK}_2(\mathcal{Q}, \mathsf{id})\big|}{\big|\mathcal{R}_{\mathsf{id}}(\mathcal{Q})\big|}\right). \qquad (15)\end{aligned}$$

We now distinguish the users that received key leaking queries from the others. Formally, let $\mathcal{U} \subseteq [u]$ be the set such that $\mathsf{id} \in \mathcal{U}$ if and only if $(\mathsf{id}, P, r) \in \mathcal{Q}_L$ for some P and r (i.e., $\mathsf{mL}_{\boldsymbol{\Delta}}(\mathsf{id}, \cdot)$ was queried). Then,

$$\Pr\big[\mathsf{Coll}\big] \leq \sum_{\mathsf{id} \in [u]} \Pr\big[\mathsf{Coll}_{\mathsf{id}}\big] = \underbrace{\sum_{\mathsf{id} \in ([u] \setminus \mathcal{U})} \Pr\big[\mathsf{Coll}_{\mathsf{id}}\big]}_{A_1} + \underbrace{\sum_{\mathsf{id} \in \mathcal{U}} \Pr\big[\mathsf{Coll}_{\mathsf{id}}\big]}_{A_2}. \qquad (16)$$

The analysis of A_1 is much simpler than that of A_2. Below we calculate their upper bounds in turn.

Bounding A_1. For any index $\mathsf{id} \in ([u] \setminus \mathcal{U})$, $\mathsf{mL}_{\boldsymbol{\Delta}}(\mathsf{id}, \cdot)$ was never queried and the key Δ_{id} in $\boldsymbol{\Delta}$ is sampled from $\mathcal{R}_{\mathsf{id}}(\mathcal{Q}_L) = \mathcal{R}_{\mathsf{id}}$. Therefore,

$$\begin{aligned}&\sum_{\mathsf{id} \in ([u] \setminus \mathcal{U})} \Pr\big[\mathsf{Coll}_{\mathsf{id}}\big]\\ &\leq \sum_{\mathsf{id} \in ([u] \setminus \mathcal{U})} \sum_{\mathcal{Q} \in \mathcal{T}} \left(\Pr\big[\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}\big] \times \frac{\big|\mathsf{BadK}_1(\mathcal{Q}, \mathsf{id})\big| + \big|\mathsf{BadK}_2(\mathcal{Q}, \mathsf{id})\big|}{\big|\mathcal{R}_{\mathsf{id}}\big|}\right)\\ &= \sum_{\mathcal{Q} \in \mathcal{T}} \Pr\big[\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}\big] \times \left(\sum_{\mathsf{id} \in ([u] \setminus \mathcal{U})} \frac{\big|\mathsf{BadK}_1(\mathcal{Q}, \mathsf{id})\big| + \big|\mathsf{BadK}_2(\mathcal{Q}, \mathsf{id})\big|}{\big|\mathcal{R}_{\mathsf{id}}\big|}\right).\end{aligned}$$

We have

$$\sum_{\mathsf{id}\in([u]\setminus\mathcal{U})}\left|\mathsf{BadK}_1(\mathcal{Q},\mathsf{id})\right| \le \sum_{(\mathsf{id},w,i,b,z)\in\mathcal{Q}_F:\mathsf{id}\in([u]\setminus\mathcal{U})}\left|\mathcal{Q}_{SRO}[i]\right|$$
$$\le \sum_{i\in\{0,1\}^n}\underbrace{\sum_{(\mathsf{id},w,i,b,z)\in\mathcal{Q}_F}}_{\le\mu}\left|\mathcal{Q}_{SRO}[i]\right|$$
$$\le \mu\cdot\sum_{i\in\{0,1\}^n}\left|\mathcal{Q}_{SRO}[i]\right| = \mu q_P. \tag{17}$$

On the other hand, if we let $q_{C,i}\le\mu$ denote the number of queries in $\mathcal{Q}_F$ using tweak i, then

$$\sum_{\mathsf{id}\in([u]\setminus\mathcal{U})}\left|\mathsf{BadK}_2(\mathcal{Q},\mathsf{id})\right| \le \sum_{(\mathsf{id},w,i,b,z),(\mathsf{id}',w',i,b',z')\in\mathcal{Q}_F:\mathsf{id}\in([u]\setminus\mathcal{U})} 1$$
$$\le \sum_{i\in\mathcal{I}}\binom{q_{C,i}}{2} \le \sum_{i\in\mathcal{I}}\frac{q_{C,i}(q_{C,i}-1)}{2}$$
$$\le (\mu-1)\sum_{i\in\mathcal{I}}\frac{q_{C,i}}{2} \le \frac{(\mu-1)\cdot q_C}{2} \tag{18}$$

To simplify notations, define a constant

$$CON := \mu q_P + \frac{(\mu-1)\cdot q_C}{2}. \tag{19}$$

It then holds

$$A_1 = \sum_{\mathsf{id}\in([u]\setminus\mathcal{U})}\Pr\left[\mathsf{Coll}_{\mathsf{id}}\right] \le \sum_{\mathcal{Q}\in\mathcal{T}}\left(\Pr\left[\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}\xrightarrow{\text{phase1}}\mathcal{Q}\right]\times\frac{CON}{\left|\mathcal{R}_{\mathsf{id}}\right|}\right)$$
$$\le \frac{CON}{\left|\mathcal{R}_{\mathsf{id}}\right|}. \tag{20}$$

Bounding A_2. For any "key leaked" index $\mathsf{id}\in\mathcal{U}$, the key Δ_{id} in $\mathbf{\Delta}$ is sampled from $\mathcal{R}_{\mathsf{id}}(\mathcal{Q}_L)\subseteq\mathcal{R}_{\mathsf{id}}$, which varies with id. For $j=1,2$, we follow the ideas of Eqs. (17) and (18) to bound $\left|\mathsf{BadK}_j(\mathcal{Q},\mathsf{id})\right|$ (instead of the sum $\sum_{\mathsf{id}\in\mathcal{U}}\left|\mathsf{BadK}_j(\mathcal{Q},\mathsf{id})\right|$):

$$\left|\mathsf{BadK}_1(\mathcal{Q},\mathsf{id})\right| \le \sum_{(w,i,b,z):(\mathsf{id},w,i,b,z)\in\mathcal{Q}_F}\left|\mathcal{Q}_{SRO}[i]\right|$$
$$\le \sum_{i\in\{0,1\}^n}\underbrace{\sum_{(\mathsf{id},w,i,b,z)\in\mathcal{Q}_F}}_{\le\mu}\left|\mathcal{Q}_{SRO}[i]\right| \le \mu q_P,$$
$$\left|\mathsf{BadK}_2(\mathcal{Q},\mathsf{id})\right| \le \sum_{(\mathsf{id},w,i,b,z),(\mathsf{id}',w',i,b',z')\in\mathcal{Q}_F} 1$$
$$\le \sum_{i\in\mathcal{I}}\binom{q_{C,i}}{2} \le \frac{(\mu-1)\cdot q_C}{2}.$$

Therefore, $|\mathsf{BadK}_1(\mathcal{Q},\mathsf{id})| + |\mathsf{BadK}_2(\mathcal{Q},\mathsf{id})| \le CON$ for the constant defined in Eq. (19). Injecting this into Eq. (15) yields

$$\begin{aligned}\Pr[\mathsf{Coll}_{\mathsf{id}}] &\le \sum_{\mathcal{Q}\in\mathcal{T}} \left(\Pr[\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}] \times \frac{|\mathsf{BadK}_1(\mathcal{Q},\mathsf{id})|}{|\mathcal{R}_{\mathsf{id}}(\mathcal{Q}_L)|}\right) \\ &\le \underbrace{\sum_{\mathcal{Q}\in\mathcal{T}} \left(\Pr[(\mathbf{ro},\mathbf{F},\mathbf{R}) \vdash \mathcal{Q}] \times \frac{CON}{|\mathcal{R}_{\mathsf{id}}(\mathcal{Q}_L)|}\right)}_{S_{\mathsf{id}}}. \end{aligned} \tag{21}$$

An upper bound for S_{id} has been given in [GWYY25, Eq. (20)]:

$$S_{\mathsf{id}} \le \frac{(q_{L,max}+1)CON}{C}, \tag{22}$$

By this and by $|\mathcal{U}| \le M$ (as per our assumption), we have

$$A_2 = \sum_{\mathsf{id}\in\mathcal{U}} \Pr[\mathsf{Coll}_{\mathsf{id}}] \le \sum_{\mathsf{id}\in\mathcal{U}} \frac{(q_{L,max}+1)CON}{C} \le \frac{M(q_{L,max}+1)CON}{C}. \tag{23}$$

Summary. Gathering Eqs. (16), (20) and (23) yields

$$\begin{aligned}&\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R}}[\mathsf{Coll} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}] \le A_1 + A_2 \\ &\le \frac{CON}{C} + \frac{M(q_{L,max}+1)CON}{C} \\ &\le \frac{(M+1)(q_{L,max}+1)}{C} \cdot \left(\mu q_P + \frac{(\mu-1)\cdot q_C}{2}\right). \end{aligned} \tag{24}$$

C.5 Indistinguishability of Σ_{inter} and Σ_{id}

Given a tuple of ideal world randomness $(\mathbf{ro},\mathbf{F},\mathbf{R})$, we use $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}_L$ to denote the event that $\mathcal{D}$ obtains the key leaking query transcript $\mathcal{Q}_L$ in phase 1 of the ideal execution $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}$. Similarly, $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}_L$ denotes the event that $\mathcal{D}$ obtains $\mathcal{Q}_L$ in phase 1 of the intermediate world execution $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$. Note that:

(i) First, whether $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}_L$ occurs is determined by $\mathbf{ro},\mathbf{F}$ and the first key vector $\boldsymbol{\Delta}^\circ$. Therefore, this event "links" the intermediate world randomness $(\mathbf{ro},\mathbf{F},\boldsymbol{\Delta}^\circ)$ and the ideal world randomness $\mathbf{ro},\mathbf{F},\mathbf{R}$;

(ii) Second, by the design of Σ_{inter} , a random tuple $(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R}) \in \mathcal{F}_{\mathcal{X},\mathcal{W}} \times \mathcal{KWC} \times \overrightarrow{\mathcal{R}} \times \overrightarrow{\mathcal{R}}$ is a "valid" Σ_{inter} tuple (that can actually be sampled by some intermediate world execution) only if $\boldsymbol{\Delta} \in \overrightarrow{\mathcal{R}}(\mathcal{Q}_L)$.

Consider two executions $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}$ such that $(\mathbf{ro},\mathbf{F},\boldsymbol{\Delta}^\circ)$ is "linked" to $\mathbf{ro},\mathbf{F},\mathbf{R}$ via some $\mathcal{Q}_L$. We then prove that they generate the same adversarial transcripts of queries and responses.

Lemma 2. *Fix $\mathcal{D}$ in arbitrary. Consider a tuple of random variables $(\mathbf{ro},\mathbf{F},\mathbf{R})$ such that*

$$\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}_L\,.$$

If $\boldsymbol{\Delta}^\circ \in \overrightarrow{\mathcal{R}}$ fulfills $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}_L$ (i.e., the two executions yield the same key leaking query transcript $\mathcal{Q}_L$ at the end of phase 1), then $\mathcal{D}$ obtains the same transcript of queries and answers in $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}$, and gives the same output.

Proof. Imagine running the two executions $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}$ in parallel. By design of the two executions, the key vectors revealed to $\mathcal{D}$ at the end of phase 1 are the same $\overrightarrow{\mathcal{R}}$. Except for this action, both the ideal world and the intermediate world executions are driven by adversarial queries. Therefore, we prove the claim via an induction on the sequence of queries of $\mathcal{D}$.

Assume that the sequence of queries and answers of $\mathcal{D}$ (which is empty at the beginning) is the same up to some point in the executions $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}$, and consider the next query. Since $\mathcal{D}$ is deterministic, it issues the same query as the next. We distinguish four cases.

- Case 1: $\mathcal{D}$'s next query is to $\mathsf{mL}_{\boldsymbol{\Delta}}(\mathsf{id}, P)$. Then, since $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}$ yield the same transcript $\mathcal{Q}_L$, the obtained responses are identical.
- Case 2: $\mathcal{D}$'s next query is to $O_2(\mathsf{id}, w, i, b)$. In both $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}$, this is replied by the same random function $\mathbf{F}$. $\mathcal{D}$ thus obtains the same response.
- Case 3: $\mathcal{D}$'s next query is to $O_1(x)$. In both $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}$, this is replied by the simulator $\mathcal{S}_{RO}$. We further distinguish two subcases.
  - Subcase 3.1: the query $O_1(x)$ occurs in phase 1. Then, by the design of $\mathcal{S}_{RO}$, in both $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}$ it simply relays the answer $y \leftarrow \mathbf{ro}(x)$. $\mathcal{D}$ thus obtains the same response.
  - Subcase 3.2: the query $O_1(x)$ occurs in phase 2. Then, by the design of $\mathcal{S}_{RO}$, in both $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}$ the response depends on four variables: $\mathcal{Q}_F$, $\boldsymbol{\Delta}$, $\mathbf{ro}$ and $\mathbf{F}$. In phase 2 of the two executions, these variables are the same. Therefore, $\mathcal{D}$ obtains the same response as well.

  Therefore, in any case, $\mathcal{D}$ obtains the same response in $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}$.

In any case, the next answers are the same in the two executions. This means $\mathcal{D}$ gains the same transcript of queries and answers in $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}$. Since $\mathcal{D}$ is deterministic, its output is determined by this transcript, and is thus the same in the two executions. □

Lemma 3. *The ideal and the intermediate worlds are perfectly indistinguishable: for any (q_C, q_P, M, $q_{L,max}$, μ)-bounded S-mTCCRL distinguisher $\mathcal{D}$, it holds*

$$\left|\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R}}\left[\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}=1\right]-\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}}\left[\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}=1\right]\right|=0.$$

Proof. Fix $\mathcal{D}$ in arbitrary. Let Θ_1 be the set of random variables $(\mathbf{ro},\mathbf{F},\mathbf{R})$ such that $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}$ outputs 1 if and only if $(\mathbf{ro},\mathbf{F},\mathbf{R}) \in \Theta_1$. Further, assume $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}_L$. By Lemma 2, if $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}_L$, then $\mathcal{D}$ obtains the same transcript of queries and answers in $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}$, and gives the same output.

Then,

$$\begin{aligned}
&\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}}\left[\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}=1\right]\\
&=\sum_{(\mathbf{ro},\mathbf{F},\mathbf{R})\in\Theta_1}\Pr\left[\mathbf{ro}^* \xleftarrow{\$} \mathcal{F}_{\mathcal{X},\mathcal{W}}, \mathbf{F}^* \xleftarrow{\$} \mathcal{KWC},\right.\\
&\qquad\qquad\left.\mathbf{R}^* \xleftarrow{\$} \overrightarrow{\mathcal{R}} : (\mathbf{ro}^*,\mathbf{F}^*,\mathbf{R}^*)=(\mathbf{ro},\mathbf{F},\mathbf{R})\right]\\
&=\frac{1}{|\mathcal{F}_{\mathcal{X},\mathcal{W}}|\times|\mathcal{KWC}|\times|\overrightarrow{\mathcal{R}}|}.
\end{aligned}\tag{25}$$

On the other hand, $\mathcal{D}$ outputs 1 in an intermediate execution $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ if and only if:

(i) The randomness in use satisfies $(\mathbf{ro}, \mathbf{F}, \mathbf{R}) \in \Theta_1$. Otherwise, $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})}$ outputs 0, and $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ also outputs 0 for any valid $\mathbf{\Delta}^\circ$ by Lemma 2.

(ii) For $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}_L$, it has $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}_L$ (which means $\mathbf{\Delta}^\circ \in \overrightarrow{\mathcal{R}}(\mathcal{Q}_L)$). Otherwise, $(\mathbf{ro}, \mathbf{F}, \mathbf{R}^\circ, \mathbf{R})$ is not a valid intermediate world randomness tuple.

Such a randomness tuple $(\mathbf{ro}, \mathbf{F}, \mathbf{R}^\circ, \mathbf{R})$ appears in an intermediate world execution if and only if:

(i) The random functions sampled at the beginning equal $\mathbf{ro}$ and $\mathbf{F}$, and

(ii) The key vector $\mathbf{\Delta}^\circ$ sampled (from $\overrightarrow{\mathcal{R}}$) at the beginning fulfills $\mathbf{\Delta}^\circ \in \overrightarrow{\mathcal{R}}(\mathcal{Q}_L)$, and

(iii) The "dummy" key vector sampled (from $\overrightarrow{\mathcal{R}}(\mathcal{Q}_L)$) at the end of phase 1 equals $\mathbf{\Delta}$.

Therefore,

$$
\begin{aligned}
&\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R}}\left[\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} = 1\right] \\
&= \sum_{(\mathbf{ro},\mathbf{F},\mathbf{R})\in\Theta_1} \Bigg(\Pr\left[\mathbf{ro}^* \xleftarrow{\$} \mathcal{F}_{\mathcal{X},\mathcal{W}}, \mathbf{F}^* \xleftarrow{\$} \mathcal{KWC} : \right. \\
&\qquad\qquad \left. (\mathbf{ro}^*, \mathbf{F}^*) = (\mathbf{ro}, \mathbf{F})\right] \\
&\qquad \times \Pr\left[\mathbf{R}^* \xleftarrow{\$} \overrightarrow{\mathcal{R}}(\mathcal{Q}_L) : \mathbf{R}^* = \mathbf{R} \mid \mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}_L\right] \qquad (26) \\
&\qquad \times \underbrace{\Pr\left[\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}_L \mid \mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}_L\right]}_{=\Pr\left[\mathbf{R}^\circ \xleftarrow{\$} \overrightarrow{\mathcal{R}} : \mathbf{R}^\circ \in \overrightarrow{\mathcal{R}}(\mathcal{Q}_L) \mid \mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})} \xrightarrow{\text{phase1}} \mathcal{Q}_L\right]} \Bigg) \\
&= \frac{1}{|\mathcal{F}_{\mathcal{X},\mathcal{W}}| \times |\mathcal{KWC}| \times |\overrightarrow{\mathcal{R}}(\mathcal{Q}_L)|} \times \frac{|\overrightarrow{\mathcal{R}}(\mathcal{Q}_L)|}{|\overrightarrow{\mathcal{R}}|} \\
&= \frac{1}{|\mathcal{F}_{\mathcal{X},\mathcal{W}}| \times |\mathcal{KWC}| \times |\overrightarrow{\mathcal{R}}|}. \qquad (27)
\end{aligned}
$$

By Eqs. (25) and (27), it then holds

$$\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R}}\left[\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} = 1\right] = \Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}}\left[\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ro},\mathbf{F},\mathbf{R})} = 1\right],$$

i.e., the ideal and the intermediate worlds are perfectly indistinguishable. □

C.6 Indistinguishability of Σ_{inter} and Σ_{re}

We use the randomness mapping argument [CHK+16, ABD+13] (we employ a formalism without explicit map) to associate real world executions to good intermediate world executions to prove their indistinguishability. For this, we first define good intermediate world executions and exhibit useful properties. We then define the association and complete the proof.

C.6.1 Good intermediate world executions and properties.

Fix a distinguisher $\mathcal{D}$ in arbitrary. We say that a tuple of random variables $(\mathbf{ro}, \mathbf{F}, \mathbf{R}^\circ, \mathbf{R})$ is *bad*, if Coll occurs in the execution of $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$. Otherwise, it is *good*. Executions using good $(\mathbf{ro}, \mathbf{F}, \mathbf{R}^\circ, \mathbf{R})$ are *good intermediate world executions*.

We say that a distinguisher *completes all chains*, if, at the end of the execution, it emulates a call to $O_1(f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i))$ for every record $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_{SF}$ (i.e., every $\mathbf{F}$-query occurred during

this execution). Clearly, for any "raw" distinguisher $\mathcal{D}$ we can construct a corresponding $\overline{\mathcal{D}}$ that completes all chains.

We use $\mathcal{D}^{\Sigma_{\text{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\text{end}} (\mathcal{Q}_F, \mathcal{Q}_{SRO}, \mathbf{\Delta}^\dagger)$ to denote the event that: (i) the vector revealed at the end of phase 1 of $\mathcal{D}^{\Sigma_{\text{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ equals $\mathbf{\Delta}^\dagger$, and (ii) the sets standing *at the end* of $\mathcal{D}^{\Sigma_{\text{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ equal $\mathcal{Q}_F$ and $\mathcal{Q}_{SRO}$. Note that it necessarily holds $\mathbf{\Delta}^\dagger = \mathbf{\Delta}$: however, keeping $\mathbf{\Delta}^\dagger$ in the outputs brings in conceptual simplifications. Also note that this slightly deviates from the notation $\xrightarrow{\text{phase1}}$ in Sect. C.5.

Consider an intermediate world tuple $(\mathbf{ro}, \mathbf{F}, \mathbf{R}^\circ, \mathbf{R})$, $\mathbf{\Delta} = (\Delta_1, ..., \Delta_u)$, such that

$$\overline{\mathcal{D}}^{\Sigma_{\text{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\text{end}} (\mathcal{Q}_F, \mathcal{Q}_{SRO}, \mathbf{\Delta}^\dagger)\,.$$

A record $(x, y) \in \mathcal{Q}_{SRO}$ such that there exists $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$ with $f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i) = x$ is called **chained**. Otherwise, $(x, y) \in \mathcal{Q}_{SRO}$ is **unchained**. Below we prove a consistency property on **chained** records.

Lemma 4. *Consider any good random tuple* $(\mathbf{ro}, \mathbf{F}, \mathbf{R}^\circ, \mathbf{R})$, $\mathbf{\Delta} = (\Delta_1, ..., \Delta_u)$, *and assume*

$$\overline{\mathcal{D}}^{\Sigma_{\text{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\text{end}} (\mathcal{Q}_F, \mathcal{Q}_{SRO}, \mathbf{\Delta}^\dagger)\,.$$

It then holds:

(i) For every $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$, *there exists a unique* **chained** *record* $(x, y) \in \mathcal{Q}_{SRO}$ *with* $f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i) = x$ *and* $y = z \oplus b \cdot \Delta_{\mathsf{id}}$.

(ii) For every **chained** *record* (x, y) *in* $\mathcal{Q}_{SRO}$, *there exists a unique record* $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$ *such that* $f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i) = x$ *and* $y \oplus b \cdot \Delta_{\mathsf{id}} = z$.

(iii) The number of **chained** *records in* $\mathcal{Q}_{SRO}$ *equals the number of* $\mathbf{F}$*-query records in* $\mathcal{Q}_F$.

Proof. We first prove claim (i). By the definition of Σ_{inter} , $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$ implies that $\overline{\mathcal{D}}$ made a query to $\mathbf{F}(\mathsf{id}, w, i, b)$ in phase 1 of $\overline{\mathcal{D}}^{\Sigma_{\text{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$. Since $(\mathbf{ro}, \mathbf{F}, \mathbf{R}^\circ, \mathbf{R})$ is good, at the end of phase 1 there is no record of the form $(f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i), \star)$ in $\mathcal{Q}_{SRO}$. Since $\overline{\mathcal{D}}$ completes all chains, in phase 2 it will query $O_1(x)$ with $x = f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i)$ at some point. By the design of $\mathcal{S}_{RO}$, it will then detect the record $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_{SF}$ with $v = f_{\mathsf{in}}^{-1}(i, x)$, $\Delta'_{\mathsf{id}} = v \oplus x$ and $\mathsf{GlobalK}_{\mathbf{\Delta}}(\mathsf{id}, \Delta'_{\mathsf{id}}) = 1$ (therefore, $\Delta'_{\mathsf{id}} = \Delta_{\mathsf{id}}$). $\mathcal{S}_{RO}$ will then add the **chained** record $(f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i), z \oplus b \cdot \Delta_{\mathsf{id}})$ to $\mathcal{Q}_{SRO}$. In addition, this record is clearly unique due to the bijectivity of f_{in}.

We then prove claim (ii). By the definition of **chained** records, there exists $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$ with $f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i) = x$. It remains to prove its uniqueness and consistency (i.e., $z = y \oplus b \cdot \Delta_{\mathsf{id}}$).

Regarding its uniqueness, assume otherwise, i.e., there exist two records

$$(\mathsf{id}, w, i, b, z), (\mathsf{id}', w', i', b', z') \in \mathcal{Q}_F\,,$$

such that $x = f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i) = f_{\mathsf{in}}(w' \oplus \Delta_{\mathsf{id}'}, i')$. Since $\mathbf{F}$-query can only be made in phase 1 of $\overline{\mathcal{D}}^{\Sigma_{\text{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$, this means the subevent (ii) of Coll occurred in $\overline{\mathcal{D}}^{\Sigma_{\text{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and contradicts the condition that $(\mathbf{ro}, \mathbf{F}, \mathbf{R}^\circ, \mathbf{R})$ is good.

With the existence of $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$ with $f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i) = x$, the proof of $z = y \oplus b \cdot \Delta_{\mathsf{id}}$ (i.e., consistency) follows the same line as the above proof for claim (i).

Finally, claims (i) and (ii) show a one-to-one correspondence between **chained** records and records in $\mathcal{Q}_F$ and prove claim (iii). □

C.6.2 Associating real world executions via randomness.

Given a good tuple $(\mathbf{ro}, \mathbf{F}, \mathbf{R}^\circ, \mathbf{R})$ such that $\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\mathrm{end}} (\mathcal{Q}_F, \mathcal{Q}_{SRO}, \mathbf{\Delta}^\dagger)$ and another random oracle $\mathbf{RO} \in \mathcal{F}_{\mathcal{X},\mathcal{W}}$, we write $\mathbf{RO} \vdash \mathcal{Q}_{SRO}$, if $\mathbf{RO}(x) = y$ for all $(x, y) \in \mathcal{Q}_{SRO}$. The real world execution $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{RO},\mathbf{\Delta})}$ with $\mathbf{RO} \vdash \mathcal{Q}_{SRO}$ and the vector $\mathbf{\Delta}$ is then expected to proceed the same as the intermediate world execution $\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$. This is formally proven as follows.

Lemma 5. *Fix $\mathcal{D}$ in arbitrary. Let $(\mathbf{ro}, \mathbf{F}, \mathbf{R}^\circ, \mathbf{R})$ be a good random tuple with respect to $\overline{\mathcal{D}}$, such that $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\mathrm{end}} (\mathcal{Q}_F, \mathcal{Q}_{SRO}, \mathbf{\Delta}^\dagger)$. Then, for any random oracle $\mathbf{RO}$ with $\mathbf{RO} \vdash \mathcal{Q}_{SRO}$, $\overline{\mathcal{D}}$ obtains the same transcript of queries and answers in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{RO},\mathbf{\Delta})}$, and gives the same output.*

Proof. Imagine running the two executions $\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{RO},\mathbf{\Delta})}$ in parallel. Similarly to Lemma 4, we prove the claim via an induction on the sequence of queries of $\overline{\mathcal{D}}$. Assume that the sequences of queries and answers of $\overline{\mathcal{D}}$ are the same up to some point in the executions $\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{RO},\mathbf{\Delta})}$, and consider the next query. Since $\overline{\mathcal{D}}$ is deterministic, it issues the same query as the next. We distinguish four cases.

- Case 1: $\overline{\mathcal{D}}$'s next query is to $\mathsf{mL}_{\mathbf{\Delta}}(\mathsf{id}, P)$. Then, since the query is answered according to the same vector $\mathbf{\Delta}$, the responses obtained in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{RO},\mathbf{\Delta})}$ are identical.
- Case 2: $\overline{\mathcal{D}}$'s next query is to $O_2(\mathsf{id}, w, i, b)$. By the definition of Σ_{inter} , in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ the query necessarily occurs in phase 1, and it was replied by $z \leftarrow \mathbf{F}(\mathsf{id}, w, i, b)$. This means $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_{SRO}$ for the set $\mathcal{Q}_{SRO}$ yielded by $\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$. By Lemma 4 claim (i), there exists $(x, y) \in \mathcal{Q}_{SRO}$ with $x = f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i)$ and $y = z \oplus b \cdot \Delta_{\mathsf{id}}$.

  Then, in the real world execution $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{RO},\mathbf{\Delta})}$, the query $O_2(\mathsf{id}, w, i, b)$ is replied by $z' \leftarrow \mathbf{RO}(f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i)) \oplus b \cdot \Delta_{\mathsf{id}}$. Since $\mathbf{RO} \vdash \mathcal{Q}_{SRO}$, the answer z' equals the aforementioned response z in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$.
- Case 3: $\overline{\mathcal{D}}$'s next query is to $O_1(x)$. Since we assumed $\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\mathrm{end}} (\mathcal{Q}_F, \mathcal{Q}_{SRO}, \mathbf{\Delta}^\dagger)$, there necessarily exists a corresponding record (x, y) in $\mathcal{Q}_{SRO}$, and y is the response in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$. In $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{RO},\mathbf{\Delta})}$, the response $y' \leftarrow \mathbf{RO}(x)$ is given by the random oracle $\mathbf{RO}$, and equals y due to the assumption $\mathbf{RO} \vdash \mathcal{Q}_{SRO}$.

In any case, the next answers are the same in the two executions. Therefore, $\overline{\mathcal{D}}$ gains the same transcript of queries and answers in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{RO},\mathbf{\Delta})}$. Since $\overline{\mathcal{D}}$ is deterministic, its output is determined by this transcript, and is thus the same in the two executions. □

C.6.3 Completing the indistinguishability proof.

As implied by Lemma 5, $\overline{\mathcal{D}}$'s output in the intermediate world is "characterized" by the final sets $\mathcal{Q}_F$, $\mathcal{Q}_{SRO}$ and the revealed keys $\mathbf{\Delta}$: if two tuples $(\mathbf{ro}, \mathbf{F}, \mathbf{R}^\circ, \mathbf{R})$ and $(\mathbf{ro}', \mathbf{F}', \mathbf{R}^{\circ'}, \mathbf{R}')$ are such that $\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\mathrm{end}} (\mathcal{Q}_F, \mathcal{Q}_{SRO}, \mathbf{\Delta}^\dagger)$ and $\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro}',\mathbf{F}',\mathbf{R}^{\circ'},\mathbf{R}')} \xrightarrow{\mathrm{end}} (\mathcal{Q}_F, \mathcal{Q}_{SRO}, \mathbf{\Delta}^\dagger)$, then $\overline{\mathcal{D}}$ outputs the same in the executions $\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$ and $\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro}',\mathbf{F}',\mathbf{R}^{\circ'},\mathbf{R}')}$. By this, let Θ_1 be the set of triples $(\mathcal{Q}_F, \mathcal{Q}_{SRO}, \mathbf{\Delta}^\dagger)$ such that there exists at least one good intermediate world tuple $(\mathbf{ro}, \mathbf{F}, \mathbf{R}^\circ, \mathbf{R})$ such that: (i) $\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})} \xrightarrow{\mathrm{end}} (\mathcal{Q}_F, \mathcal{Q}_{SRO}, \mathbf{\Delta}^\dagger)$, and (ii) $\overline{\mathcal{D}}$ outputs 1 in

$\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}$. Then, it can be seen

$$\begin{aligned}&\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}=1\wedge(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})\text{ is good}\right]\\&=\sum_{(\mathcal{Q}_F,\mathcal{Q}_{SRO},\boldsymbol{\Delta}^\dagger)\in\Theta_1}\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}\xrightarrow{\text{end}}(\mathcal{Q}_F,\mathcal{Q}_{SRO},\boldsymbol{\Delta}^\dagger)\right].\end{aligned}\tag{28}$$

On the other hand, fix a triple $(\mathcal{Q}_F,\mathcal{Q}_{SRO},\boldsymbol{\Delta}^\dagger)$ in arbitrary. Then, Lemma 5 implies that if two random oracle $\mathbf{RO}$ and $\mathbf{RO}'$ are such that $\mathbf{RO}\vdash\mathcal{Q}_{SRO}$ and $\mathbf{RO}'\vdash\mathcal{Q}_{SRO}$, then $\overline{\mathcal{D}}$ outputs the same in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{RO},\boldsymbol{\Delta}^\dagger)}$ and $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{RO}',\boldsymbol{\Delta}^\dagger)}$. Therefore,

$$\begin{aligned}&\Pr_{\mathbf{RO},\boldsymbol{\Delta}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{RO},\boldsymbol{\Delta})}=1\right]\\&\le\sum_{(\mathcal{Q}_F,\mathcal{Q}_{SRO},\boldsymbol{\Delta}^\dagger)\in\Theta_1}\Pr_{\mathbf{RO},\boldsymbol{\Delta}}\left[\mathbf{RO}\vdash\mathcal{Q}_{SRO}\wedge\boldsymbol{\Delta}=\boldsymbol{\Delta}^\dagger\right].\end{aligned}\tag{29}$$

Lemma 6. *Fix any $(q_C,q_P,M,q_{L,max},\mu)$-bounded S-mTCCRL distinguisher $\mathcal{D}$, and consider the corresponding $\overline{\mathcal{D}}$ that completes all chains. Then, for any*

$$(\mathcal{Q}_{SRO},\mathcal{Q}_F,\boldsymbol{\Delta}^\dagger)\in\Theta_1\,,$$

it holds

$$\frac{\Pr_{\mathbf{RO},\boldsymbol{\Delta}}\left[\mathbf{RO}\vdash\mathcal{Q}_{SRO}\wedge\boldsymbol{\Delta}=\boldsymbol{\Delta}^\dagger\right]}{\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}\xrightarrow{\text{end}}(\mathcal{Q}_F,\mathcal{Q}_{SRO},\boldsymbol{\Delta}^\dagger)\right]}=1.$$

Proof. Denote by α and β the number of **unchained** and **chained** records in $\mathcal{Q}_{SRO}$ respectively. Then, $|\mathcal{Q}_{SRO}|=\alpha+\beta$, $|\mathcal{Q}_F|=\beta$ by Lemma 4 claim (iii), and it holds

$$\Pr_{\mathbf{RO},\boldsymbol{\Delta}}\left[\mathbf{RO}\vdash\mathcal{Q}_{SRO}\wedge\boldsymbol{\Delta}=\boldsymbol{\Delta}^\dagger\right]=\frac{1}{|\mathcal{W}|^{\alpha+\beta}}\times\frac{1}{\prod_{\mathsf{id}=1}^{u}|\mathcal{R}_{\mathsf{id}}|}.\tag{30}$$

Fix now any triple $(\mathcal{Q}_F,\mathcal{Q}_{SRO},\boldsymbol{\Delta}^\dagger)\in\Theta_1$. One can check that for any tuple $(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})$, $\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}\xrightarrow{\text{end}}(\mathcal{Q}_F,\mathcal{Q}_{SRO},\boldsymbol{\Delta}^\dagger)$ if and only if: (i) the vector sampled and revealed at the end of phase 1 equals $\boldsymbol{\Delta}^\dagger$ (i.e., $\boldsymbol{\Delta}=\boldsymbol{\Delta}^\dagger$), and (ii) $\mathbf{F}(\mathsf{id},w,i,b)=z$ for all $(\mathsf{id},w,i,b,z)\in\mathcal{Q}_F$, which is denoted $\mathbf{F}\vdash\mathcal{Q}$ (the number of which is β); (iii) $\mathbf{ro}(x)=y$ for all **unchained** $(x,y)\in\mathcal{Q}_{SRO}$ (the number of which is α). Conditions (ii) and (iii) indicate that the randomness gained by $\mathcal{S}_{RO}$ and $\overline{\mathcal{D}}$ from $\mathbf{F}$ and $\mathbf{ro}$ is consistent with the "non-redundant" records in $\mathcal{Q}_F$ and $\mathcal{Q}_{SRO}$.

It thus follows that

$$\begin{aligned}&\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}\xrightarrow{\text{end}}(\mathcal{Q}_F,\mathcal{Q}_{SRO},\boldsymbol{\Delta}^\dagger)\right]\\&=\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R}}\left[\mathbf{ro}(x)=y\text{ for all }\textbf{unchained}\ (x,y)\in\mathcal{Q}_{SRO}\wedge\mathbf{F}\vdash\mathcal{Q}_F\wedge\boldsymbol{\Delta}=\boldsymbol{\Delta}^\circ\right]\\&=\frac{1}{|\mathcal{W}|^{\alpha}}\times\frac{1}{|\mathcal{W}|^{\beta}}\times\frac{1}{\prod_{\mathsf{id}=1}^{u}|\mathcal{R}_{\mathsf{id}}|}.\end{aligned}\tag{31}$$

Gathering Eqs. (30) and (31) yields

$$\frac{\Pr_{\mathbf{RO},\boldsymbol{\Delta}}\left[\mathbf{RO}\vdash\mathcal{Q}_{SRO}\wedge\boldsymbol{\Delta}=\boldsymbol{\Delta}^\dagger\right]}{\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}\xrightarrow{\text{end}}(\mathcal{Q}_F,\mathcal{Q}_{SRO},\boldsymbol{\Delta}^\dagger)\right]}=1.$$

This completes the proof. □

Lemma 7. *For any $(q_C, q_P, M, q_{L,max}, \mu)$-bounded S-mTCCRL distinguisher $\mathcal{D}$, it holds*

$$\left|\Pr_{\mathbf{RO},\mathbf{\Delta}}\left[\mathcal{D}^{\Sigma_{\mathrm{re}}(\mathbf{RO},\mathbf{\Delta})}=1\right]-\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R}}\left[\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}=1\right]\right| \le \frac{(M+1)(q_{L,max}+1)}{C}\cdot\left(\mu q_P+\frac{(\mu-1)\cdot q_C}{2}\right).$$

Proof. Gathering Eqs. (24), (28) and (29) and Lemma 6 yields

$$\begin{aligned}
&\left|\Pr_{\mathbf{RO},\mathbf{\Delta}}\left[\mathcal{D}^{\Sigma_{\mathrm{re}}(\mathbf{RO},\mathbf{\Delta})}=1\right]-\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R}}\left[\mathcal{D}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}=1\right]\right| \\
\le&\left|\Pr_{\mathbf{RO},\mathbf{\Delta}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{RO},\mathbf{\Delta})}=1\right]-\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}=1\right]\right| \\
\le&\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}=1\wedge(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})\text{ is bad}\right] \\
&+\Big|\Pr_{\mathbf{RO},\mathbf{\Delta}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{RO},\mathbf{\Delta})}=1\right] \\
&\qquad-\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}=1\wedge(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})\text{ is good}\right]\Big| \\
\le&\Pr_{\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R}}\left[(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})\text{ is bad}\right] \\
&+\Big|\sum_{(\mathcal{Q}_{SRO},\mathcal{Q}_F,\mathbf{\Delta}^\dagger)\in\Theta_1}\Pr_{\mathbf{RO},\mathbf{\Delta}}\left[\mathbf{RO}\vdash\mathcal{Q}_{SRO}\wedge\mathbf{\Delta}=\mathbf{\Delta}^\dagger\right] \qquad\text{(Eq. (29))} \\
&\qquad-\sum_{(\mathcal{Q}_{SRO},\mathcal{Q}_F,\mathbf{\Delta}^\dagger)\in\Theta_1}\Pr\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{inter}}(\mathbf{ro},\mathbf{F},\mathbf{R}^\circ,\mathbf{R})}\xrightarrow{\text{end}}(\mathcal{Q}_F,\mathcal{Q}_{SRO},\mathbf{\Delta}^\dagger)\right]\Big| \qquad\text{(Eq. (28))} \\
\le&\frac{(M+1)(q_{L,max}+1)}{C}\cdot\left(\mu q_P+\frac{(\mu-1)\cdot q_C}{2}\right) \qquad\text{(Eq. (24) and Lemma 6),}
\end{aligned}$$

as claimed. □

This establishes the indistinguishability of Σ_{inter} and Σ_{re} and completes the proof of Theorem 3.

D Proof of Theorem 1 (S-mTCCRL Security of $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$)

Recall that $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}(x,i) = f_{\mathsf{out}}\Big(x, i, \mathbf{E}(+, g(i), f_{\mathsf{in}}(w,i))\Big)$. To prove Theorem 1, we first describe the simulator in Sect. D.1. We then sketch our proof ideas in Sect. D.2. Our main proof is in Sect. D.3—Sect. D.6, and we refer to Sect. D.2 for a detailed roadmap.

D.1 Constructing the simulator

D.1.1 Explicit randomness.

To handily describe lazy sampling during simulation, we follow [CHK+16, ABD+13, CS15] and make the randomness used by $\mathcal{S}_G$ explicit through an ideal cipher $\mathbf{ic} : \{+,-\}\times\{0,1\}^\kappa\times\{0,1\}^n \to \{0,1\}^n$. Namely, $\mathcal{S}_G$ queries $\mathbf{ic}$ to have random values rather than straightforwardly sampling. We denote by $\mathcal{S}_G^{\mathsf{GlobalK}_{\mathbf{\Delta}},\mathbf{ic}}$ the simulator that queries $(\mathsf{GlobalK}_{\mathbf{\Delta}}, \mathbf{ic})$ to emulate. (As argued in [ABD+13], explicit randomness is an equivalent formalism of lazying sampling.)

D.1.2 Variables of the simulator.

$\mathcal{S}_G$ maintains a list $\mathcal{Q}_{SE}$ to keep track of already encountered cipher queries. The list $\mathcal{Q}_{SE}$ is initialized to empty. As the execution proceeds, $\mathcal{S}_G$ adds new records of the form (k,x,y) to $\mathcal{Q}_{SE}$, which means $\mathcal{S}_G$ has answered $O_1(+,k,x)$ with y or $O_1(-,k,y)$ with x.

To ensure that $\mathcal{Q}_{SE}$ is always consistent with a blockcipher, $\mathcal{S}_G$ preforms checking before adding every triple to $\mathcal{Q}_{SE}$. When the to-be-added triple breaks such consistency, $\mathcal{S}_G$ *aborts*. In such cases, we assume that the distinguisher $\mathcal{D}$ is consequently notified, and that $\mathcal{D}$ can then return its own output bit. Such abortions enables distinguishing (since the real world experiment never aborts), and a crucial step of our proof will be to bound the probability of abortion.

Besides $\mathcal{Q}_{SE}$, $\mathcal{S}_G$ maintains another list $\mathcal{Q}_{SF}$ to keep the **F**-query records it received via the procedure *PhaseSwitch*($\mathcal{Q}_F$).

D.1.3 Simulating strategy.

The simulator provides an interface $O_1 : \{+,-\} \times \{0,1\}^\kappa \times \{0,1\}^n \to \{0,1\}^n$ that mimics $\mathbf{E} \in \mathcal{E}_{\{0,1\}^\kappa,\{0,1\}^n}$ and is consistent with **F** when necessary. Upon $\mathcal{D}$ querying $O_1(\delta, k, v)$, $\mathcal{S}_{RO}$ first checks if its answer v' has been defined (i.e., $(k, v, v') \in \mathcal{Q}_{SE}$ when $\delta = +$, and $(k, v', v) \in \mathcal{Q}_{SE}$ when $\delta = -$), and returns v' if it is the case.

Otherwise, the (newly simulated) answer may have to be consistent with some previously appeared **F**-query record $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_{SF}$. It depends on δ:

- When $\delta = +$, $\mathcal{S}_G$ seeks for $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_{SF}$ with $g(i) = k$ and $v = f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i)$ (checked by querying $\mathsf{GlobalK}_{\boldsymbol{\Delta}}(\mathsf{id}, f_{\mathsf{in}}^{-1}(i, v) \oplus w) \stackrel{?}{=} 1$). Once such a record $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_{SF}$ is found, $\mathcal{S}_G$ adds a record (k, v, v') to $\mathcal{Q}_{SE}$ that is consistent with $(\mathsf{id}, w, i, b, z)$. Otherwise, $O_1(+, k, v)$ can be defined "freely": $\mathcal{S}_G$ acquires the answer $v' \leftarrow \mathbf{ic}(+, k, v)$ from the ideal cipher and adds (k, v, v') to $\mathcal{Q}_{SE}$.
- When $\delta = -$, $\mathcal{S}_G$ seeks for $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_{SF}$ with $g(i) = k$ and $f_{\mathsf{out}}(w \oplus \Delta_{\mathsf{id}}, i, v) \oplus b \cdot \Delta_{\mathsf{id}} = z$ (checked by querying $\mathsf{GlobalK}_{\boldsymbol{\Delta}}(\mathsf{id}, kd_y(w, i, b, y, z)) \stackrel{?}{=} 1$—the key extracting function kd_y helps). Once such a record $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_{SF}$ is found, $\mathcal{S}_G$ adds a record (k, v', v) to $\mathcal{Q}_{SE}$ that is consistent with $(\mathsf{id}, w, i, b, z)$. Otherwise, $\mathcal{S}_G$ acquires the answer $v' \leftarrow \mathbf{ic}(-, k, v)$ from the ideal cipher and adds (k, v', v) to $\mathcal{Q}_{SE}$.

In any case, $\mathcal{S}_G$ finally replies $\mathcal{D}$ with the newly defined answer v'.

Finally, the procedure *PhaseSwitch*($\mathcal{Q}_F$), once called, copies the records from $\mathcal{Q}_F$ to its own list $\mathcal{Q}_{SF}$.

D.1.4 Pseudocode description.

We now provide pseudocode to formally describe the simulator $\mathcal{S}_G^{\mathsf{GlobalK}_{\boldsymbol{\Delta}},\mathbf{ic}}$ as explained above.

Simulator $\mathcal{S}_G^{\mathsf{GlobalK}_{\boldsymbol{\Delta}},\mathbf{ic}}$
Lists $\mathcal{Q}_{SE}$ and $\mathcal{Q}_{SF}$, initialized to empty

public procedure $O_1(\delta, k, v)$
if $\delta = +$ **then** // forward blockcipher query
  $x \leftarrow v$ // Rename to improve clarity.
  if $(k, x, y) \in \mathcal{Q}_{SE}$ for some $y \in \{0,1\}^n$ **then** // $O_1(+, k, x)$ has been defined
    return y
  end if
  // $O_1(+, k, x)$ is new and has to be defined
  for all $(\mathsf{id}, i, w, b, z) \in \mathcal{Q}_{SF}$ such that $g(i) = k$ **do**
// Since $\mathcal{Q}_{SF}$ is initially empty, in phase 1 $\mathcal{S}_G$ will skip this for loop and simply respond $O_1(+, k, x)$ with $y \leftarrow \mathbf{ic}(+, k, x)$.
    $\Delta'_{\mathsf{id}} \leftarrow f_{\mathsf{in}}^{-1}(i, x) \oplus w$ // A candidate for the id-th key
    $b_R \leftarrow \mathsf{GlobalK}_{\boldsymbol{\Delta}}(\mathsf{id}, \Delta'_{\mathsf{id}})$

```
      if b_R = true then
        y ← f_out^{-1}(w ⊕ Δ'_id, i, z ⊕ b · Δ'_id)
        if (k, x', y) ∈ Q_SE for some x' ∈ {0,1}^n then
          abort
        end if
        Add (k, x, y) to Q_SE
        return y
      end if
    end for
    y ← ic(+, k, x)
    if (k, x', y) ∈ Q_SE for some x' ∈ {0,1}^n then
      abort
    end if
    Add (k, x, y) to Q_SE
    return y
  else                    // δ = −, backward blockcipher query
    y ← v                 // Rename to improve clarity.
    if (k, x, y) ∈ Q_SE for some x ∈ {0,1}^n then
      return x
    end if
    // O_1(−, k, y) is new
    for all (id, i, w, b, z) ∈ Q_SF such that g(i) = k do
  // Since Q_SF is initially empty, in phase 1 S_G will skip this for loop and simply respond O_1(−, k, y) with
  x ← ic(−, k, y).
      Δ'_id ← kd_y(w, i, b, y, z)
      b_R ← GlobalK_Δ(id, Δ'_id)
      if b_R = true then
        x ← f_in(w ⊕ Δ'_id, i)
        if (k, x, y') ∈ Q_SE for some y' ∈ {0,1}^n then
          abort
        end if
        Add (k, x, y) to Q_SE
        return x
      end if
    end for
    x ← ic(−, k, y)
    if (k, x, y') ∈ Q_SE for some y' ∈ {0,1}^n then
      abort
    end if
    Add (k, x, y) to Q_SE
    return x
  end if

  public procedure PhaseSwitch(Q_F)
  Q_SF ← Q_F
```

D.1.5 Complexity of the simulator.

The following lemma shows that the simulator runs in time polynomial in the number of queries it receives.

Lemma 8. *For any $(q_C, q_P, M, q_{L,max}, \mu)$-bounded S-mTCCRL distinguisher $\mathcal{D}$, the simulator $\mathcal{S}_G^{\mathsf{GlobalK}_\Delta, \mathbf{ic}}$ makes at most q_P queries to its randomness* **ic** *and at most μq_P queries to $\mathsf{GlobalK}_\Delta$.*

Proof. The claim on simulator queries to **ic** is clear from the design of $\mathcal{S}_G^{\mathsf{GlobalK}_{\boldsymbol{\Delta}},\mathbf{ic}}$. Regarding the other claim, consider any query $O_1(\delta, k, v)$ made by $\mathcal{D}$. By our assumption regarding the parameter μ and the injectivity of g, the number of records $(\mathsf{id}, i, w, b, z) \in \mathcal{Q}_{SF}$ with $g(i) = k$ is at most μ. For each of such records, $\mathcal{S}_G$ makes at most 1 query to $\mathsf{GlobalK}_{\boldsymbol{\Delta}}$. Thus the claim. □

D.2 Proof idea

In all the following, we consider some fixed distinguisher $\mathcal{D}$, and assume that it is deterministic (this is wlog since we consider computationally unbounded distinguishers). To simplify notations and make randomness explicit, we will denote $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$ the ideal world execution (which, according to Fig. 1, should have been $\mathbf{Exp}^{\text{S-mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\overrightarrow{\mathcal{R}},\mathcal{P},u,\mathcal{D},\mathcal{S}_G,1}$) in which:

(i) The simulator $\mathcal{S}_G$ uses $\mathbf{ic} \in \mathcal{E}_{\{0,1\}^\kappa,\{0,1\}^n}$ as its explicit randomness, and

(ii) The random function oracle is $\mathbf{F} \in \mathcal{KWC}$, and

(iii) The key vector sampled at the beginning equals $\boldsymbol{\Delta} = (R_1, ..., R_u) \in \overrightarrow{\mathcal{R}}$.

It is easy to see the tuple of random variables $(\mathbf{ic}, \mathbf{F}, \mathbf{R})$ has captured all the randomness of the ideal world execution. Similarly, denote $\mathcal{D}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$ the real world execution (i.e., $\mathbf{Exp}^{\text{S-mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\overrightarrow{\mathcal{R}},\mathcal{P},u,\mathcal{D},\mathcal{S}_G,0}$ in Fig. 1) in which the ideal cipher is $\mathbf{E} \in \mathcal{E}_{\{0,1\}^\kappa,\{0,1\}^n}$ and the key vector in use equals $\boldsymbol{\Delta} = (R_1, ..., R_u)$.

In ideal world executions, the distinguisher $\mathcal{D}$ may observe two types of inconsistencies:

(i) First, $\mathcal{S}_G$ may abort.

(ii) Second, (even if $\mathcal{S}_G$ does not abort) when the keys $\boldsymbol{\Delta} = (R_1, ..., R_u)$ are revealed at the end of phase 1, there exist two query records $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_{SF}$ and $(x, y) \in \mathcal{Q}_{SRO}$ such that $x = f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i)$ or $z = f_{\mathsf{out}}(w \oplus \Delta_{\mathsf{id}}, i, y) \oplus b \cdot \Delta_{\mathsf{id}}$. In this case, $\mathcal{S}_{RO}$ won't (re)define $O_1(+, k, x)$ or $O_1(-, k, y)$ to ensure consistency with $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_{SF}$ upon subsequent adversarial queries, and such two records (once existed) enable distinguishing.

As long as neither of the above two cases occurs, the simulated queries and answers in $\mathcal{Q}_{SE}$ are expected to be an ideal cipher consistent with $\mathbf{F}$. Therefore, we will first establish an upper bound on the probability of the two cases. The typical approach (e.g., see [CHK+16, ABD+13]) would be to identify some bad events in the *ideal world executions*, such that $\mathcal{S}_G$ does not abort as long as these events do not occur. However, as mentioned in Sect. 2.3 and Sect. 5.2, we take an "opposite" approach and define a closely related event Chain in the *real world executions* in Sect. D.3. We then relate the occurrence of Chain to breaking the mTCCRL security of $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$, and this bounds $\Pr[\mathsf{Chain} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}]$.

Then, in Sect. D.4, we employ the randomness mapping argument (developed in [CHK+16, ABD+13]) to associate some ideal world executions to the "good" real world executions in which Chain does not occur. We then show that in these associated ideal world executions, the simulator does not abort and could give consistent outputs. Finally, in Sect. D.5 we prove that "good" real world executions and associated ideal world executions occur with sufficiently close probabilities, and this completes the indistinguishability proof.

D.3 Event Chain in real world executions

D.3.1 Motivations.

While we define and analyze bad events in real world executions, it might ease understanding by beginning the discussion with the ideal world. To this end, let $\mathcal{Q}_E = \{(k_1, x_1, y_1), \ldots\}$ and $\mathcal{Q}_F = \{(\mathsf{id}_1, w_1, i_1, b_1, z_1), \ldots\}$ be the transcripts of queries and answers of $\mathcal{D}$—as defined in Sect. 3.1. Intuitively, in an ideal world execution $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$, the simulator $\mathcal{S}_G$ does not abort as long as none of the following events occur:

- (B-1) At the end of phase 1 of $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$, there exists a record $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$ such that $(g(i), f_{\mathsf{in}}(\Delta_{\mathsf{id}} \oplus w, i), \star) \in \mathcal{Q}_E$.
- (B-2) At the end of phase 1 of $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$, there exists a record $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$ such that $(g(i), \star, f^{-1}_{\mathsf{out}}(\Delta_{\mathsf{id}} \oplus w, i, z \oplus b \cdot \Delta_{\mathsf{id}})) \in \mathcal{Q}_E$.
- (B-3) At the end of phase 1 of $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$, there exist distinct records $(\mathsf{id}, w, i, b, z)$ and $(\mathsf{id}', w', i', b', z') \in \mathcal{Q}_F$ such that $g(i) = g(i')$ and $f_{\mathsf{in}}(\Delta_{\mathsf{id}} \oplus w, i) = f_{\mathsf{in}}(\Delta_{\mathsf{id}'} \oplus w', i')$.
- (B-4) At the end of phase 1 of $\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$, there exist distinct records $(\mathsf{id}, w, i, b, z)$ and $(\mathsf{id}', w', i', b', z') \in \mathcal{Q}_F$ such that $g(i) = g(i')$ and $f^{-1}_{\mathsf{out}}(\Delta_{\mathsf{id}} \oplus w, i, z \oplus b \cdot \Delta_{\mathsf{id}}) = f^{-1}_{\mathsf{out}}(\Delta_{\mathsf{id}'} \oplus w', i', z' \oplus b' \cdot \Delta_{\mathsf{id}'})$.

For example, if (B-1) occurs and $\mathcal{D}$ then queries $O_1(-, g(i), f_{\mathsf{in}}(\Delta_{\mathsf{id}} \oplus w, i))$ for $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$, then $\mathcal{S}_G$ has to reply according to the record $(g(i), f_{\mathsf{in}}(\Delta_{\mathsf{id}} \oplus w, i), y)$ already in $\mathcal{Q}_E$, and $\mathcal{S}_G$ may be unable to ensure consistency $f_{\mathsf{out}}(w \oplus \Delta_{\mathsf{id}}, i, y) \oplus b \cdot \Delta_{\mathsf{id}} = z$.

Instead of analyzing the four events in the ideal world, we now switch to the real world. It is easy to see that in a real world execution $\mathcal{D}^{\Sigma_{\mathrm{re}}(\mathbf{E},\mathbf{\Delta})}$,

- (B-1) occurs if and only if (B-2) occurs, and
- (B-3) occurs if and only if (B-4) occurs.

We thereby identify $\mathsf{Chain1} = \text{(B-1)} \wedge \text{(B-2)}$ and $\mathsf{Chain2} = \text{(B-3)} \wedge \text{(B-4)}$ as the two crucial real world events.

D.3.2 Definition.

Let $\mathcal{Q}_E = \{(k_1, x_1, y_1), \ldots\}$ and $\mathcal{Q}_F = \{(\mathsf{id}_1, w_1, i_1, b_1, z_1), \ldots\}$ be the transcripts of queries and answers of $\mathcal{D}$ (see Sect. 3.1). Motivated by the above, we identify two bad events in a real world execution $\mathcal{D}^{\Sigma_{\mathrm{re}}(\mathbf{E},\mathbf{\Delta})}$ as follows.

- Chain1: At the end of phase 1 of $\mathcal{D}^{\Sigma_{\mathrm{re}}(\mathbf{E},\mathbf{\Delta})}$, there exist $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$ and $(k, x, y) \in \mathcal{Q}_E$ such that $g(i) = k$, $f_{\mathsf{in}}(\Delta_{\mathsf{id}} \oplus w, i) = x$ and $f^{-1}_{\mathsf{out}}(\Delta_{\mathsf{id}} \oplus w, i, z \oplus b \cdot \Delta_{\mathsf{id}}) = y$;
- Chain2: At the end of phase 1 of $\mathcal{D}^{\Sigma_{\mathrm{re}}(\mathbf{E},\mathbf{\Delta})}$, there exist distinct records $(\mathsf{id}, w, i, b, z)$ and $(\mathsf{id}', w', i', b', z') \in \mathcal{Q}_F$ such that $g(i) = g(i')$, $f_{\mathsf{in}}(\Delta_{\mathsf{id}} \oplus w, i) = f_{\mathsf{in}}(\Delta_{\mathsf{id}'} \oplus w', i')$ and $f^{-1}_{\mathsf{out}}(\Delta_{\mathsf{id}} \oplus w, i, z \oplus b \cdot \Delta_{\mathsf{id}}) = f^{-1}_{\mathsf{out}}(\Delta_{\mathsf{id}'} \oplus w', i', z' \oplus b' \cdot \Delta_{\mathsf{id}'})$.

Define $\mathsf{Chain} := \mathsf{Chain1} \vee \mathsf{Chain2}$. It can be proven that if $\mathcal{D}$ can trigger the event Chain in $\mathcal{D}^{\Sigma_{\mathrm{re}}(\mathbf{E},\mathbf{\Delta})}$, then for any constant $T > 0$ we can construct a $(q_C, q_P + T, M, q_{L,max}, \mu)$-bounded distinguisher $\mathcal{B}$

breaking the mTCCRL security of $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$. Formally,

$$\begin{aligned}&\Pr_{\mathbf{E},\boldsymbol{\Delta}}\left[\mathsf{Chain} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}\right]\\ &\leq 2\mathbf{Adv}^{\mathrm{mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\vec{\mathcal{R}},\mathcal{P},u}(\mathcal{B}) + \frac{2\mu(q_P+q_C)}{2^n} + \frac{2}{T}.\end{aligned} \tag{32}$$

For clearness, the proof is deferred to Sect. D.6.

D.3.3 Good real world executions and properties.

We say that a tuple of random variables $(\mathbf{E}, \boldsymbol{\Delta})$ is *bad*, if Chain occurs in the execution of $\mathcal{D}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$. It thus holds

$$\Pr_{\mathbf{E},\boldsymbol{\Delta}}\left[(\mathbf{E}, \boldsymbol{\Delta}) \text{ is bad}\right] \leq \Pr_{\mathbf{E},\boldsymbol{\Delta}}\left[\mathsf{Chain} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}\right]. \tag{33}$$

We say that a distinguisher *completes all chains*, if, at the end of the execution, it emulates a call to $O_1(+, g(i), f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i))$ for every the $\mathbf{F}$-query records $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_{SF}$ made during the execution. Clearly, for any "raw" distinguisher $\mathcal{D}$ we can construct a corresponding $\overline{\mathcal{D}}$ that completes all chains.

Fix a distinguisher $\mathcal{D}$ in arbitrary, and consider the corresponding $\overline{\mathcal{D}}$ that completes all chains. By our security definition (see Fig. 1), the triple (or transcript) $(\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$ summarizes all information gained by $\overline{\mathcal{D}}$ during an execution (regardless of real or ideal), where $\mathcal{Q}_E = \{(k_1, x_1, y_1), \ldots\}$ and $\mathcal{Q}_F = \{(\mathsf{id}_1, w_1, i_1, b_1, z_1), \ldots\}$ are as defined in Sect. 4, and $\boldsymbol{\Delta} = (R_1, ..., R_u)$ is the key revealed in the execution.

Let Θ be the set of all *good* transcripts $(\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$, i.e., that can be obtained by $\overline{\mathcal{D}}$ in real world executions $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$ using good random pairs $(\mathbf{E}, \boldsymbol{\Delta})$. For any $(\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta}) \in \Theta$, denote by $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})} \to (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$ the event that $\overline{\mathcal{D}}$ obtains the transcripts $(\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$ at the end of the execution $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$.

Since $\overline{\mathcal{D}}$ is deterministic, if two random pairs $(\mathbf{E}, \boldsymbol{\Delta})$ and $(\mathbf{E}', \boldsymbol{\Delta}')$ yield the same transcript $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})} \to (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$ and $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E}',\boldsymbol{\Delta}')} \to (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$, then $\overline{\mathcal{D}}$ gives the same output in the two executions: because the output of $\overline{\mathcal{D}}$ is fully determined by its query transcript $(\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$. With this in mind, let $\Theta_1 \subseteq \Theta$ be the set of transcripts $(\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$ such that $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$ outputs 1 for some random pair $(\mathbf{E}, \boldsymbol{\Delta})$ such that $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})} \to (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta}) \in \Theta_1$. By the above, it holds

$$\Pr_{\mathbf{E},\boldsymbol{\Delta}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})} = 1\right] = \sum_{(\mathcal{Q}_E,\mathcal{Q}_F,\boldsymbol{\Delta})\in\Theta_1} \Pr_{\mathbf{E},\boldsymbol{\Delta}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})} \to (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})\right]. \tag{34}$$

Consider a good transcript $(\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta}) \in \Theta$, where $\boldsymbol{\Delta} = (R_1, ..., R_u)$. A record $(k, x, y) \in \mathcal{Q}_E$ such that there exists $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$ with $g(i) = k$ and $f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i) = x$ is called **chained**. Otherwise, $(k, x, y) \in \mathcal{Q}_E$ is **unchained**. In the former case, it always holds $f_{\mathsf{out}}(w \oplus \Delta_{\mathsf{id}}, i, y) \oplus b \cdot \Delta_{\mathsf{id}} = z$ (since it is a real world transcript). Note that if $(\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta}) \in \Theta$, then the event Chain does not occur in phase 1 of the corresponding real world executions, and **chained** records in $\mathcal{Q}_E$ are necessarily generated in phase 2. Below we prove one more useful property: the number of **chained** records in $\mathcal{Q}_E$ equals $|\mathcal{Q}_F|$. For this, let

$$\begin{aligned}\mathcal{Q}_E[k] &:= \{(x, y) : (k, x, y) \in \mathcal{Q}_E\},\\ \mathcal{Q}_F[i] &:= \{(\mathsf{id}, w, b, z) : (\mathsf{id}, w, i, b, z) \in \mathcal{Q}_E\}.\end{aligned} \tag{35}$$

Lemma 9. *Fix $\mathcal{D}$ in arbitrary. Let $(\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta}) \in \Theta$ for the good transcript set Θ with respect to $\overline{\mathcal{D}}$. Then, for all $i \in \mathcal{I}$, the number of* **chained** *records in $\mathcal{Q}_E[g(i)]$ equals $|\mathcal{Q}_F[i]|$, the number of $\mathbf{F}$-query records with tweak i in $\mathcal{Q}_F$.*

Proof. Let $(\mathbf{E}, \boldsymbol{\Delta})$ be an arbitrary tuple such that $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})} \to (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$.

We first prove that for every $(\mathsf{id}, w, b, z) \in \mathcal{Q}_F[i]$, there exists a unique **chained** record

$$(g(i), x, y) \in \mathcal{Q}_E\,,$$

with $f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i) = x$ and $f_{\mathsf{out}}(w \oplus \Delta_{\mathsf{id}}, i, y) \oplus b \cdot \Delta_{\mathsf{id}} = z$. Since $\overline{\mathcal{D}}$ completes all chains, in the real world execution $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$ it will query $\mathbf{E}(+, g(i), f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i))$ at some point, after which the **chained** record $(g(i), f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i), f^{-1}_{\mathsf{out}}(w \oplus \Delta_{\mathsf{id}}, i, z \oplus b \cdot \Delta_{\mathsf{id}}))$ is in $\mathcal{Q}_E$.

We then prove that for every **chained** record (k, x, y) in $\mathcal{Q}_E$, there exists a unique $i \in \mathcal{I}$ and a unique record $(\mathsf{id}, w, b, z) \in \mathcal{Q}_F[i]$ such that $g(i) = k$, $f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i) = x$ and $f_{\mathsf{out}}(w \oplus \Delta_{\mathsf{id}}, i, y) \oplus b \cdot \Delta_{\mathsf{id}} = z$. The existence of such records immediately follows from the definition of **chained** records, and it remains to prove its uniqueness. The uniqueness of i with $g(i) = k$ is obvious by the injectivity of g. Regarding the record, assume otherwise, i.e., there exist two records $(\mathsf{id}, w, b, z), (\mathsf{id}', w', b', z') \in \mathcal{Q}_F[i]$ such that $x = f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i) = f_{\mathsf{in}}(w' \oplus \Delta_{\mathsf{id}'}, i')$ and $y = f^{-1}_{\mathsf{out}}(w \oplus \Delta_{\mathsf{id}}, i, z \oplus b \cdot \Delta_{\mathsf{id}}) = f^{-1}_{\mathsf{out}}(w' \oplus \Delta_{\mathsf{id}'}, i', z' \oplus b \cdot \Delta_{\mathsf{id}'})$. Since $\mathbf{F}$-query can only be made in phase 1 of $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$, this means the event $\mathsf{Chain2}$ occurred in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$ and contradicts the condition $(\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta}) \in \Theta$ ($(\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$ is good). This completes the proof. □

D.4 Associating ideal world executions

Given a good transcript $(\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta}) \in \Theta$, $\boldsymbol{\Delta} = (R_1, ..., R_u)$, and two ideal world random primitives $\mathbf{ic}$ and $\mathbf{F}$, we write $(\mathbf{ic}, \mathbf{F}) \cong (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$, if the following are fulfilled:

- $\mathbf{F}(\mathsf{id}, w, i, b) = z$ for all records $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$;
- $\mathbf{ic}(+, k, x) = y$ for all **unchained** records $(k, x, y) \in \mathcal{Q}_E$.

In this vein, the ideal world execution $\overline{\mathcal{D}}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$ is associated with the real world execution $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})} \to (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$. Below we prove that in such associated executions, the simulator does not abort, and $\overline{\mathcal{D}}$ gives the same output.

Lemma 10. *Fix $\mathcal{D}$ in arbitrary. Let $(\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta}) \in \Theta$ for the good transcript set Θ with respect to $\overline{\mathcal{D}}$. Then, for any $(\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta}) \in \Theta$, any random tuples $(\mathbf{E}, \boldsymbol{\Delta})$ and $(\mathbf{ic}, \mathbf{F})$ such that $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})} \to (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$ and $(\mathbf{ic}, \mathbf{F}) \cong (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$, the following hold:*

(i) The simulator $\mathcal{S}_G$ never aborts during the execution $\overline{\mathcal{D}}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$, and

(ii) $\overline{\mathcal{D}}$ obtains the same transcript of queries and answers in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$ and $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$, and gives the same output.

Proof. Imagine running the two executions $\overline{\mathcal{D}}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$ and $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$ in parallel. The ideal world execution is driven by adversarial queries. Therefore, we prove the claim via an induction on the sequence of queries of $\overline{\mathcal{D}}$.

Assume that the sequence of queries and answers of $\overline{\mathcal{D}}$ (which is empty at the beginning) is the same up to some point in the executions $\overline{\mathcal{D}}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$ and $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$, and consider the next query. Since $\overline{\mathcal{D}}$ is deterministic, it issues the same query as the next. We distinguish four cases.

- Case 1: $\overline{\mathcal{D}}$'s next query is to $\mathsf{mL}_{\boldsymbol{\Delta}}(\mathsf{id}, P)$. Then, since the query is answered according to the same key vector $\boldsymbol{\Delta}$, the responses obtained in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$ and $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$ are identical.
- Case 2: $\overline{\mathcal{D}}$'s next query is to $O_2(\mathsf{id}, w, i, b)$. Since we assumed $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})} \to (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$, there necessarily exists a corresponding record $(\mathsf{id}, w, i, b, z)$ in $\mathcal{Q}_F$, and the response is z in the real world execution $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$. In the ideal world execution $\overline{\mathcal{D}}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$, this is replied by $z' \leftarrow \mathbf{F}(\mathsf{id}, w, i, b)$. It thus holds $z' = z$ by the definition of the event $(\mathbf{ic}, \mathbf{F}) \cong (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$.
- Case 3: $\overline{\mathcal{D}}$'s next query is to $O_1(+, k, x)$. Let y be the obtained answer in the execution $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$. Then, for $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})} \to (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$ it necessarily holds $(k, x, y) \in \mathcal{Q}_E$.

  In the ideal world execution $\overline{\mathcal{D}}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$, the query $y' \leftarrow O_1(+, k, x)$ is answered by the simulator $\mathcal{S}_G$. We further distinguish sub-cases as follows.

  - Sub-case 3.1: the query $O_1(+, k, x)$ occurs in phase 1. Then, in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$, the record (k, x, y) is added to $\mathcal{Q}_E$ in phase 1. Since $(\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta}) \in \Theta$ is generated by good real world executions, the event Chain does not occur in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$. By this, at the end of phase 1 of $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$, the record $(k, x, y) \in \mathcal{Q}_E$ is **unchained**. Since $\mathcal{Q}_F$ is not expanded in phase 2, the record $(k, x, y) \in \mathcal{Q}_E$ is kept **unchained** till the end of $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$. It thus holds $\mathbf{ic}(+, k, x) = y$ by the definition of $(\mathbf{ic}, \mathbf{F}) \cong (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$.

    Then, by the design of $\mathcal{S}_G$, the response y' in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$ is simply download from $y' \leftarrow \mathbf{ic}(+, k, x)$. It thus equals real world response y.
  - Sub-case 3.2: the query $O_1(+, k, x)$ occurs in phase 2, and there exists $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$ such that $g(i) = k$ and $\Delta_{\mathsf{id}} = f_{\mathsf{in}}^{-1}(i, x) \oplus w$ (in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$). Clearly, the real world answer y has $y = f_{\mathsf{out}}^{-1}(w \oplus \Delta_{\mathsf{id}}, i, z \oplus b \cdot \Delta_{\mathsf{id}})$.

    By the induction assumption, the previously appeared O_2 queries and answers are the same in

    $$\overline{\mathcal{D}}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})} \text{ and } \overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}.$$

    There thus exists $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$ in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$ with $g(i) = k$ and $\Delta_{\mathsf{id}} = f_{\mathsf{in}}^{-1}(i, x) \oplus w$ as well. Then, by querying $\mathsf{GlobalK}_{\boldsymbol{\Delta}}$, $\mathcal{S}_G$ will detect $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_{SF}$ at line 13 and set $y' \leftarrow f_{\mathsf{out}}^{-1}(w \oplus \Delta_{\mathsf{id}}, i, z \oplus b \cdot \Delta_{\mathsf{id}}) = y$.

    We argue that $\mathcal{S}_G$ will not abort at line 16, so that it replies $\overline{\mathcal{D}}$ with $y' = y$. At this point in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$, the set $\mathcal{Q}_{SE}$ of the simulator has *exactly the same* records as the blockcipher query transcript $\mathcal{Q}_E^\circ$ collected by $\overline{\mathcal{D}}$. By the induction assumption, $\mathcal{Q}_E^\circ$ is exactly the same as the ideal cipher query transcript collected by $\overline{\mathcal{D}}$ in the real world execution $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$. Since $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})} \to (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta})$, it holds $\mathcal{Q}_E^\circ \subseteq \mathcal{Q}_E$. Since (k, x, y) does not contradict the records in $\mathcal{Q}_E$, it does not contradict $\mathcal{Q}_E^\circ$ as well. Therefore, $\mathcal{S}_G$ will not abort and will reply $\overline{\mathcal{D}}$ with $y' = y$.
  - Sub-case 3.3: the query $O_1(+, k, x)$ occurs in phase 2, and there does not exist

    $$(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F,$$

    with $g(i) = k \wedge \Delta_{\mathsf{id}} = f_{\mathsf{in}}^{-1}(i, x) \oplus w$ (in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$). Then, in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})}$ the response is $y = \mathbf{E}(+, k, x)$, and the record (k, x, y) will be in $\mathcal{Q}_E$. This record will be **unchained** till the end of the execution.

In a similar vein to Sub-case 3.2, in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$ there does not exist $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$ with $g(i) = k \wedge \Delta_{\mathsf{id}} = f_{\mathsf{in}}^{-1}(i, x) \oplus w$ either. Then, by our design, $\mathcal{S}_G$ will detect this fact by querying $\mathsf{GlobalK}_{\mathbf{\Delta}}$, and will download the response $y' \leftarrow \mathbf{ic}(+, k, x)$ at line 22.

Since the aforementioned record $(k, x, y) \in \mathcal{Q}_E$ is **unchained**, it holds $\mathbf{ic}(+, k, x) = y$ by $(\mathbf{ic}, \mathbf{F}) \cong (\mathcal{Q}_E, \mathcal{Q}_F, \mathbf{\Delta})$. Meanwhile, in a similar vein to Sub-case 3.2, $\mathcal{S}_G$ will not abort at line 24. Therefore, $\mathcal{S}_G$ will reply $\overline{\mathcal{D}}$ with $y' = y$.

Therefore, in any case, the ideal world response y' is the same as the real world response y.

- Case 4: $\overline{\mathcal{D}}$'s next query is to $O_1(-, k, y)$. The analysis and conclusion are similar to Case 3 by symmetry.

In any case, the next answers are the same in the two executions, and $\mathcal{S}_G$ does not abort. Therefore, $\mathcal{S}_G$ never aborts in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$, and $\overline{\mathcal{D}}$ gains the same transcript of queries and answers in $\overline{\mathcal{D}}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}$ and $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\mathbf{\Delta})}$. Since $\overline{\mathcal{D}}$ is deterministic, its output is determined by this transcript, and is thus the same in the two executions. □

As a corollary of Lemma 10, it can be seen

$$\begin{aligned}&\Pr_{\mathbf{ic},\mathbf{F},\mathbf{R}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})} = 1\right] \\ &= \sum_{(\mathcal{Q}_E,\mathcal{Q}_F,\mathbf{\Delta}^\circ)\in\Theta_1} \Pr_{\mathbf{ic},\mathbf{F},\mathbf{R}}\left[(\mathbf{ic},\mathbf{F}) \cong (\mathcal{Q}_E,\mathcal{Q}_F,\mathbf{\Delta}^\circ) \wedge \mathbf{\Delta} = \mathbf{\Delta}^\circ\right]. \end{aligned} \tag{36}$$

D.5 Completing the indistinguishability proof

Lemma 11. *Fix any $(q_C, q_P, M, q_{L,max}, \mu)$-bounded S-mTCCRL distinguisher $\mathcal{D}$, and consider the corresponding $\overline{\mathcal{D}}$ that completes all chains. Then, for any $(\mathcal{Q}_E, \mathcal{Q}_F, \mathbf{\Delta}^\circ) \in \Theta_1$, it holds*

$$\frac{\Pr_{\mathbf{ic},\mathbf{F},\mathbf{R}}\left[(\mathbf{ic},\mathbf{F}) \cong (\mathcal{Q}_E,\mathcal{Q}_F,\mathbf{\Delta}) \wedge \mathbf{\Delta} = \mathbf{\Delta}^\circ\right]}{\Pr_{\mathbf{E},\mathbf{\Delta}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\mathbf{\Delta})} \to (\mathcal{Q}_E,\mathcal{Q}_F,\mathbf{\Delta}^\circ)\right]} \geq 1 - \frac{\mu(q_P + q_C)}{2^n}.$$

Proof. For each $k \in \{0,1\}^\kappa$, denote by α_k and β_k the number of **unchained** and **chained** records in $\mathcal{Q}_E[k]$ (see Eq. (35)) respectively. It is easy to see

$$\begin{aligned}&\Pr_{\mathbf{ic},\mathbf{F},\mathbf{R}}\left[(\mathbf{ic},\mathbf{F}) \cong (\mathcal{Q}_E,\mathcal{Q}_F,\mathbf{\Delta}^\circ) \wedge \mathbf{\Delta} = \mathbf{\Delta}^\circ\right] \\ &= \frac{1}{\prod_{\mathsf{id}=1}^{u} |\mathcal{R}_{\mathsf{id}}|} \times \frac{1}{(2^n)^{|\mathcal{Q}_F|}} \times \prod_{k\in\{0,1\}^\kappa} \prod_{i=0}^{\alpha_k - 1} \frac{1}{2^n - i} \\ &= \frac{1}{\prod_{\mathsf{id}=1}^{u} |\mathcal{R}_{\mathsf{id}}|} \times \prod_{k\in\{0,1\}^\kappa} \frac{1}{2^{\beta_k n} \times \prod_{i=0}^{\alpha_k - 1}(2^n - i)}, \end{aligned} \tag{37}$$

where the last equality follows by $\sum_{k\in\{0,1\}^\kappa} \beta_k = |\mathcal{Q}_F|$ (Lemma 9).

On the other hand, it is easy to see the event $\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\mathbf{\Delta})} \to (\mathcal{Q}_E, \mathcal{Q}_F, \mathbf{\Delta}^\circ)$ is equivalent with the event

$$\mathbf{E}(+, k, x) = y \text{ for all } (k, x, y) \in \mathcal{Q}_E \wedge \mathbf{\Delta} = \mathbf{\Delta}^\circ.$$

This principle is the basis of Patarin's H-coefficient method, and we refer to, e.g., [CS14] for a proof. This means

$$\begin{aligned}
&\Pr_{\mathbf{E},\boldsymbol{\Delta}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})} \to (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta}^\circ)\right] \\
&= \Pr_{\mathbf{E},\boldsymbol{\Delta}}\left[\mathbf{E}(+,k,x) = y \text{ for all } (k,x,y) \in \mathcal{Q}_E \wedge \boldsymbol{\Delta} = \boldsymbol{\Delta}^\circ\right] \\
&= \frac{1}{\prod_{\mathsf{id}=1}^{u} |\mathcal{R}_{\mathsf{id}}|} \times \prod_{k\in\{0,1\}^\kappa} \prod_{i=0}^{\alpha_k+\beta_k-1} \frac{1}{2^n - i}. \qquad (38)
\end{aligned}$$

Gathering Eqs. (37) and (38) and using $\sum_{k\in\{0,1\}^\kappa}(\alpha_k+\beta_k) \le q_P+q_C$ ($\overline{\mathcal{D}}$ makes at most q_P+q_C queries to O_1 and O_2 in total) and $\max_{k\in\{0,1\}^\kappa}\{\beta_k\} \le \mu$ (which is implied by Lemma 9) yield

$$\begin{aligned}
&\frac{\Pr_{\mathbf{ic},\mathbf{F},\mathbf{R}}\left[(\mathbf{ic},\mathbf{F}) \cong (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta}^\circ) \wedge \boldsymbol{\Delta} = \boldsymbol{\Delta}^\circ\right]}{\Pr_{\mathbf{E},\boldsymbol{\Delta}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})} \to (\mathcal{Q}_E, \mathcal{Q}_F, \boldsymbol{\Delta}^\circ)\right]} \\
&= \prod_{k\in\{0,1\}^\kappa} \frac{\prod_{i=\alpha_k}^{\alpha_k+\beta_k-1}(2^n - i)}{2^{\beta_k n}} \qquad (39) \\
&\ge \prod_{k\in\{0,1\}^\kappa} \left(1 - \frac{\alpha_k+\beta_k}{2^n}\right)^{\beta_k} \ge \prod_{k\in\{0,1\}^\kappa} \left(1 - \frac{\beta_k(\alpha_k+\beta_k)}{2^n}\right) \\
&\ge \left(1 - \frac{\mu\sum_{k\in\{0,1\}^\kappa}(\alpha_k+\beta_k)}{2^n}\right) \\
&\ge 1 - \frac{\mu(q_P+q_C)}{2^n}.
\end{aligned}$$

This completes the proof. □

Lemma 12. *For any $(q_C, q_P, M, q_{L,max}, \mu)$-bounded S-mTCCRL distinguisher $\mathcal{D}$ and any constant $T > 0$, there exists a $(q_C, q_P + T, M, q_{L,max}, \mu)$-bounded mTCCRL distinguisher $\mathcal{B}$ such that*

$$\begin{aligned}
&\left|\Pr_{\mathbf{E},\boldsymbol{\Delta}}\left[\mathcal{D}^{\Sigma_{\mathrm{re}}(\mathbf{E},\boldsymbol{\Delta})} = 1\right] - \Pr_{\mathbf{ic},\mathbf{F},\mathbf{R}}\left[\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})} = 1\right]\right| \\
&\le 2\mathbf{Adv}^{\mathrm{mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{B}) + \frac{3\mu(q_P+q_C)}{2^n} + \frac{2}{T}.
\end{aligned}$$

Proof. Gathering Eqs. (32), (33), (34) and (36) and Lemma 11 yields

$$
\begin{aligned}
&\left|\Pr_{\mathbf{E},\mathbf{\Delta}}\left[\mathcal{D}^{\Sigma_{\mathrm{re}}(\mathbf{E},\mathbf{\Delta})}=1\right]-\Pr_{\mathbf{ic},\mathbf{F},\mathbf{R}}\left[\mathcal{D}^{\Sigma_{\mathrm{id}}(\mathbf{ic},\mathbf{F},\mathbf{R})}=1\right]\right| \\
&\le \Pr\left[(\mathbf{E},\mathbf{\Delta}) \text{ is bad}\right] \\
&\quad+\sum_{(\mathcal{Q}_E,\mathcal{Q}_F,\mathbf{\Delta}^\circ)\in\Theta_1}\Pr\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\mathbf{\Delta})}\to(\mathcal{Q}_E,\mathcal{Q}_F,\mathbf{\Delta}^\circ)\right] \qquad \text{(by Eq. (34))} \\
&\quad-\sum_{(\mathcal{Q}_E,\mathcal{Q}_F,\mathbf{\Delta}^\circ)\in\Theta_1}\Pr_{\mathbf{ic},\mathbf{F},\mathbf{R}}\left[(\mathbf{ic},\mathbf{F})\cong(\mathcal{Q}_E,\mathcal{Q}_F,\mathbf{\Delta}^\circ)\wedge\mathbf{\Delta}=\mathbf{\Delta}^\circ\right] \text{(by Eq. (36))} \\
&\le \Pr\left[(\mathbf{E},\mathbf{\Delta}) \text{ is bad}\right]+\sum_{(\mathcal{Q}_E,\mathcal{Q}_F,\mathbf{\Delta}^\circ)\in\Theta_1}\Pr\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\mathbf{\Delta})}\to(\mathcal{Q}_E,\mathcal{Q}_F,\mathbf{\Delta}^\circ)\right] \\
&\qquad\qquad\times\left(1-\frac{\Pr_{\mathbf{ic},\mathbf{F},\mathbf{R}}\left[(\mathbf{ic},\mathbf{F})\cong(\mathcal{Q}_E,\mathcal{Q}_F,\mathbf{\Delta}^\circ)\wedge\mathbf{\Delta}=\mathbf{\Delta}^\circ\right]}{\Pr_{\mathbf{E},\mathbf{\Delta}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\mathbf{\Delta})}\to(\mathcal{Q}_E,\mathcal{Q}_F,\mathbf{\Delta}^\circ)\right]}\right) \\
&\le \underbrace{2\mathbf{Adv}^{\mathrm{mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{B})+\frac{2\mu(q_P+q_C)}{2^n}+\frac{2}{T}}_{\text{(by Eq. (33))}} \\
&\quad+\underbrace{\frac{\mu(q_P+q_C)}{2^n}\times\sum_{(\mathcal{Q}_E,\mathcal{Q}_F,\mathbf{\Delta}^\circ)\in\Theta_1}\Pr_{\mathbf{E},\mathbf{\Delta}}\left[\overline{\mathcal{D}}^{\Sigma_{\mathrm{re}}(\mathbf{E},\mathbf{\Delta})}\to(\mathcal{Q}_E,\mathcal{Q}_F,\mathbf{\Delta}^\circ)\right]}_{\text{(by Lemma 11)}} \\
&\le 2\mathbf{Adv}^{\mathrm{mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{B})+\frac{3\mu(q_P+q_C)}{2^n}+\frac{2}{T},
\end{aligned}
$$

as claimed. □

This establishes the indistinguishability of Σ_{id} and Σ_{re} and completes the proof of Theorem 1.

D.6 Proof of Eq. (32): bad events in simulation

Recall our goal: for any $(q_C, q_P, M, q_{L,max}, \mu)$-bounded S-mTCCRL distinguisher $\mathcal{D}$ and any constant $T > 0$, we can construct a $(q_C, q_P+T, M, q_{L,max}, \mu)$-bounded mTCCRL distinguisher $\mathcal{B}$, such that

$$
\Pr\left[\mathsf{Chain} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{re}}}\right]\le 2\mathbf{Adv}^{\mathrm{mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{B})+\frac{2\mu(q_P+q_C)}{2^n}+\frac{2}{T}.
$$

D.6.1 First step: high-complexity mTCCRL distinguisher $\mathcal{A}$.

We begin with constructing a mTCCRL distinguisher $\mathcal{A}$ with extremely high complexity. By Definition 2, $\mathcal{A}^{\mathbf{E},O,\mathsf{mL}_{\mathbf{\Delta}}}$ has access to three oracles $\mathbf{E}, O$ and $\mathsf{mL}_{\mathbf{\Delta}}$, where $\mathbf{\Delta}=(R_1,...,R_u)$, $O=\mathsf{mTCK}^{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}}}_{\mathbf{\Delta}}$ in the real world and $O=\mathbf{F}$ in the ideal world. It proceeds as follows.

1. Runs $\mathcal{D}$ till $\mathcal{D}$ outputs *end* (indicating the end of phase 1), replies $\mathcal{D}$'s queries using its own oracles $\mathbf{E}, O$ and $\mathsf{mL}_{\mathbf{\Delta}}$ and collects the query transcript of $\mathcal{D}$ as lists $\mathcal{Q}_E$ and $\mathcal{Q}_F$ ($\mathcal{D}$'s queries to $\mathsf{mL}_{\mathbf{\Delta}}$ are useless and not collected).

2. Initializes a set

$$
\begin{aligned}
Set\leftarrow\Big\{&((\mathsf{id},w,i,b,z),(\mathsf{id}',w',i',b',z'),\Delta^*_{\mathsf{id}},\Delta^*_{\mathsf{id}'})\in\mathcal{Q}_F^2\times\mathcal{R}_{\mathsf{id}}\times\mathcal{R}_{\mathsf{id}'}: \\
&g(i)=g(i'),\ f_{\mathsf{in}}(\Delta^*_{\mathsf{id}}\oplus w,i)=f_{\mathsf{in}}(\Delta^*_{\mathsf{id}'}\oplus w',i'), \\
&\text{and } f^{-1}_{\mathsf{out}}(\Delta^*_{\mathsf{id}}\oplus w,i,z\oplus b\cdot\Delta^*_{\mathsf{id}})=f^{-1}_{\mathsf{out}}(\Delta^*_{\mathsf{id}'}\oplus w',i',z'\oplus b'\cdot\Delta^*_{\mathsf{id}'})\Big\}.
\end{aligned}
$$

The 4-tuples in Set might satisfy the definition of $\mathsf{Chain2}$ (see Sect. D.3), which has to be checked in turn.

3. Checks if $\mathsf{Chain1}$ is (possibly) fulfilled w.r.t. the records in $\mathcal{Q}_E$ and $\mathcal{Q}_F$. Concretely, if there exist $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$ and $(k, x, y) \in \mathcal{Q}_E$ such that $g(i) = k$ and $f_{\mathsf{in}}^{-1}(i, x) \oplus w = kd_y(w, i, b, y, z)$, then outputs 1.

4. For each 4-tuple $((\mathsf{id}, w, i, b, z), (\mathsf{id}', w', i', b', z'), \Delta^*_{\mathsf{id}}, \Delta^*_{\mathsf{id}'}) \in Set$, do:

   - If $(g(i), f_{\mathsf{in}}(\Delta^*_{\mathsf{id}} \oplus w, i), \star) \in \mathcal{Q}_E$ or $(g(i), \star, f_{\mathsf{out}}^{-1}(\Delta^*_{\mathsf{id}} \oplus w, i, z \oplus b \cdot \Delta^*_{\mathsf{id}})) \in \mathcal{Q}_E$, discard this 4-tuple.
   - Else, queries $\mathbf{E}(+, g(i), f_{\mathsf{in}}(\Delta^*_{\mathsf{id}} \oplus w, i)) \to y^*$. If $y^* = f_{\mathsf{out}}^{-1}(\Delta^*_{\mathsf{id}} \oplus w, i, z \oplus b \cdot \Delta^*_{\mathsf{id}})$, then $\mathcal{A}$ decides that $\mathsf{Chain2}$ occurs and outputs 1.

5. If $\mathcal{A}$ did not output in the previous steps, then outputs $b^* \xleftarrow{\$} \{0, 1\}$.

Clearly, $\mathcal{A}$ is $(q_C, q_P + |Set|, M, q_{L,max}, \mu)$-bounded, but $|Set|$ depends on the underlying randomness. Meanwhile, when $O = \mathsf{mTCK}_{\mathbf{\Delta}}^{\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}}$, the oracles provided to $\mathcal{D}$ are indeed those presented in phase 1 of the real world experiment $\mathbf{Exp}^{\text{S-mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}, \vec{\mathcal{R}}, \mathcal{P}, u, \mathcal{D}, \mathcal{S}_G, 0}$. When $O = \mathbf{F}$, the oracles provided to $\mathcal{D}$ are $(\mathbf{E}, \mathbf{F}, \mathsf{mL}_{\mathbf{\Delta}})$. By the design of our simulator $\mathcal{S}_G$ (see Sect. D.1), the responses of $\mathbf{E}$ have the same distribution as $\mathcal{S}_G$ in phase 1 of the ideal world experiment $\mathbf{Exp}^{\text{S-mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}, \vec{\mathcal{R}}, \mathcal{P}, u, \mathcal{D}, \mathcal{S}_G, 1}$. Therefore, $\mathcal{A}$ correctly simulates the real and ideal world executions in front of $\mathcal{D}$.

Let $\mathsf{Eve1}$, resp. $\mathsf{Eve2}$, be the event that $\mathcal{A}$ outputs 1 in step 3, resp. step 4, and let $\mathsf{Eve} := \mathsf{Eve1} \vee \mathsf{Eve2}$. By construction, when Eve occurs, $\mathcal{A}$ outputs 1; when Eve does not occur, the probability that $\mathcal{A}$ outputs 1 is 1/2. Therefore,

$$
\begin{aligned}
&\Pr[\mathcal{A} = 1 \mid O = \mathsf{mTCK}_{\mathbf{\Delta}}^{\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}}] \\
&= \frac{1}{2} \times \Pr[\mathsf{Eve} \text{ does not occur in } \mathcal{D}^{\Sigma_{\mathrm{re}}}] + \Pr[\mathsf{Eve} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{re}}}] \\
&= \frac{1}{2} \times \Big(1 - \Pr[\mathsf{Eve} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{re}}}]\Big) + \Pr[\mathsf{Eve} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{re}}}] \\
&= \frac{1}{2} + \frac{1}{2} \times \Pr[\mathsf{Eve} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{re}}}], \qquad (40) \\
&\Pr[\mathcal{A} = 1 \mid O = \mathbf{F}] \\
&= \frac{1}{2} \times \Pr[\mathsf{Eve} \text{ does not occur in } \mathcal{D}^{\Sigma_{\mathrm{id}}}] + \Pr[\mathsf{Eve} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{id}}}] \\
&= \frac{1}{2} + \frac{1}{2} \times \Pr[\mathsf{Eve} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{id}}}]. \qquad (41)
\end{aligned}
$$

D.6.2 Probabilities of Eve.

Note that $\mathsf{Chain1} \Rightarrow \mathsf{Eve1}$ and $\mathsf{Chain2} \Rightarrow \mathsf{Eve2}$, but the converse may not hold. Therefore,

$$\Pr[\mathsf{Eve} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{re}}}] \geq \Pr[\mathsf{Chain} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{re}}}]. \qquad (42)$$

Below we devote to prove

$$\Pr[\mathsf{Eve} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{id}}}] \leq \frac{2\mu(q_P + q_C)}{2^n}. \qquad (43)$$

Probability of Eve1. Consider any pair of records $((\mathsf{id}, w, i, b, z), (k, x, y)) \in \mathcal{Q}_F \times \mathcal{Q}_E$ with $g(i) = k$. The condition is equivalent with $f_{\mathsf{in}}^{-1}(i, x) \oplus w = kd_y(w, i, b, y, z)$. We distinguish two cases.

- Case 1: the query $z \leftarrow \mathbf{F}(\mathsf{id}, w, i, b)$ occurs later than the query $\mathbf{E}(+, k, x)$ or $\mathbf{E}(-, k, y)$. Then, since $h_{w,i,b,y}$ is bijective (Sect. 5.1), the number of z such that $kd_y(w, i, b, y, z) = f_{\mathsf{in}}^{-1}(i, x) \oplus w$ is at most 1. Since z is uniformly distributed in $\{0, 1\}^n$, the probability to have $kd_y(w, i, b, y, z) = f_{\mathsf{in}}^{-1}(i, x) \oplus w$ is at most $1/2^n$.
- Case 2: the record (k, x, y) is produced by a forward ideal cipher query, and the query $y \leftarrow \mathbf{E}(+, k, x)$ occurs later than the query $z \leftarrow \mathbf{F}(\mathsf{id}, w, i, b)$. Then, since $h'_{w,i,b,z}$ is bijective (Sect. 5.1), the number of y with $kd_y(w, i, b, y, z) = f_{\mathsf{in}}^{-1}(i, x) \oplus w$ is at most 1. Since y is uniformly distributed in a set of size at least $2^n - q_P$, the probability to have $kd_y(w, i, b, y, z) = f_{\mathsf{in}}^{-1}(i, x) \oplus w$ is at most $1/(2^n - q_P) \le 2/2^n$ (assuming $q_P \le 2^n/2$).
- Case 3: (k, x, y) is produced by a backward query $x \leftarrow \mathbf{E}(-, k, y)$, which occurs later than the query $z \leftarrow \mathbf{F}(\mathsf{id}, w, i, b)$. Then, since f_{in} is bijective, the number of x such that $f_{\mathsf{in}}^{-1}(i, x) \oplus w = kd_y(w, i, b, y, z)$ is exactly 1, and the probability to have $kd_y(w, i, b, y, z) = f_{\mathsf{in}}^{-1}(i, x) \oplus w$ is at most $1/(2^n - q_P) \le 2/2^n$ (which resembles Case 2).

For each $(k, x, y) \in \mathcal{Q}_E$, the number of $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$ with $g(i) = k$ is at most μ. Therefore, the number of such pairs is at most μq_P. Taking a union bound yields $\Pr[\mathsf{Eve1} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{id}}}] \le 2\mu q_P/2^n$.

Probability of Eve2. Consider any 4-tuple $((\mathsf{id}, w, i, b, z), (\mathsf{id}', w', i', b', z'), \Delta^*_{\mathsf{id}}, \Delta^*_{\mathsf{id}'})$ with $g(i) = g(i')$ and $f_{\mathsf{in}}(\Delta^*_{\mathsf{id}} \oplus w, i) = f_{\mathsf{in}}(\Delta^*_{\mathsf{id}'} \oplus w', i')$. Since $z \leftarrow \mathbf{F}(\mathsf{id}, w, i, b)$ and $z' \leftarrow \mathbf{F}(\mathsf{id}', w', i', b')$ are independent and uniform, the probability to have $f_{\mathsf{out}}^{-1}(\Delta^*_{\mathsf{id}} \oplus w, i, z \oplus b \cdot \Delta^*_{\mathsf{id}}) = f_{\mathsf{out}}^{-1}(\Delta^*_{\mathsf{id}'} \oplus w', i', z' \oplus b' \cdot \Delta^*_{\mathsf{id}'})$ is $1/2^n$ by the bijectivity of $f_{\mathsf{out}}^{-1}(\Delta^*_{\mathsf{id}} \oplus w, i, \cdot)$ and $f_{\mathsf{out}}^{-1}(\Delta^*_{\mathsf{id}'} \oplus w', i', \cdot)$.

Let $k = g(i)$, $x = f_{\mathsf{in}}(\Delta^*_{\mathsf{id}} \oplus w, i)$ and $y = f_{\mathsf{out}}^{-1}(\Delta^*_{\mathsf{id}} \oplus w, i, z \oplus b \cdot \Delta^*_{\mathsf{id}})$. By our design, $\mathsf{Eve2}$ then holds w.r.t. this 4-tuple if and only if $(g(i), f_{\mathsf{in}}(\Delta^*_{\mathsf{id}} \oplus w, i), \star) \notin \mathcal{Q}_E$ and $(g(i), \star, f_{\mathsf{out}}^{-1}(\Delta^*_{\mathsf{id}} \oplus w, i, z \oplus b \cdot \Delta^*_{\mathsf{id}})) \notin \mathcal{Q}_E$ and $\mathbf{E}(+, g(i), f_{\mathsf{in}}(\Delta^*_{\mathsf{id}} \oplus w, i)) = f_{\mathsf{out}}^{-1}(\Delta^*_{\mathsf{id}} \oplus w, i, z \oplus b \cdot \Delta^*_{\mathsf{id}})$. Conditioned on that $\mathbf{E}$ gives rise to the queries and answers in $\mathcal{Q}_E$, $\mathbf{E}(+, g(i), \cdot)$ is fixed at at most $q_P \le 2^n/2$ entries.[4] By these, $\mathbf{E}(+, g(i), f_{\mathsf{in}}(\Delta^*_{\mathsf{id}} \oplus w, i))$ is uniformly distributed in a set of size at least $2^n - q_P \ge 2^n/2$. Therefore, the probability to have $\mathbf{E}(+, g(i), f_{\mathsf{in}}(\Delta^*_{\mathsf{id}} \oplus w, i)) = f_{\mathsf{out}}^{-1}(\Delta^*_{\mathsf{id}} \oplus w, i, z \oplus b \cdot \Delta^*_{\mathsf{id}})$ is at most $2/2^n$, and the probability that $\mathsf{Eve2}$ holds w.r.t. this 4-tuple is at most $2/2^{2n}$.

We next consider the number of such 4-tuples. First, the number of pairs of distinct records $((\mathsf{id}, w, i, b, z), (\mathsf{id}', w', i', b', z'))$ with $g(i) = g(i')$ is at most μq_C. For any such pairs, once Δ^*_{id} is fixed (the number of choices is at most $|\mathcal{R}_{\mathsf{id}}|$), $x = f_{\mathsf{in}}(\Delta^*_{\mathsf{id}} \oplus w, i)$ and further $\Delta^*_{\mathsf{id}'} = f_{\mathsf{in}}^{-1}(i', x') \oplus w'$ are fixed. Therefore, the number of such 4-tuples is at most $\mu q_C |\mathcal{R}_{\mathsf{id}}| \le 2^n \mu q_C$. Taking a union bound yields $\Pr[\mathsf{Eve2} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{id}}}] \le 2^{n+1} \mu q_C/2^{2n} \le 2\mu q_C/2^n$.

Gathering the two bounds yields Eq. (43).

We remark that if $\mathcal{A}$ proceeds into step 4, then $\mathsf{Eve1}$ did not happen. This means if $(g(i), f_{\mathsf{in}}(\Delta^*_{\mathsf{id}} \oplus w, i), \star) \in \mathcal{Q}_E$ or $(g(i), \star, f_{\mathsf{out}}^{-1}(\Delta^*_{\mathsf{id}} \oplus w, i, z \oplus b \cdot \Delta^*_{\mathsf{id}})) \in \mathcal{Q}_E$ then Δ^*_{id} cannot equal the "right" key Δ_{id}, and $\mathsf{Chain2}$ cannot occur w.r.t. the 4-tuple $((\mathsf{id}, w, i, b, z), (\mathsf{id}', w', i', b', z'), \Delta^*_{\mathsf{id}}, \Delta^*_{\mathsf{id}'})$. This is the reason for $\mathcal{A}$ discarding such 4-tuples.

[4] Here we used the idea of [LSS17] and the fact that the queries made in step 4 are fixed before that step.

D.6.3 To mTCCRL attacker with bounded complexity.

Gathering Eqs. (40), (41), (42) and (43) seems to yield bounds on $\mathcal{A}$'s advantage on distinguishing the real and ideal worlds in the mTCCRL setting. One may consider relating this to $\mathbf{Adv}^{\mathrm{mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\vec{\mathcal{R}},\mathcal{P},u}(\mathcal{A})$. But $\mathcal{A}$ makes $q_P+|Set|$ queries to $\mathbf{E}$ and $|Set|$ may be too large. Therefore, we need deeper discussions. We distinguish two cases.

Case 1: $\Pr[\mathcal{A}=1 \mid O=\mathsf{mTCK}^{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}}}_{\mathbf{\Delta}}] \le \Pr[\mathcal{A}=1 \mid O=\mathbf{F}]$. By Eqs. (40) and (41), this condition implies $\Pr[\mathsf{Eve}\text{ occurs in }\mathcal{D}^{\Sigma_{\mathrm{re}}}] \le \Pr[\mathsf{Eve}\text{ occurs in }\mathcal{D}^{\Sigma_{\mathrm{id}}}]$, and further

$$\begin{aligned}&\Pr[\mathsf{Chain}\text{ occurs in }\mathcal{D}^{\Sigma_{\mathrm{re}}}] \le \Pr[\mathsf{Eve}\text{ occurs in }\mathcal{D}^{\Sigma_{\mathrm{re}}}] \quad \text{by Eq. (42)}\\ &\le \Pr[\mathsf{Eve}\text{ occurs in }\mathcal{D}^{\Sigma_{\mathrm{id}}}] \le \frac{2\mu(q_P+q_C)}{2^n} \quad \text{by Eq. (43)}\ .\end{aligned} \tag{44}$$

Case 2: $\Pr[\mathcal{A}=1 \mid O=\mathsf{mTCK}^{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}}}_{\mathbf{\Delta}}] > \Pr[\mathcal{A}=1 \mid O=\mathbf{F}]$. On one hand, by the condition and Eqs. (40), (41), (42) and (43), we have

$$\begin{aligned}&\left|\Pr[\mathcal{A}=1 \mid O=\mathsf{mTCK}^{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}}}_{\mathbf{\Delta}}] - \Pr[\mathcal{A}=1 \mid O=\mathbf{F}]\right|\\ &= \Pr[\mathcal{A}=1 \mid O=\mathsf{mTCK}^{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}}}_{\mathbf{\Delta}}] - \Pr[\mathcal{A}=1 \mid O=\mathbf{F}]\\ &= \left(\frac{1}{2}+\frac{1}{2}\times\Pr[\mathsf{Eve}\text{ occurs in }\mathcal{D}^{\Sigma_{\mathrm{re}}}]\right) - \left(\frac{1}{2}+\frac{1}{2}\times\Pr[\mathsf{Eve}\text{ occurs in }\mathcal{D}^{\Sigma_{\mathrm{id}}}]\right)\\ &\ge \frac{1}{2}\times\Pr[\mathsf{Chain}\text{ occurs in }\mathcal{D}^{\Sigma_{\mathrm{re}}}] - \frac{\mu(q_P+q_C)}{2^n}.\end{aligned} \tag{45}$$

On the other hand, the condition means $\Pr[\mathcal{A}=0 \mid O=\mathsf{mTCK}^{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}}}_{\mathbf{\Delta}}] < \Pr[\mathcal{A}=0 \mid O=\mathbf{F}]$, and

$$\begin{aligned}&\left|\Pr[\mathcal{A}=1 \mid O=\mathsf{mTCK}^{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}}}_{\mathbf{\Delta}}] - \Pr[\mathcal{A}=1 \mid O=\mathbf{F}]\right|\\ &= \left|\Pr[\mathcal{A}=0 \mid O=\mathsf{mTCK}^{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}}}_{\mathbf{\Delta}}] - \Pr[\mathcal{A}=0 \mid O=\mathbf{F}]\right|\\ &= \Pr[\mathcal{A}=0\wedge|Set|\le\mu Tq_C \mid O=\mathbf{F}] + \Pr[\mathcal{A}=0\wedge|Set|>\mu Tq_C \mid O=\mathbf{F}]\\ &\quad - \Pr[\mathcal{A}=0\wedge|Set|\le\mu Tq_C \mid O=\mathsf{mTCK}^{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}}}_{\mathbf{\Delta}}]\\ &\qquad\quad - \Pr[\mathcal{A}=0\wedge|Set|>\mu Tq_C \mid O=\mathsf{mTCK}^{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}}}_{\mathbf{\Delta}}]\\ &\le \Big|\Pr[\mathcal{A}=0\wedge|Set|\le\mu Tq_C \mid O=\mathsf{mTCK}^{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}}}_{\mathbf{\Delta}}]\\ &\qquad\quad - \Pr[\mathcal{A}=0\wedge|Set|\le\mu Tq_C \mid O=\mathbf{F}]\Big|\\ &\quad + \Pr[\mathcal{A}=0\wedge|Set|>\mu Tq_C \mid O=\mathbf{F}]\end{aligned} \tag{46}$$

We then define another mTCCRL distinguisher $\mathcal{B}$: the actions are the same as $\mathcal{A}$, except that: (i) $\mathcal{B}$ immediately outputs 0 if it finds $|Set|>\mu Tq_C$ at step 2; (ii) in the other cases, $\mathcal{B}$ outputs 1 if and only if $\mathcal{A}$ outputs 0. By this, $\mathcal{B}$ is $(q_C, q_P+\mu Tq_C, M, q_{L,max}, \mu)$-bounded, and $\Pr[\mathcal{B}=1] =$

$\Pr[|Set| \le \mu T q_C \wedge \mathcal{A} = 0]$. Eq. (46) thus implies

$$\begin{aligned}
&\left|\Pr[\mathcal{A}=1 \mid O = \mathsf{mTCK}_{\boldsymbol{\Delta}}^{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}}}] - \Pr[\mathcal{A}=1 \mid O=\mathbf{F}]\right| \\
\le &\left|\Pr[\mathcal{B}=1 \mid O = \mathsf{mTCK}_{\boldsymbol{\Delta}}^{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}}}] - \Pr[\mathcal{B}=1 \mid O=\mathbf{F}]\right| \\
&+ \Pr[\mathcal{A}=0 \wedge |Set| > \mu T q_C \mid O=\mathbf{F}] \\
\le &\ \mathbf{Adv}^{\mathrm{mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\vec{\mathcal{R}},\mathcal{P},u}(\mathcal{B}) + \Pr[|Set| > \mu T q_C \mid O=\mathbf{F}]. \qquad (47)
\end{aligned}$$

Gathering Eqs. (45) and (47), we have a $(q_C, q_P + \mu T q_C, M, q_{L,max}, \mu)$-bounded mTCCRL distinguisher $\mathcal{B}$ such that

$$\begin{aligned}
&\frac{1}{2} \times \Pr[\mathsf{Chain} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{re}}}] - \frac{\mu(q_P+q_C)}{2^n} \\
\le &\left|\Pr[\mathcal{A}=1 \mid O = \mathsf{mTCK}_{\boldsymbol{\Delta}}^{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}}}] - \Pr[\mathcal{A}=1 \mid O=\mathbf{F}]\right| \\
\le &\ \mathbf{Adv}^{\mathrm{mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\vec{\mathcal{R}},\mathcal{P},u}(\mathcal{B}) + \Pr[|Set| > \mu T q_C \mid O=\mathbf{F}]. \qquad (48)
\end{aligned}$$

Bounding $\Pr[|Set| > \mu T q_C \mid O = \mathbf{F}]$. For each record $(\mathsf{id}, w, i, b, z) \in \mathcal{Q}_F$, the number of records $(\mathsf{id}', w', i', b', z') \in \mathcal{Q}_F$ with $(\mathsf{id}', w', i', b') \neq (\mathsf{id}, w, i, b)$ and $g(i) = g(i')$ is at most μ. For each such pair

$$((\mathsf{id}, w, i, b, z), (\mathsf{id}', w', i', b')),$$

the number of candidate key pair $(\Delta^*_{\mathsf{id}}, \Delta^*_{\mathsf{id}'})$ such that $f_{\mathsf{in}}(\Delta^*_{\mathsf{id}} \oplus w, i) = f_{\mathsf{in}}(\Delta^*_{\mathsf{id}'} \oplus w', i')$ is at most $|\mathcal{R}_{\mathsf{id}}| \le 2^n$ (since each $\Delta^*_{\mathsf{id}} \in \mathcal{R}_{\mathsf{id}}$ determines a unique $\Delta^*_{\mathsf{id}'} = f_{\mathsf{in}}^{-1}(i', f_{\mathsf{in}}(\Delta^*_{\mathsf{id}} \oplus w, i)) \oplus w'$). But since $z \leftarrow \mathbf{F}(\mathsf{id}, w, i, b)$ and $z' \leftarrow \mathbf{F}(\mathsf{id}', w', i', b')$ are independent and uniform in $\{0,1\}^n$, the probability to have $f_{\mathsf{out}}^{-1}(\Delta^*_{\mathsf{id}} \oplus w, i, z \oplus b \cdot \Delta^*_{\mathsf{id}}) = f_{\mathsf{out}}^{-1}(\Delta^*_{\mathsf{id}'} \oplus w', i', z' \oplus b' \cdot \Delta^*_{\mathsf{id}'})$ is $1/2^n$. Therefore,

$$\mathbb{E}[|Set| \mid O = \mathbf{F}] \le \frac{\mu q_P |\mathcal{R}_{\mathsf{id}}|}{2^n} \le \mu q_P.$$

Applying the Markov inequality yields

$$\Pr[|Set| > \mu T q_C \mid O = \mathbf{F}] \le \frac{1}{T}. \qquad (49)$$

We remark that the use of better tools such as Chebyshev or Chernoff bound seems very difficult without fine-grained assumptions on the distribution.

D.6.4 Summary.

By the above, when $\Pr[\mathcal{A}=1 \mid O = \mathsf{mTCK}_{\boldsymbol{\Delta}}^{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}}}] > \Pr[\mathcal{A}=1 \mid O=\mathbf{F}]$, gathering Eqs. (48) and (49) yields

$$\begin{aligned}
&\frac{1}{2} \times \Pr[\mathsf{Chain} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{re}}}] - \frac{\mu(q_P+q_C)}{2^n} \le \mathbf{Adv}^{\mathrm{mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\vec{\mathcal{R}},\mathcal{P},u}(\mathcal{B}) + \frac{1}{T}, \\
&\Pr[\mathsf{Chain} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{re}}}] \le 2\mathbf{Adv}^{\mathrm{mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\vec{\mathcal{R}},\mathcal{P},u}(\mathcal{B}) + \frac{2\mu(q_P+q_C)}{2^n} + \frac{2}{T}.
\end{aligned}$$

When $\Pr[\mathcal{A}=1 \mid O = \mathsf{mTCK}_{\boldsymbol{\Delta}}^{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}}}] \le \Pr[\mathcal{A}=1 \mid O=\mathbf{F}]$, Eq. (44) gives a better bound on $\Pr[\mathsf{Chain} \text{ occurs in } \mathcal{D}^{\Sigma_{\mathrm{re}}}]$. These complete the proof.

D.6.5 Discussion.

This subsection analyzed several events in the ideal world executions. This seems to contradicts our discussion in Sect. 2.3 and 5.2. However, all the events analyzed in this paper, including the real world and the ideal world events, are *not* affected by the key leaking oracle $\mathsf{mL}_{\mathbf{\Delta}}$. Consequently, the analyses are considerably simpler than those in [GWYY25]. (Our whole proof remains lengthy, but the added complexity is due to simulation.)

E Constructions for Weaker Notions in the ICM

E.1 S-mCCRL of $\widehat{\mathsf{MMO}}^{\pi}$

Using a permutation or fixed-key blockcipher $\pi : \{+,-\} \times \{0,1\}^n \to \{0,1\}^n$ and a linear orthomorphism σ, Guo et al. [GKWY20] proposed a more efficient construction

$$\widehat{\mathsf{MMO}}^{\pi}(x) := \pi(+, \sigma(x)) \oplus \sigma(x)$$

for applications that do not need the tweak. This can only achieve the "non-tweakable" variants of S-mTCCRL, i.e., S-mCCRL and S-mCRL of Definitions 8 and 9.

Note that the construction $\widehat{\mathsf{MMO}}^{\pi}$ does not fit into the model $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$ (since $g(i)$ is absent). But interestingly, its S-mCCRL bound (which implies the same S-mCRL bound) can still be derived from Theorem 1.

Corollary 2. *If σ is a linear orthomorphism and π is modeled as a random permutation, then there exists a simulator $\mathcal{S}_2$ such that the S-mCCRL advantage of $\widehat{\mathsf{MMO}}^{\pi}$ for any $(q_C, q_P, M, q_{L,max})$-bounded distinguisher $\mathcal{D}$ has upper bound*

$$\begin{aligned}&\mathbf{Adv}^{\text{S-mCCRL}}_{\widehat{\mathsf{MMO}}^{\pi},\vec{\mathcal{R}},\mathcal{P}_{\mathsf{free}},u}(\mathcal{D}, \mathcal{S}_2)\\ &\le \frac{7q_Cq_P(M+1)(q_{L,max}+1)}{C} + 6(q_C+1)\sqrt{\frac{q_C(M+1)(q_{L,max}+1)}{C}}.\end{aligned} \tag{50}$$

In addition, $\mathcal{S}_2^{\mathsf{GlobalK}_{\mathbf{\Delta}}}$ makes at most q_Cq_P queries to its oracle $\mathsf{GlobalK}_{\mathbf{\Delta}}$ and runs in time $O(q_Cq_P)$.

Proof. Consider Eq. (50) first. By Corollary 1, there exists a sound simulator $\mathcal{S}_1$ for the ideal cipher-based construction $\widehat{\mathsf{MMO}}^{\mathbf{E}}$. We construct $\mathcal{S}_2$ from $\mathcal{S}_1$ and apply Eq. (6) to bound the advantage.

In detail, fix a public constant $i^\circ \in \mathcal{I}$ in arbitrary, and let $k^\circ = g(i^\circ)$. $\mathcal{S}_2$ internally runs an instance of $\mathcal{S}_1$ to reply queries. Upon $\mathcal{D}$ querying $\mathcal{S}_2.O_1(\delta, v)$, $\mathcal{S}_2$ calls $v' \leftarrow \mathcal{S}_1.O_1(\delta, k^\circ, v)$ and replies with v'.

Now, given any distinguisher $\mathcal{D}$ against the S-mCCRL security of $\widehat{\mathsf{MMO}}^{\pi}$, we construct a distinguisher $\mathcal{B}$ against the S-mTCCRL security of $\widehat{\mathsf{MMO}}^{\mathbf{E}}$. In detail, $\mathcal{B}$ runs $\mathcal{D}$:

- When $\mathcal{D}$ queries $O_1(\delta, v)$, $\mathcal{B}$ makes a query $O_1(\delta, k^\circ, v)$ to its own oracles, and passes the response to $\mathcal{D}$;
- When $\mathcal{D}$ queries $O_2(\mathsf{id}, w, b)$, $\mathcal{B}$ makes a query $O_2(\mathsf{id}, w, i^\circ, b)$, to its own oracle O_2, and passes the response to $\mathcal{D}$;
- When $\mathcal{D}$ queries $\mathsf{mL}(\mathsf{id}, P)$, $\mathcal{B}$ simply relays the query to its own $\mathsf{mL}_{\mathbf{\Delta}}$ oracle and passes the response (including the abortion information) to $\mathcal{D}$.

By the definitions of the two constructions $\widehat{\mathsf{MMO}}^{\mathbf{E}}$ and $\widehat{\mathsf{MMO}}^{\pi}$, we have:

- When $\mathcal{B}$ is interacting with the real world oracles, $\mathcal{D}$ has access to $\mathbf{E}_{k^\circ}$ (which is a random permutation) and the multi-user keyed oracle built upon $\widehat{\mathsf{MMO}}^{\mathbf{E}_{k^\circ}}$.
- When $\mathcal{B}$ is interacting with the real world oracles, $\mathcal{D}$ is actually interacting with the aforementioned simulator $\mathcal{S}_2$.

Meanwhile, since $\mathcal{D}$ is $(q_C, q_P, M, q_{L,max})$-bounded, $\mathcal{B}$ is $(q_C, q_P, M, q_{L,max}, \mu)$-bounded (in the S-mTCCRL setting) for some μ. Since $\mathcal{B}$ insists on the same tweak i°, it has $\mu = q_C$. Therefore,

$$\mathbf{Adv}^{\text{S-mCCRL}}_{\widehat{\mathsf{MMO}}^{\pi},\mathcal{R},\mathcal{P},u}(\mathcal{D},\mathcal{S}_2) \leq \mathbf{Adv}^{\text{S-mTCCRL}}_{\widehat{\mathsf{MMO}}^{\mathbf{E}},\mathcal{R},\mathcal{P},q_C,u}(\mathcal{B},\mathcal{S}_1).$$

Injecting $\mu = q_C$ into Eq. (6) yields Eq. (50). □

F Security of Protocol Π_{LaAND}

F.1 Subroutine Functionalities

The subroutine functionalities $\mathcal{F}_{\mathsf{aShare}}$, $\mathcal{F}_{\mathsf{Zero}}$, $\mathcal{F}_{\mathsf{Com}}$, and $\mathcal{F}_{\mathsf{Rand}}$ used by protocol Π_{LaAND} are presented in Figs. 10, 11, 12, and 13.

F.2 Delayed Phase Switch

In the proof of some MPC protocols the entries in $\mathcal{Q}_F$ may be correlated with honest party's secret outputs such that the MPC simulator cannot directly construct $\mathcal{Q}_F$ in the ideal world. Nevertheless, as argued in the Introduction, we can mitigate this issue by augmenting the global key query command of the ideal functionality such that the outputs of all honest parties get leaked to the adversary once the global key is successfully guessed. Since the global key has sufficient entropy, it is statistically unlikely to be correctly guessed in the ideal world. Therefore, this change does not affect the actual security guarantee captured by the ideal functionality.

This change requires us to switch the simulator from a hybrid where the phase switch depends on the secret output of honest parties in the real world, to another hybrid where such a switch uses the outputs returned by the global key guessing queries in the ideal world. The phase switch is thus delayed. We require more properties from the primitive simulator $\mathcal{S}$ in the ideal world ($\beta = 1$). In the following definitions, we refer to $\mathcal{Q}_F$ as *legal* if $\mathcal{Q}_F$ follows the requirements in Definition 3. We say that a query record $q = ((\mathsf{id}, w, i, b), z)$ is *legal*, if it indeed can be produced in some ideal world executions. By the definition of S-mTCCRL ideal world, this means $z \in \mathcal{W}$ is in the range of the random function $\mathbf{F}$.

Definition 13 (Relevant Queries)**.** *For any legal query record $q = ((\mathsf{id}, w, i, b), z)$, there exists a set $\mathcal{S}_q$ of* **Prim** *queries, such that after $\mathcal{S}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}}}.PhaseSwitch(\{q\})$, it holds: (i) for any $q_{\mathbf{Prim}} \in \mathcal{S}_q$, $\mathcal{S}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}}}.O_1(q_{\mathbf{Prim}})$ generates a $\mathsf{GlobalK}_{\boldsymbol{\Delta}}(\mathsf{id}, \Delta_{\mathsf{id}})$ oracle query; (ii) for any $q_{\mathbf{Prim}} \notin \mathcal{S}_q$, $\mathcal{S}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}}}.O_1(q_{\mathbf{Prim}})$ would not generate a $\mathsf{GlobalK}_{\boldsymbol{\Delta}}(\mathsf{id}, \Delta_{\mathsf{id}})$ oracle query;*

We next prove that our unified model $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}} : \{0,1\}^n \times \mathcal{I} \to \{0,1\}^n$ and its simulator $\mathcal{S}_G$ in Appendix D.1 fulfill Definition 13. For completeness, we prove that the random oracle construction $\mathsf{ROH}^{\mathbf{RO}} : \{0,1\}^n \times \mathcal{I} \to \{0,1\}^n$ and its simulator $\mathcal{S}_{RO}$ in Appendix C.1 also fulfills Definition 13.

Functionality $\mathcal{F}_{\mathsf{aShare}}$

This functionality runs with parties $P_1, \ldots, P_n$. Let $\mathcal{M} \subset [n]$ be the subscript set of corrupted parties and $\mathcal{L}$ be the set of allowed selective-failure queries.

Parameter: A set $\{\mathcal{R}_i\}_{i\in[n]}$ of global-key distributions on $\mathbb{F}_{2^\lambda}$.

Initialization: Upon receiving $(\mathsf{Init}, \mathtt{sid}, i)$ from P_i,

1. If P_i is corrupted, receive $(\mathsf{GK}, \mathtt{sid}, i, \Delta_i)$ from the adversary, where $\Delta_i \in \{0,1\}^\lambda$. Otherwise, $\Delta_i \xleftarrow{\$} \mathcal{R}_i$. Store $(\mathtt{sid}, i, \Delta_i)$ and ignore all subsequent $(\mathsf{Init}, \mathtt{sid}, i)$ calls with the same $(\mathtt{sid}, i)$.

Authenticated shares: Given stored $(\mathtt{sid}, i, \Delta_i)$ for each $i \in [n]$, this functionality outputs random authenticated shares: Upon receiving $(\mathsf{Gen}, \mathtt{sid}, \mathtt{ssid}, \ell_1)$ from all parties, where $\ell_1 \in \mathbb{N}^+$,

1. **Selective-failure queries:** For each $i \in [n] \setminus \mathcal{M}$ and $j \in \mathcal{M}$ in parallel:
   (a) Receive $(\mathsf{Leak}, \mathtt{sid}, \mathtt{ssid}, i, j, L_{i,j})$ from the adversary, where $L_{i,j} \in \mathcal{L}$.
   (b) If $\Delta_i \notin L_{i,j}$, send $(\mathsf{Fail}, \mathtt{sid}, \mathtt{ssid}, i, j)$ to the adversary and abort; otherwise, do nothing.
2. **Global-key extraction with abort:** Receive $(\mathsf{Quit}, \mathtt{sid}, \mathtt{ssid}, b)$ from the adversary. If $b = 1$, send $(\mathsf{Bye}, \mathtt{sid}, \mathtt{ssid}, \{\Delta_i\}_{i\in[n]\setminus\mathcal{M}})$ to the adversary and abort. Otherwise, do nothing.
3. Sample $\boldsymbol{x} \xleftarrow{\$} \{0,1\}^{\ell_1}$ and $\{\langle\!\langle \boldsymbol{x} \rangle\!\rangle_i\}_{i\in[n]} \xleftarrow{\$} \mathsf{AuthShare}(\mathtt{sid}, \mathtt{ssid}, \boldsymbol{x}, \{\Delta_i\}_{i\in[n]})$.
4. For each $i \in [n]$, output $(\mathsf{Out}, \mathtt{sid}, \mathtt{ssid}, i, \Delta_i, \langle\!\langle \boldsymbol{x} \rangle\!\rangle_i)$ to P_i.
5. Ignore all subsequent $(\mathsf{Gen}, \mathtt{sid}, \mathtt{ssid}, \ldots)$ calls with the same $(\mathtt{sid}, \mathtt{ssid})$.

Figure 10: The functionality of authenticated shares.

Functionality $\mathcal{F}_{\mathsf{Zero}}$

This functionality runs with parties $P_1, \ldots, P_n$. Let $\mathcal{M} \subset [n]$ be the subscript set of corrupted parties.

Zero sharing: Upon receiving $(\mathsf{Zero}, \mathtt{sid}, \mathtt{ssid}, \ell)$ from all parties, where $\ell \in \mathbb{N}^+$,

1. Receive $(\mathsf{ZShr}, \mathtt{sid}, \mathtt{ssid}, \{\boldsymbol{z}^i\}_{i\in\mathcal{M}})$ from the adversary, where $\boldsymbol{z}^i \in (\{0,1\}^\lambda)^\ell$ for each $i \in \mathcal{M}$, and sample $\boldsymbol{z}^i \xleftarrow{\$} (\{0,1\}^\lambda)^\ell$ for each $i \notin \mathcal{M}$ such that $\bigoplus_{i\in[n]} \boldsymbol{z}_i = \boldsymbol{0}$.
2. For each $i \in [n]$, output $(\mathsf{Val}, \mathtt{sid}, \mathtt{ssid}, \boldsymbol{z}^i)$ to P_i.
3. Ignore all subsequent $(\mathsf{Zero}, \mathtt{sid}, \mathtt{ssid}, \ldots)$ calls with the same $(\mathtt{sid}, \mathtt{ssid})$.

Figure 11: The functionality of zero sharing.

Functionality $\mathcal{F}_{\mathsf{Com}}$

This functionality runs with parties $P_1, \ldots, P_n$.

Commit: Upon receiving $(\mathsf{Commit}, \mathtt{sid}, \mathtt{ssid}, i, x)$ from P_i, where $x \in \{0,1\}^*$, store $(\mathtt{sid}, \mathtt{ssid}, i, x)$, output $(\mathsf{Committed}, \mathtt{sid}, \mathtt{ssid}, i)$ to all parties, and ignore all subsequent $(\mathsf{Commit}, \mathtt{sid}, \mathtt{ssid}, i, \ldots)$ calls with the same $(\mathtt{sid}, \mathtt{ssid}, i)$.

Open: Given stored $(\mathtt{sid}, \mathtt{ssid}, i, x)$, this functionality outputs the committed value: Upon receiving $(\mathsf{Open}, \mathtt{sid}, \mathtt{ssid}, i)$ from P_i, output $(\mathsf{Opened}, \mathtt{sid}, \mathtt{ssid}, i, x)$ to all parties.

Figure 12: The functionality of commitment.

Functionality $\mathcal{F}_{\mathsf{Rand}}$

This functionality runs with parties $P_1, \ldots, P_n$.

Rand: Upon receiving $(\mathsf{Rand}, \mathtt{sid}, \mathtt{ssid}, \mathcal{S})$ from all parties, where $\mathcal{S}$ is a finite set along with a PPT sampling algorithm, sample $r \xleftarrow{\$} \mathcal{S}$, output $(\mathsf{Res}, \mathtt{sid}, \mathtt{ssid}, r)$ to all parties, and ignore all subsequent $(\mathsf{Rand}, \mathtt{sid}, \mathtt{ssid}, \ldots)$ calls with the same $(\mathtt{sid}, \mathtt{ssid})$.

Figure 13: The functionality of coin-tossing.

Lemma 13. *The unified model* $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}} : \{0,1\}^n \times \mathcal{I} \to \{0,1\}^n$ *in Sect. 5.1 and its simulator* $\mathcal{S}_G$ *in Appendix D.1 fulfill Definition 13.*

Proof. For any query record $q = ((\mathsf{id}, w, i, b), z)$, the set $\mathcal{S}_q$ is defined as

$$\mathcal{S}_q := \{(+, k, x), (-, k, y)\},$$

where $k = g(i)$, $x = f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i)$ and $y = f_{\mathsf{out}}^{-1}(w \oplus \Delta_{\mathsf{id}}, i, z \oplus b \cdot \Delta_{\mathsf{id}})$. By this definition and by: (i) the properties of f_{in} and f_{out}, and (ii) the definition of $\mathcal{S}_G$, we have:

- When $\mathcal{S}_G.O_1(+, k, x)$ is queried, $\mathcal{S}_G$ would compute the "right" key $\Delta_{\mathsf{id}} \leftarrow f_{\mathsf{in}}^{-1}(i, x) \oplus w$ at line 11 and then query $\mathsf{GlobalK}_{\mathbf{\Delta}}(\mathsf{id}, \Delta_{\mathsf{id}})$;
- When $\mathcal{S}_G.O_1(-, k, y)$ is queried, $\mathcal{S}_G$ would compute the "right" key $\Delta_{\mathsf{id}} \leftarrow kd_y(w, i, b, y, z)$ at line 11 and then query $\mathsf{GlobalK}_{\mathbf{\Delta}}(\mathsf{id}, \Delta_{\mathsf{id}})$.
- By the restriction at line 10, for any $k \neq g(i)$, no $\mathsf{GlobalK}_{\mathbf{\Delta}}$ would be issued. In the following, we discuss the $k = g(i)$ case. Since $f_{\mathsf{in}}(\cdot, i)$ is a bijection, the condition $\Delta_{\mathsf{id}} = f_{\mathsf{in}}^{-1}(i, x) \oplus w$ is equivalent to $x = f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i)$. Similarly, since $f_{\mathsf{out}}(w, i, \cdot)$ and $h_{w,i,b,y}(\cdot)$ are both bijective, the condition $\Delta_{\mathsf{id}} = kd_y(w, i, b, y, z)$ is equivalent to $f_{\mathsf{out}}(w \oplus \Delta_{\mathsf{id}}, i, y) \oplus b \cdot \Delta_{\mathsf{id}} = z$. Then, this condition is equivalent to $y = f_{\mathsf{out}}^{-1}(x \oplus \Delta_{\mathsf{id}}, i, z \oplus b \cdot \Delta_{\mathsf{id}})$. Therefore, for any query $q_{\mathbf{Prim}} \notin \mathcal{S}_q$, no valid $\mathsf{GlobalK}_{\mathbf{\Delta}}(\mathsf{id}, \Delta_{\mathsf{id}})$ would be generated.

Consequently, this set $\mathcal{S}_q$ indeed fulfills the condition of Definition 13. Note that this follow from both the properties of $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$ and the design of $\mathcal{S}_G$. □

Lemma 14. *The* $\mathsf{ROH}^{\mathbf{RO}} : \{0,1\}^n \times \mathcal{I} \to \{0,1\}^n$ *in Sect. B and its simulator* $\mathcal{S}_{RO}$ *in Appendix C.1 fulfill Definition 13.*

Proof. For any query record $q = ((\mathsf{id}, w, i, b), z)$, the set $\mathcal{S}_q$ is defined as

$$\mathcal{S}_q := \{(f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i))\}\,.$$

By this definition and by: (i) the properties of f_{in}, and (ii) the definition of $\mathcal{S}_{RO}$, we have:

- When $\mathcal{S}_{RO}.O_1(x)$ is queried, $\mathcal{S}_{RO}$ would compute the "right" key $\Delta_{\mathsf{id}} \leftarrow f_{\mathsf{in}}^{-1}(i, x) \oplus w$ at line 11 and then query $\mathsf{GlobalK}_{\mathbf{\Delta}}(\mathsf{id}, \Delta_{\mathsf{id}})$;
- Since $f_{\mathsf{in}}(\cdot, i)$ is a bijection, the condition $\Delta_{\mathsf{id}} = f_{\mathsf{in}}^{-1}(i, x) \oplus w$ is equivalent to $x = f_{\mathsf{in}}(w \oplus \Delta_{\mathsf{id}}, i)$. Therefore, for any query $q_{\mathbf{Prim}} \notin \mathcal{S}_q$, no valid $\mathsf{GlobalK}_{\mathbf{\Delta}}(\mathsf{id}, \Delta_{\mathsf{id}})$ would be generated.

Consequently, this set $\mathcal{S}_q$ indeed fulfills the condition of Definition 13. Note that this follow from both the properties of $\mathsf{ROH}^{\mathbf{RO}}$ and the design of $\mathcal{S}_{RO}$. □

Definition 14 (Lazy Simulation)**.** *For any legal $\mathcal{Q}_F$, define*

$$\mathcal{S}_{\mathbf{Prim}} := \bigcup_{q \in \mathcal{Q}_F} \mathcal{S}_q\,,$$

where $\mathcal{S}_q$ is the **Prim***-query set relevant to q as in Definition 13. Consider two worlds similar to Figure 1 but we only consider indistinguishability of the simulated* **Prim** *interface:*

- *This world is the $\beta = 1$ case and $\mathcal{S}^{\mathsf{GlobalK}_{\mathbf{\Delta}}}$ performs phase switch using $\mathcal{Q}_F$.*
- *In this world, we delay the phase switch. $\mathcal{S}^{\mathsf{GlobalK}_{\mathbf{\Delta}}}$ follows its* ***Phase 1*** *strategy to simulate any $q_{\mathbf{Prim}} \notin \mathcal{S}_{\mathbf{Prim}}$. Denote $q^*_{\mathbf{Prim}}$ as the first* **Prim***-query such that $q^*_{\mathbf{Prim}} \in \mathcal{S}_{\mathbf{Prim}}$. $\mathcal{S}^{\mathsf{GlobalK}_{\mathbf{\Delta}}}$ immediately performs*

  $$\mathcal{S}^{\mathsf{GlobalK}_{\mathbf{\Delta}}}.\mathit{PhaseSwitch}(\mathcal{Q}_F)\,,$$

  and then answers $q^*_{\mathbf{Prim}}$ using the ***Phase 2*** *strategy.*

The lazy simulation property states that the statistical distance between the two worlds is bounded by $\varepsilon_{\mathsf{lazy}}$.

Informally, if $\mathcal{S}^{\mathsf{GlobalK}_{\mathbf{\Delta}}}$ fulfills Definition 14, then it uses "very similar" strategies to answer $q_{\mathbf{Prim}} \notin \mathcal{S}_{\mathbf{Prim}}$ in **Phase 1** and **Phase 2**. We next prove that the simulators for both constructions in this paper, i.e., $\mathcal{S}_G$ in Appendix D.1 and $\mathcal{S}_{RO}$ in Appendix C.1, indeed fulfill Definition 14.

Lemma 15. *Against a $(q_C, q_P, M, q_{L,max}, \mu)$-bounded distinguisher $\mathcal{D}$, the simulator $\mathcal{S}_G$ in Appendix D.1 fulfill Definition 14 with*

$$\varepsilon_{\mathsf{lazy}} = 2\mathbf{Adv}^{\mathrm{mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}, \vec{\mathcal{R}}, \mathcal{P}, u}(\mathcal{B}) + \frac{3\mu(q_P + q_C)}{2^n} + \frac{2}{T}.$$

Proof. By construction, for any $(\delta, k, v) \notin \mathcal{S}_{\mathbf{Prim}}$ we have:

- When $\mathcal{S}_G.O_1(\delta, k, v)$ is queried before $\mathcal{S}^{\mathsf{GlobalK}_{\mathbf{\Delta}}, \mathbf{ic}}.\mathit{PhaseSwitch}(\mathcal{Q}_F)$ is executed, it holds $\mathcal{Q}_{SF} = \emptyset$, and $\mathcal{S}_G$ would simply query its randomness $\mathbf{ic}(\delta, k, v)$ to answer $\mathcal{S}_G.O_1(\delta, k, v)$;
- When $\mathcal{S}_G.O_1(\delta, k, v)$ is queried after $\mathcal{S}^{\mathsf{GlobalK}_{\mathbf{\Delta}}, \mathbf{ic}}.\mathit{PhaseSwitch}(\mathcal{Q}_F)$ is executed, since $(\delta, k, v) \notin \mathcal{S}_{\mathbf{Prim}}$, $\mathcal{S}_G$ would not find any "right" key candidate Δ'_{id} with $\mathsf{GlobalK}_{\mathbf{\Delta}}(\mathsf{id}, \Delta'_{\mathsf{id}}) = \mathbf{true}$. Consequently, unless $\mathcal{S}^{\mathsf{GlobalK}_{\mathbf{\Delta}}, \mathbf{ic}}$ aborts, it would query its randomness $\mathbf{ic}(\delta, k, v)$ to answer $\mathcal{S}_G.O_1(\delta, k, v)$.

Therefore, the gap between the two cases is bounded by the abort probability of $\mathcal{S}_G$, which clearly does not exceed $\mathbf{Adv}^{\text{S-mTCCRL}}_{\mathsf{GH}[f_{\mathsf{in}},g,f_{\mathsf{out}}]^{\mathbf{E}},\overrightarrow{\mathcal{R}},\mathcal{P},u}(\mathcal{D},\mathcal{S}_G)$. Thus the claim. □

Lemma 16. *Against a $(q_C, q_P, M, q_{L,max}, \mu)$-bounded distinguisher $\mathcal{D}$, the simulator $\mathcal{S}_{RO}^{\mathsf{GlobalK}_{\mathbf{\Delta}},\mathbf{ro}}$ in Appendix C.1 fulfill Definition 14 with*

$$\varepsilon_{\mathsf{lazy}} = \frac{(M+1)(q_{L,max}+1)}{C} \cdot \left(\mu q_P + \frac{(\mu-1)\cdot q_C}{2}\right).$$

Proof. By construction, for any $(x) \notin \mathcal{S}_{\mathbf{Prim}}$ we have:

- When $\mathcal{S}_{RO}^{\mathsf{GlobalK}_{\mathbf{\Delta}},\mathbf{ro}}.O_1(x)$ is queried before $\mathcal{S}_{RO}^{\mathsf{GlobalK}_{\mathbf{\Delta}},\mathbf{ro}}.\mathit{PhaseSwitch}(\mathcal{Q}_F)$ is executed, it holds $\mathcal{Q}_{SF} = \emptyset$, and $\mathcal{S}_{RO}^{\mathsf{GlobalK}_{\mathbf{\Delta}},\mathbf{ro}}$ would query its randomness $\mathbf{ro}(x)$ to answer $\mathcal{S}_{RO}^{\mathsf{GlobalK}_{\mathbf{\Delta}},\mathbf{ro}}.O_1(x)$;
- When $\mathcal{S}_{RO}.O_1(x)$ is queried after $\mathcal{S}_{RO}^{\mathsf{GlobalK}_{\mathbf{\Delta}},\mathbf{ro}}.\mathit{PhaseSwitch}(\mathcal{Q}_F)$ is executed, since $(x) \notin \mathcal{S}_{\mathbf{Prim}}$, $\mathcal{S}_{RO}^{\mathsf{GlobalK}_{\mathbf{\Delta}},\mathbf{ro}}$ would not find any "right" key candidate Δ'_{id} with $\mathsf{GlobalK}_{\mathbf{\Delta}}(\mathsf{id},\Delta'_{\mathsf{id}}) = \mathbf{true}$. Consequently, unless $\mathcal{S}_{RO}^{\mathsf{GlobalK}_{\mathbf{\Delta}},\mathbf{ro}}$ aborts, it would query its randomness $\mathbf{ro}(x)$ to answer $\mathcal{S}_{RO}^{\mathsf{GlobalK}_{\mathbf{\Delta}},\mathbf{ro}}.O_1(x)$.

Therefore, the gap between the two cases is bounded by the abort probability of $\mathcal{S}_{RO}^{\mathsf{GlobalK}_{\mathbf{\Delta}},\mathbf{ro}}$, which clearly does not exceed $\mathbf{Adv}^{\text{S-mTCCRL}}_{\mathsf{ROH}^{\mathbf{RO}},\overrightarrow{\mathcal{R}},\mathcal{P}_{\mathsf{free}},u}(\mathcal{D},\mathcal{S}_{RO})$. Thus the claim. □

F.3 Proof of Theorem 2

Theorem 2. *Given a hash function $\mathsf{H}^{\mathbf{Prim}}$, which invokes* **Prim** *c_{H} times, such that an $(\frac{n^2\ell_2}{4}, q_P + c_{\mathsf{H}} \cdot \frac{n^2\ell_2}{4}, n, n-1, 1)$-bounded distinguisher has advantage at most ε_{H} in the S-mTCCRL game (Definition 3) for at most n users and $(\lambda-1)$-bit min-entropy per key, and satisfies the $\varepsilon_{\mathsf{lazy}}$-lazy* **Prim** *simulation, and a family $\mathcal{H} = \{\mathsf{H}' : (\{0,1\}^\lambda)^{\ell_2} \to \{0,1\}^\lambda\}$ of linear ε-almost 1-universal hash functions, protocol Π_{LaAND} (Figs. 5 and 6) UC-realizes functionality $\mathcal{F}_{\mathsf{LaAND}}$ (Figs. 3 and 4) in the $(\mathcal{F}_{\mathsf{aShare}}, \mathcal{F}_{\mathsf{Zero}}, \mathcal{F}_{\mathsf{Com}}, \mathcal{F}_{\mathsf{Rand}})$-hybrid model with an advantage of at most*

$$\varepsilon_{\mathsf{H}} + 2\varepsilon + \frac{5}{2^\lambda} + \varepsilon_{\mathsf{lazy}}$$

against any adversary making at most q_P **Prim***-queries, when the command $(\mathsf{Gen}, \mathtt{sid}, \ldots)$ is invoked only once per* `sid`.

Proof. The requirement that $(\mathsf{Gen}, \mathtt{sid}, \ldots)$ command is invoked only once per `sid` ensures that, for UC security, the dummy adversary (i.e., environment) cannot use the global keys of honest parties $\{\Delta_i\}_{i\in[n]\setminus\mathcal{M}}$ output at the end of one Gen execution to break the check in another execution. For a single Gen execution, we prove the UC security via the following hybrid argument:

- $\mathbf{Hybrid}_0$. This is the real-world execution of protocol Π_{LaAND}.
- $\mathbf{Hybrid}_1$. This hybrid is identical to the previous one, except that there is a machine that plays the role of all honest parties as per Π_{LaAND} and emulates all subroutine functionalities (including $\mathcal{F}_{\mathsf{aShare}}$, $\mathcal{F}_{\mathsf{Zero}}$, $\mathcal{F}_{\mathsf{Com}}$, and $\mathcal{F}_{\mathsf{Rand}}$) for all parties. This modification is conceptual so the joint distribution in this hybrid is perfectly indistinguishable from the previous one.
- $\mathbf{Hybrid}_2$. This hybrid is identical to the previous one, except that the machine changes the way to emulate $\mathcal{F}_{\mathsf{aShare}}$ for all honest parties: it reversely samples uniform $\mathsf{M}_u[\boldsymbol{b}^j]$ and sets $\mathsf{K}_u[\boldsymbol{b}^j] := \mathsf{M}_u[\boldsymbol{b}^j] \oplus \boldsymbol{b}^j \cdot \Delta_u$ for each $j, u \in [n] \setminus \mathcal{M}$ and $u \neq j$.

  Clearly, this hybrid is perfectly indistinguishable from the previous one.

- **Hybrid**$_3$. This hybrid is identical to the previous one, except that the machine changes the way to compute the messages sent from honest parties to corrupted parties at step 5a:

  1. It defines $\mathbf{\Delta} := \{\Delta_j\}_{j\in[n]\setminus\mathcal{M}}$, invokes the universal S-mTCCRL simulator $\mathcal{S}_{\mathsf{H}}$ for hash function $\mathsf{H}^{\mathbf{Prim}}$ to emulate the ideal primitive **Prim**, and implements the oracle $\mathsf{GlobalK}_{\mathbf{\Delta}}$ accessed by $\mathcal{S}_{\mathsf{H}}$.
  2. It samples uniform $\{u_k^{j,i}\}_{i\in\mathcal{M},j\in[n]\setminus\mathcal{M},k\in[\ell_2]}$ on behalf of all honest parties at step 5a and computes the following $\mathcal{Q}_F$ to switch $\mathcal{S}_{\mathsf{H}}$ from **Phase 1** into **Phase 2** before sending the message of any honest party:

$$\mathcal{Q}_F := \left\{ \left(j, \underbrace{\mathsf{M}_j[a_k^i]}, T_{j,i,k}, \underbrace{\bigoplus_{i\in\mathcal{M}} b_k^i} \oplus \bigoplus_{j\in[n]\setminus\mathcal{M}} b_k^j\right), \gamma_{j,i,k} \right\}_{\substack{i\in\mathcal{M},\\ j\in[n]\setminus\mathcal{M},\\ k\in[\ell_2]}},$$

$$\gamma_{j,i,k} := u_k^{j,i} \oplus \mathsf{H}(\underbrace{\mathsf{M}_j[a_k^i]}, T_{j,i,k}) \oplus \bigoplus_{u\in[n]\setminus\mathcal{M},u\neq j} \left(\mathsf{M}_j[b_k^u] \oplus \mathsf{M}_u[b_k^j]\right)$$
$$\oplus z_k^j \oplus \bigoplus_{u\in\mathcal{M}} \left(\underbrace{\mathsf{M}_j[b_k^u]} \oplus \underbrace{\mathsf{K}_u[b_k^j]} \oplus b_k^j \cdot \underbrace{\Delta_u}\right),$$

     where the underlined terms are the randomness of corrupted parties and are extracted from the emulated $\mathcal{F}_{\mathsf{aShare}}$.
  3. It follows Π_{LaAND} to compute all values used by the honest parties since step 5b.

  From Lemma 17, these two hybrids are indistinguishable with advantage at most ε_{H}.

- **Hybrid**$_{3.5}$. In this hybrid, the machine constructs $\mathcal{Q}_F$ differently, so that the machine do not explicitly use the the $\mathsf{M}_j[b_k^u]$ value for $j, u \in [n] \setminus \mathcal{M}, j \neq u$ and $k \in [\ell_2]$. Consider the following two cases:

  – **Case 1:** If there is only one honest party, then the term

$$\bigoplus_{u\in[n]\setminus\mathcal{M},u\neq j} \left(\mathsf{M}_j[b_k^u] \oplus \mathsf{M}_u[b_k^j]\right),$$

    would not appear in $\mathcal{Q}_F$. Thus, the machine follows the same strategy to construct $\mathcal{Q}_F$.

  – **Case 2:** If there are more than one honest party, then the machine computes $\mathcal{Q}_F$ as

$$\mathcal{Q}_F := \left\{ \left(j, \underbrace{\mathsf{M}_j[a_k^i]}, T_{j,i,k}, \underbrace{\bigoplus_{i\in\mathcal{M}} b_k^i} \oplus \bigoplus_{j\in[n]\setminus\mathcal{M}} b_k^j\right), \gamma_{j,i,k} \right\}_{\substack{i\in\mathcal{M},\\ j\in[n]\setminus\mathcal{M},\\ k\in[\ell_2]}},$$

$$\gamma_{j,i,k} := u_k^{j,i} \oplus \mathsf{H}(\underbrace{\mathsf{M}_j[a_k^i]}, T_{j,i,k}) \oplus z_k^j$$
$$\oplus \bigoplus_{u\in\mathcal{M}} \left(\underbrace{\mathsf{M}_j[b_k^u]} \oplus \underbrace{\mathsf{K}_u[b_k^j]} \oplus b_k^j \cdot \underbrace{\Delta_u}\right),$$

    We argue that $\mathcal{Q}_F$ computed in this way follows the same distribution as the $\mathcal{Q}_F$ in **Hybrid**$_3$, thus the joint distribution in this hybrid is identical to **Hybrid**$_3$. Let us fix the index $k \in [\ell_2]$. First, $\{z_k^i\}_{i\in[n]}$ is returned by the $\mathcal{F}_{\mathsf{Zero}}$ functionality, which means that $\{z_k^j\}_{j\in[n]\setminus\mathcal{M}}$ is uniformly random conditioned on

$$\bigoplus_{j\in[n]\setminus\mathcal{M}} z_k^j = \bigoplus_{i\in\mathcal{M}} z_k^i = Z_k\,.$$

In other words, $\{z_k^j\}_{j\in[n]\setminus\mathcal{M}}$ forms a random additive secret sharing of Z_k. Let $V_j = \bigoplus_{u\in[n]\setminus\mathcal{M},u\neq j}(\mathsf{M}_j[b_k^u]\oplus\mathsf{M}_u[b_k^j])$, it holds that

$$\begin{aligned}\bigoplus_{j\in[n]\setminus\mathcal{M}} V_j &= \bigoplus_{j\in[n]\setminus\mathcal{M}}\bigoplus_{u\in[n]\setminus\mathcal{M},u\neq j} \mathsf{M}_j[b_k^u]\oplus\mathsf{M}_u[b_k^j]\,,\\ &= \bigoplus_{j,u\in[n]\setminus\mathcal{M},j\neq u} \mathsf{M}_j[b_k^u]\oplus\mathsf{M}_u[b_k^j]\,,\\ &= \bigoplus_{j,u\in[n]\setminus\mathcal{M},j\neq u} \mathsf{M}_j[b_k^u]\oplus\mathsf{M}_j[b_k^u] = 0\,,\end{aligned}$$

which implies that $\{V_j\}_{j\in[n]\setminus\mathcal{M}}$ forms an additive secret sharing of 0.

Therefore, in $\mathbf{Hybrid}_3$, under the mask of $\{z_k^j\}_{j\in[n]\setminus\mathcal{M}}$, the value

$$\left\{\bigoplus_{u\in[n]\setminus\mathcal{M},u\neq j}(\mathsf{M}_j[b_k^u]\oplus\mathsf{M}_u[b_k^j])\oplus z_k^j\right\}_{j\in[n]\setminus\mathcal{M}},$$

forms a random additive secret sharing of Z_k. In this hybrid, we change it to z_k^j. Since z_k^j does not appear elsewhere in the simulation, the $\mathcal{Q}_F$ is identically distributed in the two hybrids.

- $\mathbf{Hybrid}_4$. This hybrid is identical to the previous one, except that the machine additionally computes the following $(\{\boldsymbol{e}^j\}_{j\in[n]\setminus\mathcal{M}}, \boldsymbol{q}, \delta)$:

  1. For each $i\in\mathcal{M}, j\in[n]\setminus\mathcal{M}, k\in[\ell_2]$, it follows step 5a to compute $u_k^{i,j}\in\{0,1\}^\lambda$ and receives some $\widetilde{u_k^{i,j}}\in\{0,1\}^\lambda$ from the corrupted P_i in step 5a.
  2. For each $j\in[n]\setminus\mathcal{M}$, it computes an error $\boldsymbol{e}^j\in(\{0,1\}^\lambda)^{\ell_2}$, where for each $k\in[\ell_2]$,

  $$e_k^j := \bigoplus_{i\in\mathcal{M}}\left(\widetilde{u_k^{i,j}}\oplus u_k^{i,j}\right)\in\{0,1\}^\lambda,$$

  and follows step 5a and 5b to compute $(\widetilde{\boldsymbol{s}^j}, \widetilde{\boldsymbol{d}^j})$ (resp., $(\boldsymbol{s}^j,\boldsymbol{d}^j)$) using $\{\widetilde{u_k^{i,j}}\}_{i\in\mathcal{M}}$ (resp., $\{u_k^{i,j}\}_{i\in\mathcal{M}}$). According to the steps, it holds that

  $$\begin{aligned}\widetilde{\boldsymbol{s}^j} &= \boldsymbol{s}^j\oplus\boldsymbol{a}^j\odot\boldsymbol{e}^j\in(\{0,1\}^\lambda)^{\ell_2},\\ \widetilde{\boldsymbol{d}^j} &= \boldsymbol{d}^j\oplus\boldsymbol{a}^j\wedge\mathsf{lsb}(\boldsymbol{e}^j)\in\{0,1\},\end{aligned}$$

  where $\odot$ denotes component-wise multiplication. Note that for each $j\in[n]\setminus\mathcal{M}$, the machine uses $(\widetilde{\boldsymbol{s}^j},\widetilde{\boldsymbol{d}^j})$ to proceed as an honest P_j after step 5b.
  3. It follows step 5a and 5b to compute $\{\boldsymbol{s}^i,\boldsymbol{d}^i\}_{i\in\mathcal{M}}$, extracts $\{\widetilde{\boldsymbol{d}^i}\}_{i\in\mathcal{M}}$ from the emulated $\mathcal{F}_{\mathsf{Com}}$ at step 6, computes

  $$\begin{aligned}\boldsymbol{q} &:= \bigoplus_{i\in\mathcal{M}}\left(\widetilde{\boldsymbol{d}^i}\oplus\boldsymbol{d}^i\right)\in\{0,1\}^{\ell_2},\\ \boldsymbol{d} &:= \bigoplus_{i\in[n]}\boldsymbol{d}^i\\ &= \mathsf{lsb}\Big(\bigoplus_{i\in[n]}\boldsymbol{s}^i\Big) = (\boldsymbol{a}\wedge\boldsymbol{b}\oplus\boldsymbol{r})\cdot\mathsf{lsb}\Big(\bigoplus_{i\in[n]}\Delta_i\Big)\in\{0,1\}^{\ell_2}, \qquad (51)\\ \widetilde{\boldsymbol{d}} &:= \bigoplus_{i\in[n]}\widetilde{\boldsymbol{d}^i} = \boldsymbol{d}\oplus\boldsymbol{q}\oplus\bigoplus_{j\in[n]\setminus\mathcal{M}}\boldsymbol{a}^j\wedge\mathsf{lsb}(\boldsymbol{e}^j)\in\{0,1\}^{\ell_2}, \qquad (52)\end{aligned}$$

follows step 7 to compute $\{\boldsymbol{t}^i\}_{i\in\mathcal{M}}$ based on $(\{\boldsymbol{s}^i, \Delta_i\}_{i\in\mathcal{M}}, \boldsymbol{d})$, extracts $\{\widetilde{\alpha^i}\}_{i\in\mathcal{M}}$ from the emulated $\mathcal{F}_{\mathsf{Com}}$ at step 9, and computes

$$\delta := \bigoplus_{i\in\mathcal{M}} \left(\widetilde{\alpha^i} \oplus \mathsf{H}'(\boldsymbol{t}^i)\right) \in \{0,1\}^\lambda.$$

This modification has nothing to do with the joint distribution so that the two hybrids are perfectly indistinguishable.

- **Hybrid**$_5$. This hybrid is identical to the previous one, except that the machine changes the way to compute the messages committed by all honest parties at step 6:

  According to the public LSBs of all honest parties and the global keys of corrupted parties extracted from the emulated $\mathcal{F}_{\mathsf{aShare}}$,

  - If $\mathsf{lsb}(\bigoplus_{i\in[n]} \Delta_i) = 0$, it samples uniform $\{\widetilde{\boldsymbol{d}^j}\}_{j\in[n]\setminus\mathcal{M}}$ in the emulated $\mathcal{F}_{\mathsf{Com}}$ at step 7 on behalf of all honest parties conditioned on

    $$\bigoplus_{j\in[n]\setminus\mathcal{M}} \widetilde{\boldsymbol{d}^j} = \bigoplus_{i\in\mathcal{M}} \widetilde{\boldsymbol{d}^i} \oplus \boldsymbol{q} \oplus \bigoplus_{j\in[n]\setminus\mathcal{M}} \boldsymbol{a}^j \wedge \mathsf{lsb}(\boldsymbol{e}^j).$$

  - If $\mathsf{lsb}(\bigoplus_{i\in[n]} \Delta_i) = 1$, it samples uniform $\{\widetilde{\boldsymbol{d}^j}\}_{j\in[n]\setminus\mathcal{M}}$ in the emulated $\mathcal{F}_{\mathsf{Com}}$ at step 7 on behalf of all honest parties conditioned on

    $$\bigoplus_{j\in[n]\setminus\mathcal{M}} \widetilde{\boldsymbol{d}^j} = \bigoplus_{i\in\mathcal{M}} \widetilde{\boldsymbol{d}^i} \oplus \boldsymbol{a} \wedge \boldsymbol{b} \oplus \boldsymbol{r} \oplus \boldsymbol{q} \oplus \bigoplus_{j\in[n]\setminus\mathcal{M}} \boldsymbol{a}^j \wedge \mathsf{lsb}(\boldsymbol{e}^j).$$

  From Equation 51 and Equation 52, the distribution of $\bigoplus_{j\in[n]\setminus\mathcal{M}} \widetilde{\boldsymbol{d}^j}$ is identical in both two hybrids. In addition, if there are at least two honest parties, the distribution of the $\widetilde{\boldsymbol{d}^j}$ per honest party P_j is uniform in the previous hybrid due to the uniform mask $\mathsf{K}_j[\boldsymbol{r}^i]$ for another honest party P_i in $\widetilde{\boldsymbol{s}^j}$, having the same distribution as in this hybrid. Therefore, this hybrid is perfectly indistinguishable from the previous one.

- **Hybrid**$_6$. This hybrid is identical to the previous one, except that the machine changes the abort condition of all honest parties in step 10 as follows:

  - Let abort denote the event that $\mathsf{lsb}(\bigoplus_{i\in[n]} \Delta_i) = 0$ or

    $$\mathsf{H}'\left(\underbrace{\bigoplus_{j\in[n]\setminus\mathcal{M}} \left(\boldsymbol{a}^j \odot \boldsymbol{e}^j\right) \oplus \left(\boldsymbol{q} \oplus \bigoplus_{j\in[n]\setminus\mathcal{M}} \boldsymbol{a}^j \wedge \mathsf{lsb}(\boldsymbol{e}^j)\right) \cdot \bigoplus_{j\in[n]\setminus\mathcal{M}} \Delta_j}_{\text{denoted by } \boldsymbol{m}}\right) \neq \delta.$$

    The machine aborts all honest parties at step 10 if and only if the event abort happens.

  We claim that, after changing the abort condition, the probability of abort at step 10 in this hybrid has a negligible difference with its counterpart in the previous hybrid. As H' is linear,

the check at step 10 in the previous hybrid can be equivalently written as

$$
\begin{aligned}
0 &= \bigoplus_{i\in[n]} \widetilde{\alpha^i} \\
&= \bigoplus_{j\in[n]\setminus\mathcal{M}} \widetilde{\alpha^j} \oplus \bigoplus_{i\in\mathcal{M}} \mathsf{H}'(\boldsymbol{t}^i) \oplus \delta \\
&= \bigoplus_{j\in[n]\setminus\mathcal{M}} \mathsf{H}'\Big(\widetilde{\boldsymbol{s}^j} \oplus \widetilde{\boldsymbol{d}} \cdot \Delta_j\Big) \oplus \bigoplus_{i\in\mathcal{M}} \mathsf{H}'\Big(\boldsymbol{s}^i \oplus \boldsymbol{d} \cdot \Delta_i\Big) \oplus \delta \\
&= \bigoplus_{j\in[n]\setminus\mathcal{M}} \mathsf{H}'\left(\boldsymbol{s}^j \oplus \boldsymbol{a}^j \odot \boldsymbol{e}^j \oplus \left(\boldsymbol{d} \oplus \boldsymbol{q} \oplus \bigoplus_{j\in[n]\setminus\mathcal{M}} \boldsymbol{a}^j \wedge \mathsf{lsb}(\boldsymbol{e}^j)\right) \cdot \Delta_j\right) \\
&\quad \oplus \bigoplus_{i\in\mathcal{M}} \mathsf{H}'\Big(\boldsymbol{s}^i \oplus \boldsymbol{d} \cdot \Delta_i\Big) \oplus \delta \\
&= \mathsf{H}'\Big(\bigoplus_{i\in[n]} \boldsymbol{s}^i \oplus \boldsymbol{d} \cdot \bigoplus_{i\in[n]} \Delta_i\Big) \oplus \mathsf{H}'(\boldsymbol{m}) \oplus \delta \\
&= \mathsf{H}'\Big(\overline{\mathsf{lsb}(\bigoplus_{i\in[n]} \Delta_i)} \cdot (\boldsymbol{a} \wedge \boldsymbol{b} \oplus \boldsymbol{r}) \cdot \bigoplus_{i\in[n]} \Delta_i\Big) \oplus \mathsf{H}'(\boldsymbol{m}) \oplus \delta.
\end{aligned}
\tag{53}
$$

If $\mathsf{lsb}(\bigoplus_{i\in[n]} \Delta_i) = 1$, all honest parties behave identically for abort in both two hybrids. If $\mathsf{lsb}(\bigoplus_{i\in[n]} \Delta_i) = 0$, Equation 53 in the previous hybrid is further equivalent to

$$
\begin{aligned}
&\mathsf{H}'\left(\boldsymbol{a} \wedge \boldsymbol{b} \oplus \boldsymbol{r} \oplus \boldsymbol{q} \oplus \bigoplus_{j\in[n]\setminus\mathcal{M}} \boldsymbol{a}^j \wedge \mathsf{lsb}(\boldsymbol{e}^j)\right) \cdot \bigoplus_{j\in[n]\setminus\mathcal{M}} \Delta_j \\
&\qquad = \delta \oplus \mathsf{H}'\left(\bigoplus_{j\in[n]\setminus\mathcal{M}} \boldsymbol{a}^j \odot \boldsymbol{e}^j\right) \oplus \mathsf{H}'(\boldsymbol{a} \wedge \boldsymbol{b} \oplus \boldsymbol{r}) \cdot \bigoplus_{i\in\mathcal{M}} \Delta_i
\end{aligned}
\tag{54}
$$

where H' is applied to vectors over $\{0,1\}^{\ell_2}$ by first casting them to $(\{0,1\}^{\lambda})^{\ell_2}$.

Without loss of generality, consider there is only one honest party P_{j_0} as the adversary has a lower advantage to ensure Equation 54 if there are at least two honest parties. First, the randomness $\boldsymbol{r}$ guarantees that the left-hand hash input is zero with probability $2^{-\ell_2} \leq 2^{-\lambda}$. Second, conditioned on this input is non-zero, the hash output is zero with probability at most ε since H' is ε-almost universal. Third, conditioned on both the hash input and output are non-zero, Equation 54 can hold with probability at most $\frac{1}{2^{\lambda-1}}$.

To see the third probability, on the one hand, we move to compute the probability that the adversary successfully guesses exact $0 \leq c \leq \lambda - 1$ non-LSB bits of Δ_{j_0}. This probability upper bounds the probability that δ or $\{\boldsymbol{e}^j\}_{j\in[n]\setminus\mathcal{M}}$ can depend on c non-LSB bits of Δ_{j_0}. We stress that, in the previous hybrid, the adversary can only guess some bits of Δ_{j_0} via the selective-failure queries in the emulated $\mathcal{F}_{\mathsf{aShare}}$ as its received messages are now independent of Δ_{j_0} before step 10. Thus, this probability is exactly $\frac{1}{2^c}$. On the other hand, the remaining $\lambda - 1 - c$ non-LSB bits of Δ_{j_0} are uniform and independent of δ or $\{\boldsymbol{e}^j\}_{j\in[n]\setminus\mathcal{M}}$ on the right hand (otherwise, the adversary must successfully guess more than c non-LSB bits of Δ_{j_0}), this randomness ensures that these bits comply with Equation 54 with probability $\frac{1}{2^{\lambda-1-c}}$. Putting the two cases together, for any c, the third probability is at most $\frac{1}{2^c} \cdot \frac{1}{2^{\lambda-1-c}} = \frac{1}{2^{\lambda-1}}$.

Therefore, if $\mathsf{lsb}(\bigoplus_{i\in[n]} \Delta_i) = 0$, the check at step 10 passes with probability at most $\frac{3}{2^\lambda} + \varepsilon$, while it passes with zero probability in this hybrid.

In conclusion, the two hybrids are statistically indistinguishable with an statistical distance at most $\frac{3}{2^\lambda} + \varepsilon$.

- **Hybrid**$_7$. This hybrid is identical to the previous one, except that the machine proceeds as follows if and only if $\mathsf{lsb}(\bigoplus_{i\in[n]}\Delta_i)=0$ (this event is identified by the machine at step 1):

  1. It samples uniform $\{\boldsymbol{a}^j\}_{j\in[n]\setminus\mathcal{M}}$ for all honest parties, instead of using the values from the emulated $\mathcal{F}_{\mathsf{aShare}}$.
  2. It does not compute $\{(\widetilde{\boldsymbol{s}^j},\widetilde{\boldsymbol{d}^j})\}_{j\in[n]\setminus\mathcal{M}}$ for all honest parties at step 5b but uses uniform $\{\widetilde{\boldsymbol{d}^j}\}_{j\in[n]\setminus\mathcal{M}}$ in the emulated $\mathcal{F}_{\mathsf{Com}}$ at step 7 conditioned on
  $$\bigoplus_{j\in[n]\setminus\mathcal{M}}\widetilde{\boldsymbol{d}^j}=\bigoplus_{i\in\mathcal{M}}\widetilde{\boldsymbol{d}^i}\oplus\boldsymbol{q}\oplus\bigoplus_{j\in[n]\setminus\mathcal{M}}\boldsymbol{a}^j\wedge\mathsf{lsb}(\boldsymbol{e}^j).$$
  3. It computes $\boldsymbol{a}:=\bigoplus_{i\in[n]}\boldsymbol{a}^i$, where $\{\boldsymbol{a}^i\}_{i\in\mathcal{M}}$ are extracted from the emulated $\mathcal{F}_{\mathsf{aShare}}$, and samples $\boldsymbol{b},\boldsymbol{r}\xleftarrow{\$}\{0,1\}^{\ell_2}$ instead.
  4. It samples $\{\Delta_j\}_{j\in[n]\setminus\mathcal{M}}\xleftarrow{\$}\{\mathcal{R}_j\}_{j\in[n]\setminus\mathcal{M}}$ conditioned on that $\{\Delta_j\}_{j\in[n]\setminus\mathcal{M}}$ comply with the selective-failure queries in the emulated $\mathcal{F}_{\mathsf{aShare}}$.
  5. It does not compute $\{\widetilde{\boldsymbol{t}^j}\}_{j\in[n]\setminus\mathcal{M}}$ (resp., $\{\widetilde{\alpha^j}\}_{j\in[n]\setminus\mathcal{M}}$) at step 7 (resp., 9) for all honest parties, but uses uniform $\{\widetilde{\alpha^j}\}_{j\in[n]\setminus\mathcal{M}}$ in the emulated $\mathcal{F}_{\mathsf{Com}}$ at step 10 conditioned on
  $$\bigoplus_{j\in[n]\setminus\mathcal{M}}\widetilde{\alpha^j}=\bigoplus_{i\in\mathcal{M}}\widetilde{\alpha^i}\oplus\mathsf{H}'\Big((\boldsymbol{a}\wedge\boldsymbol{b}\oplus\boldsymbol{r})\cdot\bigoplus_{i\in[n]}\Delta_i\Big)\oplus\mathsf{H}'(\boldsymbol{m})\oplus\delta.$$

  In this and the previous hybrid, all honest parties abort without any output in the case of $\mathsf{lsb}(\bigoplus_{i\in[n]}\Delta_i)=0$. The circumstances where the view of the adversary could differ include the selective-failure queries against $\{\Delta_j\}_{j\in[n]\setminus\mathcal{M}}$ in the emulated $\mathcal{F}_{\mathsf{aShare}}$, $\{\widetilde{\boldsymbol{d}^j}\}_{j\in[n]\setminus\mathcal{M}}$ in the emulated $\mathcal{F}_{\mathsf{Com}}$ at step 7, and $\{\widetilde{\alpha^j}\}_{j\in[n]\setminus\mathcal{M}}$ in the emulated $\mathcal{F}_{\mathsf{Com}}$ at step 10.

  First, the sampling of $\{\Delta_j\}_{j\in[n]\setminus\mathcal{M}}$ in this hybrid can be consistent with the selective-failure queries as in the previous hybrid. Second, the distribution of $\bigoplus_{j\in[n]\setminus\mathcal{M}}\widetilde{\boldsymbol{d}^j}$ is identical in the two hybrids due to Equation 52 and the identical distribution of $\{\boldsymbol{a}^j\}_{j\in[n]\setminus\mathcal{M}}$. If there are at least two honest parties, the $\widetilde{\boldsymbol{d}^j}:=\mathsf{lsb}(\widetilde{\boldsymbol{s}^j})$ of per honest party P_j is uniform in the previous hybrid due to the uniform mask $\mathsf{K}_j[\boldsymbol{r}^i]$ for another honest party P_i in $\widetilde{\boldsymbol{s}^j}$, having the same distribution as in this hybrid. Third, the distribution of $\bigoplus_{j\in[n]\setminus\mathcal{M}}\widetilde{\alpha^j}$ is identical in the two hybrids due to Equation 53 and the identical distribution of $\{(\boldsymbol{a}^j,\Delta_j)\}_{j\in[n]\setminus\mathcal{M}}$ and $(\boldsymbol{a},\boldsymbol{b},\boldsymbol{r})$. If there are at least two honest parties, the $\widetilde{\alpha^j}$ of per honest party P_j is uniform in previous hybrid due to the uniform zero share u^i, having the same distribution as in this hybrid.

  Putting this argument of all honest parties' outputs and the adversarial view together, we can see that these two hybrids have perfectly indistinguishable joint distribution.

- **Hybrid**$_8$. This hybrid is identical to the previous one, except that the machine proceeds as follows if and only if $\mathsf{lsb}(\bigoplus_{i\in[n]}\Delta_i)=1$ (this event is identified by the machine at step 1):

  1. It does not compute $\{(\widetilde{\boldsymbol{s}^j},\widetilde{\boldsymbol{d}^j})\}_{j\in[n]\setminus\mathcal{M}}$ for all honest parties at step 5b but uses uniform $\{\widetilde{\boldsymbol{d}^j}\}_{j\in[n]\setminus\mathcal{M}}$ in the emulated $\mathcal{F}_{\mathsf{Com}}$ at step 7.
  2. It does not compute $\{\widetilde{\boldsymbol{t}^j}\}_{j\in[n]\setminus\mathcal{M}}$ (resp., $\{\widetilde{\alpha^j}\}_{j\in[n]\setminus\mathcal{M}}$) at step 7 (resp., 9) for all honest parties, but proceeds as follows:
     Case 1: If the event $\mathsf{H}'(\boldsymbol{m})\neq\delta$ is identified by the machine at step 9,

(a) It samples uniform $\{\boldsymbol{a}^j\}_{j\in[n]\setminus\mathcal{M}}$ for all honest parties, instead of using the values from the emulated $\mathcal{F}_{\mathsf{aShare}}$.

(b) It samples $\{\Delta_j\}_{j\in[n]\setminus\mathcal{M}} \xleftarrow{\$} \{\mathcal{R}_j\}_{j\in[n]\setminus\mathcal{M}}$ conditioned on that (i) $\{\Delta_j\}_{j\in[n]\setminus\mathcal{M}}$ comply with the selective-failure queries in the emulated $\mathcal{F}_{\mathsf{aShare}}$, and (ii) $\mathsf{H}'(\boldsymbol{m}) \oplus \delta$ from the sampled $\{(\boldsymbol{a}^j, \Delta_j)\}_{j\in[n]\setminus\mathcal{M}}$ is non-zero. Let $X \neq 0$ denote such a value.

(c) It uses uniform $\{\widetilde{\alpha^j}\}_{j\in[n]\setminus\mathcal{M}}$ in the emulated $\mathcal{F}_{\mathsf{Com}}$ at step 10 conditioned on

$$\bigoplus_{j\in[n]\setminus\mathcal{M}} \widetilde{\alpha^j} = \bigoplus_{i\in\mathcal{M}} \widetilde{\alpha^i} \oplus X.$$

Case 2: If the event $\mathsf{H}'(\boldsymbol{m}) = \delta$ is identified by the machine at step 9,

(a) It uses uniform $\{\widetilde{\alpha^j}\}_{j\in[n]\setminus\mathcal{M}}$ in the emulated $\mathcal{F}_{\mathsf{Com}}$ at step 10 conditioned on

$$\bigoplus_{j\in[n]\setminus\mathcal{M}} \widetilde{\alpha^j} = \bigoplus_{i\in\mathcal{M}} \widetilde{\alpha^i}.$$

(b) Instead of computing the authenticated shares at step 11, the machine directly uses $\{\Delta_j\}_{j\in[n]\setminus\mathcal{M}}$ at step 12 to sample $\{\boldsymbol{c}^j\}_{j\in[n]\setminus\mathcal{M}}$ such that

$$\bigoplus_{i\in[n]} \boldsymbol{c}^i = \bigoplus_{i\in[n]} \boldsymbol{a}^i \wedge \bigoplus_{i\in[n]} \boldsymbol{b}^i,$$

and generates $\langle\!\langle \boldsymbol{c} \rangle\!\rangle_j = (\boldsymbol{c}^j, \{\mathsf{M}_i[\boldsymbol{c}^j], \mathsf{K}_j[\boldsymbol{c}^i]\}_{i\neq j})$ for each $j \in [n] \setminus \mathcal{M}$ such that

$$\mathsf{K}_j[\boldsymbol{c}^i] := \begin{cases} \mathsf{M}_j[\boldsymbol{c}^i] \oplus \boldsymbol{c}^i \cdot \Delta_j, & P_i \text{ is corrupted and } (\boldsymbol{c}^i, \mathsf{M}_j[\boldsymbol{c}^i]) \text{ is honestly computed from } P_i\text{'s view} \\ \text{uniform}, & P_i \text{ is honest} \end{cases}$$

$$\mathsf{M}_i[\boldsymbol{c}^j] := \begin{cases} \mathsf{K}_i[\boldsymbol{c}^j] \oplus \boldsymbol{c}^j \cdot \Delta_i, & P_i \text{ is corrupted and } (\Delta_i, \mathsf{K}_i[\boldsymbol{c}^j]) \text{ is honestly computed from } P_i\text{'s view} \\ \mathsf{K}_i[\boldsymbol{c}^j] \oplus \boldsymbol{c}^j \cdot \Delta_i, & P_i \text{ is honest} \end{cases}$$

First, the event $\mathsf{lsb}(\bigoplus_{i\in[n]} \Delta_i) = 1$ implies that Equation 52 in the previous hybrid equals

$$\bigoplus_{j\in[n]\setminus\mathcal{M}} \widetilde{\boldsymbol{d}^j} = (\boldsymbol{a} \wedge \boldsymbol{b}) \oplus \boldsymbol{r} \oplus \bigoplus_{i\in\mathcal{M}} \widetilde{\boldsymbol{d}^i} \oplus \boldsymbol{q} \oplus \bigoplus_{j\in[n]\setminus\mathcal{M}} \boldsymbol{a}^j \wedge \mathsf{lsb}(\boldsymbol{e}^j),$$

which is uniform as $\boldsymbol{r}$ is uniform. In this hybrid, this sum is uniform as well. If there are at least two honest parties, the $\widetilde{\boldsymbol{d}^j} := \mathsf{lsb}(\widetilde{\boldsymbol{s}^j})$ of per honest party P_j is uniform in the previous hybrid due to the uniform mask $\mathsf{K}_j[\boldsymbol{r}^i]$ for another honest party P_i in $\widetilde{\boldsymbol{s}^j}$, having the same distribution as in this hybrid.

Then, consider **Case 1**. In this case, all honest parties abort without any output in two hybrids. Conditioned on the argued same distribution of $\{\widetilde{\boldsymbol{d}^j}\}_{j\in[n]\setminus\mathcal{M}}$, the circumstances where the view of the adversary may differ include the selective-failure queries against $\{\Delta_j\}_{j\in[n]\setminus\mathcal{M}}$ in the emulated $\mathcal{F}_{\mathsf{aShare}}$ and $\{\widetilde{\alpha^j}\}_{j\in[n]\setminus\mathcal{M}}$ in the emulated $\mathcal{F}_{\mathsf{Com}}$ at step 10. On the one hand, the sampling of $\{\Delta_j\}_{j\in[n]\setminus\mathcal{M}}$ in this hybrid yields the same distribution as in the previous hybrid, i.e., respecting the selective-failure queries under the event $\mathsf{H}'(\boldsymbol{m}) \neq \delta$. On the other hand, the distribution of $\bigoplus_{j\in[n]\setminus\mathcal{M}} \widetilde{\alpha^j}$ is identical in the two hybrids due to Equation 53 and

the same distribution of $\{(\boldsymbol{a}^j, \Delta_j)\}_{j\in[n]\setminus\mathcal{M}}$. If there are at least two honest parties, the $\widetilde{\alpha^j}$ of per honest party P_j is uniform in previous hybrid due to the uniform zero share u^i, having the same distribution as in this hybrid.

Finally, consider **Case 2**. In this case, all honest parties do not abort in both two hybrids. Conditioned on the argued same distribution of $\{\widetilde{\boldsymbol{d}^j}\}_{j\in[n]\setminus\mathcal{M}}$, the places where the view of the adversary may differ include $\{\widetilde{\alpha^j}\}_{j\in[n]\setminus\mathcal{M}}$ in the emulated $\mathcal{F}_{\mathsf{Com}}$ at step 10 and the joint distribution. The distributions of $\bigoplus_{j\in[n]\setminus\mathcal{M}} \widetilde{\alpha^j}$ and each $\widetilde{\alpha^j}$ are identical in the two hybrids based on a similar argument in the **Case 1**. For the joint distribution, let bad denote the event that all honest parties' outputs yield $\bigoplus_{i\in[n]} \boldsymbol{c}^i \neq \bigoplus_{i\in[n]} \boldsymbol{a}^i \wedge \bigoplus_{i\in[n]} \boldsymbol{b}^i$. As $\bigoplus_{i\in[n]} \boldsymbol{c}^i = \bigoplus_{i\in[n]} \boldsymbol{a}^i \wedge \bigoplus_{i\in[n]} \boldsymbol{b}^i \oplus \boldsymbol{q} \oplus \bigoplus_{j\in[n]\setminus\mathcal{M}} \boldsymbol{a}^j \wedge \mathsf{lsb}(\boldsymbol{e}^j)$ in the previous hybrid due to Equation 52, bad occurs in the previous hybrid if and only if $\boldsymbol{q} \oplus \bigoplus_{j\in[n]\setminus\mathcal{M}} \boldsymbol{a}^j \wedge \mathsf{lsb}(\boldsymbol{e}^j) \neq \boldsymbol{0}$. Using this equivalence and a similar argument in $\mathbf{Hybrid}_6$, we have in the previous hybrid

$$
\begin{aligned}
\Pr[\mathsf{bad}] &= \Pr\left[\mathsf{bad} \cap \left(\mathsf{H}'(\boldsymbol{m}) = \delta\right) \cap \left(\mathsf{lsb}(\textstyle\bigoplus_{i\in[n]} \Delta_i) = 1\right)\right] \\
&= \Pr\left[\mathsf{H}'(\boldsymbol{m}) = \delta \;\middle|\; \mathsf{bad} \cap \left(\mathsf{lsb}(\textstyle\bigoplus_{i\in[n]} \Delta_i) = 1\right)\right] \\
&\quad \cdot \Pr\left[\mathsf{bad} \cap \left(\mathsf{lsb}(\textstyle\bigoplus_{i\in[n]} \Delta_i) = 1\right)\right] \\
&\leq \Pr\left[\mathsf{H}'(\boldsymbol{m}) = \delta \;\middle|\; \mathsf{bad} \cap \left(\mathsf{lsb}(\textstyle\bigoplus_{i\in[n]} \Delta_i) = 1\right)\right] \\
&= \Pr\left[\mathsf{H}'(\boldsymbol{m}) = \delta \;\middle|\; \begin{array}{c}\left(\boldsymbol{q} \oplus \bigoplus_{j\in[n]\setminus\mathcal{M}} \boldsymbol{a}^j \wedge \mathsf{lsb}(\boldsymbol{e}^j) \neq \boldsymbol{0}\right) \\ \cap \left(\mathsf{lsb}(\bigoplus_{i\in[n]} \Delta_i) = 1\right)\end{array}\right] \\
&\leq \varepsilon + \frac{1}{2^{\lambda-1}}.
\end{aligned}
$$

That is, in the previous hybrid, all honest parties have erroneous outputs with probability at most $\varepsilon + \frac{1}{2^{\lambda-1}}$. In this hybrid, this probability is zero. Conditioned on the consistent outputs, the joint distribution in previous hybrid is identical to that in this hybrid given the uniform $\{\langle\!\langle \boldsymbol{r} \rangle\!\rangle_j\}_{j\in[n]\setminus\mathcal{M}}$ to compute the shares $\{\langle\!\langle \boldsymbol{c} \rangle\!\rangle_j\}_{j\in[n]\setminus\mathcal{M}}$. As a corollary, the statistical distance between the two hybrids in this case is at most $\varepsilon + \frac{1}{2^{\lambda-1}}$.

In summary, in either case, the two hybrids are statistically indistinguishable with advantage at most $\varepsilon + \frac{1}{2^{\lambda-1}}$.

- $\mathbf{Hybrid}_9$. This hybrid is identical to the previous one, except that the machine no longer plays the roles of all honest parties but interacts with functionality $\mathcal{F}_{\mathsf{LaAND}}$ to complete the execution. The machine in this hybrid differs from the previous one as follows:

  1. It uses the interfaces given in functionality $\mathcal{F}_{\mathsf{LaAND}}$ for the initialization, the selective-failure queries, and the global-key extraction with abort in the emulated $\mathcal{F}_{\mathsf{aShare}}$. Meanwhile, it still emulates $\mathcal{F}_{\mathsf{Zero}}$ for both honest and corrupted parties.
  2. It uses the share queries with selective failure in functionality $\mathcal{F}_{\mathsf{LaAND}}$ to possibly abort all honest parties.
     Specifically, it sends $(\mathsf{LeakA}, \mathtt{sid}, \mathtt{ssid}\|0, \{\boldsymbol{e}^j\}_{j\in[n]\setminus\mathcal{M}}, \boldsymbol{q})$ to $\mathcal{F}_{\mathsf{LaAND}}$ at step 7, obtains $(\mathsf{Hash}, \mathtt{sid}, \mathtt{ssid}, \mathsf{H}')$, sends $(\mathsf{Res}, \mathtt{sid}, \mathtt{ssid}, \mathsf{H}')$ in the emulated $\mathcal{F}_{\mathsf{Rand}}$, and finally sends $(\mathsf{LeakA}, \mathtt{sid}, \mathtt{ssid}\|1, \delta)$ to $\mathcal{F}_{\mathsf{LaAND}}$. As the machine can observe whether all honest parties abort in this interaction, it can simulate the messages sent by all honest parties as in the previous hybrid depending on this event.

3. It delays switching $\mathcal{S}_{\mathsf{H}}$ from **Phase 1** into **Phase 2** and changes the computation of the S-mTCCRL transcript $\mathcal{Q}_F$. Instead of knowing the outputs of all honest parties (in particular, $\{\boldsymbol{b}^j\}_{j\in[n]\setminus\mathcal{M}}$) to compute $\mathcal{Q}_F$, it extracts them via the global-key queries in $\mathcal{F}_{\mathsf{LaAND}}$. At step 5a, for $i \in \mathcal{M}, j \in [n]\setminus\mathcal{M}, k \in [\ell_2]$, the machine still samples $u_k^{j,i}$ uniformly at random. Then, for $(b_k^j, b_k) \in \{0,1\}^2$, the machine initializes 4 independent simulators $\mathcal{S}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}}}_{(b_k^j,b_k)}$, and defines:

$$\mathcal{Q}_F^{(b_k^j,b_k)} := \left\{\left(j, \underbrace{\mathsf{M}_j[a_k^i]}, T_{j,i,k}, b_k\right), \gamma_{j,i,k}\right\},$$
$$\gamma_{j,i,k} := u_k^{j,i} \oplus \mathsf{H}(\underbrace{\mathsf{M}_j[a_k^i]}, T_{j,i,k})$$
$$\oplus\, z_k^j \oplus \bigoplus\nolimits_{u\in\mathcal{M}} \left(\underbrace{\mathsf{M}_j[b_k^u]} \oplus \underbrace{\mathsf{K}_u[b_k^j]} \oplus b_k^j \cdot \underbrace{\Delta_u}\right).$$

The machine runs $\mathcal{S}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}}}_{(b_k^j,b_k)}.\mathit{PhaseSwitch}(\mathcal{Q}_F^{(b_k^j,b_k)})$.

Therefore, after step 5a, the machine initialized $4(n-|\mathcal{M}|)\,|\mathcal{M}|\,\ell_2$ **Prim**-simulators and changed their simulation to **Phase 2** using the possible combination of (b_k^j, b_k) values. For every subsequent **Prim** query $q_{\mathbf{Prim}}$, the machine first runs $\mathcal{S}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}}}_{(b_k^j,b_k)}.O_1(q_{\mathbf{Prim}})$ for every $i \in \mathcal{M}, j \in [n]\setminus\mathcal{M}, k \in [\ell_2]$. For every $\mathsf{GlobalK}_{\boldsymbol{\Delta}}$ oracle query (j, Δ'_j) issued by $\mathcal{S}_{(b_k^j,b_k)}$, the machine sends $(\mathsf{Guess}, \mathtt{sid}, j, \Delta'_j)$ to $\mathcal{F}_{\mathsf{LaAND}}$. If it receives back $(\mathsf{Success}, \mathtt{sid}, \{A_j\}_{j\in[n]\setminus\mathcal{M}})$, it uses $\{A_j\}_{j\in[n]\setminus\mathcal{M}}$ to compute $\mathcal{Q}_F$ as in the previous hybrid, switches $\mathcal{S}_{\mathsf{H}}$ from **Phase 1** into **Phase 2**, and answers $q_{\mathbf{Prim}}$ using the **Phase 2** $\mathcal{S}_{\mathsf{H}}$; otherwise, it continues to simulate $q_{\mathbf{Prim}}$ using the **Phase 1** $\mathcal{S}_{\mathsf{H}}$.

We note that the joint distribution in the two hybrids only differs by the simulation strategy of $\mathcal{S}_{\mathsf{H}}$. In Lemma 18, we show that under the lazy simulation strategy of $\mathcal{S}_{\mathsf{H}}$, the statistical distance between the two hybrids is bounded by $\varepsilon_{\mathsf{lazy}}$.

One can check that $\mathbf{Hybrid}_9$ is essentially the ideal-world execution where the machine plays as the simulator. Therefore, the theorem is proved. □

Lemma 17. *Given a hash function* $\mathsf{H}^{\mathbf{Prim}}$*, which invokes* **Prim** c_{H} *times, such that an* $(\frac{n^2\ell_2}{4}, q_P + c_{\mathsf{H}} \cdot \frac{n^2\ell_2}{4}, n, n-1, 1)$*-bounded distinguisher has advantage at most* ε_{H} *in the S-mTCCRL game (Definition 3) for at most* n *users and* $(\lambda-1)$*-bit min-entropy per key, and satisfies the* $\varepsilon_{\mathsf{lazy}}$*-lazy* **Prim** *simulation,* $\mathbf{Hybrid}_2$ *and* $\mathbf{Hybrid}_3$ *in the proof of Theorem 2 are computationally indistinguishable with advantage at most* ε_{H}*.*

Proof. For contradiction, assume that there exists an environment $\mathcal{Z}$ that can distinguish the two hybrids with advantage more than ε_{H}. We can construct a distinguisher $\mathcal{D}$ to break the S-mTCCRL security of H. Let $\mathcal{S}_{\mathsf{H}}$ denote the universal simulator in the S-mTCCRL experiment (in Figure 1) regarding H and $\overrightarrow{\mathcal{R}} := \{\mathcal{R}_j\}_{j\in[n]\setminus\mathcal{M}}$. Each component of $\overrightarrow{\mathcal{R}}$ has min-entropy $\lambda-1$. According to the experiment (see Figure 1), $\mathcal{D}$ has oracle access to $(\mathcal{O}_1, \mathcal{O}_2, \mathsf{mL}_{\boldsymbol{\Delta}})$, where $\boldsymbol{\Delta} \xleftarrow{\$} \overrightarrow{\mathcal{R}}$ and

$$(\mathcal{O}_1, \mathcal{O}_2) = \begin{cases} (\mathbf{Prim}, \mathsf{mTCK}^{\mathsf{H}}_{\boldsymbol{\Delta}}), & \text{if } \beta = 0 \\ (\mathcal{S}_{\mathsf{H}}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}}}.O_1, \mathbf{F}), & \text{if } \beta = 1 \end{cases}$$

It invokes $\mathcal{Z}$ in a black-box way and simulates the execution in either hybrid (depending on β) to interacts with $\mathcal{Z}$ as follows:

1. $\mathcal{D}$ emulates $\mathcal{F}_{\mathsf{aShare}}$, $\mathcal{F}_{\mathsf{Zero}}$, $\mathcal{F}_{\mathsf{Com}}$, and $\mathcal{F}_{\mathsf{Rand}}$ for the parties corrupted by $\mathcal{Z}$. In particular, it emulates the selective-failure queries in $\mathcal{F}_{\mathsf{aShare}}$ using its given $\mathsf{mL}_{\boldsymbol{\Delta}}$ and aborts the emulated $\mathcal{F}_{\mathsf{aShare}}$ if the oracle $\mathsf{mL}_{\boldsymbol{\Delta}}$ aborts. If $\mathcal{Z}$ asks for the global-key extraction with abort in the emulated $\mathcal{F}_{\mathsf{aShare}}$, $\mathcal{D}$ switches from **Phase 1** to **Phase 2** to obtain $\boldsymbol{\Delta} = \{\Delta_j\}_{j\in[n]\setminus\mathcal{M}}$, sends $(\mathsf{Bye}, \mathtt{sid}, \mathtt{ssid}, \boldsymbol{\Delta})$ to $\mathcal{Z}$, and aborts the emulated $\mathcal{F}_{\mathsf{aShare}}$.

2. $\mathcal{D}$ uses $\mathcal{O}_1$ to emulate **Prim** for the parties corrupted by $\mathcal{Z}$ during the whole reduction.

3. $\mathcal{D}$ behaves as the machine in $\mathbf{Hybrid}_2$ for all honest parties except that (i) in the emulation of $\mathcal{F}_{\mathsf{aShare}}$ for all honest parties, it only samples

$$\{(\boldsymbol{a}^j, \boldsymbol{b}^j)\}_{j\in[n]\setminus\mathcal{M}}, \{\mathsf{M}_u[\boldsymbol{b}^j]\}_{j,u\in[n]\setminus\mathcal{M},u\neq j},$$

   (ii) for each $i \in \mathcal{M}$, $j \in [n] \setminus \mathcal{M}$, and $k \in [\ell_2]$, it computes

$$\begin{aligned} u_k^{j,i} := \mathcal{O}_2\Big(j, \underbrace{\mathsf{M}_j[a_k^i]}, T_{j,i,k}, \underbrace{\textstyle\bigoplus_{i\in\mathcal{M}} b_k^i} \oplus \textstyle\bigoplus_{j\in[n]\setminus\mathcal{M}} b_k^j\Big) \oplus \mathsf{H}(\underbrace{\mathsf{M}_j[a_k^i]}, T_{j,i,k}) \\ \oplus \textstyle\bigoplus_{u\in[n]\setminus\mathcal{M},u\neq j} \Big(\mathsf{M}_j[b_k^u] \oplus \mathsf{M}_u[b_k^j]\Big) \oplus z_k^j \\ \oplus \textstyle\bigoplus_{u\in\mathcal{M}} \Big(\underbrace{\mathsf{M}_j[b_k^u]} \oplus \underbrace{\mathsf{K}_u[b_k^j]} \oplus b_k^j \cdot \underbrace{\Delta_u}\Big), \end{aligned}$$

   where the underlined terms are the randomness of corrupted parties extracted from the emulated $\mathcal{F}_{\mathsf{aShare}}$, and (iii) before sending the message of any honest party at step 5a, it switches itself from **Phase 1** to **Phase 2** to get $\boldsymbol{\Delta} = \{\Delta_j\}_{j\in[n]\setminus\mathcal{M}}$, uses $\boldsymbol{\Delta}$ to sample the undefined parts in the emulated $\mathcal{F}_{\mathsf{aShare}}$ for all honest parties, and honestly plays all honest parties as per protocol Π_{LaAND}.

4. $\mathcal{D}$ outputs whatever $\mathcal{Z}$ outputs.

In this construction, $\mathcal{D}$ makes at most $q_P + c_{\mathsf{H}} \cdot |\mathcal{M}| \cdot (n - |\mathcal{M}|) \cdot \ell_2 \leq q_P + c_{\mathsf{H}} \cdot \frac{n^2\ell_2}{4}$ queries to $\mathcal{O}_1$ since $\mathcal{Z}$ only makes at most q_P queries to **Prim**, at most $|\mathcal{M}| \cdot (n - |\mathcal{M}|) \cdot \ell_2 \leq \frac{n^2\ell_2}{4}$ queries to H, and at most $|\mathcal{M}| \leq n - 1$ queries to the $\mathsf{mL}_{\boldsymbol{\Delta}}$ oracle for at most $n - |\mathcal{M}| \leq n$ distinct honest parties. Moreover, at most 1 query to $\mathcal{O}_2$ is made for each distinct tweak. That is, $\mathcal{D}$ is $(\frac{n^2\ell_2}{4}, q_P + c_{\mathsf{H}} \cdot \frac{n^2\ell_2}{4}, n, n-1, 1)$-bounded.

When $\beta = 0$, one can check that

$$\begin{aligned} u_k^{j,i} &:= \mathsf{H}(\mathsf{M}_j[a_k^i], T_{j,i,k}) \oplus \mathsf{H}(\mathsf{M}_j[a_k^i] \oplus \Delta_j, T_{j,i,k}) \oplus b_k \cdot \Delta_j \\ &\quad \oplus \textstyle\bigoplus_{u\in[n]\setminus\mathcal{M},u\neq j} \Big(\mathsf{K}_j[b_k^u] \oplus b_k^u \cdot \Delta_j \oplus \mathsf{M}_u[b_k^j]\Big) \oplus z_k^j \\ &\quad \oplus \textstyle\bigoplus_{u\in\mathcal{M}} \Big(\mathsf{K}_j[b_k^u] \oplus b_k^u \cdot \Delta_j \oplus \mathsf{M}_u[b_k^j]\Big) \\ &= \mathsf{H}(\mathsf{K}_j[a_k^i], T_{j,i,k}) \oplus \mathsf{H}(\mathsf{K}_j[a_k^i] \oplus \Delta_j, T_{j,i,k}) \oplus b_k^j \cdot \Delta_j \\ &\quad \oplus \textstyle\bigoplus_{u\neq j} \Big(\mathsf{K}_j[b_k^u] \oplus \mathsf{M}_u[b_k^j]\Big) \oplus z_k^j, \end{aligned}$$

which is defined as in $\mathbf{Hybrid}_2$. Moreover, according to the identical distribution in the emulated functionalities and the fact that $\boldsymbol{\Delta}$ can never be used before step 5a in $\mathbf{Hybrid}_2$, the interaction between $\mathcal{D}$ and $\mathcal{Z}$ when $\beta = 0$ yields the same joint distribution as $\mathbf{Hybrid}_2$.

When $\beta = 1$, each response from the oracle $\mathcal{O}_2$ is independently uniform as $\mathcal{O}_2 = \mathbf{F}$ is a random function in this case and the tweaks are pairwise distinct. As a result, the interaction between $\mathcal{D}$ and $\mathcal{Z}$ when $\beta = 1$ yields the same joint distribution as $\mathbf{Hybrid}_3$.

Therefore, this $(\frac{n^2\ell_2}{4}, q_P + c_{\mathsf{H}} \cdot \frac{n^2\ell_2}{4}, n, n-1, 1)$-bounded distinguisher $\mathcal{D}$ can break the S-mTCCRL security of H with same advantage as that the q_P-bounded $\mathcal{Z}$ can distinguish the two hybrids, i.e., at least ε_{H}, leading to a contradiction and completing this proof. □

Lemma 18. $\mathbf{Hybrid}_8$ *and* $\mathbf{Hybrid}_9$ *in the proof of Theorem 2 are statistically indistinguishable with an advantage at most* $\varepsilon_{\mathsf{lazy}}$.

Proof. In $\mathbf{Hybrid}_8$, the phase switching input for $\mathcal{S}_{\mathsf{H}}$ is constructed as

$$\mathcal{Q}_F := \left\{ \left(j, \underbrace{\mathsf{M}_j[a_k^i]}, T_{j,i,k}, \underbrace{\bigoplus_{i \in \mathcal{M}} b_k^i} \oplus \bigoplus_{j \in [n] \setminus \mathcal{M}} b_k^j \right), \gamma_{j,i,k} \right\}_{\substack{i \in \mathcal{M}, \\ j \in [n] \setminus \mathcal{M}, \\ k \in [\ell_2]}},$$

$$\gamma_{j,i,k} := u_k^{j,i} \oplus \mathsf{H}(\underbrace{\mathsf{M}_j[a_k^i]}, T_{j,i,k}) \oplus z_k^j \oplus \bigoplus_{u \in \mathcal{M}} \left(\underbrace{\mathsf{M}_j[b_k^u]} \oplus \underbrace{\mathsf{K}_u[b_k^j]} \oplus b_k^j \cdot \underbrace{\Delta_u} \right).$$

Let $\mathcal{S} := \bigcup_{q \in \mathcal{Q}_F} \mathcal{S}_q$ be the relevant **Prim**-queries (Definition 13). The lazy simulation property of $\mathcal{S}_{\mathsf{H}}$ implies that delaying the $\mathcal{S}_{\mathsf{H}}.\mathit{PhaseSwitch}(\mathcal{Q}_F)$ computation until before the first query in $\mathcal{S}$ only causes $\varepsilon_{\mathsf{lazy}}$ statistical changes in the behavior of $\mathcal{S}_{\mathsf{H}}$. We then argue that this is indeed the case in $\mathbf{Hybrid}_9$. In the following discussion, for $k \in [\ell_2]$, denote $b_k = (\underbrace{\bigoplus_{i \in \mathcal{M}} b_k^i}) \oplus (\bigoplus_{j \in [n] \setminus \mathcal{M}} b_k^j)$.

In $\mathbf{Hybrid}_8$, for every index combination $i \in \mathcal{M}, j \in [n] \setminus \mathcal{M}, k \in [\ell_2]$, the corresponding $q \in \mathcal{Q}_F$ is defined as

$$q := \left(j, \underbrace{\mathsf{M}_j[a_k^i]}, T_{j,i,k}, b_k, \gamma_{j,i,k} \right),$$

$$\gamma_{j,i,k} := u_k^{j,i} \oplus \mathsf{H}(\underbrace{\mathsf{M}_j[a_k^i]}, T_{j,i,k}) \oplus z_k^j \oplus \bigoplus_{u \in \mathcal{M}} \left(\underbrace{\mathsf{M}_j[b_k^u]} \oplus \underbrace{\mathsf{K}_u[b_k^j]} \oplus b_k^j \cdot \underbrace{\Delta_u} \right).$$

In $\mathbf{Hybrid}_9$ for the corresponding i, j, k index set, $\mathcal{Q}_F^{(b_k^j, b_k)}$ follows the same distribution as $\{q\}$, and the machine has run the following after step 5a:

$$\mathcal{S}_{(b_k^j, b_k)}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}}}.\mathit{PhaseSwitch}(\mathcal{Q}_F^{(b_k^j, b_k)}).$$

Therefore, for any $q_{\mathbf{Prim}} \in \mathcal{S}_q$, by Definition 13, $\mathcal{S}_{(b_k^j, b_k)}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}}}(q_{\mathbf{Prim}})$ would generate a correct oracle query $\mathsf{GlobalK}_{\boldsymbol{\Delta}}(j, \Delta_j')$. Since in $\mathbf{Hybrid}_9$, the machine prepared a simulator for every $q \in \mathcal{Q}_F$ and possible $(b_k^j, b_k) \in \{0,1\}^2$, by the above argument, for every $q_{\mathbf{Prim}} \in \mathcal{S}_q$, the machine can get the correct Δ_j. From $(\mathsf{Guess}, \mathtt{sid}, j, \Delta_j)$ command output, the machine then constructs $\mathcal{Q}_F$ as in $\mathbf{Hybrid}_8$.

On the other hand, for any $q_{\mathbf{Prim}} \notin \mathcal{S}_q$, we argue that the simualtion strategy in $\mathbf{Hybrid}_9$ would not switch $\mathcal{S}_{\mathsf{H}}$ into **Phase 2**. Suppose by contradiction that some simulator $\mathcal{S}_{(b_k^j, b_k)}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}}}(q_{\mathbf{Prim}})$ instantiated in $\mathbf{Hybrid}_9$ for key extraction and associated with $q = ((\mathsf{id}, w, i, b), z)$ submitted a correct $\mathsf{GlobalK}_{\boldsymbol{\Delta}}(\mathsf{id}, \Delta_{\mathsf{id}})$ query. By Definition 13, this implies that $q_{\mathbf{Prim}} \in \mathcal{S}_q$, which implies that $q_{\mathbf{Prim}} \in \mathcal{S}$. □

Functionality $\mathcal{F}_{\mathsf{COT}}$

This functionality runs with a sender P_1 and a receiver P_2. Let $\mathcal{L}$ be the set of allowed selective-failure queries.

Initialize: Upon receiving $(\mathsf{Init}, \mathtt{sid})$ from P_1 and P_2, receive $(\mathsf{GK}, \mathtt{sid}, \Delta)$ from the adversary if P_1 is corrupted; otherwise, sample $\Delta \xleftarrow{\$} \mathbb{F}_{2^\lambda}$ and send $(\mathsf{GK}, \mathtt{sid}, \Delta)$ to P_1. Store $(\mathtt{sid}, \Delta)$ and ignore all subsequent $(\mathsf{Init}, \mathtt{sid})$ commands with the same $\mathtt{sid}$.

Extend: Given stored $(\mathtt{sid}, \Delta)$, this functionality outputs random COT tuples: Upon receiving $(\mathsf{Extend}, \mathtt{sid}, \mathtt{ssid}, n)$ from P_1 and P_2, where $n \in \mathbb{N}^+$,

1. If P_1 is corrupted, receive $\boldsymbol{v} \in \mathbb{F}_{2^\lambda}^n$ from the adversary. Otherwise, $\boldsymbol{v} \leftarrow \mathbb{F}_{2^\lambda}^n$.
2. If P_2 is corrupted, receive $(\boldsymbol{u}, \boldsymbol{w}) \in \mathbb{F}_2^n \times \mathbb{F}_{2^\lambda}^n$ from the adversary and redefine $\boldsymbol{v} := \boldsymbol{w} \oplus \boldsymbol{u} \cdot \Delta \in \mathbb{F}_{2^\lambda}^n$; otherwise, $\boldsymbol{u} \xleftarrow{\$} \mathbb{F}_2^n$ and compute $\boldsymbol{w} := \boldsymbol{v} \oplus \boldsymbol{u} \cdot \Delta \in \mathbb{F}_{2^\lambda}^n$.
3. Output $(\mathsf{Out}, \mathtt{sid}, \mathtt{ssid}, \Delta, \boldsymbol{v})$ to P_1 and $(\mathsf{Out}, \mathtt{sid}, \mathtt{ssid}, \boldsymbol{w})$ to P_2.
4. Ignore all subsequent $(\mathsf{Extend}, \mathtt{sid}, \mathtt{ssid}, \ldots)$ calls with the same $(\mathtt{sid}, \mathtt{ssid})$.

Selective-failure query: If P_2 is corrupted, the adversary can execute the following selective failure command after initialization.

1. Receive $(\mathsf{Leak}, \mathtt{sid}, L)$ from the adversary, where $L \in \mathcal{L}$.
2. If $\Delta \notin L$, send $(\mathsf{Fail}, \mathtt{sid})$ to the adversary and abort; otherwise, do nothing.

Figure 14: The functionality of correlated oblivious transfer.

G UC-Secure Ferret Single-Point COT from S-mTCCRL

In Sect. 2.1, we have recalled that, even in the standalone model, the Ferret protocol [YWL+20] cannot simply base its security on some variant of correlation robustness since the global key Δ is revealed to the distinguisher as a part of the honest sender's output. However, we find that this issue can be patched using our S-mTCCRL notion, which includes an implicit programming to maintain the consistency regarding Δ and make the proof go through.

In Figure 15 and Figs. 16, 17, we respectively recall the functionality $\mathcal{F}_{\mathsf{SPCOT}}$ and the protocol Π_{SPCOT} of single-point COT, where the subroutine functionality $\mathcal{F}_{\mathsf{COT}}$ used by Π_{SPCOT} is given in Figure 14. Notice that we prove the UC security in Theorem 4, which only focuses on a corrupted receiver. The UC security against a corrupted sender can follow from the original proof of [YWL+20].

Theorem 4. *Given a hash function $\mathsf{H}^{\mathbf{Prim}}$, which invokes* **Prim** *c_{H} times, such that an $(1, q_P + c_{\mathsf{H}}, 1, 1, 1)$-bounded distinguisher has advantage at most ε_{H} in the S-mTCCRL game (Definition 3) for one user and λ-bit min-entropy of its key, and an ε_G-secure PRG G, the protocol Π_{SPCOT} (Figs. 16 and 17) UC-realizes functionality $\mathcal{F}_{\mathsf{SPCOT}}$ (Figure 15) in the $\mathcal{F}_{\mathsf{COT}}$-hybrid model with an advantage at most $\varepsilon_{\mathsf{H}} + h_{\mathsf{max}} \cdot \varepsilon_G$ against any corrupted receiver making at most q_P* **Prim***-queries, when the command $(\mathsf{Extend}, \mathtt{sid}, \ldots)$ is only invoked once per $\mathtt{sid}$ and $n \leq 2^{h_{\mathsf{max}}}$, where n is the extension length.*

Proof. We prove the UC security via the following hybrid argument.

Functionality $\mathcal{F}_{\mathsf{SPCOT}}$

This functionality runs with a sender P_1 and a receiver P_2. Let $\mathcal{L}$ be the set of allowed selective-failure queries.

Initialize: Upon receiving $(\mathsf{Init}, \mathtt{sid})$ from P_1 and P_2, receive $(\mathsf{GK}, \mathtt{sid}, \Delta)$ from the adversary if P_1 is corrupted; otherwise, sample $\Delta \xleftarrow{\$} \mathbb{F}_{2^\lambda}$. Store $(\mathtt{sid}, \Delta)$ and ignore all subsequent $(\mathsf{Init}, \mathtt{sid})$ commands with the same $\mathtt{sid}$.

Extend: Given stored $(\mathtt{sid}, \Delta)$, this functionality outputs random single-point COTs: Upon receiving $(\mathsf{Extend}, \mathtt{sid}, \mathtt{ssid}, n)$ from P_1 and $(\mathsf{Extend}, \mathtt{sid}, \mathtt{ssid}, n, \alpha)$ from P_2, where $n \in \mathbb{N}^+$ is a power of two and $\alpha \in [0, n)$,

1. If P_1 is corrupted, receive $\boldsymbol{v} \in \mathbb{F}_{2^\lambda}^n$ from the adversary. Otherwise, $\boldsymbol{v} \leftarrow \mathbb{F}_{2^\lambda}^n$.
2. Define $\boldsymbol{u} \in \mathbb{F}_2^n$ as a weight-1 vector with exactly one non-zero entry at α.
3. If P_2 is corrupted, receive $\boldsymbol{w} \in \mathbb{F}_{2^\lambda}^n$ from the adversary and recompute $\boldsymbol{v} := \boldsymbol{w} \oplus \boldsymbol{u} \cdot \Delta \in \mathbb{F}_{2^\lambda}^n$. Otherwise, compute $\boldsymbol{w} := \boldsymbol{v} \oplus \boldsymbol{u} \cdot \Delta \in \mathbb{F}_{2^\lambda}^n$.
4. **Selective failure against α:** If P_1 is corrupted, proceed as follows:
   (a) Receive $(\mathsf{Leak}, \mathtt{sid}, \mathtt{ssid}, I)$ from the adversary, where $I \subseteq [0, n)$.
   (b) If $\alpha \notin I$, send $(\mathsf{Fail}, \mathtt{sid}, \mathtt{ssid})$ to the adversary and abort. Otherwise, do nothing.
5. Output $(\mathsf{Out}, \mathtt{sid}, \mathtt{ssid}, \Delta, \boldsymbol{v})$ to P_1 and $(\mathsf{Out}, \mathtt{sid}, \mathtt{ssid}, \boldsymbol{w})$ to P_2.
6. Ignore all subsequent $(\mathsf{Extend}, \mathtt{sid}, \mathtt{ssid}, \ldots)$ calls with the same $(\mathtt{sid}, \mathtt{ssid})$.

Global-key queries: Upon receiving $(\mathsf{Guess}, \mathtt{sid}, \Delta')$ from the adversary, where $\Delta' \in \mathbb{F}_{2^\lambda}$, if P_1 is honest and $\Delta' = \Delta$, send $(\mathsf{Success}, \mathtt{sid})$ to the adversary; otherwise, do nothing.

Selective-failure query: If P_2 is corrupted, the adversary can execute the following selective failure command after initialization.

1. Receive $(\mathsf{Leak}, \mathtt{sid}, L)$ from the adversary, where $L \in \mathcal{L}$.
2. If $\Delta \notin L$, send $(\mathsf{Fail}, \mathtt{sid})$ to the adversary and abort; otherwise, do nothing.

Figure 15: The functionality of single-point correlated OT.

- **Hybrid**$_0$. This is the real-world execution of protocol Π_{SPCOT}.
- **Hybrid**$_1$. This hybrid is identical to the previous one, except that there is a machine playing the role of the honest sender P_1 and emulating the subroutine functionality $\mathcal{F}_{\mathsf{COT}}$ for the corrupted receiver. This modification is conceptual, so the joint distribution in this hybrid is perfectly indistinguishable from the previous one.

  We note that, since this hybrid, the machine can extract $(\boldsymbol{r}, \boldsymbol{t})$ and $(\boldsymbol{x}^*, \boldsymbol{z}^*)$ from the emulated $\mathcal{F}_{\mathsf{COT}}$ for the corrupted receiver. The machine can also extract α from Step 4.
- **Hybrid**$_{1.5}$. This hybrid is identical to the previous one, except that the machine changes the way to compute honest P_1's output as follows. After computing P_2's view, the machine computes $\boldsymbol{w}$ using the simulated $\{(M_0^i, M_1^i)\}_{i \in [h]}$, c, and then computes $\boldsymbol{v} = \boldsymbol{w} \oplus \boldsymbol{u} \cdot \Delta$, where

Protocol Π_{SPCOT}

Parameters: A length-doubling PRG $G : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$. A S-mTCCRL hash function $\mathsf{H} : \{0,1\}^{2\lambda} \to \{0,1\}^\lambda$. A random oracle $H' : \{0,1\}^* \to \{0,1\}^{2\lambda}$.

Initialize: The two parties execute this procedure only once.

1. P_1 and P_2 sends $(\mathsf{Init}, \mathtt{sid})$ to $\mathcal{F}_{\mathsf{COT}}$, which outputs $(\mathsf{GK}, \mathtt{sid}, \Delta)$ to P_1.

Extend: This procedure can be executed multiple times. Let $n = 2^h$ for some $h \in \mathbb{N}^+$ and $\alpha \in [0, n)$. Let $\alpha_1, ..., \alpha_h$ be the binary decomposition of α.

2. P_1 and P_2 send $(\mathsf{Extend}, \mathtt{sid}, \mathtt{ssid} \| 0, h)$ to $\mathcal{F}_{\mathsf{COT}}$, which outputs $(\mathsf{Out}, \mathtt{sid}, \mathtt{ssid} \| 0, \boldsymbol{q})$ to P_1 and $(\mathsf{Out}, \mathtt{sid}, \mathtt{ssid} \| 0, \boldsymbol{r}, \boldsymbol{t})$ to P_2, where $\boldsymbol{q} \in \mathbb{F}_{2^\lambda}^h$ and $(\boldsymbol{r}, \boldsymbol{t}) \in \mathbb{F}_2^h \times \mathbb{F}_{2^\lambda}^h$ such that $\boldsymbol{t} = \boldsymbol{q} \oplus \boldsymbol{r} \cdot \Delta$.

3. P_1 picks a random $s_0^0 \in \{0,1\}^\lambda$. For each $i \in \{1, \ldots, h\}, j \in [0, 2^{i-1})$, P_1 computes $(s_{2j}^i, s_{2j+1}^i) := G(s_j^{i-1})$. For each $i \in \{1, \ldots, h\}$, P_1 then computes $K_0^i := \bigoplus_{j \in [2^{i-1}]} s_{2j}^i$ and $K_1^i := \bigoplus_{j \in [2^{i-1}]} s_{2j+1}^i$.

4. For $i \in \{1, \ldots, h\}$, P_2 sends a bit $b_i := r_i \oplus \alpha_i \oplus 1$ to P_1, and then P_1 sends $M_0^i := K_0^i \oplus \mathsf{H}(q_i \oplus b_i \Delta, i \| \mathtt{sid} \| \mathtt{ssid})$, $M_1^i := K_1^i \oplus \mathsf{H}(q_i \oplus \overline{b_i} \Delta, i \| \mathtt{sid} \| \mathtt{ssid})$ to P_2. In parallel, P_1 defines a vector $\boldsymbol{v} := (s_0^h, \ldots, s_{n-1}^h) \in \mathbb{F}_{2^\lambda}^n$ and sends $c := \Delta \oplus \bigoplus_{i \in [0,n)} v_i \in \mathbb{F}_{2^\lambda}$ to P_2.

5. For $i \in \{1, \ldots, h\}$, P_2 proceeds as follows for a prefix $\alpha_i^* := \alpha_1 \cdots \alpha_{i-1} \overline{\alpha_i}$:

   (a) Compute $K_{\overline{\alpha_i}}^i := M_{\overline{\alpha_i}}^i \oplus \mathsf{H}(t_i, i \| \mathtt{sid} \| \mathtt{ssid})$. If $i = 1$, define $s_{\overline{\alpha_1}}^1 := K_{\overline{\alpha_1}}^1$.

   (b) If $i \geq 2$, for $j \in [0, 2^{i-1}), j \neq \alpha_1 \cdots \alpha_{i-1}$, compute $(s_{2j}^i, s_{2j+1}^i) := G(s_j^{i-1})$.

   (c) Compute $s_{\alpha_i^*}^i := K_{\overline{\alpha_i}}^i \oplus (\bigoplus_{j \in [0,2^{i-1}), j \neq \alpha_1 \cdots \alpha_{i-1}} s_{2j+\overline{\alpha_i}}^i)$.

   (d) Compute $w_i := s_i^h$ for $i \in [0, n) \setminus \{\alpha\}$, and $w_\alpha := c \oplus \bigoplus_{i \in [0,n) \setminus \{\alpha\}} w_i$.

Figure 16: The protocol of single-point correlated OT from Ferret [YWL+20] (Part I).

$\boldsymbol{u}$ is the weight-1 vector with exactly one non-zero entry at α. Since the machine simply re-orders computation, this hybrid is identically distributed to the previous one.

- **Hybrid**$_2$. This hybrid is identical to the previous one, except that the machine changes the expansion of the GGM trees owned by the sender as follows: In the $(\mathsf{Extend}, \mathtt{sid}, \mathtt{ssid}, n, \alpha)$ execution, define $\alpha = \alpha_1 ... \alpha_h$ and $\tilde{\alpha}_i = \alpha_1 ... \alpha_i$ for each $i \in [h]$ ($\tilde{\alpha}_0 = 0$), and replace $(s_{2\tilde{\alpha}_i}^{i+1}, s_{2\tilde{\alpha}_i+1}^{i+1}) := G(s_{\tilde{\alpha}_i}^i)$ by sampling $(s_{2\tilde{\alpha}_i}^{i+1}, s_{2\tilde{\alpha}_i+1}^{i+1}) \xleftarrow{\$} \{0,1\}^{2\lambda}$ for each $i \in [h]$.

  Note that $n \leq 2^{h_{\mathsf{max}}}$. Using a standard hybrid argument (see Goldreich et al. [GGM84]), this hybrid is computationally indistinguishable from the previous one with advantage $h_{\mathsf{max}} \cdot \varepsilon_G$.

- **Hybrid**$_3$. This hybrid is identical to the previous one, except that the machine replaces, for each $i \in \{1, \ldots, h\}$ in the $(\mathsf{Extend}, \mathtt{sid}, \mathtt{ssid}, n, \alpha)$ execution, the two messages (M_0^i, M_1^i) with uniform values. The message $c \in \mathbb{F}_{2^\lambda}$ in Step 4 is computed as:

$$c = M_0^h \oplus M_1^h \oplus \mathsf{H}(t_h, h \| \mathtt{sid} \| \mathtt{ssid}) \oplus \mathsf{H}(t_h \oplus \Delta, h \| \mathtt{sid} \| \mathtt{ssid}) \oplus \Delta \,.$$

Protocol Π_{SPCOT}

Then, the two parties perform a consistency check for malicious security.

6. P_1 and P_2 send $(\mathsf{Extend}, \mathtt{sid}, \mathtt{ssid}\|1, \lambda)$ to $\mathcal{F}_{\mathsf{COT}}$, which outputs $(\mathsf{Out}, \mathtt{sid}, \mathtt{ssid}\|1, \boldsymbol{y}^*)$ to P_1 and $(\mathsf{Out}, \mathtt{sid}, \mathtt{ssid}\|1, \boldsymbol{x}^*, \boldsymbol{z}^*)$ to P_2, where $\boldsymbol{y}^* \in \mathbb{F}_{2^\lambda}^\lambda$ and $(\boldsymbol{x}^*, \boldsymbol{z}^*) \in \mathbb{F}_2^\lambda \times \mathbb{F}_{2^\lambda}^\lambda$ such that $\boldsymbol{z}^* = \boldsymbol{y}^* \oplus \boldsymbol{x}^* \cdot \Delta$.
7. P_2 samples $\{\chi_i\}_{i\in[0,n)} \xleftarrow{\$} \mathbb{F}_{2^\lambda}^n$ and interprets χ_α as $\boldsymbol{x} \in \mathbb{F}_2^\lambda$ such that $\chi_\alpha = \bigoplus_{i\in[0,\lambda)} x_i \cdot X^i \in \mathbb{F}_{2^\lambda}$. P_2 sends $\{\chi_i\}_{i\in[0,n)}$ and $\boldsymbol{x}' := \boldsymbol{x} \oplus \boldsymbol{x}^* \in \mathbb{F}_2^\lambda$ to P_1.
8. P_1 computes $\boldsymbol{y} := \boldsymbol{y}^* \oplus \boldsymbol{x}' \cdot \Delta \in \mathbb{F}_{2^\lambda}^\lambda$, $Y := \bigoplus_{i\in[0,\lambda)} y_i \cdot X^i \in \mathbb{F}_{2^\lambda}$, and $V := \bigoplus_{i\in[0,n)} \chi_i \cdot v_i \oplus Y \in \mathbb{F}_{2^\lambda}$. P_1 sends $H'(V\|\mathtt{sid}\|\mathtt{ssid})$ to P_2.
9. P_2 computes $Z := \bigoplus_{i\in[0,\lambda)} z_i^* \cdot X^i \in \mathbb{F}_{2^\lambda}$ and $W := \bigoplus_{i\in[0,n)} \chi_i \cdot w_i \oplus Z \in \mathbb{F}_{2^\lambda}$. P_2 checks if $H'(V\|\mathtt{sid}\|\mathtt{ssid}) = H'(W\|\mathtt{sid}\|\mathtt{ssid})$. If this check fails, P_2 aborts; otherwise, P_2 does nothing.
10. P_1 outputs $(\mathsf{Out}, \mathtt{sid}, \mathtt{ssid}, \Delta, \boldsymbol{v})$ and P_2 outputs $(\mathsf{Out}, \mathtt{sid}, \mathtt{ssid}, \boldsymbol{w})$.

Figure 17: The protocol of single-point correlated OT from Ferret [YWL+20] (Part II).

In the previous hybrid, for $i \in \{1, \ldots, h\}$, $j \in \{0, 1\}$, the values K_j^i, M_j^i are distributed subject to $K_j^i \oplus M_j^i = \mathsf{H}(q_i \oplus (b_i \oplus j)\Delta, i\|\mathtt{sid}\|\mathtt{ssid})$. Moreover, in the previous hybrid, the (K_0^i, K_1^i) values are already uniformly random. Therefore, by sampling (M_0^i, M_1^i) uniformly random and re-compute c accordingly, this hybrid is identically distributed as the previous one.

- **Hybrid**$_4$. We change the simulation of the message c in Step 4 of the command

$$(\mathsf{Extend}, \mathtt{sid}, \mathtt{ssid}, n, \alpha)\,,$$

and the ideal primitive **Prim** in this hybrid. At the beginning of the simulation, the machine invokes the S-mTCCRL simulator $\mathcal{S}_{\mathsf{H}}$ to simulate **Prim**. At Step 4, the machine samples $c \leftarrow \mathbb{F}_{2^\lambda}$. Then the machine defines

$$\mathcal{Q}_F = \left\{\left(1, t_h, h\|\mathtt{sid}\|\mathtt{ssid}, 1\right), c \oplus M_0^h \oplus M_1^h \oplus \mathsf{H}(t_h, h\|\mathtt{sid}\|\mathtt{ssid})\right\}\,,$$

and switches $\mathcal{S}_{\mathsf{H}}$ from **Phase 1** to **Phase 2** using $\mathcal{Q}_F$.

Then, the machine continues to simulate **Prim** with the simulator $\mathcal{S}_{\mathsf{H}}$ (now switched to Phase 2).

From Lemma 19, these two hybrids are indistinguishable with advantage at most ε_{H}.

- **Hybrid**$_5$. In this hybrid, we use the lazy sampling approach to simulate the random oracle H'. The machine initializes an empty list $\mathcal{Q}$ at the beginning of the simulation. We identify three types of random oracle queries.
  - **Regular queries before extension**. In $(\mathsf{Extend}, \mathtt{sid}, \mathtt{ssid}, n, \alpha)$, before Step 8, for each random oracle query x, the machine samples $h \leftarrow \{0,1\}^\lambda$ and appends (x, h) to $\mathcal{Q}$.
  - **Implicit queries during extension**. In $(\mathsf{Extend}, \mathtt{sid}, \mathtt{ssid}, n, \alpha)$, to simulate the Step 8, the simulator proceeds as follows.

1. Compute α's bit decomposition $\{\alpha_i := b_i \oplus r_i \oplus 1\}_{i\in\{1,\dots,h\}}$ at Step 4.
2. Follow the honest receiver's procedure to compute $\boldsymbol{w}$.
3. Receive $\{\chi_i\}$ and $\boldsymbol{x}'$ from the adversary at Step 7. Let χ_α be the α-th value in $\{\chi_i\}$. Compute $\boldsymbol{x} := \boldsymbol{x}^* \oplus \boldsymbol{x}' \in \mathbb{F}_2^\lambda$ and $\chi'_\alpha := \bigoplus_{i\in[0,\lambda)} x_i \cdot X^i \in \mathbb{F}_{2^\lambda}$.
4. Compute $W = \bigoplus_{i\in[n]} \chi_i \cdot w_i \oplus \bigoplus_{i\in[\lambda]} z_i^* \cdot X^i$.

We identify the following two sub-cases.

* If $\chi'_\alpha = \chi_\alpha$, then send $H'(W\|\mathtt{sid}\|\mathtt{ssid})$ to the corrupted receiver at Step 8.
* If $\chi'_\alpha \neq \chi_\alpha$, then for each entry $(x\|\mathtt{sid}\|\mathtt{ssid}, h') \in \mathcal{Q}$, compute $\Delta' = (W \oplus x) \cdot (\chi'_\alpha \oplus \chi_\alpha)^{-1}$ and the machine checks if $\Delta' = \Delta$. If so, then the machine sends h' to the corrupted receiver. Otherwise, the machine samples a uniformly random $h \xleftarrow{\$} \{0,1\}^{2\lambda}$ and records $(W, \chi'_\alpha \oplus \chi_\alpha, \mathtt{sid}\|\mathtt{ssid}, h)$, and sends h to the corrupted receiver.

– **Regular queries after extension**. For every query x after Step 4, the machine answers as follows.

* If there exists a record $(x, h) \in \mathcal{Q}$, then answers with h.
* Let $(W, \chi'_\alpha \oplus \chi_\alpha, \mathtt{sid}\|\mathtt{ssid}, h)$ be the implicit query record. If x has the form $x = x'\|\mathtt{sid}\|\mathtt{ssid}$, then the machine computes $\Delta' = (W \oplus x') \cdot (\chi'_\alpha \oplus \chi_\alpha)^{-1}$ and checks if $\Delta' = \Delta$. If so, then the machine answers with h.
* Otherwise, the machine samples $h \leftarrow \{0,1\}^{2\lambda}$ and answers with h.

By Lemma 20, the environment's advantage in this hybrid is identical to the advantage in the previous one.

- **Hybrid**$_6$. This hybrid is identical to the previous one, except that the machine no longer plays the role of the honest sender but interacts with functionality $\mathcal{F}_{\mathsf{SPCOT}}$ to complete the execution. More specifically, it uses the interface of global-key queries in $\mathcal{F}_{\mathsf{SPCOT}}$ to check if a computed $\Delta' = \Delta$. Moreover, it uses the $(\mathsf{Leak}, \mathtt{sid}, L)$ interface of $\mathcal{F}_{\mathsf{SPCOT}}$ to answer the selective failure query of the $\mathcal{F}_{\mathsf{COT}}$ functionality.

  This hybrid is perfectly indistinguishable from the previous one.

It is clear that **Hybrid**$_6$ is the ideal-world execution, completing the proof.

□

Lemma 19. *Given a hash function* $\mathsf{H}^{\mathbf{Prim}}$*, which invokes* **Prim** c_{H} *times, such that an* $(1, q_P + c_{\mathsf{H}}, 1, 1, 1)$*-bounded distinguisher has advantage at most* ε_{H} *in the S-mTCCRL game (Definition 3) for one user and* λ*-bit min-entropy of its key,* **Hybrid**$_3$ *and* **Hybrid**$_4$ *in the proof of Theorem 4 are computationally indistinguishable with advantage of at most* ε_{H}*.*

Proof. We argue that an environment $\mathcal{Z}$ distinguishing **Hybrid**$_3$ and **Hybrid**$_4$ can be turned into a $(1, q_P + c_{\mathsf{H}}, 1, 1, 1)$-bounded distinguisher $\mathcal{D}$ against the S-mTCCRL property of $\mathsf{H}^{\mathbf{Prim}}$. We construct the distinguisher $\mathcal{D}$ as follows. According to the experiment against $\overrightarrow{\mathcal{R}}$ of length 1 with min-entropy λ, $\mathcal{D}$ has oracle access to $(\mathcal{O}_1, \mathcal{O}_2, \mathsf{mL}_{\boldsymbol{\Delta}})$, where $\boldsymbol{\Delta} \xleftarrow{\$} \overrightarrow{\mathcal{R}}$ and

$$(\mathcal{O}_1, \mathcal{O}_2) = \begin{cases} (\mathbf{Prim}, \mathsf{mTCK}^{\mathsf{H}}_{\boldsymbol{\Delta}}), & \text{if } \beta = 0 \\ (\mathcal{S}^{\mathsf{GlobalK}_{\boldsymbol{\Delta}}}_{\mathsf{H}}.O_1, \mathbf{F}), & \text{if } \beta = 1 \end{cases}$$

$\mathcal{D}$ invokes $\mathcal{Z}$ in a black-box way and simulates the execution in either hybrid (depending on β) to interact with $\mathcal{Z}$ as follows:

1. $\mathcal{D}$ emulates the $\mathcal{F}_{\mathsf{COT}}$ for the corrupted receiver. In particular, it emulates the selective-failure queries in $\mathcal{F}_{\mathsf{COT}}$ using it's given $\mathsf{mL}_{\boldsymbol{\Delta}}$ oracle and aborts the emulated $\mathcal{F}_{\mathsf{COT}}$ if the oracle $\mathsf{mL}_{\boldsymbol{\Delta}}$ aborts.

2. $\mathcal{D}$ uses $\mathcal{O}_1$ to emulate the ideal primitive **Prim** during the whole reduction.

3. $\mathcal{D}$ behaves as the machine in $\mathbf{Hybrid}_3$ for the honest sender except that in the command $(\mathsf{extend}, \mathtt{sid}, \mathtt{ssid}, n)$, it computes

$$c = M_0^h \oplus M_1^h \oplus \mathsf{H}(t_h, h \| \mathtt{sid} \| \mathtt{ssid}) \oplus \mathcal{O}_2(1, t_h, h \| \mathtt{sid} \| \mathtt{ssid}, 1)\,.$$

   Then, $\mathcal{D}$ switches itself from **Phase 1** to **Phase 2** and uses the Δ value to continue the simulation.

4. $\mathcal{D}$ outputs whatever $\mathcal{Z}$ outputs.

Consider the resources that $\mathcal{D}$ uses in this construction.

- $\mathcal{D}$ makes 1 query to $\mathcal{O}_2$;
- $\mathcal{D}$ makes at most $q_P + c_{\mathsf{H}}$ queries to $\mathcal{O}_1$ since $\mathcal{Z}$ makes at most q_P queries to **Prim** and at most c_{H} **Prim** queries to compute c;
- The global key vector $\boldsymbol{\Delta}$ consists of only 1 key of the honest sender;
- $\mathcal{D}$ makes at most 1 query to $\mathsf{mL}_{\boldsymbol{\Delta}}$
- For each distinct tweak, $\mathcal{D}$ only makes 1 query to $\mathcal{O}_2$.

Therefore, $\mathcal{D}$ is $(1, q_P + c_{\mathsf{H}}, 1, 1, 1)$-bounded.

If $\beta = 0$, one can check that

$$\begin{aligned} c &= M_0^h \oplus M_1^h \oplus \mathsf{H}(t_h, h \| \mathtt{sid} \| \mathtt{ssid}) \oplus \mathsf{H}(t_h, h \| \mathtt{sid} \| \mathtt{ssid}) \oplus \Delta \\ &= \Delta \oplus K_0^h \oplus K_1^h \\ &= \Delta \oplus \bigoplus_{i \in [0,n)} v_i\,, \end{aligned}$$

which is defined as in $\mathbf{Hybrid}_3$.

When $\beta = 1$, we have $\mathcal{O}_2 = \mathbf{F}$ which implies that c is uniformly random, as in $\mathbf{Hybrid}_4$. Therefore, we conclude that $\mathcal{Z}$ distinguishing $\mathbf{Hybrid}_3$ and $\mathbf{Hybrid}_4$ can be transformed into a $(1, q_P + c_{\mathsf{H}}, 1, 1, 1)$-bounded distinguisher $\mathcal{D}$. In other words, $\mathbf{Hybrid}_3$ and $\mathbf{Hybrid}_4$ are ε_{H}-indistinguishable against an adversary making at most q_P **Prim** queries.

□

Lemma 20. *In the ROM,* $\mathbf{Hybrid}_4$ *and* $\mathbf{Hybrid}_5$ *in the proof of Theorem 4 are identically distributed.*

Proof. We argue that the lazy sampling approach in $\mathbf{Hybrid}_5$ is perfect to prove this lemma. The **regular queries before extension** are handled identically in $\mathbf{Hybrid}_4$ and $\mathbf{Hybrid}_5$. Consider

the machine's action following the honest sender's program in $\mathbf{Hybrid}_4$. In Step 8, we have

$$\begin{aligned}
V &:= \bigoplus_{i\in[0,n)} \chi_i \cdot v_i \oplus Y \\
&= \bigoplus_{i\in[0,n)} \chi_i \cdot w_i \oplus \chi_\alpha \cdot \Delta \oplus \bigoplus_{i\in[0,\lambda)} (y_i^* \oplus x_i' \cdot \Delta) \cdot X^i \\
&= \bigoplus_{i\in[0,n)} \chi_i \cdot w_i \oplus \chi_\alpha \cdot \Delta \oplus \bigoplus_{i\in[0,\lambda)} (y_i^* \oplus x_i^* \cdot \Delta \oplus x_i^* \cdot \Delta \oplus x_i' \cdot \Delta) \cdot X^i \\
&= \bigoplus_{i\in[0,n)} \chi_i \cdot w_i \oplus \bigoplus_{i\in[0,\lambda)} z_i^* \cdot X^i \oplus \chi_\alpha \cdot \Delta \oplus \bigoplus_{i\in[0,\lambda)} (x_i^* \oplus x_i')X^i \cdot \Delta \\
&= W \oplus (\chi_\alpha \oplus \chi_\alpha')\Delta
\end{aligned}$$

Therefore, if $\chi_\alpha = \chi_\alpha'$, we have $V = W$, showing that the simulation strategy of sending $\mathsf{H}'(W\|\mathtt{sid}\|\mathtt{ssid})$ in $\mathbf{Hybrid}_5$ is perfect.

If $\chi_\alpha \neq \chi_\alpha'$, the machine in $\mathbf{Hybrid}_5$ computes every possible Δ' value from the previous random oracle transcript and checks if $\Delta' = \Delta$. If so, then it implies the underlying random oracle response has been sampled before, so the machine answers with the previously sampled value. Otherwise, it samples $h \leftarrow \{0,1\}^{2\lambda}$, answers with this new h value, and records the corresponding information $(W, \chi_\alpha' \oplus \chi_\alpha, \mathtt{sid}\|\mathtt{ssid}, h)$. Notably, this response implicitly defines a random oracle preimage that is related to the Δ value.

For every subsequent random oracle query, the machine in $\mathbf{Hybrid}_5$ not only uses the transcript $\mathcal{Q}$ to give consistent answers, but also checks the query against the implicit entry. Therefore, with the help of the Δ checking procedure, the machine in $\mathbf{Hybrid}_5$ never gives inconsistent random oracle responses. We conclude that the random oracle simulation is perfect.

□

Contents

1 Introduction **1**
- 1.1 Our Contributions . . . 3
  - 1.1.1 New CR Notions for MPC. . . . 3
  - 1.1.2 Provably Secure Constructions. . . . 4
  - 1.1.3 Applications of S-mTCCRL. . . . 5
  - 1.1.4 Organizations. . . . 5

2 Technical Overview **5**
- 2.1 Limitations of Game-based Correlation Robustness . . . 5
- 2.2 S-mTCCRL: Simulatable mTCCRL . . . 7
- 2.3 S-mTCCRL Secure Hash Constructions . . . 9
  - 2.3.1 Security of Blockcipher-based Constructions via a Unified Approach. . . . 9
  - 2.3.2 Security of Random Oracle-based Constructions. . . . 10
- 2.4 Applications of S-mTCCRL in MPC . . . 10

3 Preliminaries **14**
- 3.1 Multi-user tweakable circular correlation robust hashing . . . 14

4 S-mTCCRL: Simulatable TCCRL Hashing **15**

5 S-mTCCRL Hashing in the Ideal Cipher Model **17**
5.1 Unified Model of 1-call Blockcipher-based Hash . . . 17
5.1.1 Restrictions. . . . 18
5.2 S-mTCCRL Security of $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$ . . . 18
5.3 Concrete Results: S-mTCCRL security of $\widehat{\mathsf{MMO}}^{\mathbf{E}}$ . . . 19

6 UC-Secure Authenticated AND Triple Generation from S-mTCCRL **21**
6.1 Basic Primitives . . . 21
6.2 Leaky Authenticated AND Triples . . . 21

A Deferred Security Notions **31**
A.1 Formal Definitions of the Basic Notions CR and S-CR . . . 31
A.2 Zoo of weaker variants of mTCCRL and S-mTCCRL . . . 31
A.2.1 Key-whitened oracles without circularity or tweaks. . . . 32
A.2.2 Multi-user CR definition zoo. . . . 32
A.2.3 Multi-user S-CR definition zoo. . . . 34
A.2.4 Brief relations within the zoo. . . . 35

B S-mTCCRL Hashing in the Random Oracle Model **35**

C Proof of Theorem 3 (S-mTCCRL Security of $\mathsf{ROH}^{\mathbf{RO}}$) **36**
C.1 Constructing the simulator . . . 36
C.1.1 Variables of the simulator. . . . 36
C.1.2 Simulating strategy. . . . 37
C.1.3 Pseudocode description. . . . 37
C.1.4 Complexity of the simulator. . . . 37
C.2 Proof idea . . . 38
C.3 Intermediate world . . . 39
C.4 Event Coll in intermediate world executions . . . 39
C.5 Indistinguishability of Σ_{inter} and Σ_{id} . . . 43
C.6 Indistinguishability of Σ_{inter} and Σ_{re} . . . 45
C.6.1 Good intermediate world executions and properties. . . . 45
C.6.2 Associating real world executions via randomness. . . . 47
C.6.3 Completing the indistinguishability proof. . . . 47

D Proof of Theorem 1 (S-mTCCRL Security of $\mathsf{GH}[f_{\mathsf{in}}, g, f_{\mathsf{out}}]^{\mathbf{E}}$) **49**
D.1 Constructing the simulator . . . 49
D.1.1 Explicit randomness. . . . 49
D.1.2 Variables of the simulator. . . . 49
D.1.3 Simulating strategy. . . . 50
D.1.4 Pseudocode description. . . . 50
D.1.5 Complexity of the simulator. . . . 51
D.2 Proof idea . . . 52
D.3 Event Chain in real world executions . . . 53
D.3.1 Motivations. . . . 53
D.3.2 Definition. . . . 53
D.3.3 Good real world executions and properties. . . . 54
D.4 Associating ideal world executions . . . 55

D.5 Completing the indistinguishability proof . . . 57
D.6 Proof of Eq. (32): bad events in simulation . . . 59
D.6.1 First step: high-complexity mTCCRL distinguisher $\mathcal{A}$. . . . 59
D.6.2 Probabilities of Eve. . . . 60
D.6.3 To mTCCRL attacker with bounded complexity. . . . 62
D.6.4 Summary. . . . 63
D.6.5 Discussion. . . . 64

E Constructions for Weaker Notions in the ICM **64**
E.1 S-mCCRL of $\widehat{\mathsf{MMO}}^{\pi}$ . . . 64

F Security of Protocol Π_{LaAND} **65**
F.1 Subroutine Functionalities . . . 65
F.2 Delayed Phase Switch . . . 65
F.3 Proof of Theorem 2 . . . 69

G UC-Secure Ferret Single-Point COT from S-mTCCRL **80**