Introduction to Indifferentiability

P. Farshim

Many practical protocols have been designed assuming (intuitively) that a well-designed hash function (such as SHA) has "random-looking" outputs. This methodology has been formalized via the **random-oracle model** by Bellare and Rogaway [BR93].

The Random-Oracle Model

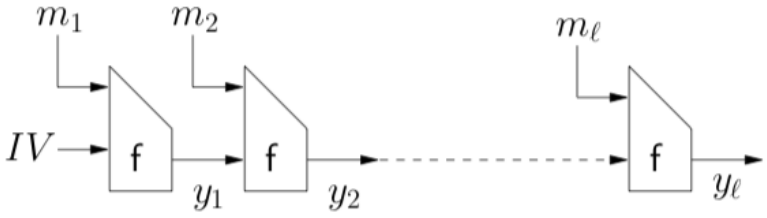
Given a domain D and range R, define Func[D,R] := {f: f is function D ⟶ R}. The **random oracle** (RO) is the uniform distribution on Func[D,R]. The **random-oracle model** is obtained when all parties, honest or otherwise, are given oracle access to a function chosen at random f ⟵ Func[D,R]. Access to f models an **ideal hash function** whose outputs are uniform and independent (as in the intuition above). This methodology has been very successful as many practical protocols have been proven secure in the RO model (FDH, RSA-OAEP, Fiat-Shamir, IBE, etc.).

Remark: Similarly, we can define:

1) The **random-permutation model** by looking at the uniform distribution over Perm[D] := $\{\pi^{\pm} : \pi^{+}$ a permutation D ⟶ D with inverse $\pi^{-}\}$
2) The **ideal-cipher model** by looking at all *keyed* permutations (aka block-ciphers) Block[K,D] := $\{E^{\pm}(\cdot,\cdot)$: for all k in K $E^{+}(k, \cdot)$ is a permutation on D with inverse $E^{-}(k,.)\}$.

The Gap

Concrete hash functions are **non-monolithic** objects as they come with extra internal structure. For example, a popular approach to build hash functions is to start with a simpler primitive, such a compression function f: $\{0,1\}^{2n} \longrightarrow \{0,1\}^{n}$, and then build a full-fledged (variable-input-length; VIL) hash function H: $\{0,1\}^{*} \longrightarrow \{0,1\}^{n}$. A classical way to do this is, for example, via the Merkle--Damgård (MD) chaining:



But this **leaves a gap** in the overall security analyses: A proof of security for, say, the unforgeability of FDH (a signature scheme using random oracles) in the RO model assumes a monolithic RO. It is then unclear if this proof would still hold once we replace the monolithic RO with a non-monolithic one, like that build above via MD chaining.

Two ways to go:

1) A **construction-specific** approach: re-do security analysis with a given hash construction.
2) A **modular** approach: define what it means for a hash construction to realize a random oracle. Prove that a given construction meets this definition, and apply the old (and simpler) analysis in RO.

The intuitive approach is (2): If f is a (small) RO then our hope is that MD^f is a big (VIL) RO. To formalize this, however, we need to **define the problem**:

What does it mean to realize a RO?

Developing the Definition

Attempt 1: A construction C^f realizes a random oracle if $C^f(\cdot)$ is **indistinguishable** from RO(·):

$$C^f \approx \text{RO}.$$

Although reasonable, it misses a crucial point: the adversary **does not** get access to f when trying to distinguish C^f from RO. This is a serious shortcoming: **f is publicly available** and being an integral part of the realization of RO could be central to distinguishing attacks. And, of course, not giving access to f goes against the RO methodology: *all parties, honest or otherwise, get access to ideal primitive.*

Attempt 2: A construction C^f realizes a random oracle if C^f is indistinguishable from RO *even in the presence of f***:**

$$(C^f, f) \approx (\text{RO}, f)\ .$$

Question: Can this ever hold?
Answer: No, because the distinguisher can simple compute $h = C^{O2}(X)$ using the second oracle O2 for some X and check if h = O1(X). In the left world they would match, in the right world, whp, they won't.

Attempt 3: The problem is that we are comparing (C^f, f) to a world (RO, f) **doesn't quite make sense**. When we analyze FDH in the ROM there is no f; we just have the RO.

What we want is a way to compare attacks on C^f with access to f to attacks on "**the ideal functionality"** RO. That is, what we really want is that for any attack on C^f there is an **equivalent attack** on RO. This is inspired by the **real/ideal paradigm**, notably the UC framework.

Let's develop the definition. Take an arbitrary security 0/1-game G for hash functions (we don't want to focus on any specific game for modularity). The game looks like

$$G^{C^f, A^f}.$$

We want to say that there is an **ideal attacker** B against the RO in the same game, i.e., $G^{RO,B^{RO}}$ that has a similar behavior. This leads to:

Def1: C^f realizes RO if for any G and any A there is a B such that

$$\Pr[G^{C^f, A^f} = 1] \approx \Pr[G^{RO, B^{RO}} = 1].$$

We are qualifying over all G and A. We can this absorb A into G to simplify. With this change, let's rename G to a distinguisher D and B to a **simulator** S. The new definition is:

Def2: C^f realizes RO if for any D there is an S such that

$$\Pr[D^{C^f, f} = 1] \approx \Pr[D^{RO, S^{RO}} = 1].$$

To see the equivalence more formally:

1. Def1 $\Longrightarrow$ Def2: Trivial: Given D in Def2, define G := D and A := dummy adversary in for Def1. Take an ideal attacker B for (G,A). Then S := B is a simulator for D.
2. Def2 $\Longrightarrow$ Def1: Easy: Given (G,A) in Def1, consider D that runs G together with A (absorption). If S is a simulator for D, then B := A^S would be an ideal attacker in G for A.

So Def2, basically says that

$$(C^f, f) \approx (RO, S^{RO})$$

are indistinguishable, and this is the definition of **indifferentiability** of C^f from RO. We define as usual:

$$Adv^{indiff}_{C,D,S} := \Pr[D^{C^f, f} = 1] - \Pr[D^{RO, S^{RO}} = 1].$$

As mentioned, this definition has its roots in the UC framework of Canetti and reactive systems of Pfitzmann and Waidner… It was formulated with respect to the more general random systems by **Maurer, Renner, and Holenstein** [**MRH04**]. It was then made explicit for hash functions by Coron, Dodis, Malinaud, and Puniya [CDMP05].

Remark: More precisely, this is the definition of **weak** indifferentiability. **Strong** indifferentiability reverses the order of the quantifies and requires a **universal simulator** S for all D. As far as I know, all indifferentiability proofs are strong.

Problem: Come up with a separating example or prove equivalence.

Remarks:

1) What the definition (informally) says is: S^{RO} "cooks up" f-values as if RO was built via C from these values.
2) The simulator does **not** get to see the queries of D to RO. If we allowed this, we get a variant called **public indifferentiability**, which is sufficient for some (but not all) games, where all RO queries are known by the adversary.

Composition

What did we just do? We developed a definition of indifferentiability by looking for a (sufficient) condition that allows **composition** in arbitrary games (that is, one that allows replacing RO with C^f in arbitrary settings G. Let's state and prove this once more.

Theorem (Composition): If C^f is indifferentiable from RO then for any game G, whenever RO is G-secure so is C^f, even when the adversary gets access to f (that is, indifferentiability is sufficient for composition). The converse implication also holds (indifferentiability is also necessary for composition).

Proof:

- ($\Longrightarrow$) G^{C^f,A^f} $= D^{C^f,f}$ where D := run G with A
  $= D^{RO, S^{RO}} + Adv^{indiff}$
  $= G^{RO,A^{S^{RO}}} + Adv^{indiff}$
  $= G^{RO,B^{RO}} + Adv^{indiff}$ where $B := A^S$.

- ($\Longleftarrow$) Take the dummy A so that the game is $G^{C^f,f}$. Let the indifferentiability simulator S be B, the ideal attacker for the dummy A.

Exercise: For which class of games *indistinguishability* guarantees composition?

Concrete security: Note that an A that places q queries to f against C^f is translated to a qq_S-query adversary B against the RO where q_S is an upper bound on the number of RO queries that S places per invocation.

Example: If $q_S = q$ (e.g., because the simulator goes through some list and queries RO on all entries) then $qq_S = q^2$. That is a q-query adversary against C^f translates to a q^2-query adversary against RO. We know, for example, that RO:

2n⟶ n is collision-resistant with bound $q^2/2^n$. This means the indifferentiable construction is only guaranteed to be collision resistant with bound $(q^2)^2/2^n$, which is quartic. Thus, efficient (low query) simulators are important.

The Scope of Composition

We can extend the syntax of the games and still get composition. The games could take the from:

$$G^{C^f, A1^f, A2^{C^f}, ..., An^{C^f}}$$

for adversaries A1, ..., An which maintain **their own local state** (and cannot necessarily communicate freely via). This is because $A2^{C^f}$, ..., An^{C^f} can be all merged into G since G has access to C^f.

How expressive is this class?

- With just a single $A1^f$ we get many standard games: One-wayness, collision resistance, IND-CPA/CCA, unforgeability, etc. all have a **central adversary**.
- With two adversaries $A1^f$, $A2^{C^f}$ we get more. In particular we would get security against related-key attacks and key-dependent message security where the related-key functions or message-deriving function even when they depend on C^f. This includes many practical cases (e.g., xor-RKAs). There are many other games in this class.

Question: Does composition extend further to games of the form

$$G^{C^f, A1^f, A2^f, ..., An^f} ?$$

(If it did then we can extend the reach of composition to a larger class of games, for example, RKA where the key deriving functions directly depend on f.)

Answer: No, composition does **not** go through. This was observed by Ristenpart, Shacham, and Shrimpton [RSS11]. We would need to replace each Ai using a single simulator S. This gives

$$G^{RO, A1^{S^{RO}}, A2^{S^{RO}}, ..., An^{S^{RO}}}$$

where S would need to **keep state across different invocations to maintain consistency.** But if Ai cannot communicate with each other freely, then the simulator won't be able to keep consistent state. Indeed, there is a **separating example**: There is a game with two adversaries that is hard to win with respect to a RO, but easy to win with respect to an indifferentiable construction.

Problem: Are there also separating examples for the RKA and/or KDM games?

Reflections

With indifferentiability we get security with respect to a large class of games in a **single** proof. These include many known games, and new yet unforeseen ones. The unforeseen ones could come in the form of a **new protocols** using the RO in a new way, or a **new security model** for hash functions. Both are covered as long as they fall within the extended game syntax above.

Problem: Is there an ideal primitive f and construction C such that C^f is *indistinguishable* from RO but *every* construction is *differentiable* from RO?

Basic Indifferentiability Questions

The general template for a typical question in indifferentiability is similar to the rest of crypto:

Given primitives P and Q, is there an indifferentiable construction C^P of Q?

A partial catalogue of primitives $\mathbf{K} \times \mathbf{D} \rightarrow \{0,1\}^n$:

	$\mathbf{K} = \{\varepsilon\}$ $\mathbf{D} = \{0,1\}^n$	$\mathbf{K} = \{\varepsilon\}$ $\mathbf{D} = \{0,1\}^*$	$\mathbf{K} = \{0,1\}^k$ $\mathbf{D} = \{0,1\}^n$	$\mathbf{K} = \{0,1\}^k$ $\mathbf{D} = \{0,1\}^*$
Functions F	*ideal compression*	*RO*	ideal keyed compression	keyed RO
Permutations P	*random permutation*	wide random permutation	*ideal cipher*	wide ideal cipher
Injections I	random injection	wide random injection	ideal encryption	wide ideal encryption

Converting:

- F1 to F2, F3 or F4 are done via **Merkle-Damgård**.
- F1 to P1 is done via **Feistel**.
- P1 to P2 is done via **Confusion-Diffusion** (CD).
- P1 to P3 is done via **Even-Mansour** construction.
- P1 to P4 is done via **Even-Mansour + CD** (that is, AES). Open!
- F1 to I4 is the subject of building **highly-secure AE** schemes**.**

We want the constructions to be as efficient and as secure as possible.

1) **Construction**: minimize various resources: e.g., the sequential and/or parallel query complexity of the construction.
2) **Security**: 2.1) minimize the query complexity of the simulator.
   2.2) minimize the indifferentiability advantage bound.

Or if you cannot go beyond some bounds, show evidence for its hardness, i.e., give an **indifferentiability lower bound** (on the efficiency of the construction and/or simulator). If there is time, I'll give an example on Friday.

Problem(s): A lot of the lower bounds remain open. Resolve them!

Domain Extension for ROs

We are given a small hash function f and we want to build a big hash function C^f that is indifferentiable from RO.

Question: Let H be a VIL *collision-resistant hash function* (which is non-ideal). Define.

$$C^f(X) = f(H(X)),$$

i.e., hash the input first and the apply the small RO to it. Is this a VIL-RO?

Answer: No. $(C^f, f) \approx (RO, S^{RO})$. Call the first oracle Const and second Prim. Consider the following **differentiator**.

1. Sample a random and long X.
2. Query Const(X) to get Y. (This is either f(H(X)) or RO(X).)
3. Compute h := H(X) locally (H is non-ideal).
4. Query the Prim(h) to get Y'. (This is either f(h) or S(h).)
5. Return (Y = Y').

In the real (left) world Y = Y' with probability 1. In the ideal (right) world, however, the simulator would need to guess Y. It is however only given h = H(X). Since H is compressing, the probability of guessing X used to compute RO(X) is negligible. Outside this, the only hope is to guess Y, which is also negligible.

Note: Even an **inefficient** simulator (which even has all of RO) will **not** succeed.

Exercise: Intuitively, what property of the RO does the above construction not achieve? If H is a random oracle (and ideal), the construction should be intuitively indifferentiable from RO. In this case, why can the simulator guess X in this case?

The Length-Extension Attack

The MD transform allows us to extend the domain, but only for arbitrary but **fixed-length** messages. This, however, does not mean that we also a **variable-input-length** RO (i.e., where input lengths can vary). Inconsistencies across "different branches of the simulator" could arise, for example is some message is a **prefix** of another and hence queries to f coincide.

Indeed, in the plain MD construction given $h^* = MD^f(M1,M2)$ (and not M1 or M2) one can easily compute $MD^f(M1,M2,M3)$ for any M3. This is the so-called length-extension attack. Obviously, length extension is not possible with respect to the RO. This in turn leads to the following differentiator:

1. Pick random M1, M2, and M3.
2. Query $h^* \leftarrow Const(M1,M2)$ and $h \leftarrow Const(M1,M2,M3)$.
3. Query $h' \leftarrow Prim(h^*,M3)$.
4. Return $(h' = h)$.

In the real world, the differentiator returns 1 with probability 1. In the ideal world, a simulator given (h*,M3) would have to guess RO(M1,M2,M3). This is information theoretically impossible (even given all of RO) since h* misses n bits of information about (M1,M2). Outside guessing (M1, M2), the simulator would have to guess h', which happens with negligible probability.

Exercise: Is the plain MD^f construction *publicly* indifferentiable? Is this enough to use MD^f within FDH? What about as a MAC?

Using a **prefix-free** encoding of the input we can avoid inter-dependencies across simulators for different message lengths (Fig. 6). This encoding however can be somewhat wasteful. There are more efficient alternatives, the most important being the HMAC and NMAC constructions (Fig. 8).

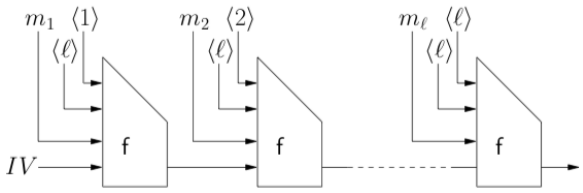


Fig. 6. Merkle-Damgård with a particular prefix-free encoding.

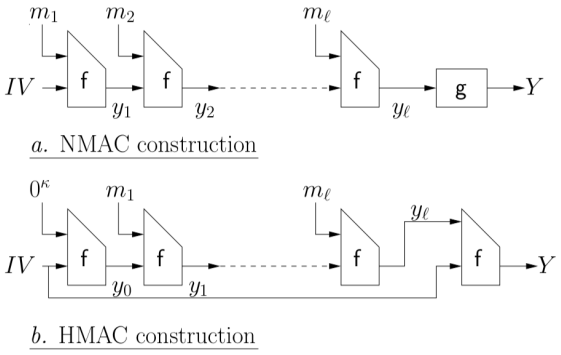


Fig. 8. The NMAC and HMAC constructions

References

[MRH04] U.M. Maurer, R. Renner, and C. Holenstein. Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. *TCC 2004.*

[CDMP05] J.S. Coron, Y. Dodis, C. Malinaud, and P. Puniya. Merkle-Damgård revisited: How to construct a hash function. *CRYPTO 2005.*

[RSS11] T. Ristenpart, H. Shacham, and T. Shrimpton. Careful with composition: Limitations of the indifferentiability framework. *EUROCRYPT 2011.*