A preliminary version of this paper appears in the proceedings of INDOCRYPT 2020. This is the full version.

Incremental Cryptography Revisited: PRFs, Nonces and Modular Design

Vivek Arte[1] Mihir Bellare[2] Louiza Khati[3]

October 2020

Abstract

This paper gives the first definitions and constructions for incremental pseudo-random functions (IPRFs). The syntax is nonce based. (Algorithms are deterministic but may take as input a non-repeating quantity called a nonce.) The design approach is modular. First, given a scheme secure only in the single-document setting (there is just one document on which incremental updates are being performed) we show how to generically build a scheme that is secure in the more realistic multi-document setting (there are many documents, and they are simultaneously being incrementally updated). Then we give a general way to build an IPRF from (1) an incremental hash function with weak collision resistance properties and (2) a symmetric encryption scheme. (This adapts the classic Carter-Wegman paradigm used to build message authentication schemes in the non-incremental setting.) This leads to many particular IPRFs. Our work has both practical and theoretical motivation and value: Incremental PRFs bring the benefits of incrementality to new applications (such as incremental key derivation), and the movement from randomized or stateful schemes to nonce based ones, and from UF (unforgeability) to PRF security, bring incremental symmetric cryptography up to speed with the broader field of symmetric cryptography itself.

[1] Department of Computer Science & Engineering, University of California San Diego, USA. Email: `varte@eng.ucsd.edu`. URL: `cseweb.ucsd.edu/~varte/`. Supported in part by the grants of the second author.

[2] Department of Computer Science & Engineering, University of California San Diego, USA. Email: `mihir@eng.ucsd.edu`. URL: `cseweb.ucsd.edu/~mihir/`. Supported in part by NSF grant CNS-1717640 and a gift from Microsoft corporation.

[3] ANSSI, France. Email: `louiza.khati@ens.fr`. URL: `www.di.ens.fr/~khati/`.

Contents

1 Introduction

Data isn't static. We routinely edit our videos, photos, MS-Word / Apple-Pages files or text files. (We'll use the term "documents" to cover these types of data and more.) Incremental cryptography [BGG94, BGG95] was conceived to harmonize cryptography with this world of dynamic data. The idea is that, just as we edit the document, we can "edit" the already-computed result of a cryptographic function of the document, via a special, fast "update" algorithm, to obtain the result of the cryptographic function on the edited document much more quickly than if we had computed it from scratch. The challenge is not only to give schemes allowing updates, but to ensure that the updates are secure.

The relevance of incremental cryptography is even greater now in the world of big data, where the volume and size of documents makes from-scratch re-computations of cryptographic functions prohibitively expensive. In this light we revisit the subject.

Contributions in brief. Incremental cryptography has previously been considered for many primitives [BGG94, BGG95, Fis97a, BM97, BKY02, MPRS12, MGS15, SY16, GP17, KV19]. But one for which it has surprisingly *not* so far been considered is Pseudo-Random Functions (PRFs) [GGM86], arguably the most basic primitive, and important building block, in symmetric cryptography. Our work fills this gap by giving the first definitions and constructions for incremental pseudo-random functions (IPRFs).

We do this, not in isolation, but as part of a broader effort. Its first component is a *new framework*. We adopt a nonce-based setting [RBBK01, Rog02] (algorithms are deterministic but may take input a non-repeating quantity called a nonce), defining a syntactic object called an incremental function family (iFF). For it we define two security metrics, namely incremental unforgeability (IUF) and incremental pseudo-randomness (IPRF), and show that the latter implies the former. The second component is *modular design*. Where the most related prior work (on incremental message authentication) gave ad hoc, dedicated schemes, we instead give general transforms. First we show how, given a scheme secure only in the single-document setting (there is just one document to which updates are being applied), to build a scheme secure in the more realistic multi-document setting (the scheme can handle many documents on which updates are being performed simultaneously). Then we show how to achieve security in the single-document setting through an extension of the Carter-Wegman paradigm [WC81]. (Recall that the latter has been extensively used to obtain non-incremental UF-secure schemes such as [BHK$^+$99, Rog95, Sho96, HK97].) The result is that, even ignoring PRF security and considering only the UF security goal targeted in incremental message authentication work [BGG94, BGG95, Fis97a, KV19], we bring stronger schemes (able to use, and be secure with, any non-repeating nonce) and modular ways of building and analyzing them.

Background and context. Recall that a function family F takes a key K and input X to deterministically return an output $Y = \mathsf{F}(K, X)$. For this syntax, one can define both PRF and UF security. Classical examples include HMAC [BCK96] and CMAC [Dwo05], but these are not incremental. To achieve incrementality, schemes, starting with [BGG94, BGG95] and continuing with [Fis97a, KV19], use randomization or state (a counter), making them, even just as syntactic objects, different from function families. They are usually called message authentication schemes because the goal is UF. But PRFs are inherently deterministic and stateless, so that PRF security cannot even be defined, let alone achieved, under this syntax. Our solution is to move to a nonce-based setting. iFF algorithms are deterministic but may take input a nonce. Security will require only that nonces are not reused. Now it becomes possible to define both UF and PRF security, and consider achieving them, either via nonce-based extensions of prior schemes, or in other ways.

Why IPRFs? IPRF security (as opposed to just IUF) is valuable because it brings the possibility of efficiency improvement via incrementality to a broader range of applications, namely ones that, like key-derivation, require pseudo-randomness. For example, an application holding key K may at regular intervals $i = 1, 2, \ldots$ derive a sub-key K_i by applying a PRF to a quantity X_i that contains both static data (application-dependent context) and dynamic data (the counter i). An incremental PRF allows the application to update K_i to K_{i+1} in a way that is faster than computing K_{i+1} from scratch.

Why nonces? Nonce-based schemes are valuable (beyond allowing IPRF security) because, in practice, randomness is fragile (system RNGs are prone to failure) and state may not be maintainable (due to system resets), so schemes that maintain security for arbitrary (non-repeating) nonces are more versatile and robust, and correspondingly less error-prone.

Beyond this, the movement from randomized or stateful schemes to nonce based ones, and from UF to PRF, "modernizes" incremental message authentication, bringing it up to speed with the broader field of symmetric cryptography in which, over the last 20 years, we have seen these same movements, not just in academic work but also in standards and deployment. The rest of this Introduction discusses our contributions, and related work, in more detail.

New framework. As indicated above, prior work considered many types of syntax (randomized, stateful, deterministic). We define and consider just a single type: nonce-based. We call the object in question an incremental function family (iFF). It provides tagging, update and verification algorithms that are all deterministic, the first two (but not the last) taking as input a nonce. Thus, the tagging algorithm of an iFF iF takes the key K, a nonce N, document identifier id, and document D to return a tag. The update algorithm takes K, N, id, D, a description (consisting of an operation code op and corresponding argument arg) of the edit to be performed on the document, and current tag t, to return an updated tag t' for the edited document. Verification algorithm Ver takes K, id, D, t to return a boolean decision on the validity of t.

We define incremental unforgeability (IUF) of an iFF iF via a game that gives the adversary access to oracles $\textsc{Tag}, \textsc{Upd}, \textsc{Vf}$ for tagging, update and verification (respectively) under a game-chosen key, winning requiring forging, via the last oracle, a valid tag for a new document. More novel and interesting is that this syntax permits us to define, for the first time in the incremental setting, a notion of PRF security, that we denote IPRF. The oracles in our game formalizing IPRF security have the same names as for IUF, but the first two return either real or random tags depending on a challenge bit that the adversary has to determine to win, while the last returns either real verifications or just rejects. Both definitions require that nonces cannot be re-used, except in the degenerate case that the scheme itself allows just one choice of nonce. (The exception allows us to capture nonce-free schemes as a special case.)

Crafting these definitions was delicate, in part due to our wanting PRF security to imply UF security. Recall that for regular (no nonces, no incrementality) function families, the implication is true [BKR00, GGM86]. But nonces can disrupt this. Indeed, for the prevailing definitions of PRF and UF security for nonce-based (not incremental) function families, the implication fails [PS16]. Through the (unusual) step of including a verification oracle in the IPRF game, we obtain (cf. Proposition 2) the "best of both worlds:" Nonces are allowed yet IPRF security does imply IUF security. This is valuable because establishing the former now obviates us from having to separately establish the latter.

In addition to a standard correctness requirement, we define strong correctness, which asks that updated tags produced by the update algorithm be identical to the ones that would have been computed by tagging the edited document from scratch with the tagging algorithm. For schemes with this property, Proposition 1 says that updates (the $\textsc{Upd}$ oracle) may effectively be neglected in proving security. The security of updates having been, historically, the main new security concern brought by incrementality [BGG94, BGG95], Proposition 1 results in significant proof simplification.

From single- to multi-document security. With new (and stronger) target definitions in place, we turn to designing efficient schemes that meet them. We aim for modularity as a way to simplify both design and analysis. The first angle we consider here is single-document (sd) versus multi-document (md) security.

The tagging, update and verification algorithms in our syntax all take as input a document identifier *id* (for example `myphoto.jpg`) that names the document on which the operation is to be performed. In the sd setting, there is only one allowed choice of *id*, as though your filesystem contained just one file that you keep editing. In the more realistic md setting, any number of document identifiers may coexist, the adversary choosing them in its queries. The treatment in prior work (the two settings originate in [BGG95]) has been ad hoc, with schemes and proofs given first for the sd setting, then separately for md. We step back to take a higher-level view. We show how sd-secure iFFs can be generically turned into md-secure ones, giving for this purpose two general "bootstrapping" transforms, **StM1** and **StM2**. Each turns a given iFF $\mathsf{iF}_{\mathsf{sd}}$ that is IUF/IPRF-secure in the sd setting into an iFF $\mathsf{iF}_{\mathsf{md}}$ that is IUF/IPRF-secure in the md-setting. The first transform is simple and natural, working for all choices of document edit operations, but the reduction (Theorem 3) is not tight. The second transform allows a tight reduction (Theorem 4). It requires strong correctness (discussed above) of $\mathsf{iF}_{\mathsf{sd}}$ and also that the document edit operations are what we call "translating," but the first is met by our constructions discussed below, and the second is generous enough to include common operations like *replace*, *insert*, *delete*.

Incremental Carter-Wegman. The above has simplified our task of designing iFFs that are IUF/IPRF-secure in the md setting, reducing it to the same task in the sd setting. We now further simplify the latter task, reducing it, via an extension of the Carter-Wegman paradigm, to the task of designing incremental hash functions satisfying weak collision-resistance properties.

The standard Carter-Wegman (CW) paradigm [WC81] builds a nonce-based message authentication scheme by first applying a hash function to the message, and then masking the result h in some way using the key and nonce to get the tag. In our "incremental Hash-then-Encrypt" variant **iHtE**, the hash function iHF is assumed incremental for some set of edit operations. Now we need to extend CW so that (1) the incrementality of iHF is inherited by iF, and (2) IPRF security (as opposed to just the UF provided by CW) is provided. The change is in the masking. The general CW paradigm does not require this to be reversible, but, to allow incremental updates, we do, accordingly using for the purpose a symmetric encryption scheme SE. Furthermore, SE is required to meet the NBE2 syntax of [BNT19]. This means the nonce is not required for decryption. (Otherwise, one has to include the nonce in the tag. This would provide IUF but violate IPRF.) The full construction in Section 5 also uses a key-derivation function that we omit here. Theorem 6 says that **iHtE** provides IPRF security assuming iHF is cau-secure and SE is AE2 secure. (Recall that cau [Bel06], the computational relaxation of the almost universality notion of [WC81], is a very weak form of collision resistance for secretly-keyed hash functions. AE2 is the notion of authenticated encryption security for NBE2 schemes from [BNT19].)

Instantiations. We give many choices of cau-secure hash functions that are incremental for the *replace* operation, yielding, via **iHtE**, corresponding iFFs that are IPRF secure in the sd setting and incremental for the same operation. These hash functions are implicit in message authentication schemes already in the literature, and we only need to extract them.

Section 6 takes a systematic approach. We look at different message authentication schemes in the literature including XOR-MACs [BGR95], GMAC [MV04], Poly1305-AES [Ber05], PMAC

[BR02], PWC [PS16], ZHASH [IMPS17] and more. For each, we extract an underlying incremental hash function. In some cases (eg. [BR02, Ber05]), the authors have already noted that their algorithms allow incremental updates, but stopped short of showing that any formal notion of incremental security is met. To fill this gap, we cast their schemes as iFFs. Their results in some cases can be used to conclude IPRF security of our iFF. But more often, they only yield IUF security (because of inclusion of the nonce in the tag). In this case, we go back to the underlying hash function and use **iHtE** to get an IPRF. Some of the original message authentication schemes, however, are not themselves incremental, usually due to a non-invertible masking function in an (implicit or explicit) use of CW paradigm. In these cases, again, we go back to the underlying hash function and use **iHtE** to get an IPRF. Figure 8 summarizes the instantiations obtained.

LIMITATIONS AND EXTENSIONS. Incrementality is with respect to some set of edit operations on the documents. (This is formalized as a *document editing system* in the body of this paper.) Our "boosting" results —from sd security to md via **StM1**, **StM2**, or from incremental cau-hash functions to incremental IPRFs via **iHtE**— are general in this regard, preserving the allowed set of operations. (That is, if the starting scheme is incremental for some set of edit operations, the constructed scheme is incremental for the same set, with the above-discussed caveat that for **StM2** the operations must be translating.) However, we currently know of examples of incremental cau-secure hash functions only for the *replace* operation, so obtain IPRFs only for this. Via nonce-based extensions of the randomized schemes of [BGG95, KV19], we can obtain iFFs that are incremental for *insert*, *delete* and that provide IUF security. These, however, will not provide IPRF security. We leave IPRFs for *insert*, *delete* as an open problem.

For incremental message authentication, [BGG95] considered security that was either "basic" (the adversary can request updates only on documents resulting from prior tagging or update operations) or "tamper-proof" (the restriction is dropped). Our definitions and results are all for basic security. This already suffices for many applications. We can define and achieve tamper-proof IUF security, but for IPRFs we do not know how to do this extension, and leave it as an open problem.

Overall we view our work as initiating the study of incremental PRFs, and leave extensions to future work.

RELATED WORK. Incrementality has been considered for (UF-secure) message authentication [BGG94, BGG95, Fis97a, KV19], encryption [BGG95, BKY02, SY16], collision-resistant hashing [BGG94, BM97, MGS15], digital signatures [BGG94, Mic97, Fis97b], deterministic PKE [MPRS12], program obfuscation [GP17] and beyond [ACJ17]. Early work on incremental symmetric encryption [BGG95, BKY02] used the classical randomized setting. Sasaki and Yasuda [SY16] were the first to bring nonces into this, treating nonce-based authenticated encryption. We follow in their vein, bringing nonces also to UF and PRF security.

2 Preliminaries

NOTATION. By $[1..n]$ we abbreviate the set $\{1, \ldots, n\}$ and by $[i \ldots j]$ the set $\{i, \ldots, j\}$, for integers $n \geq 1$ and $j \geq i$. We denote the number of coordinates of a vector D by $|D|$ and its i-th coordinate by $D[i]$. By B^* we denote the set of all vectors over B, meaning vectors D with $D[i] \in B$ for all $i \in [1..|D|]$. The empty vector is denoted $()$. The empty string is denoted ε. If $x \in \{0,1\}^*$ is a string then $|x|$ is its length and $x[i]$ is its i-th bit. We let $x[i..j] = x[i] \ldots x[j]$ be the concatenation of bits i through j of x if $i \leq j$ and ε otherwise. If S is a finite set then $|S|$ its is size or cardinality. We use $\perp$ (bot) as a special symbol to denote rejection, and it is assumed to not be in $\{0,1\}^*$.

By $\mathrm{FUNC}(D, R)$ we denote the set of all functions $f : D \to R$ and by $\mathrm{PERM}(D)$ the set of all permutations $\pi : D \to D$.

If X is a finite set, we let $x \twoheadleftarrow X$ denote picking an element of X uniformly at random and assigning it to x. Algorithms may be randomized unless otherwise indicated. If A is an algorithm, we let $y \leftarrow A^{\mathrm{O}_1,\cdots}(x_1, \ldots ; \omega)$ denote running A on inputs $x_1, \ldots$ and coins ω, with oracle access to $\mathrm{O}_1, \ldots$, and assigning the output to y. By $y \twoheadleftarrow A^{\mathrm{O}_1,\cdots}(x_1, \ldots)$ we denote picking ω at random and letting $y \leftarrow A^{\mathrm{O}_1,\cdots}(x_1, \ldots ; \omega)$. We let $\mathrm{Out}(A^{\mathrm{O}_1,\cdots}(x_1, \ldots))$ denote the set of all possible outputs of A when run on inputs $x_1, \ldots$ and with oracle access to $\mathrm{O}_1, \ldots$. An adversary is an algorithm. Running time is worst case, which for an algorithm with access to oracles means across all possible replies from the oracles.

<u>Games.</u> We use the code-based game-playing framework of BR [BR06]. A game G (see Figure 1 for an example) starts with an optional Init procedure, followed by a non-negative number of additional procedures called oracles, and ends with a Fin procedure. Execution of adversary A with game G consists of running A with oracle access to the game procedures, with the restrictions that A's first call must be to Init (if present), its last call must be to Fin, and it can call these procedures at most once. The output of the execution is the output of Fin. By $\Pr[\mathrm{G}(A)]$ we denote the probability that the execution of game G with adversary A results in this output being the boolean true.

The running time of an adversary playing some game, as referred to in theorem statements, is defined as the worst-case time of the execution of the adversary in the game, so that the time for game procedures to compute responses to oracle queries is included. This convention means that reductions usually preserve adversary running time, up to small terms that we will ignore.

Note that our adversaries have no inputs or outputs. The role of what in other treatments is the adversary input is, for us, played by the response to the Init query, and the role of what in other treatments is the adversary output is, for us, played by the query to Fin.

Different games may have procedures (oracles) with the same names. If we need to disambiguate, we may write G.O to refer to oracle O of game G. In games, integer variables, set variables, boolean variables and string variables are assumed initialized, respectively, to 0, the empty set $\emptyset$, the boolean false and $\perp$.

<u>Security.</u> We generally say a scheme is X-secure (for some definition of X-security usually provided by a game) if the x-advantage (as defined along with the game for X-security) of any "efficient" adversary is "small." In an asymptotic setting, "efficient" would mean polynomial time and "small" would mean negligible in the security parameter, but in our concrete setting, the quantities in quotes, and thus the notion of a scheme being "secure," remain informal, and theorems explicitly bound adversary x-advantage as a function of its resources, for results of more direct value in practice.

<u>PRFs and MACs.</u> Recall that a function family $\mathsf{F} : \mathsf{F.KS} \times \mathsf{F.I} \to \mathsf{F.O}$ is a deterministic algorithm. Here, $\mathsf{F.KS}$ is the key space, $\mathsf{F.I}$ is the input space and $\mathsf{F.O}$ (required to be finite) is the output space. For this syntax, one can define PRF security as well as UF (MAC) security. Game $\mathbf{G}^{\mathrm{uf}}_{\mathsf{F}}$ on the left of Figure 1 defines UF security of function family F. The UF advantage of adversary A_{uf} is $\mathbf{Adv}^{\mathrm{uf}}_{\mathsf{F}}(A_{\mathrm{uf}}) = \Pr[\mathbf{G}^{\mathrm{uf}}_{\mathsf{F}}(A_{\mathrm{uf}})]$. Game $\mathbf{G}^{\mathrm{prf}}_{\mathsf{F}}$ on the right of Figure 1 defines PRF security of function family F. The PRF advantage of adversary A_{prf} is $\mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(A_{\mathrm{prf}}) = 2\Pr[\mathbf{G}^{\mathrm{prf}}_{\mathsf{F}}(A_{\mathrm{prf}})] - 1$. In these games the adversary is required to be domain respecting in the sense that $X \in \mathsf{F.I}$ across all its queries. It is well known that PRF security implies UF security as long as the set $\mathsf{F.O}$ is large [BKR00, GGM86].

<u>CR hashing.</u> A variable output length keyless hash function H takes a string $X \in \{0,1\}^*$ and a desired output length ℓ and returns an ℓ-bit string $\mathsf{H}(X, \ell)$. The cr-advantage $\mathbf{Adv}^{\mathrm{cr}}_{\mathsf{H},\ell}(B)$ of an

```
Game $\mathbf{G}^{\mathrm{uf}}_{\mathsf{F}}$

oracle INIT
  $K \twoheadleftarrow \mathsf{F.KS}$ ; $f \leftarrow \mathsf{F}(K, \cdot)$

oracle FN($X$)
  $\mathrm{UX} \leftarrow \mathrm{UX} \cup \{X\}$ ; **return** $f(X)$

oracle VF($X, Y$)
  If $X \in \mathrm{UX}$ then return $\bot$
  $Y' \leftarrow f(X)$
  If $Y' = Y$ then $\mathsf{win} \leftarrow \mathsf{true}$
  **return** $(Y' = Y)$

oracle FIN
  **return** $\mathsf{win}$
```

```
Game $\mathbf{G}^{\mathrm{prf}}_{\mathsf{F}}$

oracle INIT
  $b \twoheadleftarrow \{0, 1\}$
  **if** $b = 1$ then
      $K \twoheadleftarrow \mathsf{F.KS}$ ; $f \leftarrow \mathsf{F}(K, \cdot)$
  Else $f \twoheadleftarrow \mathrm{FUNC}(\mathsf{F.I}, \mathsf{F.O})$

oracle FN($X$)
  **return** $f(X)$

oracle FIN($b'$)
  **return** $(b = b')$
```

Figure 1: Games defining UF (left) and PRF (right) security for function family F.

op	arg	$\mathsf{Ed}(D, op, arg)$
replace	i, x	$(D[1], \ldots, D[i-1], x, D[i+1], \ldots, D[\mathsf{nb}])$
insert	i, x	$(D[1], \ldots, D[i-1], x, D[i], \ldots, D[\mathsf{nb}])$
delete	i	$(D[1], \ldots, D[i-1], D[i+1], \ldots, D[\mathsf{nb}])$

Figure 2: Examples of edit operations. The first column shows the edit-operation code, the second column shows the arguments and the third shows the resulting, edited document. Here $i \in [1..\mathsf{nb}]$, where $\mathsf{nb} = |D|$, and $x \in \mathsf{BS}$.

adversary B against H for output-length ℓ is defined as the probability that $\mathsf{H}(X_1, \ell) = \mathsf{H}(X_2, \ell)$ and $X_1 \neq X_2$ when $(X_1, X_2) \twoheadleftarrow B(\ell)$. Obviously, this can only be small if ℓ is large. In our usage, ℓ is either large and we assume $\mathsf{H}(\cdot, \ell)$ is collision-resistant, or $\ell = 0$, in which case $\mathsf{H}(X, \ell) = \varepsilon$ for all X.

3 Framework: iFFs, IUF and IPRF

Here we give a framework of definitions and basic results that has two main new elements. The first is that the setting is nonce based, and the second is that, besides defining incremental UF security, we give the first definitions for incremental PRFs.

Nonce-based means algorithms in our iFF syntax are deterministic and may take as input a quantity called a nonce that, for security, is only required to be non-repeating. This generalizes and extends prior schemes, that used randomness or counters, yielding schemes that are more versatile and robust. An added benefit is that this setting allows us to define PRF security and also to define and achieve a strong form of correctness which asks that updated tags coincide with ones that would have been computed from scratch. This in turn allows us to neglect updates in proving security.

We start with document editing systems, then give our syntax of nonce-based incremental function families, then define UF and PRF security, and then give some basic results and relations.

DOCUMENT EDITING SYSTEMS. An incremental cryptographic scheme works for (sits atop) what we call a document editing system. Extending what [BGG95] call text modification operations, a document editing system describes the format of documents and a set of modification (edit) operations on them. It is an entirely non-cryptographic object.

The specification of a document editing system $\mathsf{DE} = (\mathsf{bl}, \mathsf{BS}, \mathsf{OpC}, \mathsf{OpA}, \mathsf{Ed})$ starts with a block length $\mathsf{bl} \geq 1$. The block space is then set to $\mathsf{BS} = \{0,1\}^{\mathsf{bl}}$. Documents (also called messages) are vectors over BS, meaning members of BS^*. There is a set OpC of edit-operation codes, which are names or formal symbols to indicate different edit operations on documents. The actual edit operations are defined and performed by a deterministic edit algorithm $\mathsf{Ed} : \mathsf{BS}^* \times \mathsf{OpC} \times \mathsf{OpA} \to \mathsf{BS}^*$ which takes, as inputs, a document $D \in \mathsf{BS}^*$, an operation code $op \in \mathsf{OpC}$ and arguments $arg \in \mathsf{OpA}$ to return an updated document $\mathsf{Ed}(D, op, arg) \in \mathsf{BS}^*$. If necessary to disambiguate, we write $\mathsf{DE.bl}$, $\mathsf{DE.BS}, \mathsf{DE.OpC}, \mathsf{DE.OpA}, \mathsf{DE.Ed}$ in place of $\mathsf{bl}, \mathsf{BS}, \mathsf{OpC}, \mathsf{OpA}, \mathsf{Ed}$.

Figure 2 shows three common edit operations, namely replace, insert and delete. Their operation codes, denoted respectively by *insert*, *replace* and *delete*, are shown in the first column. By nb we denote the number of blocks in the starting document D. The *insert* operation allows inserting a block $x \in \mathsf{BS}$ at position $i \in [1..\mathsf{nb}]$ in the document D, the *delete* operation allows deletion of the i-th block of D, and the *replace* operation allows replacement of the i-th block of D by the block x. Of course a scheme which is incremental for the *insert* and *delete* operations is also incremental for the *replace* operation (the latter can be implemented by using the former two). Other possible operations are append or prepend of a block to a document. (They are special cases of insert, but some schemes are incremental for append or prepend, but not for insert [BGR95, SY16].)

Incremental function families. We define incremental function families as the syntax that will underly both incremental MACs and PRFs. An incremental function family $\mathsf{iF} = (\mathsf{KS}, \mathsf{NS}, \mathsf{Rng}, \mathsf{Tg}, \mathsf{Up}, \mathsf{Ver})$ for a document editing system $\mathsf{DE} = (\mathsf{bl}, \mathsf{BS}, \mathsf{OpC}, \mathsf{OpA}, \mathsf{Ed})$ specifies numerous algorithms and sets, as follows:

— A key space KS, a nonce space NS, and an output space Rng.

— A tagging algorithm $\mathsf{Tg} : \mathsf{KS} \times \mathsf{NS} \times \{0,1\}^* \times \mathsf{BS}^* \to \mathsf{Rng}$ that takes the key K, a nonce N, document identifier id, and document D to deterministically return a tag $t \leftarrow \mathsf{Tg}(K, N, id, D)$.

— A tag update algorithm $\mathsf{Up} : \mathsf{KS} \times \mathsf{NS} \times \{0,1\}^* \times \mathsf{BS}^* \times \mathsf{OpC} \times \mathsf{OpA} \times \mathsf{Rng} \to \mathsf{Rng}$ that takes the key K, a nonce N, a document identifier id, a document D, an operation code op, the operation arguments arg, and a current tag t to deterministically return an updated tag $t' \leftarrow \mathsf{Up}(K, N, id, D, op, arg, t)$.

— A tag verification algorithm $\mathsf{Ver} : \mathsf{KS} \times \{0,1\}^* \times \mathsf{BS}^* \times \{0,1\}^* \to \{\mathsf{true}, \mathsf{false}\}$ that takes a key K, a document identifier id, a document D and a candidate tag t to deterministically return either true or false.

We say that iF has (fixed) nonce-length nl if $\mathsf{NS} = \{0,1\}^{\mathsf{nl}}$. We require that if $|\mathsf{iF.NS}| = 1$ then $\mathsf{iF.NS} = \{\varepsilon\}$, meaning has nonce-length zero. In this case, we refer to iF as nonce-less.

Update time. For an iFF to have practical value, the update time should be less than the time to compute the tag, on the modified document, from scratch via the tagging algorithm. The actual time for updates varies across schemes, and no formal condition on it is mandated. Ideally this time is proportional only to the time to perform the modification and the number of blocks modified, but in fact even an update time linear in the length of the document can be interesting if it is cheaper than from-scratch tagging. Most of our results are general transforms that preserve update time.

Correctness and strong correctness. Correctness requires that tags generated by the tagging and update algorithms are accepted by the verification algorithm. It is a little more delicate to define than usual because it is required only for tags that arise through legitimate applications of

```
Games $\mathbf{G}^{\mathrm{corr}}_{\mathsf{iF,DE}}$, $\mathbf{G}^{\mathrm{scorr}}_{\mathsf{iF,DE}}$

oracle INIT
$K \twoheadleftarrow \mathsf{KS}$ ; return $K$

oracle TAG$(N, id, D)$
$D_{id} \leftarrow D$ ; $t_{id} \leftarrow \mathsf{Tg}(K, N, id, D_{id})$
if $(\mathsf{Ver}(K, id, D_{id}, t_{id}) = \mathsf{false})$ then win $\leftarrow$ true
return $\perp$

oracle UPD$(N, id, op, arg)$
if $(D_{id} = \perp)$ then return $\perp$
$D'_{id} \leftarrow \mathsf{Ed}(D_{id}, op, arg)$ ; $t'_{id} \leftarrow \mathsf{Up}(K, N, id, D_{id}, op, arg, t_{id})$
if $(\mathsf{Ver}(K, id, D'_{id}, t'_{id}) = \mathsf{false})$ then win $\leftarrow$ true
$t''_{id} \leftarrow \mathsf{Tg}(K, N, id, D'_{id})$  // Game $\mathbf{G}^{\mathrm{scorr}}_{\mathsf{iF,DE}}$
if $t'_{id} \neq t''_{id}$ then win $\leftarrow$ true  // Game $\mathbf{G}^{\mathrm{scorr}}_{\mathsf{iF,DE}}$
return $\perp$

oracle FIN
return win
```

Figure 3: Games defining correctness and strong correctness for an incremental function family iF. If a line indicates a game it means that line is included only in that game.

the tagging or update algorithms. The clearest way we know to formalize this is via a game. We say that $\mathsf{iF} = (\mathsf{KS}, \mathsf{NS}, \mathsf{Rng}, \mathsf{Tg}, \mathsf{Up}, \mathsf{Ver})$ satisfies correctness relative to $\mathsf{DE} = (\mathsf{bl}, \mathsf{BS}, \mathsf{OpC}, \mathsf{OpA}, \mathsf{Ed})$ if $\Pr[\mathbf{G}^{\mathrm{corr}}_{\mathsf{iF,DE}}(A)] = 0$ for all adversaries A (regardless of their running time), where game $\mathbf{G}^{\mathrm{corr}}_{\mathsf{iMA,DE}}$ is shown in Figure 3. (Lines 8, 9 are excluded from this game. The game including them will be discussed next.) As per INIT, the adversary is given the key K. Correctness is required regardless of whether or not nonces repeat. What oracles TAG, UPD return doesn't matter since they are deterministic, so we have them return $\perp$.

We also introduce a notion of *strong correctness*. It asks that tags returned by the update algorithm are the same as if the updated document had instead been tagged directly, from scratch, via the tagging algorithm. Formally we say that $\mathsf{iF} = (\mathsf{KS}, \mathsf{NS}, \mathsf{Rng}, \mathsf{Tg}, \mathsf{Up}, \mathsf{Ver})$ satisfies strong correctness for $\mathsf{DE} = (\mathsf{bl}, \mathsf{BS}, \mathsf{OpC}, \mathsf{OpA}, \mathsf{Ed})$ if $\Pr[\mathbf{G}^{\mathrm{scorr}}_{\mathsf{iF,DE}}(A)] = 0$ for all adversaries A (regardless of their running time), where game $\mathbf{G}^{\mathrm{scorr}}_{\mathsf{iF,DE}}$ is shown in Figure 3. Strong correctness implies correctness, since the lines of the latter game are present in the former. But there are two additional, important dividends. The first is Proposition 1, which says that when strong correctness is present, we can, in evaluating security, ignore the UPD oracle. This will significantly simplify proofs. The second dividend is that strong correctness implies privacy of updates, meaning updated tags do not reveal the modification history of the document [Mic97].

Correctness will be the default, unstated assumption. If strong correctness is assumed or achieved, we will say so explicitly.

IUF SECURITY. We define Incremental Unforgeability (IUF) of an incremental function family $\mathsf{iF} = (\mathsf{KS}, \mathsf{NS}, \mathsf{Rng}, \mathsf{Tg}, \mathsf{Up}, \mathsf{Ver})$, relative to document editing system $\mathsf{DE} = (\mathsf{bl}, \mathsf{BS}, \mathsf{OpC}, \mathsf{OpA}, \mathsf{Ed})$, extending the notion of basic security [BGG94] to our nonce-based setting. Consider game $\mathbf{G}^{\mathrm{iuf}}_{\mathsf{iF,DE}}$ of Figure 4 and let $\mathbf{Adv}^{\mathrm{iuf}}_{\mathsf{iF,DE}}(A) = \Pr[\mathbf{G}^{\mathrm{iuf}}_{\mathsf{iF,DE}}(A)]$ be the iuf-advantage of an adversary A. For any "live" document identity id, the game maintains: (1) D_{id}, the current version of the document associated to id (2) t_{id}, its tag (3) NL_{id}, a set storing nonces used so far for id, and (4) DL_{id}, a set storing

```
Game $\mathbf{G}^{\mathrm{iuf}}_{\mathsf{iF},\mathsf{DE}}$

oracle Init
 1 $K \twoheadleftarrow \mathsf{KS}$

oracle Tag($N, id, D$)
 2 if ($N \in \mathrm{NL}_{id}$ and $|\mathsf{NS}| \neq 1$) then
 3     return $\perp$
 4 $D_{id} \leftarrow D$ ; $t_{id} \leftarrow \mathsf{Tg}(K, N, id, D_{id})$
 5 $\mathrm{NL}_{id} \leftarrow \mathrm{NL}_{id} \cup \{N\}$
 6 $\mathrm{DL}_{id} \leftarrow \mathrm{DL}_{id} \cup \{D_{id}\}$
 7 return $t_{id}$

oracle Upd($N, id, op, arg$)
 8 if $D_{id} = \perp$ then return $\perp$
 9 if ($N \in \mathrm{NL}_{id}$ and $|\mathsf{NS}| \neq 1$) then
10     return $\perp$
11 $t_{id} \leftarrow \mathsf{Up}(K, N, id, D_{id}, op, arg, t_{id})$
12 $D_{id} \leftarrow \mathsf{Ed}(D_{id}, op, arg)$
13 $\mathrm{NL}_{id} \leftarrow \mathrm{NL}_{id} \cup \{N\}$
14 $\mathrm{DL}_{id} \leftarrow \mathrm{DL}_{id} \cup \{D_{id}\}$
15 return $t_{id}$

oracle Vf($id, D, t$)
16 if $D \in \mathrm{DL}_{id}$ then return $\perp$
17 $d \leftarrow \mathsf{Ver}(K, id, D, t)$
18 if $d$ then $\mathsf{win} \leftarrow \mathsf{true}$
19 return $d$

oracle Fin
20 return $\mathsf{win}$
```

```
Game $\mathbf{G}^{\mathrm{iprf}}_{\mathsf{iF},\mathsf{DE}}$

oracle Init
 1 $b \twoheadleftarrow \{0,1\}$ ; $K \twoheadleftarrow \mathsf{KS}$
 2 $f \twoheadleftarrow \mathrm{FUNC}(\mathsf{NS} \times \{0,1\}^* \times \mathsf{BS}^*, \mathsf{Rng})$

oracle Tag($N, id, D$)
 3 if ($N \in \mathrm{NL}_{id}$ and $|\mathsf{NS}| \neq 1$) then
 4     return $\perp$
 5 $D_{id} \leftarrow D$ ; $t^1_{id} \leftarrow \mathsf{Tg}(K, N, id, D_{id})$
 6 $t^0_{id} \leftarrow f(N, id, D_{id})$
 7 $\mathrm{NL}_{id} \leftarrow \mathrm{NL}_{id} \cup \{N\}$
 8 $\mathrm{DL}_{id} \leftarrow \mathrm{DL}_{id} \cup \{D_{id}\}$
 9 return $t^b_{id}$

oracle Upd($N, id, op, arg$)
10 if $D_{id} = \perp$ then return $\perp$
11 if ($N \in \mathrm{NL}_{id}$ and $|\mathsf{NS}| \neq 1$) then
12     return $\perp$
13 $t^1_{id} \leftarrow \mathsf{Up}(K, N, id, D_{id}, op, arg, t^1_{id})$
14 $D_{id} \leftarrow \mathsf{Ed}(D_{id}, op, arg)$
15 $t^0_{id} \leftarrow f(N, id, D_{id})$
16 $\mathrm{NL}_{id} \leftarrow \mathrm{NL}_{id} \cup \{N\}$
17 $\mathrm{DL}_{id} \leftarrow \mathrm{DL}_{id} \cup \{D_{id}\}$
18 return $t^b_{id}$

oracle Vf($id, D, t$)
19 if $D \in \mathrm{DL}_{id}$ then return $\perp$
20 if $b = 1$ then return $\mathsf{Ver}(K, id, D, t)$
21 else return $\mathsf{false}$

oracle Fin($b'$)
22 return ($b' = b$)
```

Figure 4: Games defining IUF (left) and IPRF (right) security of an incremental function family $\mathsf{iF} = (\mathsf{KS}, \mathsf{NS}, \mathsf{Rng}, \mathsf{Tg}, \mathsf{Up}, \mathsf{Ver})$ relative to document editing system $\mathsf{DE} = (\mathsf{bl}, \mathsf{BS}, \mathsf{OpC}, \mathsf{OpA}, \mathsf{Ed})$.

versions of the document with identity id tagged so far. Variable D_{id} starts out $\perp$. An adversary initializes an id via its Tag oracle. The adversary is returned the tag as computed by the tagging algorithms Tg. Now id is live, and the adversary can make Upd queries, and also further Tag queries, with it. An Upd query for id takes a nonce and the description of the update. Document D_{id} is updated (edited) according to the latter, and the tag computed by the Up algorithm is returned to the adversary. Each Tag and Upd query adds entries to the sets $\mathrm{NL}_{id}, \mathrm{DL}_{id}$, thus keeping track of which nonces have been used and which documents have been tagged. Lines 2, 8 disallow nonce reuse for any individual document identity, except if the scheme is nonce-less, in which case this restriction is dropped. The latter is important to capture nonce-less schemes as special case of our framework. It is always permitted to re-use the same nonce across different document identities. To win the adversary must make a query to Vf that is successful (algorithm Ver accepts) and non-trivial (the document was not previously tagged for this identity). Any number of Vf queries are allowed and they may be interleaved arbitrarily with other queries. The adversary cannot make an Upd query with some document identity prior to having initialized that identity via a Tag query with that identity, but can make a Vf query without such initialization.

IPRF SECURITY. We define Incremental Pseudorandom Function (IPRF) security of an incremental function family $\mathsf{iF} = (\mathsf{KS}, \mathsf{NS}, \mathsf{Rng}, \mathsf{Tg}, \mathsf{Up}, \mathsf{Ver})$, relative to document editing system $\mathsf{DE} = (\mathsf{bl}, \mathsf{BS}, \mathsf{OpC}, \mathsf{OpA}, \mathsf{Ed})$. Consider game $\mathbf{G}^{\mathrm{iprf}}_{\mathsf{iF},\mathsf{DE}}$ of Figure 4 and let $\mathbf{Adv}^{\mathrm{iprf}}_{\mathsf{iF},\mathsf{DE}}(A) = 2\Pr[\mathbf{G}^{\mathrm{iprf}}_{\mathsf{iF},\mathsf{DE}}(A)] - 1$ be the iprf-advantage of an adversary A. The game picks a random challenge bit $b \in \{0, 1\}$, samples a key K from the key space, and picks a random function with domain the Cartesian product of the nonce space, the document identity space, and the message space, and range the output space Rng. The game responds to Tag oracle queries depending on the value of the bit b, either generating tags using the tagging algorithm (when $b = 1$) or using the random function f (when $b = 0$). Similarly for responses to Upd, with the document used in the $b = 0$ case at line 13 being the updated one. The Vf oracle verifies as prescribed by iF if $b = 1$ and otherwise returns false. Inclusion of this oracle is important for showing that IPRF security implies IUF security. The adversary ends with a Finalize query that is a bit b' representing its guess as to the challenge bit b, and the game returns true iff this guess is correct.

DROPPING UPDATES. In giving the first security definitions for incremental cryptography, [BGG94, BGG95] are at pains to warn that one must allow the adversary Upd queries, because updated tags may be different from from-scratch ones, and allow forgery. Their definitions and analyses reflect this. Nonetheless, below, we show that, for both IUF and IPRF security, we can assume that adversaries make no queries to their Upd oracles if the function family satisfies strong correctness. This will simplify later proofs. We provide the proof of the following Proposition in Appendix A.

Proposition 1 *Let* $\mathsf{iF} = (\mathsf{KS}, \mathsf{NS}, \mathsf{Rng}, \mathsf{Tg}, \mathsf{Up}, \mathsf{Ver})$ *be an incremental function family satisfying strong correctness relative to document editing system* DE*. Let* $(\mathrm{x}, \mathrm{X}) \in \{(\mathrm{iuf}, \mathrm{IUF}), (\mathrm{iprf}, \mathrm{IPRF})\}$*. Suppose* A *is an adversary against the* X*-security of* iF *making* q_t*,* q_u*,* q_v *queries to its* Tag*,* Upd*,* Vf *oracles. Then we can construct an adversary* A_0*, also against the* X*-security of* iF*, making* $q_t + q_u$ *queries to its* Tag *oracle, zero queries to its* Upd *oracle, and* q_v *queries to its* Vf *oracle such that* $\mathbf{Adv}^{\mathrm{x}}_{\mathsf{iF},\mathsf{DE}}(A_0) = \mathbf{Adv}^{\mathrm{x}}_{\mathsf{iF},\mathsf{DE}}(A)$*. Adversary* A_0 *has the same running time as* A*.*

ADVERSARY CLASSES. If $\mathcal{A}$ is a class (set) of adversaries, then we will say that an incremental function family iF is IPRF[$\mathcal{A}$] (resp. IUF[$\mathcal{A}$]) -secure relative to DE if the iprf (resp. iuf) -advantage of adversaries in $\mathcal{A}$ is small. Considering different classes of adversaries gives us a precise way to discuss special cases of our broad notion. Some of the adversary classes we consider are as follows:

— $\mathcal{A}_{\mathrm{sd}}$ — The class of single-document adversaries. These, across all their queries, use only one document identity, that we can (wlog) assume to be ε. By restricting to this class we capture what prior work called the single-document setting .

— $\mathcal{A}_{\mathrm{md}}$ — The class of multi-document adversaries, this is simply the class of all adversaries. We give it this name just for later emphasis.

— $\mathcal{A}_{\mathrm{1T}}$ — Adversaries in this class make only one Tag query per document identity, as per the definition of [BGG95].

— $\mathcal{A}_{\mathrm{1V}}$ — Adversaries in this class make only one Vf query. This is the case in all prior work. However we know that in general security for adversaries making multiple verification queries is a strictly stronger requirement than security for adversaries making just one such query [BGM04] and, even when a hybrid argument works to show there is no qualitative difference, there is a significant quantitative difference that makes it worth allowing multiple verification queries in the definition.

— $\mathcal{A}_{\rm rn}$ — The class of random-nonce adversaries. These are ones whose choices of nonces, across all queries, are made independently and uniformly at random . This restriction allows us to recover (in our nonce-based setting) the setting of randomized schemes used in all prior work.

Recovering prior notions. We recover some prior notions in the literature using our notation for adversary classes, as follows:

— IUF$[\mathcal{A}_{\rm 1T} \cap \mathcal{A}_{\rm 1V} \cap \mathcal{A}_{\rm rn}]$ — This is the basic security notion of [BGG95]. Their definition allows only a single Tag query per document identifier. (We will see below that this is *strictly* weaker than allowing multiple Tag queries.) They also allow only one Vf query and schemes are randomized, not nonce-based.

— IUF$[\mathcal{A}_{\rm 1T} \cap \mathcal{A}_{\rm 1V} \cap \mathcal{A}_{\rm rn} \cap \mathcal{A}_{\rm sd}]$ — This is the notion of [BGG94], which additionally restricts to the single document setting.

— IUF$[\mathcal{A}_{\rm 1V} \cap \mathcal{A}_{\rm rn}]$ — This is the basic security notion of [Fis97a].

Prior works have considered both basic and tamper-proof security [BGG95, Fis97a]. We clarify that our formulation of IUF here only covers basic, where documents submitted for update cannot be tampered by the adversary.

IPRF security implies IUF security. For regular (no nonces, no incrementality) function families, it is well known that PRF security implies UF security [BKR00, GGM86]. This is useful, since establishing the former obviates establishing the latter, and we would like to preserve it. Nonces, however, bring a challenge here. Indeed, prior work has defined notions of PRF and UF security for nonce-based (not incremental) function families [PS16], but (as they point out), PRF does not imply UF under their definitions. It is to remedy this that our IPRF game, unusually for a PRF definition, included a verification oracle. This allows us to recover the implication. The following says that an iFF that is IPRF-security will also be IUF-secure. The proof is in Appendix B. A useful consequence is that an iFF which is shown to satisfy IPRF security can directly be used to perform incremental message authentication.

Proposition 2 *Let* $\mathsf{iF} = (\mathsf{KS}, \mathsf{NS}, \mathsf{Rng}, \mathsf{Tg}, \mathsf{Up}, \mathsf{Ver})$ *be an incremental function family relative to document editing system* DE. *Let* $A_{\rm uf}$ *be an adversary against the IUF security of* iF *making* q_t, q_u, q_v *queries to its* Tag, Upd, Vf *oracles, respectively. Then we can construct an adversary* $A_{\rm prf}$ *against the IPRF security of* iF *making* q_t, q_u, q_v *queries to its* Tag, Upd, Vf *oracles, respectively, such that* $\mathbf{Adv}^{\rm iuf}_{\mathsf{iF},\mathsf{DE}}(A_{\rm uf}) = \mathbf{Adv}^{\rm iprf}_{\mathsf{iF},\mathsf{DE}}(A_{\rm prf})$. *Adversary* $A_{\rm prf}$ *has about the same running time as* $A_{\rm uf}$.

The result of [BKR00], saying that PRF implies UF for regular function families, requires that the size of the range set $\mathsf{F.O}$ of the function family F be large. This shows up as an added $q_v/|\mathsf{F.O}|$ in the concrete bound. This term is absent in Proposition 2 because our definition of IPRF security has the verification oracle reject in the random case. But in practice, one should still ensure that the range set is large enough to avoid forgery by tag guessing.

4 From Single- to Multi-Document Security

The work that introduced incremental message authentication [BGG94] considered security only in the single-document (SD) setting. In practice one would however expect the user to have many documents that it wants to incrementally process with a single key. This lead [BGG95] to introduce the multi-document (MD) setting. (In our definitions, this is the default.) They, and later [Fis97a], gave and analyzed schemes directly for this setting.

We take a more abstract and high-level view, asking how IPRF and IUF security in the single and multi document settings relate. First we give a separation, showing that there exist schemes secure in the SD setting but insecure in the MD setting. This shows that the latter is a strictly stronger requirement than the former, and motivates making it the target. Next, to reach this target, we give two general "boosting" results: given a scheme secure only in the SD setting, we (efficiently) transform it into a scheme secure in the MD setting. This lets us simplify design and analysis by, in the end, doing this only in the original and simpler single-document setting, relying on our transform to boost security to the MD setting. The difference between the two transforms is in the tightness of the reductions, as we will see below.

Separation result. Let $\mathsf{iF} = (\mathsf{KS}, \mathsf{NS}, \mathsf{Rng}, \mathsf{Tg}, \mathsf{Up}, \mathsf{Ver})$ be an incremental function family that is $\mathrm{IPRF}[\mathcal{A}_{\mathrm{sd}}]$ relative to the document editing system $\mathsf{DE} = (\mathsf{bl}, \mathsf{BS}, \mathsf{OpC}, \mathsf{OpA}, \mathsf{Ed})$. Let $\mathrm{X} \in \{\mathrm{IUF}, \mathrm{PRF}\}$. We modify iF to an incremental function family $\mathsf{iF}' = (\mathsf{KS}, \mathsf{NS}, \mathsf{Rng}, \mathsf{Tg}', \mathsf{Up}', \mathsf{Ver}')$ —the key space, nonce space and range are unchanged, but the algorithms are changed— that has the following properties. (1) iF' remains $\mathrm{X}[\mathcal{A}_{\mathrm{sd}}]$-secure relative to document editing system DE, but (2) iF' is not X-secure, meaning there is an attack showing that it is insecure in the MD setting.

The modification is simple, namely have the algorithms of iF' ignore their id input and run the corresponding iF algorithms with the document identity set to the empty string. In detail, let $\mathsf{Tg}'(K, N, id, D)$ return $\mathsf{Tg}(K, N, \varepsilon, D)$ and let $\mathsf{Up}'(K, N, id, D, op, arg, t)$ return $\mathsf{Up}(K, N, \varepsilon, D, op, arg, t)$. Then (1) is true because queries of an adversary $A \in \mathcal{A}_{\mathrm{sd}}$ already only have $id = \varepsilon$, so nothing is really changed from the perspective of such an adversary. For (2), we give separate attacks IPRF and IUF, beginning with the former. Let id_1, id_2 be some two distinct document identities, N some nonce and D some document. An adversary A can easily obtain an iprf-advantage of $(1 - 1/|\mathsf{Rng}|)$ as follows. Having started with its mandatory $\textsc{Init}$ query, it then makes query $t_1 \leftarrow \textsc{Tag}(N, id_1, D)$, followed by query $t_2 \leftarrow \textsc{Tag}(N, id_2, D)$. (Although the nonce N is the same in both queries, it is for two different identities, so the adversary is nonce-respecting.) Then A calls $\textsc{Fin}(1)$ if $t_1 = t_2$, and $\textsc{Fin}(0)$ otherwise. For the IUF case, an adversary A can obtain an iuf-advantage of 1 as follows. It starts with its mandatory $\textsc{Init}$ query, then makes query $t_1 \leftarrow \textsc{Tag}(N, id_1, D)$, followed by query $d \leftarrow \textsc{Vf}(id_2, D, t_1)$. The adversary then ends with its $\textsc{Fin}$ call.

Boosting results. The above says that a scheme having sd security need not have md. We now show, however, that the sd-secure scheme can be generically transformed into one that is md secure. We give two transforms. The first uses a natural technique, namely to apply a PRF under the key K to the document-id to obtain a sub-key under which the SD scheme may be used. It works, but security degrades by a factor equal to the number of document identities. The second transform gives a tight reduction by a different technique. It uses, as an auxiliary tool, a variable-output-length hash function. The relevant definitions for the auxiliary tools (function families satisfying PRF security in the first case, and CR hash functions in the second) are provided in Appendix **??**.

StM1. Given an incremental function family $\mathsf{iF}_{\mathsf{sd}}$ for a document editing system DE that is IPRF-secure only in the single-document setting, our transform **StM1** uses as auxiliary tool a PRF $\mathsf{F} : \mathsf{F.KS} \times \{0,1\}^* \rightarrow \mathsf{iF}_{\mathsf{sd}}.\mathsf{KS}$ to construct the incremental function family $\mathsf{iF}_{\mathsf{md}} = \mathbf{StM1}[\mathsf{iF}_{\mathsf{sd}}, \mathsf{F}]$ that is defined as follows. Its key space $\mathsf{iF}_{\mathsf{md}}.\mathsf{KS} = \mathsf{F.KS}$ is that of the PRF . Its nonce space $\mathsf{iF}_{\mathsf{md}}.\mathsf{NS} = \mathsf{iF}_{\mathsf{sd}}.\mathsf{NS}$ is that of the given scheme. Its algorithms are shown in the top panel of Figure 5. The following theorem says that if $\mathsf{iF}_{\mathsf{sd}}$ provides security in the single-document setting and F is a secure PRF then $\mathsf{iF}_{\mathsf{md}}$ provides security in the multi-document setting. The proof is in Appendix C.

```
Alg $\mathsf{iF_{md}.Tg}(K, N, id, D)$:
    $K_{id} \leftarrow \mathsf{F}(K, id)$ ; return $\mathsf{iF_{sd}.Tg}(K_{id}, N, \varepsilon, D)$
Alg $\mathsf{iF_{md}.Up}(K, N, id, D, op, arg, t)$:
    $K_{id} \leftarrow \mathsf{F}(K, id)$ ; $t' \leftarrow \mathsf{iF_{sd}.Up}(K_{id}, N, \varepsilon, D, op, arg, t)$ ; return $t'$
Alg $\mathsf{iF_{md}.Ver}(K, id, D, t_{id})$:
    $K_{id} \leftarrow \mathsf{F}(K, id)$ ; return $\mathsf{iMA_{sd}.Ver}(K_{id}, \varepsilon, D, t)$
```

```
Alg $\mathsf{iF_{md}.Tg}(K, N, id, D)$:
    $d \leftarrow \mathsf{H}(id, \mathsf{bl})$ ; $D' \leftarrow \mathrm{Prepend}(d, D)$ ; $N' \leftarrow \mathsf{H}(id \| N, \mathsf{nl})$
    return $\mathsf{iF_{sd}.Tg}(K, N', \varepsilon, D')$
Alg $\mathsf{iF_{md}.Up}(K, N, id, D, op, arg, t)$:
    $d \leftarrow \mathsf{H}(id, \mathsf{bl})$ ; $D' \leftarrow \mathrm{Prepend}(d, D)$ ; $N' \leftarrow \mathsf{H}(id \| N, \mathsf{nl})$
    $t' \leftarrow \mathsf{iF_{sd}.Up}(K, N', \varepsilon, D', op, \mathrm{OpTr}(op, arg), t)$ ; return $t'$
Alg $\mathsf{iF_{md}.Ver}(K, id, D, t_{id})$:
    $d \leftarrow \mathsf{H}(id, \mathsf{bl})$ ; $D' \leftarrow \mathrm{Prepend}(d, D)$ ; return $\mathsf{iF_{sd}.Ver}(K, \varepsilon, D', t)$
```

Figure 5: Algorithms of the incremental function family $\mathsf{iF_{md}} = \mathbf{StM1}[\mathsf{iF_{sd}}, \mathsf{F}]$ (top) and $\mathsf{iF_{md}} = \mathbf{StM2}[\mathsf{iF_{sd}}, \mathsf{H}]$ (bottom).

Theorem 3 *Let* $(\mathrm{x}, \mathrm{X}) \in \{(\mathrm{iuf}, \mathrm{IUF}), (\mathrm{iprf}, \mathrm{IPRF})\}$. *Let* $\mathsf{iF_{sd}}$ *is an incremental function family for the document editing system* DE. *Let* $\mathsf{F} : \mathsf{F.KS} \times \{0,1\}^* \to \mathsf{iF_{sd}.KS}$ *be a PRF. Let* $\mathsf{iF_{md}} = \mathbf{StM1}[\mathsf{iF_{sd}}, \mathsf{F}]$ *be the scheme constructed as above. Suppose we are given an adversary* A_{md} *against the* X *security of* $\mathsf{iF_{md}}$ *making* q_t, q_u, q_v *queries per document to its* $\textsc{Tag}, \textsc{Upd}, \textsc{Vf}$ *oracles, respectively. Let* q *denote the number of distinct document identifiers across all these queries. Then we construct an adversary* $A_{\mathrm{sd}} \in \mathcal{A}_{\mathrm{sd}}$ *against the* X *security of* $\mathsf{iF_{sd}}$ *and an adversary* B *against the PRF security of* F *such that*

$$\mathbf{Adv}^{\mathrm{x}}_{\mathsf{iF_{md}},\mathsf{DE}}(A_{\mathrm{md}}) \leq q \cdot \mathbf{Adv}^{\mathrm{x}}_{\mathsf{iF_{sd}},\mathsf{DE}}(A_{\mathrm{sd}}) + \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(B) \; .$$

Adversary A_{sd} *makes* q_t, q_u, q_v *queries to its* $\textsc{Tag}, \textsc{Upd}, \textsc{Vf}$ *oracles, respectively, all involving just the one document identity* ε*, and its running time is about that of* A_{md}*. The number of distinct* $\textsc{Fn}$ *queries of* B *is* q*, and its running time is about that of* A_{md}*. If* $A_{\mathrm{md}} \in \mathcal{A}_{\mathrm{rn}}$ *then* $A_{\mathrm{sd}} \in \mathcal{A}_{\mathrm{rn}}$.

StM2. Our second transform **StM2** *tightly* reduces the IPRF security of the constructed multi-document function family $\mathsf{iF_{md}}$ to the IPRF security of the given single-document function family $\mathsf{iF_{sd}}$, meaning the factor q in Theorem 3 disappears. This requires that $\mathsf{iF_{sd}}$ satisfies strong correctness and the operations of $\mathsf{DE} = (\mathsf{bl}, \mathsf{BS}, \mathsf{OpC}, \mathsf{OpA}, \mathsf{Ed})$ satisfy a translation condition we will define. We assume $\mathsf{iF_{sd}}$ has a fixed nonce length $\mathsf{nl} \geq 0$. The transform uses as auxiliary tool a variable output length hash function H as defined in Section 2, constructing $\mathsf{iF_{md}} = \mathbf{StM2}[\mathsf{iF_{sd}}, \mathsf{H}]$ as follows. The key space $\mathsf{iF_{md}.KS} = \mathsf{iF_{sd}.KS}$ is that of the starting scheme. The nonce space $\mathsf{iF_{md}.NS}$ is $\{\varepsilon\}$ if $\mathsf{nl} = 0$ and is $\{0,1\}^*$ otherwise. Its algorithms are shown in the bottom panel of Figure 5.

Here, if D is a document and d is a block, we have defined $\mathrm{Prepend}(d, D)$ to be the document $(d, D[1], \ldots, D[|D|])$ obtained by pre-pending d to D as its first block. If $D' \in \mathsf{BS}^*$ is a document then we let $\mathrm{First}(D')$ be its first block and $\mathrm{Rest}(D')$ the rest. Thus, in the $\mathsf{iF_{md}.Tg}$ algorithm in Figure 5, we have $\mathrm{First}(D') = d$ and $\mathrm{Rest}(D') = D$. One must take care, however, that what is the "document" differs for $\mathsf{iF_{md}}$ —for which it is D, to which we want to apply *op*— and for $\mathsf{iF_{sd}}$ —for which it is $D' = \mathrm{Prepend}(d, D)$. So, for example, if the requested operation is *replace* on block 3 of

D, we need to perform the *replace* on block 4 of D'. We call this operation translation, and assume it is possible.

More precisely, we assume there is a function OpTr such that if we compute $X \leftarrow \mathsf{Ed}(D', op, \mathrm{OpTr}(op, arg))$ then $\mathrm{First}(X) = D'[1]$ —the first block is unchanged— and $\mathrm{Rest}(X) = \mathsf{Ed}(\mathrm{Rest}(D'), op, arg)$ —the edit is performed correctly on the remaining document. For example, $\mathrm{OpTr}(\mathit{replace}, (i, v)) = (i+1, v)$, $\mathrm{OpTr}(\mathit{insert}, (i, v)) = (i+1, v)$ and $\mathrm{OpTr}(\mathit{delete}, i) = i+1$, showing that these operations translate. (Not all operations translate, however. For example, *prepend*, the operation of pre-pending a block, does not, because it changes the first block of the document.) Note translation is a property of, and assumption on, the document editing system, not the incremental function families.

The following theorem gives the tight reduction result. The proof is in Appendix D.

Theorem 4 *Let* $(\mathsf{x}, \mathrm{X}) \in \{(\mathrm{iuf}, \mathrm{IUF}), (\mathrm{iprf}, \mathrm{IPRF})\}$. *Let* DE *be a document editing system whose operations are translatable, and let* OpTr *denote the translation function. Let* $\mathsf{iF}_{\mathsf{sd}}$ *be an incremental function family that satisfies strong correctness for* DE. *Let* H *be a variable output length hash function. Let* $\mathsf{iF}_{\mathsf{md}} = \mathbf{StM2}[\mathsf{iF}_{\mathsf{sd}}, \mathsf{H}]$ *be the incremental function family constructed as above. Suppose we are given an adversary* A_{md} *against the* X *security of* $\mathsf{iF}_{\mathsf{md}}$*, making* q_t, q_u, q_v *queries to its* Tag, Upd, Vf *oracles, respectively. Then we construct an adversary* $A_{\mathrm{sd}} \in \mathcal{A}_{\mathrm{sd}}$ *against the* X *security of* $\mathsf{iF}_{\mathsf{sd}}$ *and adversaries* B_1, B_2 *against the CR security of* H *such that*

$$\mathbf{Adv}^{\mathrm{x}}_{\mathsf{iF}_{\mathsf{md}},\mathsf{DE}}(A_{\mathrm{md}}) \leq \mathbf{Adv}^{\mathrm{x}}_{\mathsf{iF}_{\mathsf{sd}},\mathsf{DE}}(A_{\mathrm{sd}}) + \mathbf{Adv}^{\mathrm{cr}}_{\mathsf{H},\mathsf{bl}}(B_1) + \epsilon \ ,$$

where $\epsilon = \mathbf{Adv}^{\mathrm{cr}}_{\mathsf{H},\mathsf{nl}}(B_2)$ *if* $\mathsf{nl} \neq 0$ *and* $\epsilon = 0$ *otherwise. Adversary* A_{sd} *makes* $q_t + q_u$ *queries to its* Tag *oracle,* q_v *queries to its* Vf *oracle and zero queries to its* Upd *oracle, all involving just one document identity* ε. *The running times of the constructed adversaries is about the same as that of* A_{iF}. *If* $A_{\mathrm{md}} \in \mathcal{A}_{\mathrm{rn}}$ *then* $A_{\mathrm{sd}} \in \mathcal{A}_{\mathrm{rn}}$.

5 Incremental Hash-then-Encrypt (iHtE) construction

In this Section we give a construction of an incremental PRF for a document editing system DE from the following ingredients: (1) an incremental hash function iHF for DE (2) a symmetric encryption scheme SE (2) a key-derivation function KDF. The construction is called **iHtE** for "incremental Hash-then-Encrypt," and we write $\mathsf{iF} = \mathbf{iHtE}[\mathsf{iHF}, \mathsf{SE}, \mathsf{KDF}]$. The construction adapts the Carter-Wegman paradigm. We target security in the single-document setting, since the results of Section 4 can be used to boost security to the multi-document setting.

We show that one can obtain many examples of iHF, SE, KDF such that iHF is incremental for *replace* and $\mathbf{iHtE}[\mathsf{iHF}, \mathsf{SE}, \mathsf{KDF}]$ is $\mathrm{IPRF}[\mathcal{A}_{\mathrm{sd}}]$-secure. Thus , we obtain, via **iHtE**, many particular constructions of incremental PRFs for *replace*.

We do not currently know of hash functions iHF that are incremental for operations other than *replace*, but, if these are found, **iHtE** would yield incremental PRFs for these operations. Meanwhile, we will see later how to obtain results for *insert*, *delete* in other ways.

Ingredients. Our **iHtE** construction will need the following three objects:

— An incremental hash function iHF for document editing system $\mathsf{DE} = (\mathsf{bl}, \mathsf{BS}, \mathsf{OpC}, \mathsf{OpA}, \mathsf{Ed})$. It specifies a key space iHF.KS and an output length iHF.ol. Then, via $h \leftarrow \mathsf{iHF.Hsh}(K_{\mathsf{iHF}}, D)$, the deterministic hash computation algorithm determines the iHF.ol-bit hash of a message $D \in \mathsf{BS}^*$. Via $h' \leftarrow \mathsf{iHF.Up}(K_{\mathsf{iHF}}, D, op, arg, h)$, the deterministic hash update algorithm can update a hash value h. The update must be correct: $\mathsf{iHF.Up}(K_{\mathsf{iHF}}, D, op, arg, \mathsf{iHF.Hsh}(K_{\mathsf{iHF}}, D)) = \mathsf{iHF.Hsh}(K_{\mathsf{iHF}}, \mathsf{Ed}(D, op, arg))$.

```
Alg iF.Tg$(K, N, id, D)$:
  $(K_{\mathsf{iHF}}, K_{\mathsf{SE}}) \leftarrow \mathsf{KDF}(K)$ ; $h \leftarrow \mathsf{iHF.Hsh}(K_{\mathsf{iHF}}, D)$ ; $t \leftarrow \mathsf{SE.Enc}(K_{\mathsf{SE}}, N, h)$
  return $t$
Alg iF.Up$(K, N, id, D, op, arg, t)$:
  $(K_{\mathsf{iHF}}, K_{\mathsf{SE}}) \leftarrow \mathsf{KDF}(K)$ ; $h \leftarrow \mathsf{SE.Dec}(K_{\mathsf{SE}}, t)$
  $h' \leftarrow \mathsf{iHF.Up}(K_{\mathsf{iHF}}, D, op, arg, h)$ ; $t' \leftarrow \mathsf{SE.Enc}(K_{\mathsf{SE}}, N, h')$ ; return $t'$
Alg iF.Ver$(K, id, D, t)$:
  $(K_{\mathsf{iHF}}, K_{\mathsf{SE}}) \leftarrow \mathsf{KDF}(K)$ ; $h \leftarrow \mathsf{iHF.Hsh}(K_{\mathsf{iHF}}, D)$ ; $h' \leftarrow \mathsf{SE.Dec}(K_{\mathsf{SE}}, t)$
  return $(h = h')$
```

Figure 6: Algorithms of the incremental function family $\mathsf{iF} = \mathbf{iHtE}[\mathsf{iHF}, \mathsf{SE}, \mathsf{KDF}]$.

— A symmetric encryption scheme SE. This has a key space $\mathsf{SE.KS}$, nonce length $\mathsf{SE.nl}$ and a ciphertext space denoted Rng. We encrypt via $t \leftarrow \mathsf{SE.Enc}(K_{\mathsf{SE}}, N, h)$. Decryption, operating as $h \leftarrow \mathsf{SE.Dec}(K_{\mathsf{SE}}, t)$, does not take the nonce. (This is the NBE2 syntax of [BNT19].) We require of course that decryption reverses encryption: $\mathsf{SE.Dec}(K_{\mathsf{SE}}, \mathsf{SE.Enc}(K_{\mathsf{SE}}, N, h)) = h$.

— A key-derivation function $\mathsf{KDF} : \mathsf{KDF.KS} \to \mathsf{iHF.KS} \times \mathsf{SE.KS}$ that via $(K_{\mathsf{iHF}}, K_{\mathsf{SE}}) \leftarrow \mathsf{KDF}(K)$ maps a base key K into keys for iHF and for SE. The two may be related, or even the same , which is why a KDF is needed.

One important way in which the above differs from, or extends, the classical Carter-Wegman paradigm, is that, in the latter, the object playing the role of SE is not required to be invertible, and in some cases is not invertible. For allowing updates (incrementality), it is not only crucial that SE is invertible (that is, the $\mathsf{SE.Dec}$ operation above exists) but also, as assumed above, that decryption does not require the nonce, meaning the syntax is NBE2. Also, in the usual Carter-Wegman paradigm, keys for hashing and encryption are independent. The above extends this by introducing a key-derivation function, which allows the hashing and encryption keys to be related, as happens in some constructions.

We clarify that we do not, at this point, mandate any particular security conditions for $\mathsf{iHF}, \mathsf{SE}, \mathsf{KDF}$. Different results (eg. Theorem 6) or constructions may assume different things that they will state as necessary.

iHtE construction. We specify incremental function family $\mathsf{iF} = \mathbf{iHtE}[\mathsf{iHF}, \mathsf{SE}, \mathsf{KDF}]$. We set the key space to $\mathsf{iF.KS} = \mathsf{KDF.KS}$. The nonce space is that of SE: we set $\mathsf{iF.NS} = \{0,1\}^{\mathsf{SE.nl}}$. The range is Rng. The tagging, update and verification algorithms are in Figure 6. The idea for updates is to decrypt t via $\mathsf{SE.Dec}$ to recover the iHF-hash h, exploit incrementality of iHF to update h to h', and then re-encrypt h' to get the updated tag t'. It is crucial that decryption is possible and also that decryption does not use the nonce used for encryption. In classical Carter-Wegman, the nonce is placed in the tag, but we cannot do this for (I)PRF security, so we must be able to decrypt without the nonce.

The following proposition, which we prove in Appendix E, says that **iHtE** provides strong correctness. This allows us to exploit Theorem 4 to obtain a IPRF$[\mathcal{A}_{\mathrm{md}}]$-secure scheme without loss of quantitative security.

Proposition 5 *Let* $\mathsf{iF} = \mathbf{iHtE}[\mathsf{iHF}, \mathsf{SE}, \mathsf{KDF}]$ *be the incremental function family built as above, and let* DE *be the underlying document editing system. Then* iF *satisfies strong correctness relative to* DE.

Game $\mathbf{G}^{\mathrm{cau}}_{\mathsf{iHF}}$

```
oracle INIT
1  $K \twoheadleftarrow \mathsf{iHF.KS}$

oracle HASH($D$)
2  $n \leftarrow n+1$ ; $D_n \leftarrow D$
3  $h_n \leftarrow \mathsf{iHF.Hsh}(K, D_n)$
4  return $\perp$

oracle FIN
5  for $1 \le i < j \le n$ do
6      if $((h_i = h_j)$ and $(D_i \neq D_j))$
7          then $\mathsf{win} \leftarrow \mathsf{true}$
8  return $\mathsf{win}$
```

Game $\mathbf{G}^{\mathrm{ae2}}_{\mathsf{SE}}$

```
oracle INIT
1  $b \twoheadleftarrow \{0,1\}$ ; $K \twoheadleftarrow \mathsf{SE.KS}$
2  $f \twoheadleftarrow \mathrm{FUNC}(\mathsf{NS} \times \{0,1\}^{\mathsf{ol}}, \mathsf{Rng})$

oracle ENC($N, h$)
3  if $(N \in \mathrm{NL})$ then return $\perp$
4  $c_0 \leftarrow f(N, h)$ ; $c_1 \leftarrow \mathsf{SE.Enc}(K, N, h)$
5  $\mathrm{NL} \leftarrow \mathrm{NL} \cup \{N\}$ ; $\mathrm{HT}[c_b] \leftarrow h$
6  return $c_b$

oracle DEC($c$)
7  if $(\mathrm{HT}[c] \neq \perp)$ then return $\mathrm{HT}[c]$
8  if $(b = 0)$ then return $\perp$
9  return $\mathsf{SE.Dec}(K, c)$

oracle FIN($b'$)
10 return $(b' = b)$
```

Figure 7: Left: Game defining CAU security for an incremental hash function iHF. Right: Game defining AE2 security for a symmetric encryption scheme SE.

IPRF SECURITY FROM THE **iHtE** CONSTRUCTION. We now proceed to provide a result showing how to achieve IPRF security from specific security notions for the incremental hash function and the symmetric encryption scheme. To do this, we use a notion of computationally almost universal (CAU) security [Bel06] for the incremental hash function, and a notion of AE2 security [BNT19] for the symmetric encryption scheme. We start by defining these notions formally.

CAU SECURITY. Consider the game $\mathbf{G}^{\mathrm{cau}}_{\mathsf{iHF}}$ described on the left in Figure 7, and let $\mathbf{Adv}^{\mathrm{cau}}_{\mathsf{iHF}}(A) = \Pr\left[\mathbf{G}^{\mathrm{cau}}_{\mathsf{iHF}}(A)\right]$ be the cau-advantage of an adversary A. The game starts by sampling a key at random from the key space of the incremental hash function. When the adversary queries a document to the HASH oracle, the game stores the queried document along with the hash for that document, and returns $\perp$ to the adversary (that is, the adversary does not receive any output from the HASH oracle). When the adversary calls the FIN procedure, the game checks for collisions between hashes of distinct documents queried by the adversary. The adversary wins the game iff such a collision exists. Note that the standard definition corresponds to the case where the adversary is restricted to making exactly two HASH queries. We use the more general definition as it simplifies proofs. It follows from the union bound that the advantage of a q-query adversary is at most $\binom{q}{2}$ times the advantage of a 2-query adversary.

AE2 SECURITY. Next consider the game $\mathbf{G}^{\mathrm{ae2}}_{\mathsf{SE}}$ on the right in Figure 7, and let $\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}}(A) = 2\Pr\left[\mathbf{G}^{\mathrm{ae2}}_{\mathsf{SE}}(A)\right] - 1$ be the ae2-advantage of an adversary A. The game picks a random bit $b \in \{0,1\}$, and samples a key at random from the key space $\mathsf{SE.KS}$. The game responds to ENC queries either by performing the real encryption (when $b = 1$) or by picking an element of the ciphertext range Rng at random (when $b = 0$). Note (line 2) that nonce re-use is not allowed during ENC queries. The game responds to DEC queries either by performing real decryption (when $b = 1$), or by always returning $\perp$ (when $b = 0$), unless the output of an ENC query is asked, in which case the stored decryption value corresponding to the queried ciphertext is returned. The adversary wins if it is able to correctly guess the bit b in its mandatory FIN query.

Message Authentication Scheme (M)	**iFF**	**Security**	
		IUF	**IPRF**
PMAC1 [Rog04]	$\mathsf{iF_M}$	Yes	Yes
PMAC [BR02]	$\mathsf{iF_M}$	Yes	Yes
XORMAC [BGR95]	$\mathsf{iF_{M\text{-}1}}$	Yes	No
	$\mathsf{iF_{M\text{-}2}}$	Yes	Yes
GMAC [MV04]	$\mathsf{iF_{M\text{-}1}}$	Yes	No
	$\mathsf{iF_{M\text{-}2}}$	Yes	Yes
Poly1305-AES [Ber05]	$\mathsf{iF_{M\text{-}1}}$	Yes	No
	$\mathsf{iF_{M\text{-}2}}$	Yes	Yes
PWC [PS16]	$\mathsf{iF_{M\text{-}1}}$	Yes	No
	$\mathsf{iF_{M\text{-}2}}$	Yes	Yes
PMAC_Plus [Yas11]	$\mathsf{iF_M}$	Yes	Yes
ZMAC [IMPS17]	$\mathsf{iF_M}$	Yes	Yes

Figure 8: Table summarizing the constructed iFFs for different instantiations.

Given the above definitions, the following shows that **iHtE** provides IPRF security in the sd setting. The proof of this is deferred to Appendix F.

Theorem 6 *Let hash function* iHF *and symmetric encryption scheme* SE *be as above. Let* KDF *return* $(K_{\mathsf{iHF}}, K_{\mathsf{SE}})$ *with the two keys chosen independently and at random. Let* $\mathsf{iF} = \mathbf{iHtE}[\mathsf{iHF}, \mathsf{SE}, \mathsf{KDF}]$ *be the incremental function family built as above, and let* DE *be the underlying document editing system. Suppose we are given an adversary* $A_{\mathsf{iF}} \in \mathcal{A}_{\mathrm{sd}}$ *against the IPRF security of* iF*, making* q_t, q_u, q_v *queries to its* Tag, Upd, Vf *oracles, respectively, and let* $q = q_t + q_u + q_v$*. Then we construct an adversary* A_{H} *against the CAU security of* iHF *and an adversary* A_{SE} *against the AE2 security of* SE *such that*

$$\mathbf{Adv}^{\mathrm{iprf}}_{\mathsf{iF},\mathsf{DE}}(A_{\mathsf{iF}}) \leq 2 \cdot \mathbf{Adv}^{\mathrm{cau}}_{\mathsf{iHF}}(A_{\mathsf{H}}) + 2 \cdot \mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}}(A_{\mathsf{SE}}) \ .$$

Adversary A_{H} *makes* q *queries to its* Hash *oracle and adversary* A_{SE} *makes* $q_t + q_u$ *queries to its* Enc *oracle, and* q_v *queries to its* Dec *oracle. The running times of the constructed adversaries is about the same as that of* A_{iF}*.*

6 Instantiations

Armed with the tools from the previous sections, we turn to finding specific IUF/IPRF-secure iFFs with efficient update algorithms. In the following, incrementality is always for the *replace* operation, and security always means in the single document setting.

The first examples of incremental message authentication schemes, given in [BGG95], were the XORMACs of [BGR95], but these are not nonce based and need to be recast in our syntax before we can even talk of security. We begin by identifying, underlying these schemes, a hash function iHF

that is incremental. Thence, we directly obtain an iFF $\mathsf{iF}_{\mathsf{XORMAC\text{-}1}}$ that is the natural nonce-based extension of the original schemes. This is IUF-secure, but (due to inclusion of the nonce in the tag), not IPRF-secure. To get an IPRF-secure iFF (viz. $\mathsf{iF}_{\mathsf{XORMAC\text{-}2}}$), we use **iHtE** and Theorem 6 in conjunction with the identified iHF.

Many message authentication schemes have appeared subsequent to [BGG94]. We divide them into groups. The first group is ones (eg. PMAC [BR02], $\mathsf{PMAC1}$ [Rog04], $\mathsf{Poly1305\text{-}AES}$ [Ber05]) that come with explicit claims of incrementality by their authors. The works however stop short of establishing that the schemes meet any formal notion of incremental security. We revisit these schemes to fill these gaps and/or go beyond to obtain IUF/IPRF-secure iFFs. We use, broadly, the same approach as outlined above for XORMACs. First, we identify, underlying the schemes, a hash function iHF that is incremental. Thence, we attempt first to recover the natural expression of the original scheme as an iFF. In some cases, existing results can be used to show this iFF is IPRF secure, but this is rare. (The only examples we have are $\mathsf{PMAC1}$ and PMAC.) In other cases (eg. $\mathsf{Poly1305\text{-}AES}$), this iFF is IUF secure but not IPRF secure. (Because of inclusion of the nonce in the tag.) In this case, we go back to the incremental hash function iHF, and use **iHtE** and Theorem 6 to obtain an IPRF-secure iFF.

In the second group are existing message authentication schemes that are not incremental but underlying which we can nevertheless identify an incremental hash function iHF. (These schemes, explicitly or implicitly, use the CW paradigm. Incrementality fails due to the masking step not being invertible.) In these cases, we again use **iHtE** and Theorem 6 in conjunction with the identified iHF to obtain IPRFs. Figure 8 summarizes our analysis of different works . We now provide details of our analysis. In the following, let $\mathsf{DE} = (\mathsf{bl}, \mathsf{BS}, \mathsf{OpC}, \mathsf{OpA}, \mathsf{Ed})$ be the underlying document editing system. Recall that the only operation supported is *replace*.

$\underline{\mathsf{iF}_{\mathsf{XORMAC\text{-}1}}, \mathsf{iF}_{\mathsf{XORMAC\text{-}2}}.}$ The original XORMAC [BGR95] comes in two forms, randomized and counter based. We generalize these to create nonce-based iFFs.

Let $E : \{0,1\}^k \times \mathcal{T} \times \{0,1\}^{\mathsf{bl}} \to \{0,1\}^{\mathsf{bl}}$ be a tweakable blockcipher [LRW11] with tweak space $\mathcal{T} = \mathbb{N}$. Define $\mathsf{iHF} : \{0,1\}^k \times \mathsf{BS}^* \to \{0,1\}^{\mathsf{bl}}$ to take input K, D and return $h \leftarrow E(K, 1, D[1]) \oplus \cdots \oplus E(K, m, D[m])$, where $m \leftarrow |D|$ is the number of blocks in D. This hash function is easily seen to be incremental for *replace* [BGG95], and we denote the update algorithm by $\mathsf{iHF.Up}$.

Define $\mathsf{iF}_{\mathsf{XORMAC\text{-}1}}$ as follows. The nonce space is $\{0,1\}^{\mathsf{bl}}$. The tagging algorithm Tg takes K, N, ε, D —recall we are in the sd setting, so the document id is ε— and returns tag (N, c) where $c \leftarrow E(K, 0, N) \oplus \mathsf{iHF}(K, D)$. The update algorithm Up takes $K, N', \varepsilon, D, \textit{replace}, \textit{arg}, (N, c)$ and returns (N', c') obtained by setting $h' \leftarrow \mathsf{iHF.Up}(K, D, \textit{replace}, \textit{arg}, E(K, 0, N) \oplus c)$ and $c' \leftarrow E(K, 0, N') \oplus h'$. The verification algorithm Ver takes $K, \varepsilon, D, (N, c)$ and returns true iff $\mathsf{Tg}(K, N, \varepsilon, D) = (N, c)$. We can see that the iFF is strongly correct, whence the proofs of [BGR95] (for their randomized and counter-based message authentication schemes) extend to our nonce-based setting to show that $\mathsf{iF}_{\mathsf{XORMAC\text{-}1}}$ is IUF secure.

The inclusion of the nonce N in the tag $t = (N, c)$ in $\mathsf{iF}_{\mathsf{XORMAC\text{-}1}}$, however, precludes its being IPRF secure. To obtain an iFF that is IPRF secure, we can simply use the underlying incremental hash function iHF in **iHtE**. Namely, we pick some suitable symmetric encryption SE —there are many choices— and apply Theorem 6 to get an IPRF $\mathsf{iF}_{\mathsf{XORMAC\text{-}2}} = \mathbf{iHtE}[\mathsf{iHF}, \mathsf{SE}, \mathsf{KDF}]$ that is incremental for DE.

$\underline{\mathsf{iF}_{\mathsf{PMAC1}}, \mathsf{iF}_{\mathsf{PMAC}}.}$ We show how to cast $\mathsf{PMAC1}$ [Rog04] as an iFF, that we denote $\mathsf{iF}_{\mathsf{PMAC1}}$, that is IPRF secure and incremental for DE. PMAC [BR02] can be treated analogously. The versions of $\mathsf{PMAC1}, \mathsf{PMAC}$ we consider are without tag truncation.

Assume bl is even. Let $E : \{0,1\}^k \times \mathcal{T} \times \{0,1\}^{\mathsf{bl}} \to \{0,1\}^{\mathsf{bl}}$ be a tweakable blockcipher with tweak space $\mathcal{T} = \{0,1\}^{\mathsf{bl}} \times [1..2^{\mathsf{bl}/2}] \times \{2,3,4\} \to \{0,1\}^{\mathsf{bl}}$. Define $\mathsf{iHF} : \{0,1\}^k \times \mathsf{BS}^* \to \{0,1\}^{\mathsf{bl}}$ to take input K, D and return $h \leftarrow E(K, (0^{\mathsf{bl}}, 1, 2), D[1]) \oplus \cdots \oplus E(K, (0^{\mathsf{bl}}, m-1, 2), D[m-1]) \oplus D[m]$, where $m \leftarrow |D|$ is the number of blocks in D. This hash function is easily seen to be incremental for *replace* [Rog04], and we denote the update algorithm by $\mathsf{iHF.Up}$.

The nonce space of $\mathsf{iF}_{\mathsf{PMAC1}}$ is $\{\varepsilon\}$. The tagging algorithm Tg takes $K, \varepsilon, \varepsilon, D$ and returns tag $t \leftarrow E(K, (0^n, |D|, 3), \mathsf{iHF}(K, D))$. The update algorithm Up takes $K, \varepsilon, \varepsilon, D, \mathit{replace}, \mathit{arg}$, t and returns t' obtained by setting $h' \leftarrow \mathsf{iHF.Up}(K, D, \mathit{replace}, \mathit{arg}, E^{-1}(K, (0^n, |D|, 3), t))$ and $t' \leftarrow E(K, (0^n, |D|, 3), h')$. The verification algorithm Ver takes K, ε, D, t and returns true iff $\mathsf{Tg}(K, \varepsilon, \varepsilon, D) = t$. Then $\mathsf{iF}_{\mathsf{PMAC1}}$ is just $\mathsf{PMAC1}$ recast in our syntax, and the proof of PRF security of the latter from [Rog04], along with the strong correctness of the former and Proposition 1, imply IPRF security of the former. (Note that $\mathsf{iF}_{\mathsf{PMAC1}}$ is nonce-less, so here we are exploiting the fact that our definitions dealt with this as a special case in which the nonce non-repetition requirement was dropped.) We do not need **iHtE** in this case.

$\underline{\mathsf{iF}_{\mathsf{GMAC\text{-}1}},\ \mathsf{iF}_{\mathsf{GMAC\text{-}2}}.}$ The GMAC scheme [MV04] is nonce-based, and is claimed to be incremental by the authors. Let $E : \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$ be a blockcipher, and let $a \cdot b$ denote multiplication in the field $\mathrm{GF}(2^{128})$. We set $\mathsf{bl} \leftarrow 128$ to be the block length. Define $\mathsf{iHF} : \{0,1\}^{128} \times \mathsf{BS}^* \to \{0,1\}^{128}$ to take input K, D and return $h \leftarrow D[1] \cdot H^{m+1} \oplus D[2] \cdot H^m \oplus \cdots \oplus D[m] \cdot H^2 \oplus \langle m \rangle \cdot H$, where $H \leftarrow E(K, 0^{128})$, $m \leftarrow |D|$ is the number of blocks in D, and $\langle m \rangle$ is the representation of $m \mod 2^{128}$ as an m-bit string. This hash function is incremental for *replace*, and we denote the update algorithm by $\mathsf{iHF.Up}$. Note that this hash function is denoted as the GHASH function in [MV04].

We define $\mathsf{iF}_{\mathsf{GMAC1}}$ as follows. The key space is $\{0,1\}^{128}$ and the nonce space is $\{0,1\}^{96}$. The tagging algorithm takes K, N, ε, D and returns a tag (N, c) with $c \leftarrow E(K, N||0^{31}1) \oplus \mathsf{iHF}(K, D)$. The update algorithm Up takes $K, N', \varepsilon, D, \mathit{replace}, \mathit{arg}, (N, c)$ and returns (N', c') obtained by setting $h' \leftarrow \mathsf{iHF.Up}(K, D, \mathit{replace}, \mathit{arg}, E(K, N||0^{31}1) \oplus c)$ and $c' \leftarrow E(K, N'||0^{31}1) \oplus h'$. The verification algorithm Ver takes $K, \varepsilon, D, (N, c)$ and returns true iff $\mathsf{Tg}(K, N, \varepsilon, D) = (N, c)$. It is easy to show that $\mathsf{iF}_{\mathsf{GMAC1}}$ satisfies strong correctness, after which we can use Proposition 1 and the security proofs of [MV04] to show that the scheme is IUF secure.

Note however, that the inclusion of the nonce in the tag does not allow for IPRF security. To obtain IPRF security, we instead use the **iHtE** transform with the underlying incremental hash function defined above, and a suitable symmetric encryption scheme SE, and then use Theorem 6 to get an IPRF $\mathsf{iF}_{\mathsf{GMAC2}} = \textbf{iHtE}[\mathsf{iHF}, \mathsf{SE}, \mathsf{KDF}]$ that is incremental for DE.

$\underline{\mathsf{iF}_{\mathsf{Poly1305\text{-}AES\text{-}1}}, \mathsf{iF}_{\mathsf{Poly1305\text{-}AES\text{-}2}}.}$ The Poly1305-AES scheme [Ber05] is nonce-based, and is claimed to be incremental by the authors. Let $E : \{0,1\}^{128} \times \{0,1\}^{128} \to \{0,1\}^{128}$ be a blockcipher (which is set to be AES for concreteness by the authors). Let $a \cdot b$ and $a + b$ denote multiplication and addition modulo $p = 2^{130} - 5$. We set $\mathsf{bl} \leftarrow 128$ to be the block length. Define $\mathsf{iHF} : \{0,1\}^{106} \times \mathsf{BS}^* \to \{0,1\}^{128}$ to take input K, D and do the following. It parses the key K as $r_0||r_1||r_2||r_3 \leftarrow K$ where $r_0 \in \{0,1\}^{28}$, $r_1, r_2, r_3 \in \{0,1\}^{26}$. It then computes $r \leftarrow r_0 + r_1 \cdot 2^{34} + r_2 \cdot 2^{66} + r_3 \cdot 2^{98}$, and returns a hash value $h \leftarrow ((D[1]||0^71) \cdot r^m + (D[2]||0^71) \cdot r^{m-1} + \ldots + (D[m]||0^71) \cdot r \mod p) \mod 2^{128}$. Note that at each point, bitstrings are decoded as integers in little-endian representation. This hash function is incremental for *replace*, and we denote the update algorithm by $\mathsf{iHF.Up}$.

We define $\mathsf{iF}_{\mathsf{Poly1305\text{-}AES1}}$ as follows. The key space is $\{0,1\}^{234}$ and the nonce space is $\{0,1\}^{128}$. The tagging algorithm takes K, N, ε, D, splits the key into K_1, K_{iHF}, and returns a tag (N, c) with $c \leftarrow (E(K_1, N) + \mathsf{iHF}(K_{\mathsf{iHF}}, D)) \mod 2^{128}$. The update algorithm Up takes $K, N', \varepsilon, D, \mathit{replace}$, $\mathit{arg}, (N, c)$ and returns (N', c') obtained by setting $h' \leftarrow \mathsf{iHF.Up}(K, D, \mathit{replace}, \mathit{arg}, (E(K_1, N) + c) \mod 2^{128})$ and $c' \leftarrow (E(K_1, N') + h') \mod 2^{128}$. The verification algorithm Ver takes $K, \varepsilon, D, (N, c)$ and returns true iff $\mathsf{Tg}(K, N, \varepsilon, D) = (N, c)$. It is easy to show that $\mathsf{iF}_{\mathsf{Poly1305\text{-}AES1}}$ satisfies strong correctness, after which we can use Proposition 1 and the security proofs of [Ber05] to show that the scheme is IUF secure.

The inclusion of the nonce in the tag does not allow for IPRF security. To obtain IPRF security, we instead use the **iHtE** transform with the underlying incremental hash function defined above, and a suitable symmetric encryption scheme SE, and then use Theorem 6 to get an IPRF $\mathsf{iF}_{\mathsf{Poly1305\text{-}AES2}} = \mathbf{iHtE}[\mathsf{iHF}, \mathsf{SE}, \mathsf{KDF}]$ that is incremental for DE.

$\underline{\mathsf{iF}_{\mathsf{PWC\text{-}1}}, \mathsf{iF}_{\mathsf{PWC\text{-}2}}.}$ The PWC scheme [PS16] is a nonce-based scheme. We start by extracting a hash function iHF from the scheme description. Let $E : \{0,1\}^k \times \mathcal{T} \times \{0,1\}^{\mathsf{bl}} \to \{0,1\}^{\mathsf{bl}}$ be a tweakable blockcipher with tweak space $\mathcal{T} = \{2,3\} \times [1..L]$. L here denotes the maximum number of blocks that can be in a single document. We define $\mathsf{iHF} : \{0,1\}^k \times \mathsf{BS}^* \to \{0,1\}^{\mathsf{bl}}$ to take as input K, D and return $h \leftarrow E(K,(3,1),D[1]) \oplus \ldots \oplus E(K,(3,m),D[m])$, where $m \leftarrow |D|$ is the number of blocks in the document. This hash function is incremental for *replace*, and we denote the update algorithm by $\mathsf{iHF.Up}$.

We define $\mathsf{iF}_{\mathsf{PWC1}}$ as follows. The nonce space is $\{0,1\}^{\mathsf{bl}}$. The tagging algorithm takes K, N, ε, D, and returns a tag (N,c) with $c \leftarrow E(K,(2,0),N) \oplus E(K,(2,1),N) \oplus \mathsf{iHF}(K_{\mathsf{iHF}}, D)$. The update algorithm Up takes $K, N', \varepsilon, D, \mathit{replace}, \mathit{arg}, (N,c)$ and returns (N',c') obtained by setting $h' \leftarrow \mathsf{iHF.Up}(K, D, \mathit{replace}, \mathit{arg}, (E(K,(2,0),N) \oplus E(K,(2,1),N) \oplus c))$ and $c' \leftarrow E(K,(2,0),N') \oplus E(K,(2,1),N') \oplus h'$. The verification algorithm Ver takes $K, \varepsilon, D, (N,c)$ and returns true iff $\mathsf{Tg}(K,N,\varepsilon,D) = (N,c)$. This iFF satisfies strong correctness, and this allows us to use Proposition 1 and extend the proofs of [PS16] to show the IUF security of this iFF.

Again, the inclusion of the nonce in the tag does not allow for IPRF security. To obtain IPRF security, we instead use the **iHtE** transform with the underlying incremental hash function defined above, and a suitable symmetric encryption scheme SE, and then use Theorem 6 to get an IPRF $\mathsf{iF}_{\mathsf{PWC2}} = \mathbf{iHtE}[\mathsf{iHF}, \mathsf{SE}, \mathsf{KDF}]$ that is incremental for DE.

$\underline{\mathsf{iF}_{\mathsf{PMAC\_Plus}}.}$ We next study the $\mathsf{PMAC\_Plus}$ scheme [Yas11]. The original scheme is not incremental due to the non-invertible nature of the masking. However, we identify an incremental hash function present within the construction. This allows us to then provide a nonce-based iFF. There are two versions of the scheme, one which uses three different keys, and another which uses the same key, and uses different tweaks of a tweakable blockcipher. However, the two schemes differ only in the masking phase, and therefore we get the same iFF in both cases.

Let $E : \{0,1\}^k \times \{0,1\}^{\mathsf{bl}} \to \{0,1\}^{\mathsf{bl}}$ be a blockcipher, and let $a \cdot x$ denote multiplication in the field $\mathrm{GF}(2^{\mathsf{bl}})$. We define the incremental hash function $\mathsf{iHF} : \{0,1\}^k \times \mathsf{BS}^* \to \{0,1\}^{2\mathsf{bl}}$ to take as input K, D and return $h \leftarrow (Y_1 \oplus Y_2 \oplus \cdots \oplus Y_m) \,||\, (Y_1 \oplus 2 \cdot Y_2 \oplus \cdots \oplus 2^{m-1} \cdot Y_m)$, where $m \leftarrow |D|$ is the number of blocks in D, and $Y_i \leftarrow E(K, D[i] \oplus 2^i \cdot \Delta_0 \oplus 2^{2i} \cdot \Delta_1)$ is defined for each $i \in [1..m]$, with $\Delta_0 \leftarrow E(K, 0^{\mathsf{bl}})$ and $\Delta_1 \leftarrow E(K, 0^{\mathsf{bl}-1}1)$. This hash function is incremental for *replace*, and we denote the update algorithm by $\mathsf{iHF.Up}$.

Now, to obtain an iFF that is IPRF secure, we can use the above defined incremental hash function iHF in **iHtE**, along with some suitable symmetric encryption scheme, as in Theorem 6. We denote this iFF by $\mathsf{iF}_{\mathsf{PMAC\_Plus}}$.

The constructions $\mathsf{PMAC\_TBC3k}$ and $\mathsf{PMAC\_TBC1k}$ from [Nai15] can also be studied in the same fashion, with only minor changes in the definition of the incremental hash function.

$\underline{\mathsf{iF}_{\mathsf{ZMAC}}.}$ The universal hash function used in the construction of the ZMAC is called ZHASH [IMPS17]. Note that the ZMAC does not use nonces, and that the masking (that is, the finalization procedure ZFIN in [IMPS17]) is non-invertible. As a result, the original scheme is not incremental. We cast ZHASH as an incremental hash function – for simplicity we focus on the case where the size of the tweak is not greater than the size of the blockcipher input (i.e. the $t \leq n$ case in [IMPS17]) – the other case can be studied analogously.

Let $E : \mathcal{K} \times \mathcal{T} \times \{0,1\}^n \to \{0,1\}^n$ be a tweakable blockcipher with tweak space $\mathcal{T} = \{0,1\}^t \times [0..9]$. Let the operation $a \cdot b$ denote multiplication in the field $\mathrm{GF}(2^n)$. The block length is $\mathsf{bl} = n + t$. Define $\mathsf{iHF} : \mathcal{K} \times \mathsf{BS} \to \{0,1\}^{\mathsf{bl}}$ to take as input K, D, split each document block into n and t bit parts as $D_\ell[i] || D_r[i] \leftarrow D[i]$ for $i \in [1..m]$ where $m \leftarrow |D|$ is the number of blocks in the document (where $D_\ell[i] \in \{0,1\}^n$ and $D_r[i] \in \{0,1\}^t$), and return $h \leftarrow (2^m \cdot C_\ell[1] \oplus \ldots \oplus 2 \cdot C_\ell[m]\ ,\ C_r[1] \oplus \ldots \oplus C_r[m])$, where for each $i \in [1..m]$, $C_\ell[i] \leftarrow E(K, (2^{i-1}L_r \oplus D_r[i], 8), 2^{i-1}L_\ell \oplus D_\ell[i])$ and $C_r[i] \leftarrow C_\ell[i] \oplus D_r[i]$. In the above, $L_\ell \leftarrow E(K, (0^t, 9), 0^n)$ and $L_r \leftarrow E(K, (0^{t-1}1, 9), 0^n)$. It is easy to see that this hash function is incremental for the *replace* operation, and we denote the update algorithm by $\mathsf{iHF.Up}$.

In order to obtain an iFF that is IPRF secure, we can use the above defined incremental hash function iHF in **iHtE**, along with some suitable symmetric encryption scheme, as in Theorem 6. We denote this iFF by $\mathsf{iF_{ZMAC}}$.

Acknowledgments

We thank Damien Vergnaud for discussions and comments in early stages of this work. We thank the IndoCrypt 2020 reviewers for their comments.

References

[ACJ17] Prabhanjan Ananth, Aloni Cohen, and Abhishek Jain. Cryptography with updates. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *EUROCRYPT 2017, Part II*, volume 10211 of *LNCS*, pages 445–472. Springer, Heidelberg, April / May 2017. (Cited on page 6.)

[BCK96] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. Keying hash functions for message authentication. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 1–15. Springer, Heidelberg, August 1996. (Cited on page 3.)

[Bel06] Mihir Bellare. New proofs for NMAC and HMAC: Security without collision-resistance. In Cynthia Dwork, editor, *CRYPTO 2006*, volume 4117 of *LNCS*, pages 602–619. Springer, Heidelberg, August 2006. (Cited on page 5, 18.)

[Ber05] Daniel J. Bernstein. The poly1305-AES message-authentication code. In Henri Gilbert and Helena Handschuh, editors, *FSE 2005*, volume 3557 of *LNCS*, pages 32–49. Springer, Heidelberg, February 2005. (Cited on page 5, 6, 19, 20, 21, 22.)

[BGG94] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography: The case of hashing and signing. In Yvo Desmedt, editor, *CRYPTO'94*, volume 839 of *LNCS*, pages 216–233. Springer, Heidelberg, August 1994. (Cited on page 3, 5, 6, 10, 12, 13, 20.)

[BGG95] Mihir Bellare, Oded Goldreich, and Shafi Goldwasser. Incremental cryptography and application to virus protection. In *27th ACM STOC*, pages 45–56. ACM Press, May / June 1995. (Cited on page 3, 5, 6, 9, 12, 13, 19, 20.)

[BGM04] Mihir Bellare, Oded Goldreich, and Anton Mityagin. The power of verification queries in message authentication and authenticated encryption. Cryptology ePrint Archive, Report 2004/309, 2004. http://eprint.iacr.org/2004/309. (Cited on page 12.)

[BGR95] Mihir Bellare, Roch Guérin, and Phillip Rogaway. XOR MACs: New methods for message authentication using finite pseudorandom functions. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 15–28. Springer, Heidelberg, August 1995. (Cited on page 5, 9, 19, 20.)

[BHK$^+$99] John Black, Shai Halevi, Hugo Krawczyk, Ted Krovetz, and Phillip Rogaway. UMAC: Fast and secure message authentication. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 216–233. Springer, Heidelberg, August 1999. (Cited on page 3.)

[BKR00] Mihir Bellare, Joe Kilian, and Phillip Rogaway. The security of the cipher block chaining message authentication code. *Journal of Computer and System Sciences*, 61(3):362–399, 2000. (Cited on page 4, 7, 13.)

[BKY02] Enrico Buonanno, Jonathan Katz, and Moti Yung. Incremental unforgeable encryption. In Mitsuru Matsui, editor, *FSE 2001*, volume 2355 of *LNCS*, pages 109–124. Springer, Heidelberg, April 2002. (Cited on page 3, 6.)

[BM97] Mihir Bellare and Daniele Micciancio. A new paradigm for collision-free hashing: Incrementality at reduced cost. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 163–192. Springer, Heidelberg, May 1997. (Cited on page 3, 6.)

[BNT19] Mihir Bellare, Ruth Ng, and Björn Tackmann. Nonces are noticed: AEAD revisited. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 235–265. Springer, Heidelberg, August 2019. (Cited on page 5, 17, 18.)

[BR02] John Black and Phillip Rogaway. A block-cipher mode of operation for parallelizable message authentication. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, volume 2332 of *LNCS*, pages 384–397. Springer, Heidelberg, April / May 2002. (Cited on page 6, 19, 20.)

[BR06] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 409–426. Springer, Heidelberg, May / June 2006. (Cited on page 7, 26, 32.)

[Dwo05] Morris Dworkin. Recommendation for block cipher modes of operation: the CMAC mode for authentication. NIST Special Publication 800-38B, 2005. (Cited on page 3.)

[Fis97a] Marc Fischlin. Incremental cryptography and memory checkers. In Walter Fumy, editor, *EUROCRYPT'97*, volume 1233 of *LNCS*, pages 293–408. Springer, Heidelberg, May 1997. (Cited on page 3, 6, 13, 14.)

[Fis97b] Marc Fischlin. Lower bounds for the signature size of incremental schemes. In *38th FOCS*, pages 438–447. IEEE Computer Society Press, October 1997. (Cited on page 6.)

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986. (Cited on page 3, 4, 7, 13.)

[GP17] Sanjam Garg and Omkant Pandey. Incremental program obfuscation. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part II*, volume 10402 of *LNCS*, pages 193–223. Springer, Heidelberg, August 2017. (Cited on page 3, 6.)

[HK97] Shai Halevi and Hugo Krawczyk. MMH: Software message authentication in the Gbit/second rates. In Eli Biham, editor, *FSE'97*, volume 1267 of *LNCS*, pages 172–189. Springer, Heidelberg, January 1997. (Cited on page 3.)

[IMPS17] Tetsu Iwata, Kazuhiko Minematsu, Thomas Peyrin, and Yannick Seurin. ZMAC: A fast tweakable block cipher mode for highly secure message authentication. In Jonathan Katz and Hovav Shacham, editors, *CRYPTO 2017, Part III*, volume 10403 of *LNCS*, pages 34–65. Springer, Heidelberg, August 2017. (Cited on page 6, 19, 22.)

[KV19] Louiza Khati and Damien Vergnaud. Analysis and improvement of an authentication scheme in incremental cryptography. In Carlos Cid and Michael J. Jacobson Jr:, editors, *SAC 2018*, volume 11349 of *LNCS*, pages 50–70. Springer, Heidelberg, August 2019. (Cited on page 3, 6.)

[LRW11] Moses Liskov, Ronald L. Rivest, and David Wagner. Tweakable block ciphers. *Journal of Cryptology*, 24(3):588–613, July 2011. (Cited on page 20.)

[MGS15] Hristina Mihajloska, Danilo Gligoroski, and Simona Samardjiska. Reviving the idea of incremental cryptography for the zettabyte era use case: Incremental hash functions based on sha-3. In *International Workshop on Open Problems in Network Security*, pages 97–111. Springer, 2015. (Cited on page 3, 6.)

[Mic97] Daniele Micciancio. Oblivious data structures: Applications to cryptography. In *29th ACM STOC*, pages 456–464. ACM Press, May 1997. (Cited on page 6, 10.)

[MPRS12] Ilya Mironov, Omkant Pandey, Omer Reingold, and Gil Segev. Incremental deterministic public-key encryption. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, volume 7237 of *LNCS*, pages 628–644. Springer, Heidelberg, April 2012. (Cited on page 3, 6.)

[MV04] David A. McGrew and John Viega. The security and performance of the Galois/counter mode (GCM) of operation. In Anne Canteaut and Kapalee Viswanathan, editors, *INDOCRYPT 2004*, volume 3348 of *LNCS*, pages 343–355. Springer, Heidelberg, December 2004. (Cited on page 5, 19, 21.)

[Nai15] Yusuke Naito. Full PRF-secure message authentication code based on tweakable block cipher. In Man Ho Au and Atsuko Miyaji, editors, *ProvSec 2015*, volume 9451 of *LNCS*, pages 167–182. Springer, Heidelberg, November 2015. (Cited on page 22.)

[PS16] Thomas Peyrin and Yannick Seurin. Counter-in-tweak: Authenticated encryption modes for tweakable block ciphers. In Matthew Robshaw and Jonathan Katz, editors, *CRYPTO 2016, Part I*, volume 9814 of *LNCS*, pages 33–63. Springer, Heidelberg, August 2016. (Cited on page 4, 6, 13, 19, 22.)

[RBBK01] Phillip Rogaway, Mihir Bellare, John Black, and Ted Krovetz. OCB: A block-cipher mode of operation for efficient authenticated encryption. In Michael K. Reiter and Pierangela Samarati, editors, *ACM CCS 2001*, pages 196–205. ACM Press, November 2001. (Cited on page 3.)

[Rog95] Phillip Rogaway. Bucket hashing and its application to fast message authentication. In Don Coppersmith, editor, *CRYPTO'95*, volume 963 of *LNCS*, pages 29–42. Springer, Heidelberg, August 1995. (Cited on page 3.)

[Rog02] Phillip Rogaway. Authenticated-encryption with associated-data. In Vijayalakshmi Atluri, editor, *ACM CCS 2002*, pages 98–107. ACM Press, November 2002. (Cited on page 3.)

[Rog04] Phillip Rogaway. Efficient instantiations of tweakable blockciphers and refinements to modes OCB and PMAC. In Pil Joong Lee, editor, *ASIACRYPT 2004*, volume 3329 of *LNCS*, pages 16–31. Springer, Heidelberg, December 2004. (Cited on page 19, 20, 21.)

[Sho96] Victor Shoup. On fast and provably secure message authentication based on universal hashing. In Neal Koblitz, editor, *CRYPTO'96*, volume 1109 of *LNCS*, pages 313–328. Springer, Heidelberg, August 1996. (Cited on page 3.)

[SY16] Yu Sasaki and Kan Yasuda. A new mode of operation for incremental authenticated encryption with associated data. In Orr Dunkelman and Liam Keliher, editors, *SAC 2015*, volume 9566 of *LNCS*, pages 397–416. Springer, Heidelberg, August 2016. (Cited on page 3, 6, 9.)

[WC81] Mark N. Wegman and Larry Carter. New hash functions and their use in authentication and set equality. *Journal of Computer and System Sciences*, 22:265–279, 1981. (Cited on page 3, 5.)

[Yas11] Kan Yasuda. A new variant of PMAC: Beyond the birthday bound. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 596–609. Springer, Heidelberg, August 2011. (Cited on page 19, 22.)

A Proof of Proposition 1

There are two versions of the statement of Proposition 1, corresponding to the two choices of $(\mathrm{x}, \mathrm{X}) \in \{(\mathrm{iuf}, \mathrm{IUF}), (\mathrm{iprf}, \mathrm{IPRF})\}$. We proceed to prove the version where $(\mathrm{x}, \mathrm{X}) = (\mathrm{iprf}, \mathrm{IPRF})$, and then discuss how to adapt the proof to the remaining version, where $(\mathrm{x}, \mathrm{X}) = (\mathrm{iuf}, \mathrm{IUF})$.

Proof of Proposition 1, $(\mathrm{x}, \mathrm{X}) = (\mathrm{iprf}, \mathrm{IPRF})$**:** Consider the game G_0 described in Figure 9. This game is exactly the IPRF security game for the incremental function family iF and the document editing system DE. Games G_0 and G_1 are identical-until-bad games, which means that, by the Fundamental Lemma of Game Playing [BR06], we have

$$\Pr[\mathrm{G}_0(A)] - \Pr[\mathrm{G}_1(A)] \leq \Pr[\mathrm{G}_0(A) \text{ sets } \mathsf{bad}] \; .$$

Further, we can see that $\Pr[\mathrm{G}_0(A) \text{ sets } \mathsf{bad}] = \Pr[\mathbf{G}^{\mathrm{scorr}}_{\mathsf{iF},\mathsf{DE}}(A)] = 0$ due to the strong correctness of the incremental function family iF relative to the document editing system DE. This means that $\Pr[\mathrm{G}_0(A)] = \Pr[\mathrm{G}_1(A)]$.

We then construct, in Figure 10, the adversary A_0, which runs the adversary A, giving the latter its own $\textsc{Init}$, $\textsc{Vf}$, and $\textsc{Fin}$ oracles, and simulating the $\textsc{Tag}$ and $\textsc{Upd}$ oracle. $\textsc{Tag}_0$ perfectly simulates the $\textsc{Tag}$ oracle for A, while $\textsc{Upd}_0$ perfectly simulates the $\textsc{Upd}$ oracle for A in game G_1. This gives us $\Pr[\mathrm{G}_1(A)] = \Pr[\mathrm{G}_1(A_0)]$ with A_0 making $q_t + q_u$ $\textsc{Tag}$ queries and zero $\textsc{Upd}$ queries. We again invoke the strong correctness property to assert that $\Pr[\mathrm{G}_1(A_0)] = \Pr[\mathrm{G}_0(A_0)]$. Putting this together gives us $\mathbf{Adv}^{\mathrm{iprf}}_{\mathsf{iF},\mathsf{DE}}(A_0) = \mathbf{Adv}^{\mathrm{iprf}}_{\mathsf{iF},\mathsf{DE}}(A)$, which completes the proof. ▮

We now discuss adapting the above proof for the version of the proposition where $(\mathrm{x}, \mathrm{X}) = (\mathrm{iuf}, \mathrm{IUF})$. The games used in this version are analogous to those in the above proof – game G_0 is the

```
Game $G_0$, $\boxed{G_1}$

oracle INIT
  $b \twoheadleftarrow \{0,1\}$ ; ; $K \twoheadleftarrow \mathsf{KS}$ ; $f \twoheadleftarrow \mathrm{FUNC}(\mathsf{NS} \times \{0,1\}^* \times \mathsf{BS}^*, \mathsf{Rng})$

oracle TAG$(N, id, D)$
  if $(N \in \mathrm{NL}_{id}$ and $|\mathsf{NS}| \neq 1)$ then return $\perp$
  $D_{id} \leftarrow D$ ; $t^1_{id} \leftarrow \mathsf{Tg}(K, N, id, D_{id})$ ; $t^0_{id} \leftarrow f(N, id, D_{id})$
  $\mathrm{NL}_{id} \leftarrow \mathrm{NL}_{id} \cup \{N\}$ ; $\mathrm{DL}_{id} \leftarrow \mathrm{DL}_{id} \cup \{D_{id}\}$ ; return $t^b_{id}$

oracle UPD$(N, id, op, arg)$
  if $D_{id} = \perp$ then return $\perp$
  if $(N \in \mathrm{NL}_{id}$ and $|\mathsf{NS}| \neq 1)$ then return $\perp$
  $t^1_{id} \leftarrow \mathsf{Up}(K, N, id, D_{id}, op, arg, t^1_{id})$ ; $D_{id} \leftarrow \mathsf{Ed}(D_{id}, op, arg)$
  $t \leftarrow \mathsf{Tg}(K, N, id, D[]id)$ ; $t^0_{id} \leftarrow f(N, id, D_{id})$
  if $t \neq t^1_{id}$ then bad $\leftarrow$ true $\boxed{; t^1_{id} \leftarrow t}$
  $\mathrm{NL}_{id} \leftarrow \mathrm{NL}_{id} \cup \{N\}$ ; $\mathrm{DL}_{id} \leftarrow \mathrm{DL}_{id} \cup \{D_{id}\}$ ; return $t^b_{id}$

oracle VF$(id, D, t)$
  if $D \in \mathrm{DL}_{id}$ then return $\perp$
  if $b = 1$ then return $\mathsf{Ver}(K, id, D, t)$ else return false

oracle FIN$(b')$
  return $(b' = b)$
```

Figure 9: Games G_0 and G_1 for the proof of Proposition 1. The boxed code is only included in G_1.

```
Adversary $A_0^{\mathrm{INIT,TAG,UPD,VF,FIN}}$:
  $A^{\mathrm{INIT,TAG_0,UPD_0,VF,FIN}}$

subroutine TAG$_0(N, id, D)$
  $D^*_{id} \leftarrow D$
  return TAG$(N, id, D)$

subroutine UPD$_0(N, id, op, arg)$
  if $D^*_{id} = \perp$ then return $\perp$
  $D^*_{id} \leftarrow \mathsf{Ed}(D^*_{id}, op, arg)$
  return TAG$(N, id, D^*_{id})$
```

Figure 10: Adversary A_0 for the proof of Proposition 1. It runs A and answers the oracle queries of the latter via the shown subroutines.

IUF security game, and games G_0, G_1 are identical-until-bad, where G_1, during UPD queries, sets the tag to be the result of the tagging algorithm rather than the update algorithm. The adversary A'_0 is constructed by running A', giving the latter its own INIT, VF, and FIN oracles and simulating the TAG and UPD oracle, in the same manner as in the above proof. The analysis proceeds as before, and therefore we get the claimed equation between the advantages of the adversaries as in the first version of the proposition.

B Proof of Proposition 2

Proof of Proposition 2: We assume the adversary A_{uf} does not make any trivial queries, such as repeating nonces across its TAG, UPD queries, querying VF(id, D, t) if $D \in \mathrm{DL}_{id}$, and querying UPD with id such that $D_{id} = \perp$. This allows us to remove the checks for these occurrences and

```
Adversary $A_{\mathrm{prf}}^{\textsc{Init},\textsc{Tag},\textsc{Upd},\textsc{Vf},\textsc{Fin}}$:
1 $A_{\mathrm{uf}}^{\textsc{Init},\textsc{Tag},\textsc{Upd},\textsc{Vf}_{\mathrm{M}},\textsc{Fin}_M}$

subroutine $\textsc{Vf}_{\mathrm{M}}(id, D, t)$
2 $d \leftarrow \textsc{Vf}(id, D, t)$
3 if $d$ then win $\leftarrow$ true
4 return $d$

subroutine $\textsc{Fin}_M$
5 $\textsc{Fin}(\mathsf{win})$
```

Figure 11: Adversary A_{prf} for the proof of Proposition 2. It runs A_{uf} and answers the oracle queries of the latter via the shown subroutines.

simplify the IUF game. The adversary A_{prf} passes to A_{uf} its $\textsc{Init}, \textsc{Tag}, \textsc{Upd}$ oracles, and simulates the latters $\textsc{Vf}$ queries as described by $\textsc{Vf}_{\mathrm{M}}$ in Figure 11. From the definition of the irf-advantage of an adversary, we have

$$\begin{aligned}\mathbf{Adv}^{\mathrm{iprf}}_{\mathsf{iF},\mathsf{DE}}(A_{\mathrm{prf}}) &= 2\Pr[\mathbf{G}^{\mathrm{iprf}}_{\mathsf{iF},\mathsf{DE}}(A_{\mathrm{prf}})] - 1\\ &= 2\Pr[\mathsf{win} = b] - 1\\ &= \Pr[\mathsf{win}|b=1] - \Pr[\mathsf{win}|b=0]\\ &= \mathbf{Adv}^{\mathrm{iuf}}_{\mathsf{iMA},\mathsf{DE}}(A_{\mathrm{uf}})\end{aligned}$$

This completes the proof.▌

C Proof of Theorem 3

We prove the case where $(\mathrm{x}, (X)) = (\mathrm{iprf}, \mathrm{IPRF})$.

Proof of Theorem 3, $(\mathrm{x}, (X)) = (\mathrm{iprf}, \mathrm{IPRF})$**:** We assume A_{md} makes no trivial queries. This means it does not query $\textsc{Upd}$ with id such that $D_{id} = \bot$. This allows us to simplify the games and adversaries by removing the check related to this item. Now consider games $\mathrm{G}_0, \mathrm{G}_1$ of Figure 12. The only difference is in the choice of g made in $\textsc{Init}$. We have

$$\begin{aligned}\mathbf{Adv}^{\mathrm{iprf}}_{\mathsf{iF}_{\mathsf{md}},\mathsf{DE}}(A_{\mathrm{md}}) &= \Pr[\mathrm{G}_0(A_{\mathrm{md}})]\\ &= \Pr[\mathrm{G}_1(A_{\mathrm{md}})] + (\Pr[\mathrm{G}_0(A_{\mathrm{md}})] - \Pr[\mathrm{G}_1(A_{\mathrm{md}})])\ .\end{aligned}$$

We design adversary B so that

$$\Pr[\mathrm{G}_0(A_{\mathrm{md}})] - \Pr[\mathrm{G}_1(A_{\mathrm{md}})] \le \mathbf{Adv}^{\mathrm{prf}}_{\mathsf{F}}(B)\ .$$

The design is standard. Briefly, adversary B runs A_{md}, responding to the latter's $\textsc{Tag}, \textsc{Upd}$ queries as per the code of the corresponding oracles in G_1 except that calls to g are substituted by calls to B's own $\textsc{Fn}$ oracle. When A_{md} ends by calling $\textsc{Fin}$, adversary B returns 1 if $b' = b$ and 0 otherwise.

In game G_1, the keys of distinct document identities are uniformly and independently distributed. Game G_2 picks them directly that way, no longer using g. Therefore we have

$$\Pr[\mathrm{G}_2(A_{\mathrm{md}})] = \Pr[\mathrm{G}_1(A_{\mathrm{md}})]\ .$$

We now proceed via a hybrid argument. It is easy to see that

```
Games $G_0, G_1, G_2$

oracle INIT
 1  $b \twoheadleftarrow \{0,1\}$ ; $f \twoheadleftarrow \mathrm{FUNC}(\mathsf{NS} \times \{0,1\}^* \times \mathsf{BS}^*, \mathsf{Rng})$
 2  $K \twoheadleftarrow \mathsf{KS}$ ; $g \leftarrow \mathsf{F}(K, \cdot)$  // Game $G_0$
 3  $g \twoheadleftarrow \mathrm{FUNC}(\{0,1\}^*, \mathsf{iF_{sd}.KS})$  // Game $G_1$
 4  For $j = 1, \ldots, q$ do $L_j \twoheadleftarrow \mathsf{iF_{sd}.KS}$  // Game $G_2$

oracle TAG$(N, id, D)$
 5  $K_{id} \leftarrow g(id)$  // Game $G_0, G_1$
 6  if $K_{id} = \varepsilon$ then $i \leftarrow i+1$ ; $K_{id} \leftarrow L_i$  // Game $G_2$
 7  $D_{id} \leftarrow D$ ; $t^1_{id} \leftarrow \mathsf{iF_{sd}.Tg}(K_{id}, N, \varepsilon, D_{id})$ ; $t^0_{id} \leftarrow f(N, id, D_{id})$ ; return $t^b_{id}$

oracle UPD$(N, id, op, arg)$
 8  $K_{id} \leftarrow g(id)$  // Game $G_0, G_1$
 9  $t^1_{id} \leftarrow \mathsf{iF_{sd}.Up}(K_{id}, N, \varepsilon, D_{id}, op, arg, t^1_{id})$
10  $D_{id} \leftarrow \mathsf{Ed}(D_{id}, op, arg)$ ; $t^0_{id} \leftarrow f(N, id, D_{id})$ ; return $t^b_{id}$

oracle VF$(id, D, t)$
11  $K_{id} \leftarrow g(id)$  // Game $G_0, G_1$
12  if $K_{id} = \varepsilon$ then $i \leftarrow i+1$ ; $K_{id} \leftarrow L_i$  // Game $G_2$
13  if $b = 1$ then return $\mathsf{iF_{sd}.Ver}(K_{id}, \varepsilon, D, t)$
14  else return false

oracle FIN$(b')$
15  return $(b' = b)$
```

```
Games $G_{3,\ell}$ where $0 \le \ell \le q$

oracle INIT
 1  $f \twoheadleftarrow \mathrm{FUNC}(\mathsf{NS} \times \{0,1\}^* \times \mathsf{BS}^*, \mathsf{Rng})$
 2  For $j = 1, \ldots, q$ do $L_j \twoheadleftarrow \mathsf{iF_{sd}.KS}$

oracle TAG$(N, id, D)$
 3  if $K_{id} = \varepsilon$ then $i \leftarrow i+1$ ; $K_{id} \leftarrow L_i$ ; $C[id] \leftarrow i$
 4  $D_{id} \leftarrow D$ ; $t^1_{id} \leftarrow \mathsf{iF_{sd}.Tg}(K_{id}, N, \varepsilon, D_{id})$ ; $t^0_{id} \leftarrow f(N, id, D_{id})$
 5  if $(C[id] \le \ell)$ then return $t^0_{id}$ else return $t^1_{id}$

oracle UPD$(N, id, op, arg)$
 6  $t^1_{id} \leftarrow \mathsf{iF_{sd}.Up}(K_{id}, N, \varepsilon, D_{id}, op, arg, t^1_{id})$
 7  $D_{id} \leftarrow \mathsf{Ed}(D_{id}, op, arg)$ ; $t^0_{id} \leftarrow f(N, id, D_{id})$
 8  if $(C[id] \le \ell)$ then return $t^0_{id}$ else return $t^1_{id}$

oracle VF$(id, D, t)$
 9  if $K_{id} = \varepsilon$ then $i \leftarrow i+1$ ; $K_{id} \leftarrow L_i$ ; $C[id] \leftarrow i$
10  if $(C[id] \le \ell)$ then return false
11  return $\mathsf{iF_{sd}.Ver}(K_{id}, \varepsilon, D, t)$

oracle FIN$(b')$
12  return $(b' = 1)$
```

Figure 12: Top: Games G_0, G_1, G_2 for the proof of Theorem 3. Line 2 is included only in game G_0, line 3 is included only in line G_1, while lines 4 and 6 are included only in game G_2. Lines 5 and 8 are included only in games G_0 and G_1. Bottom: Hybrid games $G_{3,\ell}$ for the proof of Theorem 3.

```
Adversary $A_j^{\textsc{Init},\textsc{Tag},\textsc{Upd},\textsc{Fin}}$:
$L_{j+1},\ldots,L_q \twoheadleftarrow \mathsf{iF_{sd}.KS}$ ; $f \twoheadleftarrow \mathrm{FUNC}(\mathsf{NS}\times\{0,1\}^*\times\mathsf{BS}^*,\mathsf{Rng})$
$A_{\mathrm{md}}^{\textsc{Init},\textsc{Tag}_{\mathrm{md}},\textsc{Upd}_{\mathrm{md}},\textsc{Vf}_{\mathrm{md}},\textsc{Fin}}$

subroutine $\textsc{Tag}_{\mathrm{md}}(N, id, D)$
$D_{id} \leftarrow D$
if $(K_{id} = \varepsilon)$ then $i \leftarrow i+1$ ; $C[id] \leftarrow i$
if $(C[id] < j)$ then return $f(N, id, D)$
if $(C[id] = j)$ then return $\textsc{Tag}(N, \varepsilon, D)$
$K_{id} \leftarrow L_i$ ; $t_{id} \leftarrow \mathsf{iF_{sd}.Tg}(K_{id}, N, \varepsilon, D_{id})$ ; return $t_{id}$

subroutine $\textsc{Upd}_{\mathrm{md}}(N, id, op, arg)$
if $(C[id] = j)$ then return $\textsc{Upd}(N, \varepsilon, op, arg)$
if $(C[id] < j)$ then $D' \leftarrow \mathsf{Ed}(D_{id}, op, arg)$ ; return $f(N, id, D')$
$t_{id} \leftarrow \mathsf{iF_{sd}.Up}(K_{id}, N, \varepsilon, D_{id}, op, arg, t)$
$D_{id} \leftarrow \mathsf{Ed}(D_{id}, op, arg)$ ; return $t_{id}$

subroutine $\textsc{Vf}_{\mathrm{md}}(id, D, t)$
if $(K_{id} = \varepsilon)$ then $i \leftarrow i+1$ ; $C[id] \leftarrow i$
if $(C[id] = j)$ then return $\textsc{Vf}(\varepsilon, D, t)$
if $(C[id] < j)$ then return false
$K_{id} \leftarrow L_i$ ; return $\mathsf{iF_{sd}.Ver}(K_{id}, \varepsilon, D, t)$
```

Figure 13: Adversary $A_j \in \mathcal{A}_{\mathrm{sd}}$ against the IRF security of $\mathsf{iF_{sd}}$ for the proof of Theorem 3. It runs A_{md} and answers the oracle queries of the latter via the shown subroutines.

$$\begin{aligned}\Pr[\mathrm{G}_2(A_{\mathrm{md}})] &= \Pr[\mathrm{G}_{3,0}(A_{\mathrm{md}})] - \Pr[\mathrm{G}_{3,q}(A_{\mathrm{md}})]\\ &= (\Pr[\mathrm{G}_{3,0}(A_{\mathrm{md}})] - \Pr[\mathrm{G}_{3,1}(A_{\mathrm{md}})]) + (\Pr[\mathrm{G}_{3,1}(A_{\mathrm{md}})] - \Pr[\mathrm{G}_{3,2}(A_{\mathrm{md}})])\\ &\quad + \ldots + (\Pr[\mathrm{G}_{3,q-1}(A_{\mathrm{md}})] - \Pr[\mathrm{G}_{3,q}(A_{\mathrm{md}})]) \;,\end{aligned}$$

where $\mathrm{G}_{3,\ell}$, $0 \le \ell \le q$ are the hybrid games described in Figure 12. We now design adversary A_j in Figure 13 such that

$$\Pr[\mathrm{G}_{3,j-1}(A_{\mathrm{md}})] - \Pr[\mathrm{G}_{3,j}(A_{\mathrm{md}})] \le \mathbf{Adv}^{\mathrm{iprf}}_{\mathsf{iF_{sd}},\mathsf{DE}}(A_j) \;.$$

We let A_{sd} denote the adversary among A_j, $1 \le j \le q$ that achieves the maximum iprf-advantage. Then, we can write

$$\Pr[\mathrm{G}_2(A_{\mathrm{md}})] \le q \cdot \mathbf{Adv}^{\mathrm{iprf}}_{\mathsf{iF_{sd}},\mathsf{DE}}(A_{\mathrm{sd}}) \;.$$

Using this in the earlier equations gives us the bound claimed in the theorem, and completes the proof.▮

We now discuss adapting to prove for the case where $(\mathrm{x}, \mathrm{X}) = (\mathrm{iuf}, \mathrm{IUF})$. The game transitions used are analogous to those in the above proof, but for the IUF case – game G_0 is the IUF security game, game G_1 differs from game G_0 in the use of a random function in place of the PRF, while game G_2 samples the sub-keys at random just as in the above proof. The hybrid games $\mathrm{G}_{3,j}$ and adversaries A_j are adapted in the same manner. Then, the analysis proceeds along the same lines as the above case, and we get the claimed equation as in the first version of the proposition.

```
Games $G_0, G_1, G_2, G_3$

oracle INIT
  $b \twoheadleftarrow \{0,1\}$ ; $f \twoheadleftarrow \mathrm{FUNC}(\mathsf{NS} \times \{0,1\}^* \times \mathsf{BS}^*, \mathsf{Rng})$ ; $K \twoheadleftarrow \mathsf{KS}$

oracle TAG$(N, id, D)$
  $D_{id} \leftarrow D$ ; $d \leftarrow \mathsf{H}(id, \mathsf{bl})$
  if $\mathrm{HT}_1[d] \notin \{\perp, id\}$ then $\mathsf{coll}_1 \leftarrow$ true
  $\mathrm{HT}_1[d] \leftarrow id$ ; $D' \leftarrow \mathrm{Prepend}(d, D)$ ; $N' \leftarrow \mathsf{H}(id \| N, \mathsf{nl})$
  if $\mathrm{HT}_2[N'] \notin \{\perp, id||N\}$ then $\mathsf{coll}_2 \leftarrow$ true
  $\mathrm{HT}_2[N'] \leftarrow id||N$ ; $t^1_{id} \leftarrow \mathsf{iF_{sd}}.\mathsf{Tg}(K, N', \varepsilon, D')$
  $t^0_{id} \leftarrow f(N, id, D_{id})$ ; return $t^b_{id}$   // Games $G_0, G_1, G_2$
  $t^0_{id} \twoheadleftarrow \mathsf{Rng}$ ; return $t^b_{id}$   // Game $G_3$

oracle VF$(id, D, t)$
  $d \leftarrow \mathsf{H}(id, \mathsf{bl})$
  if $\mathrm{HT}_1[d] \notin \{\perp, id\}$ then $\mathsf{coll}_1 \leftarrow$ true
  $\mathrm{HT}_1[d] \leftarrow id$ ; $D' \leftarrow \mathrm{Prepend}(d, D)$
  if $b = 1$ then return $\mathsf{iF_{sd}}.\mathsf{Ver}(K, \varepsilon, D', t)$
  else return false

oracle FIN$(b')$
  if $\mathsf{coll}_1$ then $\mathsf{bad}_1 \leftarrow$ true $\boxed{\text{; return false}}$
  if $\mathsf{coll}_2$ then $\mathsf{bad}_2 \leftarrow$ true ; return false
  return $(b' = b)$
```

Figure 14: Games G_0, G_1, G_2, G_3 for the proof of Theorem 4. The $\boxed{\text{boxed code}}$ is excluded in game G_0, while the highlighted code is excluded in games G_0, G_1.

```
Adversary $A_{\mathrm{sd}}^{\mathrm{INIT,TAG,FIN}}$:
  $A_{\mathrm{md}}^{\mathrm{INIT,TAG_{md},VF_{md},FIN}}$

subroutine TAG$_{\mathrm{md}}(N, id, D)$
  $d \leftarrow \mathsf{H}(id, \mathsf{bl})$ ; $D' \leftarrow \mathrm{Prepend}(d, D)$
  $N' \leftarrow \mathsf{H}(id \| N, \mathsf{nl})$ ; return TAG$(N', \varepsilon, D')$

subroutine VF$_{\mathrm{md}}(id, D, t)$
  $d \leftarrow \mathsf{H}(id, \mathsf{bl})$ ; $D' \leftarrow \mathrm{Prepend}(d, D)$
  return VF$(\varepsilon, D, t)$
```

Figure 15: Adversary $A_{\mathrm{sd}} \in \mathcal{A}_{\mathrm{sd}}$ against the IPRF security of $\mathsf{iF_{sd}}$ for the proof of Theorem 4. It runs A_{md} and answers the oracle queries of the latter via the shown subroutines.

D Proof of Theorem 4

We prove the case where $(\mathrm{x}, \mathrm{X}) = (\mathrm{iprf}, \mathrm{IPRF})$, and where $\mathsf{nl} \neq 0$, .

Proof of Theorem 4, $(\mathrm{x}, (X)) = (\mathrm{iprf}, \mathrm{IPRF})$, $\mathsf{nl} \neq 0$: We begin by recalling Proposition 1, which tells us that since $\mathsf{iF_{sd}}$ satisfies strong correctness, we can replace any adversary making q_t TAG queries and q_u UPD queries with an adversary that makes $q_t + q_u$ TAG queries and zero UPD queries. Therefore, we can assume without loss of generality that the adversary A_{iF} does not make any UPD queries, which allows us to remove this oracle from consideration for this proof. We further assume

A_{md} makes no trivial queries. This allows us to simplify the games and adversaries by removing various checks related the reuse of nonces. Consider the game G_0 described in Figure 14, which is essentially the IPRF game along with some additional bookkeeping. We have that

$$\begin{aligned}\mathbf{Adv}^{\mathrm{iprf}}_{\mathsf{iF_{md}},\mathsf{DE}}(A_{\mathrm{md}}) &= \Pr[\mathrm{G}_0(A_{\mathrm{md}})] \\ &= \Pr[\mathrm{G}_1(A_{\mathrm{md}})] + (\Pr[\mathrm{G}_0(A_{\mathrm{md}})] - \Pr[\mathrm{G}_1(A_{\mathrm{md}})])\ .\end{aligned}$$

Now, games G_0 and G_1 are identical-until-bad_1 games, and therefore we use the Fundamental Lemma of Game Playing [BR06] to say

$$\Pr[\mathrm{G}_0(A_{\mathrm{md}})] - \Pr[\mathrm{G}_1(A_{\mathrm{md}})] \leq \Pr\left[\mathrm{G}_0(A_{\mathrm{md}}) \text{ sets } \mathsf{bad}_1\right]\ .$$

We design adversary B_1 such that

$$\Pr\left[\mathrm{G}_0(A_{\mathrm{md}}) \text{ sets } \mathsf{bad}_1\right] \leq \mathbf{Adv}^{\mathrm{cr}}_{\mathsf{H},\mathsf{bl}}(B_1)\ .$$

The adversary is standard – it runs adversary A_{md} in the game G_0, and stores hash queries in the table HT_1 as in the game. In the event of the coll_1 flag being set to true, the adversary returns the pair consisting of the document identity already in the table for that hash value, and the currently queried document identity.

Next, note that games G_1 and G_2 are identical-until-bad_2 games, and therefore we again use the Fundamental Lemma of Game Playing to say

$$\Pr[\mathrm{G}_1(A_{\mathrm{md}})] - \Pr[\mathrm{G}_2(A_{\mathrm{md}})] \leq \Pr\left[\mathrm{G}_1(A_{\mathrm{md}}) \text{ sets } \mathsf{bad}_2\right]\ .$$

We can design adversary B_2 along the lines of the strategy described for adversary B_1, such that

$$\Pr\left[\mathrm{G}_1(A_{\mathrm{md}}) \text{ sets } \mathsf{bad}_2\right] \leq \mathbf{Adv}^{\mathrm{cr}}_{\mathsf{H},\mathsf{nl}}(B_2)\ .$$

The game G_3 picks the tags at random rather than using the random function f in the Tag and Upd queries. Since every input to the random function is new (since the adversary A_{md} is nonce-respecting), the games G_2 and G_3 are equivalent, and we have that

$$\Pr[\mathrm{G}_2(A_{\mathrm{md}})] = \Pr[\mathrm{G}_3(A_{\mathrm{md}})]\ .$$

We now provide an adversary $A_{\mathrm{sd}} \in \mathcal{A}_{\mathrm{sd}}$ in Figure 15 such that

$$\Pr[\mathrm{G}_3(A_{\mathrm{md}})] \leq \mathbf{Adv}^{\mathrm{iprf}}_{\mathsf{iF_{sd}},\mathsf{DE}}(A_{\mathrm{sd}})\ .$$

Note that in game G_3, we can assume that the choice of nonces and document identitities picked by the adversary A_{md} is such that no collisions are generated (as otherwise the Fin procedure would return false). This means that every input to the $\mathrm{Tag}_{\mathrm{md}}$ and $\mathrm{Upd}_{\mathrm{md}}$ queries made by A_{sd} is unique, and therefore we can assume t^0_{id} to be picked freshly at random for each query. Putting these equations together achieves the claimed inequality, and completes the proof. ▌

The proof in the case where $(\mathrm{x}, \mathrm{X}) = (\mathrm{iuf}, \mathrm{IUF})$ follows from an adaptation of the games above to the IUF setting. The analysis then proceeds in a similar manner to the proof above, which gives us the claimed equation. When $\mathsf{nl} = 0$, then we are in the setting where the hash function H always returns ε. Note that the underlying single-document scheme also does not take nonces as input. This allows us to skip the game G_2 in the game transitions, and jump directly from game G_1 to game G_3. This leads to the term ϵ being set to 0 in this setting, and gives us the claimed equation.

```
Game G_0

oracle INIT
  $K \twoheadleftarrow \mathsf{KS}$ ; $(K_{\mathsf{iHF}}, K_{\mathsf{SE}}) \leftarrow \mathsf{KDF}(K)$ ; return $K$

oracle TAG$(N, id, D)$
  $D_{id} \leftarrow D$ ; $h \leftarrow \mathsf{iHF.Hsh}(K_{\mathsf{iHF}}, D_{id})$
  $t_{id} \leftarrow \mathsf{SE.Enc}(K_{\mathsf{SE}}, N, h)$ ; $h' \leftarrow \mathsf{SE.Dec}(K_{\mathsf{SE}}, t_{id})$
  if $(h \neq h')$ then win $\leftarrow$ true
  return $\perp$

oracle UPD$(N, id, op, arg)$
  $D'_{id} \leftarrow \mathsf{Ed}(D_{id}, op, arg)$ ; $h \leftarrow \mathsf{SE.Dec}(K_{\mathsf{SE}}, t_{id})$
  $h' \leftarrow \mathsf{iHF.Up}(K_{\mathsf{iHF}}, D_{id}, op, arg, h)$ ; $t'_{id} \leftarrow \mathsf{SE.Enc}(K_{\mathsf{SE}}, N, h')$
  $h_1 \leftarrow \mathsf{iHF.Hsh}(K_{\mathsf{iHF}}, D'_{id})$ ; $h'_1 \leftarrow \mathsf{SE.Dec}(K_{\mathsf{SE}}, t'_{id})$
  if $(h_1 \neq h'_1)$ then win $\leftarrow$ true
  $t''_{id} \leftarrow \mathsf{SE.Enc}(K_{\mathsf{SE}}, N, h_1)$
  if $t'_{id} \neq t''_{id}$ then win $\leftarrow$ true
  return $\perp$

oracle FIN
  return win
```

Figure 16: Game G_0 for the proof of Proposition 5.

E Proof of Proposition 5

Proof of Proposition 5: We assume that the adversary makes no trivial queries – that is, it does not make UPD queries with *id* such that $D_{id} = \perp$. We can then simplify the games and adversaries by removing the checks associated with them.

Consider game G_0, which is the strong correctness game rewritten for the function family $\mathsf{iF} = \mathbf{iHtE}[\mathsf{iHF}, \mathsf{SE}, \mathsf{KDF}]$. An adversary A can win by setting the win flag to true in any of lines 4, 9, and 11. For line 4, the requirement on SE that decryption reverses encryption ensures that $h = h'$ for any TAG query made by A. From the correctness requirement of the incremental hash function and the requirement on SE that decryption reverses encryption, we know that $h' = h_1$, where h' is defined on line 7 and h_1 is defined on line 8. Therefore, analogous to line 4, in line 9 we are assured that $h_1 = h'_1$ in any UPD query made by A due to the requirement on SE that decryption reverses encryption. Further, since the encryption in SE is deterministic, we also have that $t'_{id} = t''_{id}$ in line 11, for any UPD query made by A.

All these together give us that for any adversary A, $\Pr[\mathbf{G}^{\mathrm{scorr}}_{\mathsf{iF},\mathsf{DE}}(A)] = 0$, and therefore the constructed incremental function family iF satisfies strong correctness. This completes the proof.∎

F Proof of Theorem 6

Proof of Theorem 6: We begin by recalling Proposition 1, which tells us that since the **iHtE** construction provides strong correctness (Proposition 5), we can replace any adversary making q_t TAG queries and q_u UPD queries with an adversary that makes $q_t + q_u$ TAG queries and zero UPD queries. Therefore, we can assume without loss of generality that the adversary A_{iF} does not

```
Games $G_0$, $G_1$, $G_2$, $\boxed{G_3, G_4}$

oracle INIT
  $b \twoheadleftarrow \{0,1\}$ ; $K_{\mathsf{iHF}} \twoheadleftarrow \mathsf{iHF.KS}$ ; $K_{\mathsf{SE}} \twoheadleftarrow \mathsf{SE.KS}$
  $f \twoheadleftarrow \mathrm{FUNC}(\mathsf{NS} \times \{\varepsilon\} \times \mathsf{BS}^*, \mathsf{Rng})$
  $g \twoheadleftarrow \mathrm{FUNC}(\mathsf{NS} \times \{0,1\}^{\mathsf{ol}}, \mathsf{Rng})$

oracle TAG$(N, id, D)$
  $t^0 \leftarrow f(N, id, D)$  // Game $G_0$
  $t^0 \twoheadleftarrow \mathsf{Rng}$  // Games $G_1$, $G_2$, $G_3$, $G_4$
  $h \leftarrow \mathsf{iHF.Hsh}(K_{\mathsf{iHF}}, D)$
  if $(\mathrm{HT}[h] \neq \{\perp, D\})$ then bad $\leftarrow$ true $\boxed{\text{; return } t^0}$
  $\mathrm{HT}[h] \leftarrow D$
  $t^1 \leftarrow \mathsf{SE.Enc}(K_{\mathsf{SE}}, N, h)$  // Games $G_0$, $G_1$
  $t^1 \leftarrow g(N, h)$  // Games $G_2$, $G_3$
  $t^1 \twoheadleftarrow \mathsf{Rng}$  // Game $G_4$
  return $t^b$

oracle VF$(id, D, t)$
  $h \leftarrow \mathsf{iHF.Hsh}(K_{\mathsf{iHF}}, D)$
  if $(\mathrm{HT}[h] \neq \{\perp, D\})$ then bad $\leftarrow$ true $\boxed{\text{; return false}}$
  $\mathrm{HT}[h] \leftarrow D$
  $h' \leftarrow \mathsf{SE.Dec}(K_{\mathsf{SE}}, t)$  // Games $G_0$, $G_1$
  $h' \leftarrow \perp$  // Games $G_2$, $G_3$, $G_4$
  if $(b = 1)$ then return $(h = h')$  // Games $G_0$, $G_1$, $G_2$, $G_3$
  if $(b = 1)$ then return false  // Game $G_4$
  return false

oracle FIN$(b')$
  return $(b' = b)$
```

Figure 17: Games for the proof of Theorem 6. The $\boxed{\text{boxed code}}$ is excluded in games G_0, G_1 and G_2.

make any UPD queries, which allows us to remove this oracle from consideration for this proof. We also assume the adversary does not make any trivial queries, such as re-using nonces between TAG queries, and querying VF with a document that has already been queried to the TAG oracle. This allows us to remove the checks for these conditions and further simplifies the games for the proof. Note that since we are in the sd setting, the document identity is assumed to always be the empty string ε.

Consider the game G_0 described in Figure 17. This is exactly the $\mathbf{G}^{\mathrm{iprf}}_{\mathsf{iF},\mathsf{DE}}$ expanded for the construction $\mathsf{iF} = \mathbf{iHtE}[\mathsf{iHF}, \mathsf{SE}, \mathsf{KDF}]$. The game G_1 differs from game G_0 in that the tag t^0 is sampled at random instead of picked as the output of a random function f. Since the input arguments to f are always different, the two games are equivalent, and we have that $\Pr[G_0(A_{\mathsf{iF}})] = \Pr[G_1(A_{\mathsf{iF}})]$. We then get

$$\begin{aligned}\mathbf{Adv}^{\mathrm{iprf}}_{\mathsf{iF},\mathsf{DE}}(A_{\mathsf{iF}}) &= 2 \cdot \Pr[G_0(A_{\mathsf{iF}})] - 1 \\ &= 2 \cdot \Pr[G_1(A_{\mathsf{iF}})] - 1 + 2 \cdot (\Pr[G_0(A_{\mathsf{iF}})] - \Pr[G_1(A_{\mathsf{iF}})]) \\ &= 2 \cdot \Pr[G_1(A_{\mathsf{iF}})] - 1 \,.\end{aligned}$$

Game G_2 differs from game G_1 in two ways – (1) the tag t^1 is now the output of a random function

```
Adversary $A_{\mathsf{SE}}^{\textsc{Init},\textsc{Enc},\textsc{Dec},\textsc{Fin}}$:
  $b \twoheadleftarrow \{0,1\}$ ; $K_{\mathsf{iHF}} \twoheadleftarrow \mathsf{iHF.KS}$ ; $f \twoheadleftarrow \mathrm{FUNC}(\mathsf{NS} \times \{\varepsilon\} \times \mathsf{BS}^*, \mathsf{Rng})$
  $A_{\mathsf{iF}}^{\textsc{Init},\textsc{Tag}_{\mathrm{SE}},\textsc{Vf}_{\mathrm{SE}},\textsc{Fin}_{\mathrm{SE}}}$

subroutine $\textsc{Tag}_{\mathrm{SE}}(N, id, D)$
  $t^0 \twoheadleftarrow \mathsf{Rng}$ ; $h \leftarrow \mathsf{iHF.Hsh}(K_{\mathsf{iHF}}, D)$
  if $(\mathrm{HT}[h] \neq \{\bot, D\})$ then return $t^0$
  $\mathrm{HT}[h] \leftarrow D$ ; $t^1 \leftarrow \textsc{Enc}(N, h)$ ; return $t^b$

subroutine $\textsc{Vf}_{\mathrm{SE}}(id, D, t)$
  $h \leftarrow \mathsf{iHF.Hsh}(K_{\mathsf{iHF}}, D)$
  if $(\mathrm{HT}[h] \neq \{\bot, D\})$ then return false
  $\mathrm{HT}[h] \leftarrow D$ ; $h' \leftarrow \textsc{Dec}(t)$
  if $(b = 1)$ then return $(h = h')$
  return false

subroutine $\textsc{Fin}_{\mathrm{SE}}(b')$
  $\textsc{Fin}$ ; return $(b' = b)$
```

```
Adversary $A_{\mathsf{H}}^{\textsc{Init},\textsc{Hash},\textsc{Fin}}$:
  $b \twoheadleftarrow \{0,1\}$ ; $K_{\mathsf{SE}} \twoheadleftarrow \mathsf{SE.KS}$ ; $f \twoheadleftarrow \mathrm{FUNC}(\mathsf{NS} \times \{\varepsilon\} \times \mathsf{BS}^*, \mathsf{Rng})$
  $g \twoheadleftarrow \mathrm{FUNC}(\mathsf{NS} \times \{0,1\}^{\mathsf{ol}}, \mathsf{Rng})$ ; $A_{\mathsf{iF}}^{\textsc{Init},\textsc{Tag}_{\mathrm{H}},\textsc{Vf}_{\mathrm{H}},\textsc{Fin}_{\mathrm{H}}}$

subroutine $\textsc{Tag}_{\mathrm{H}}(N, id, D)$
  $\textsc{Hash}(D)$ ; $t \twoheadleftarrow \mathsf{Rng}$ ; return $t$

subroutine $\textsc{Vf}_{\mathrm{H}}(id, D, t)$
  $\textsc{Hash}(D)$ ; return false

subroutine $\textsc{Fin}_{\mathrm{H}}(b')$
  $\textsc{Fin}$ ; return $(b' = b)$
```

Figure 18: Adversaries for the proof of Theorem 6.

g instead of the output of the encryption algorithm of the symmetric scheme, and (2) the variable h' is set to $\bot$ instead of the output of the decryption algorithm of the symmetric scheme. We construct an adversary A_{SE} such that

$$\Pr[\mathrm{G}_1(A_{\mathsf{iF}})] - \Pr[\mathrm{G}_2(A_{\mathsf{iF}})] \leq \mathbf{Adv}_{\mathsf{SE}}^{\mathrm{ae2}}(A_{\mathsf{SE}}) .$$

This gives us that

$$\begin{aligned}
\mathbf{Adv}_{\mathsf{iF},\mathsf{DE}}^{\mathrm{iprf}}(A_{\mathsf{iF}}) &= 2 \cdot \Pr[\mathrm{G}_1(A_{\mathsf{iF}})] - 1 \\
&= 2 \cdot \Pr[\mathrm{G}_2(A_{\mathsf{iF}})] - 1 + 2 \cdot (\Pr[\mathrm{G}_1(A_{\mathsf{iF}})] - \Pr[\mathrm{G}_2(A_{\mathsf{iF}})]) \\
&\leq 2 \cdot \Pr[\mathrm{G}_2(A_{\mathsf{iF}})] - 1 + 2\mathbf{Adv}_{\mathsf{SE}}^{\mathrm{ae2}}(A_{\mathsf{SE}}) .
\end{aligned}$$

The games G_2 and G_3 are identical-until-bad games, which means that by the Fundamental Lemma of Game Playing, we can write

$$\Pr[\mathrm{G}_2(A_{\mathsf{iF}})] - \Pr[\mathrm{G}_3(A_{\mathsf{iF}})] \leq \Pr[\mathrm{G}_3(A_{\mathsf{iF}}) \text{ sets } \mathsf{bad}] .$$

We construct an adversary A_{H} against cau-security such that

$$\Pr[\mathrm{G}_3(A_{\mathsf{iF}}) \text{ sets } \mathsf{bad}] \leq \mathbf{Adv}_{\mathsf{iHF}}^{\mathrm{cau}}(A_{\mathsf{H}}) .$$

This gives us that

$$\begin{aligned}\mathbf{Adv}^{\mathrm{iprf}}_{\mathsf{iF},\mathsf{DE}}(A_{\mathsf{iF}}) &\leq 2\Pr\left[\mathrm{G}_2(A_{\mathsf{iF}})\right] - 1 + 2 \cdot \mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}}(A_{\mathsf{SE}}) \\ &= 2\Pr\left[\mathrm{G}_3(A_{\mathsf{iF}})\right] - 1 + 2\left(\Pr\left[\mathrm{G}_2(A_{\mathsf{iF}})\right] - \Pr\left[\mathrm{G}_3(A_{\mathsf{iF}})\right]\right) + 2\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}}(A_{\mathsf{SE}}) \\ &\leq 2\Pr\left[\mathrm{G}_3(A_{\mathsf{iF}})\right] - 1 + 2\mathbf{Adv}^{\mathrm{cau}}_{\mathsf{iHF}}(A_{\mathsf{H}}) + 2\mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}}(A_{\mathsf{SE}})\ .\end{aligned}$$

The game G_4 differs from game G_3 in two ways – (1) the VF oracle now returns false when $b = 1$, and (2) t^1 is sampled at random from the range set. Games G_3 and G_4 are equivalent, since the output of the incremental hash function will never be $\perp$, and since the h values queried to g will always be unique, and therefore can be sampled lazily. Therefore, we have that $\Pr\left[\mathrm{G}_3(A_{\mathsf{iF}})\right] = \Pr\left[\mathrm{G}_4(A_{\mathsf{iF}})\right]$. Furthermore, notice that the game G_4 responds in the same fashion irrespective of whether the bit b is 0 or 1. This means that $\Pr\left[\mathrm{G}_4(A_{\mathsf{iF}})\right] = 1/2$. We then get

$$\begin{aligned}\mathbf{Adv}^{\mathrm{iprf}}_{\mathsf{iF},\mathsf{DE}}(A_{\mathsf{iF}}) &\leq 2\Pr\left[\mathrm{G}_3(A_{\mathsf{iF}})\right] - 1 + 2 \cdot \mathbf{Adv}^{\mathrm{cau}}_{\mathsf{iHF}}(A_{\mathsf{H}}) + 2 \cdot \mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}}(A_{\mathsf{SE}}) \\ &= 2\Pr\left[\mathrm{G}_4(A_{\mathsf{iF}})\right] - 1 + 2 \cdot \mathbf{Adv}^{\mathrm{cau}}_{\mathsf{iHF}}(A_{\mathsf{H}}) + 2 \cdot \mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}}(A_{\mathsf{SE}}) \\ &= 2 \cdot \mathbf{Adv}^{\mathrm{cau}}_{\mathsf{iHF}}(A_{\mathsf{H}}) + 2 \cdot \mathbf{Adv}^{\mathrm{ae2}}_{\mathsf{SE}}(A_{\mathsf{SE}})\ ,\end{aligned}$$

which is the claimed inequality. Note that the adversary A_{H} makes $q_t + q_u + q_v$ queries to its HASH oracle, and the adversary A_{SE} makes $q_t + q_u$ queries to its ENC oracle and q_v queries to its DEC oracle. This completes the proof. ▮