Incremental Cryptography Revisited: PRFs, Nonces and Modular Design

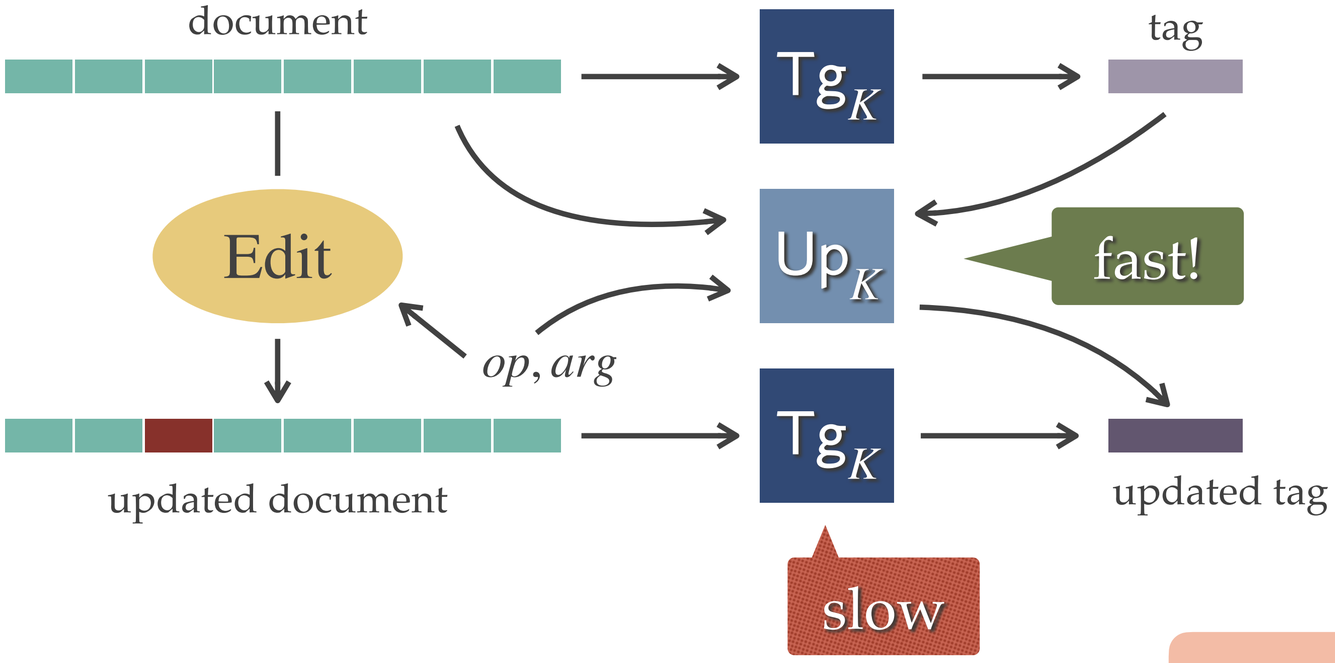
Vivek Arte - UCSD

Mihir Bellare - UCSD

Louiza Khati - ANSSI

https://eprint.iacr.org/2020/1360

Motivation



WITHOUT INCREMENTALITY

tag needs to be computed from scratch, using the entire document!

WITH INCREMENTALITY [BGG95]

use document, previous tag and edit details to construct updated tag

This can be pretty inefficient for large documents that require frequent updates

message authentication
[BGG94, BGG95, Fis97a, KV19]

encryption
[BGG95, BKY02]

authenticated encryption
[SY16]

collision-resistant hashing
[BGG94, BM97, MGS15]

incremental cryptography

digital signatures
[BGG94, Mic97, Fis97b]

program obfuscation
[GP17]

deterministic public-key encryption
[MPRS12]

pseudo-random functions
[this work]

Why IPRFs?

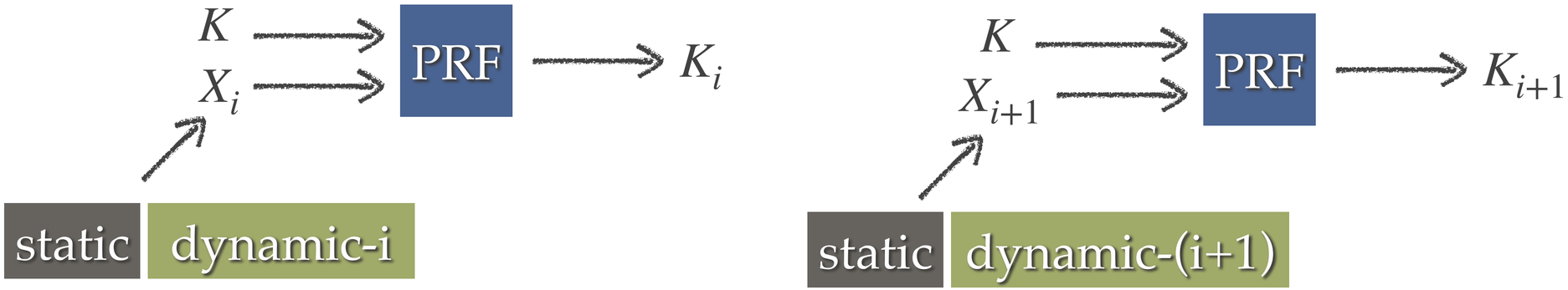
incremental cryptography

pseudo-random functions
[this work]

Allows for efficiency improvement via incrementality in a broader range of applications

those needing pseudo-randomness

Consider **key-derivation**:



An incremental PRF will allow will allow updating from K_i to K_{i+1} faster than computing it from scratch

Contributions

Definitions

Tools for Modular Design

Constructions

Contributions

Definitions

- *introduce* incremental pseudorandom functions (IPRFs) *and* incremental function families (iFFs)

STAY TUNED!

nonce-based

- security notions *for iFFs*

IUF : incremental unforgeability

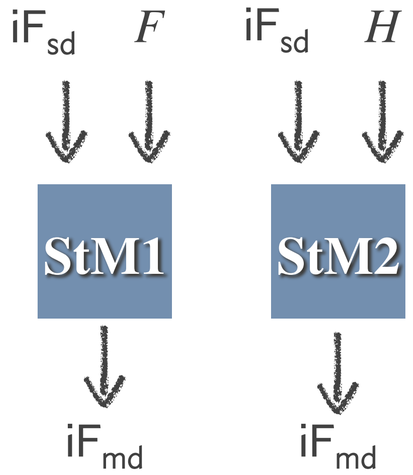
IPRF : incremental pseudorandomness

Contributions

Tools for Modular Design

Single-document schemes : *secure when only a single document is considered*

Multi-document schemes : *secure even when multiple documents are considered*



- transforms *to convert single-document schemes into multi-document schemes*
  - **StM1** — *works for* all *edit operations*, non-tight security reduction
  - **StM2** — *works for* a large class of *edit operations*, tight security reduction
  - *work for both IUF and IPRF security*

- transform *to build single-document IPRF schemes from incremental hash functions*
  - *extends the Carter-Wegman paradigm* [WC81] *to the incremental setting*
  - **iHtE** uses incremental hash function iHF and symmetric encryption scheme SE

SE uses the NBE2 syntax of [BNT19]

Contributions

Constructions

- *applies modular design tools to build IPRFs out of existing message authentication schemes*
  - extract *underlying incremental hash function, then use* **iHtE** *to build single-document IPRF*

Document Editing Systems

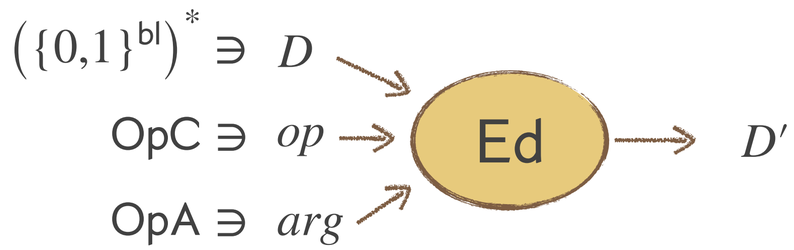
non-cryptographic object

- describes document format
- defines set of **edit operations**

Examples of edit-operations on document $D =$

$D[1]$	…	$D[i-1]$	$D[i]$	$D[i+1]$	…	$D[m]$

op	*arg*	D'
replace	(i, x)	$D[1]$ … $D[i-1]$ $D[i]$ $D[i+1]$ … $D[m]$



Document Editing Systems

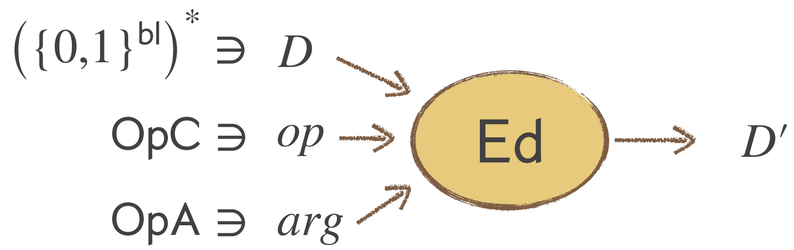
non-cryptographic object

- describes document format
- defines set of **edit operations**

Examples of edit-operations on document $D =$

$D[1]$	…	$D[i-1]$	$D[i]$	$D[i+1]$	…	$D[m]$

op	*arg*	D'
replace	(i, x)	$D[1]$ … $D[i-1]$ x $D[i+1]$ … $D[m]$
insert	(i, x)	$D[1]$ … $D[i-1]$ $D[i]$ $D[i+1]$ … $D[m]$



Document Editing Systems

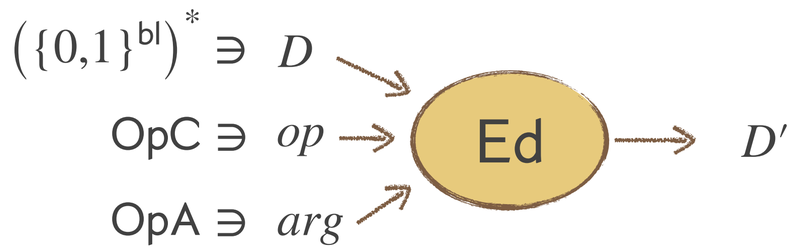
non-cryptographic object

- describes document format
- defines set of **edit operations**

Examples of edit-operations on document $D =$

$D[1]$	…	$D[i-1]$	$D[i]$	$D[i+1]$	…	$D[m]$

op	*arg*	D'
replace	(i, x)	$D[1]$ … $D[i-1]$ x $D[i+1]$ … $D[m]$
insert	(i, x)	$D[1]$ … $D[i-1]$ x $D[i]$ $D[i+1]$ … $D[m]$
delete	i	$D[1]$ … $D[i-1]$ $D[i]$ $D[i+1]$ … $D[m]$

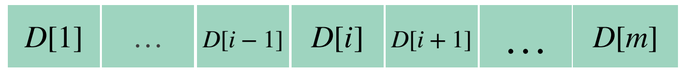


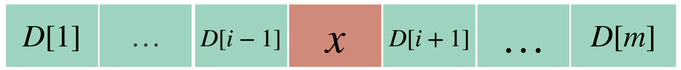
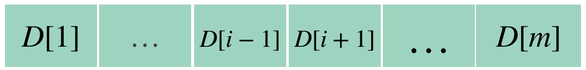
Document Editing Systems

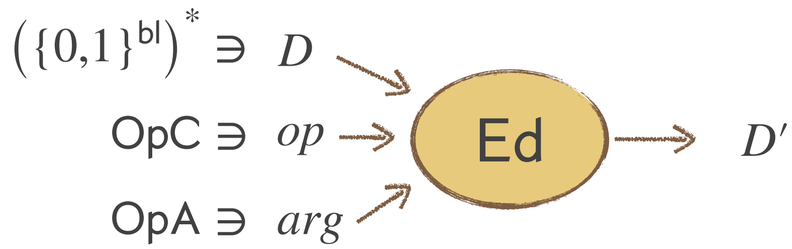
non-cryptographic object

- describes document format
- defines set of **edit operations**

Examples of edit-operations on document $D =$



op	*arg*	*D′*
replace	(i, x)	
insert	(i, x)	
delete	i	



$$DE = (bl, BS, OpC, OpA, Ed)$$

- block length
- block space $= \{0,1\}^{bl}$
- edit-operation codes
- edit-operation arguments
- edit function

Incremental Function Families (iFFs)

Tagging algorithm — Takes the key, a nonce, the document ID, and the document, and produces a tag

$t \leftarrow \mathsf{Tg}(K, N, id, D)$

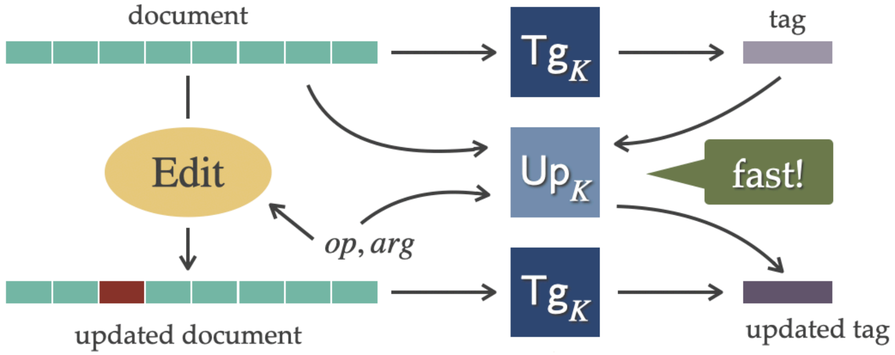
Update algorithm — Takes the key, a nonce, the document ID, the document, the edit details, and the original tag, and updates the tag

$t' \leftarrow \mathsf{Up}(K, N, id, D, op, arg, t)$

Verification algorithm — Takes the key, the document ID, the document, and the tag and returns whether or not the verification succeeded

$d \leftarrow \mathsf{Ver}(K, id, D, t)$

Verification does not use the nonce!



An iFF is defined for a document editing system DE

$$\mathsf{iF} = (\mathsf{KS}, \mathsf{NS}, \mathsf{Rng}, \mathsf{Tg}, \mathsf{Up}, \mathsf{Ver})$$

key space, nonce space, output space, tagging algorithm, update algorithm, verification algorithm

We assume the identity space to be the set of all possible bitstrings, {0,1}*

Nonces [RBBK01, Rog02]

non-repeating quantities that may be picked by an adversary

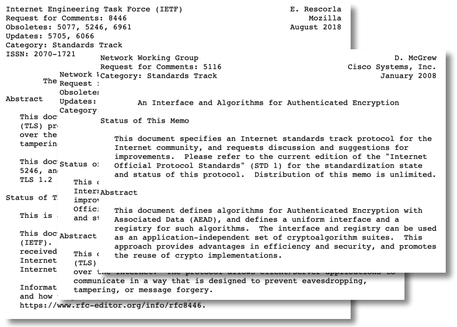
algorithms are deterministic but take a nonce as input

Randomized algorithms can be captured by picking the nonce at random and having the algorithm use the nonce as the randomness

Stateful algorithms can be captured by letting the nonce be the state

Using nonces improves robustness by maintaining security for arbitrary (non-repeating) nonces — precluding issues arising due to randomness failure

Nonce-based PRFs : allow for capturing more constructions, and increase applicability


Internet Engineering Task Force (IETF) E. Rescorla
Request for Comments: 8446 Mozilla
Obsoletes: 5077, 5246, 6961 August 2018
Updates: 5705, 6066
Category: Standards Track
ISSN: 2070-1721

Network Working Group D. McGrew
Request for Comments: 5116 Cisco Systems, Inc.
Category: Standards Track January 2008

An Interface and Algorithms for Authenticated Encryption

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Abstract

This document defines algorithms for Authenticated Encryption with Associated Data (AEAD), and defines a uniform interface and a registry for such algorithms. The interface and registry can be used as an application-independent set of cryptoalgorithm suites. This approach provides advantages in efficiency and security, and promotes the reuse of crypto implementations.

Abstract

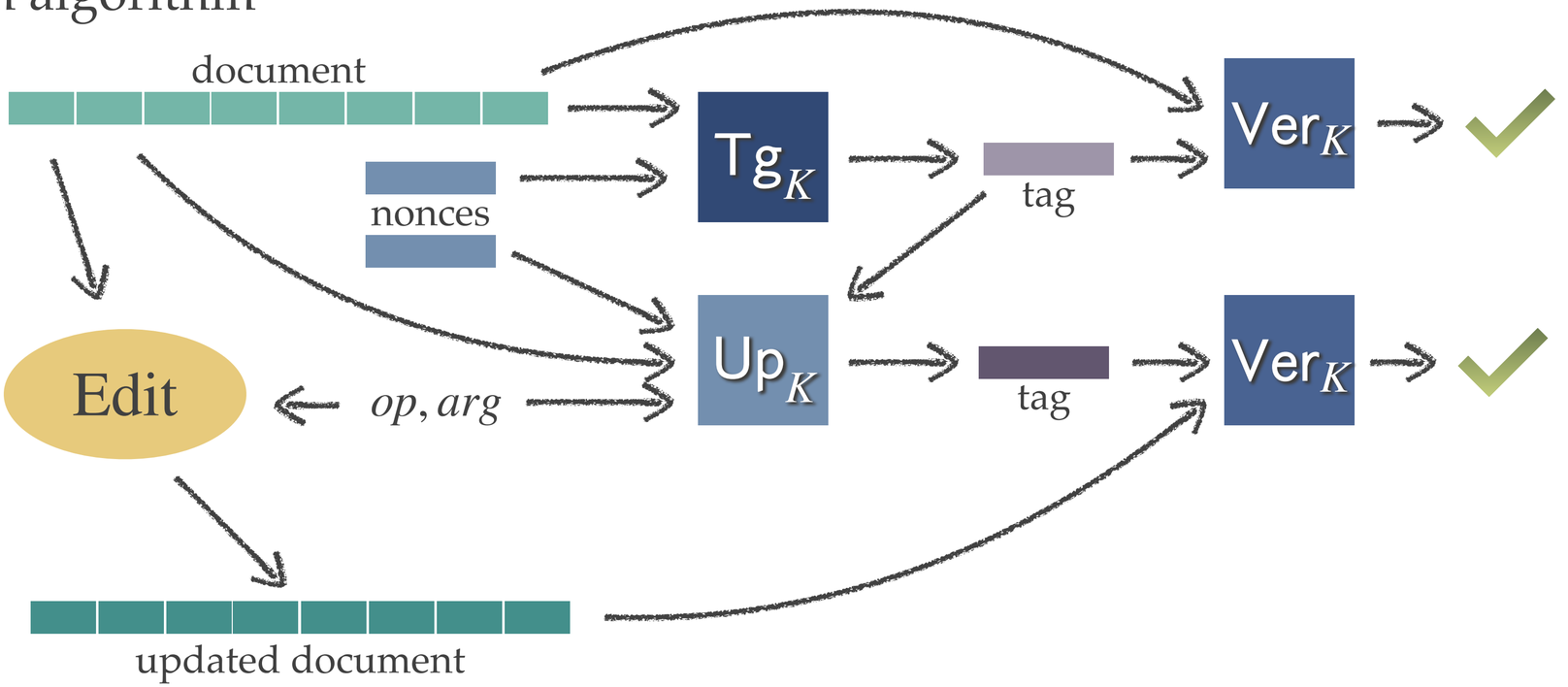
The
Network
Request
Obsolete
Updates:
Category
Status of
This
Inter
impro
Offic
and st
This
(IETF).
received
Internet
(TLS)
over
communicate in a way that is designed to prevent eavesdropping,
tampering, or message forgery.
Informat
and how
https://www.rfc-editor.org/info/rfc8446.

Nonces are **widely used** in various standards such as those for authenticated encryption [RFC 5116] and in the TLS standard [RFC 5246, RFC 8446]

Correctness of Updates

Tags generated by legitimate applications of the tagging and update algorithms must be accepted by the verification algorithm

This should hold even if nonces are repeated



Correctness of Updates

Tags generated by legitimate applications of the tagging and update algorithms must be accepted by the verification algorithm

This should hold even if nonces are repeated

STRONG CORRECTNESS

The value of this is that it allows updates to be dropped

We introduce this notion

Here, tags produced via the update algorithm must exactly match tags produced by the tagging algorithm **with the same nonce**

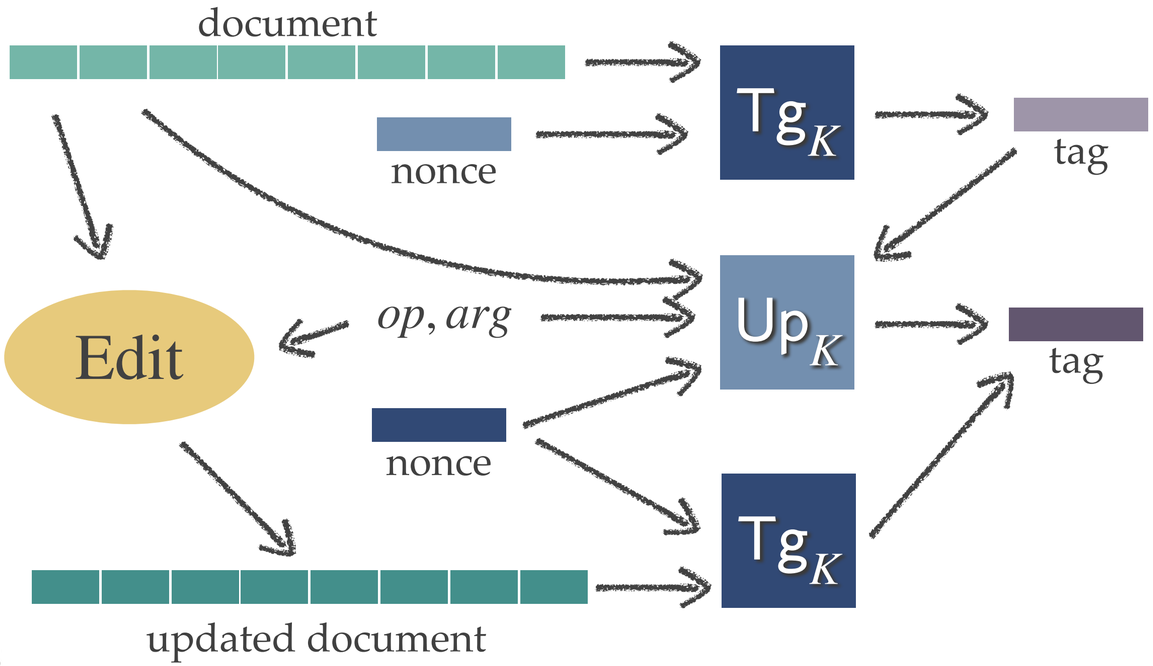
$$t = \mathsf{Tg}(K, N', D)$$

$$t' = \mathsf{Upd}(K, N, D, op, arg, t)$$

$$t' = \mathsf{Tg}(K, N, D_{new})$$

where

$$D_{new} = \mathsf{Ed}(D, op, arg)$$



Incremental UF security

This corresponds to the notion of basic security in [BGG95]

The game captures the ability of an adversary to generate a valid tag for a new document after seeing the tags for it's choice of documents and updates

The adversary wins if it makes a VF query with a new document-*id* pair and a tag that verifies

The adversary is not allowed to repeat nonces

For an *id*, UPD queries must be made after an initial TAG query

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

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 win $\leftarrow$ true
19 **return** d

oracle FIN
20 **return** win

$$\mathbf{Adv}^{\mathrm{iuf}}_{\mathsf{iF,DE}}(A) = \Pr\left[G^{\mathrm{iuf}}_{\mathsf{iF,DE}}(A)\right]$$

Incremental PRF security

The game captures the ability of an adversary to distinguish between an IPRF and a random function after seeing the outputs for it's choice of documents and updates

The adversary wins if it is able to distinguish between the IPRF and a random function

In the case of a random function, the VF oracle always returns false

The adversary is not allowed to repeat nonces

For an *id*, UPD queries must be made after an initial TAG query

Game $G^{\text{iprf}}_{\text{iF,DE}}$

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

oracle TAG(N, id, D)
3 **if** $(N \in \text{NL}_{id}$ 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 $\text{NL}_{id} \leftarrow \text{NL}_{id} \cup \{N\}$
8 $\text{DL}_{id} \leftarrow \text{DL}_{id} \cup \{D_{id}\}$
9 **return** t^b_{id}

oracle UPD(N, id, op, arg)
10 **if** $D_{id} = \perp$ **then return** $\perp$
11 **if** $(N \in \text{NL}_{id}$ 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 $\text{NL}_{id} \leftarrow \text{NL}_{id} \cup \{N\}$
17 $\text{DL}_{id} \leftarrow \text{DL}_{id} \cup \{D_{id}\}$
18 **return** t^b_{id}

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

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

$$\mathbf{Adv}^{\text{iprf}}_{\text{iF,DE}}(A) = 2 \cdot \Pr\left[G^{\text{iprf}}_{\text{iF,DE}}(A)\right] - 1$$

Why do we need to consider Updates separately? [BGG95]

The adversary may have access to previous versions of documents and their tags

Further, it may be able to issue edit commands to existing documents and obtain new incremental signatures

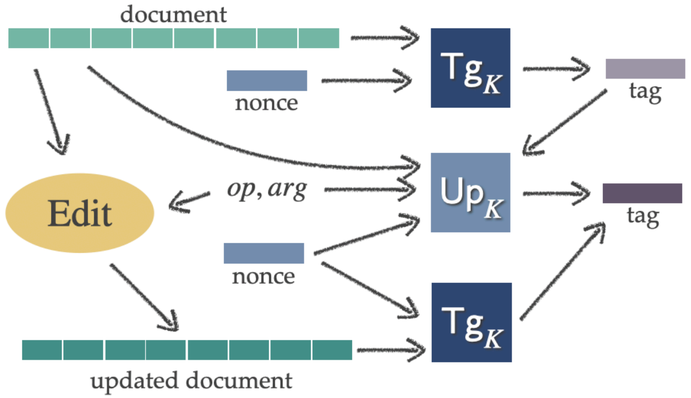
This may allow for attacks that break schemes that cannot be broken when restricted to not using the incremental update algorithm

DROPPING UPDATES

Updates can be dropped in the case where the scheme under consideration satisfies **strong correctness**

This holds for both IUF and IPRF security

Calls for updates will be answered by updating the document, and then using the tagging algorithm



IPRF $\Longrightarrow$ IUF

PREVIOUS DEFINITIONS

Regular setting (no nonces, no incrementality) — PRF $\Longrightarrow$ UF [BKR00, GGM86]

Nonce-based setting (no incrementality) — PRF $\not\Rightarrow$ UF [PS16]

Our IPRF security notion implies the IUF security notion

Let iF be an incremental function family for document editing system DE. Let A_{uf} be an adversary against IUF security of iF. Then we can construct adversary A_{prf} against IPRF security of iF such that

$$\mathbf{Adv}^{\text{iprf}}_{\text{iF,DE}}\left(A_{\text{prf}}\right) = \mathbf{Adv}^{\text{iuf}}_{\text{iF,DE}}\left(A_{\text{uf}}\right)$$

A_{prf} makes the same number of queries as A_{uf} and has similar running time

The benefit is that an incremental function family shown to satisfy IPRF security can directly be used for message authentication

In order to achieve this implication, we include a verification oracle in our IPRF game

Single-document and Multi-document

Single-document scheme : when only one document is being edited

definitions of [BGG94]

Multi-document scheme : when many documents, with different *id*s are being edited

defined in [BGG95]

more useful in practice

stronger

We provide **two transforms** that take a scheme that is secure in the single-document setting, and return a scheme secure in the multi-document setting

Transform	Auxiliary tool	Security Reduction
StM1	PRF	non-tight
StM2	hash function	tight

Transform **StM1**

Let $\mathsf{iF}_{\mathrm{md}}$ denote the multi-document iFF constructed by the transform using $\mathsf{iF}_{\mathrm{sd}}$, a single-document iFF, and F, a PRF.

$$\mathsf{iF}_{\mathrm{md}} = \mathbf{StM1}\left[\mathsf{iF}_{\mathrm{sd}}, F\right]$$

Algorithm generates $K_{id} \leftarrow F(K, id)$

Uses the single-document scheme with the above different key for each distinct id

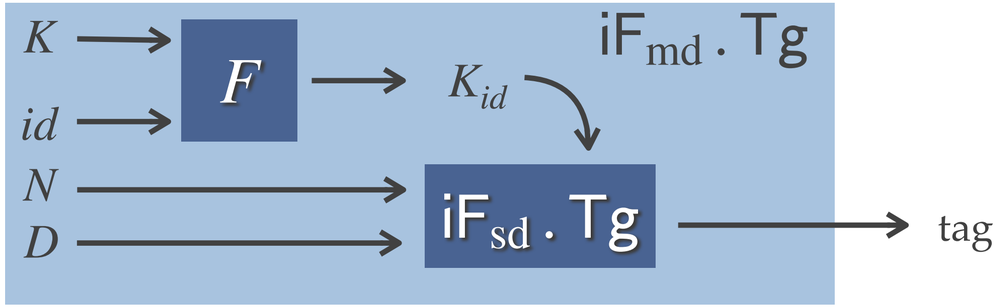
Given adversary A against the IPRF security of $\mathsf{iF}_{\mathrm{md}}$ relative to DE, we can construct adversary A_1 against the IPRF security of $\mathsf{iF}_{\mathrm{sd}}$ relative to DE and adversary B against the PRF security of F such that

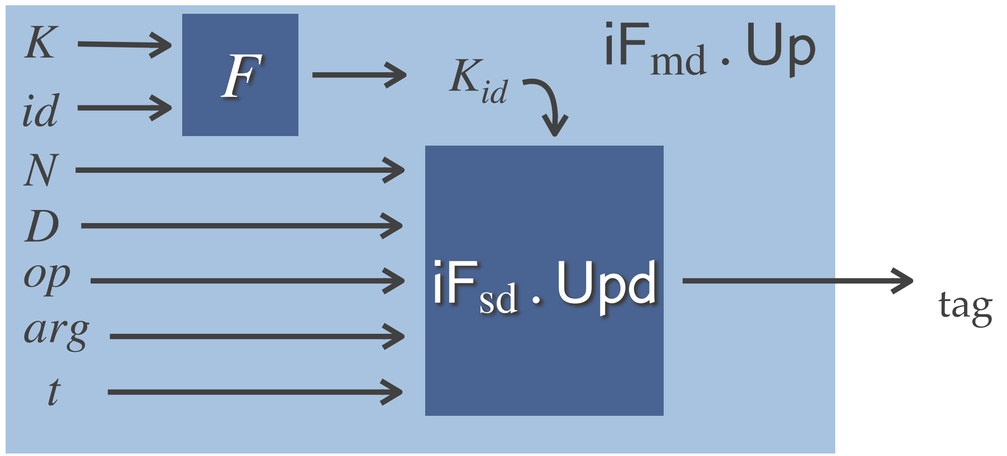
$$\mathbf{Adv}^{\mathrm{iprf}}_{\mathsf{iF}_{\mathrm{md}},\mathsf{DE}}(A) \le q \cdot \mathbf{Adv}^{\mathrm{iprf}}_{\mathsf{iF}_{\mathrm{sd}},\mathsf{DE}}(A_1) + \mathbf{Adv}^{\mathrm{prf}}_{F}(B)$$

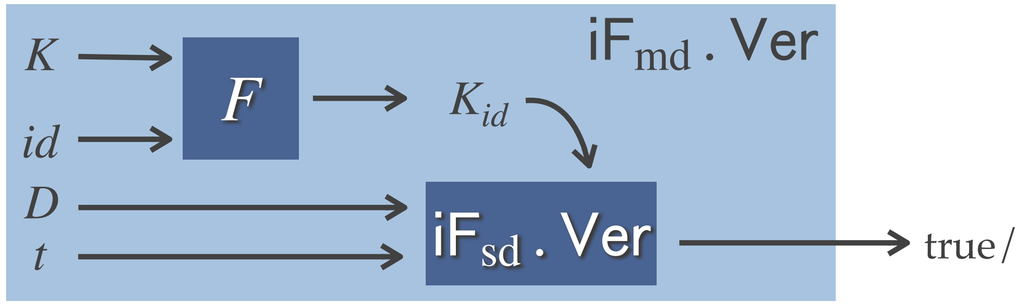
number of distinct identities queried

Similar result holds for IUF security

The reduction is not tight







Transform **StM2**

Let $\mathsf{iF}_{\mathrm{md}}$ denote the multi-document iFF constructed by the transform using $\mathsf{iF}_{\mathrm{sd}}$, a single-document iFF, and H, a variable length hash function.

$$\mathsf{iF}_{\mathrm{md}} = \mathbf{StM2}\left[\mathsf{iF}_{\mathrm{sd}}, H\right]$$

Use the variable-length hash function to hash the *id* and the nonce as follows

$$d \leftarrow H(id, \mathsf{bl}) \qquad N' \leftarrow H(id \| N, \mathsf{nl})$$

Prepend d to the start of the document

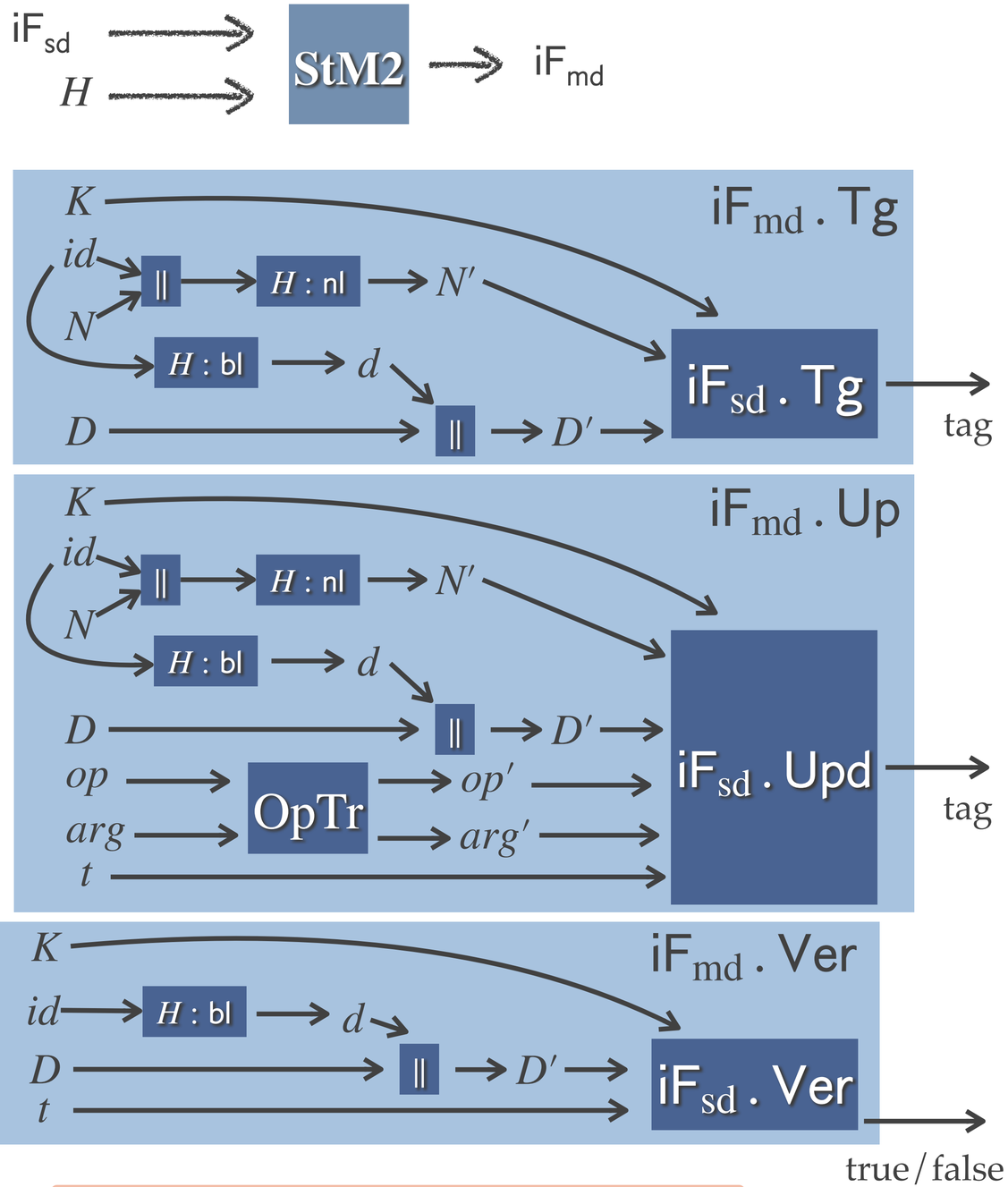
Use the single-document scheme with the resulting document and the new nonce N'

Given adversary A against the IPRF security of $\mathsf{iF}_{\mathrm{md}}$ relative to DE, we can construct adversary A_1 against the IPRF security of $\mathsf{iF}_{\mathrm{sd}}$ relative to DE and adversaries B_1, B_2 against the collision-resistance security of H such that

$$\mathbf{Adv}^{\mathrm{iprf}}_{\mathsf{iF}_{\mathrm{md}},\mathsf{DE}}(A) \le \mathbf{Adv}^{\mathrm{iprf}}_{\mathsf{iF}_{\mathrm{sd}},\mathsf{DE}}(A_1) + \mathbf{Adv}^{\mathrm{cr}}_{H,\mathsf{bl}}(B_1) + \mathbf{Adv}^{\mathrm{cr}}_{H,\mathsf{nl}}(B_2)$$

Similar result holds for IUF security

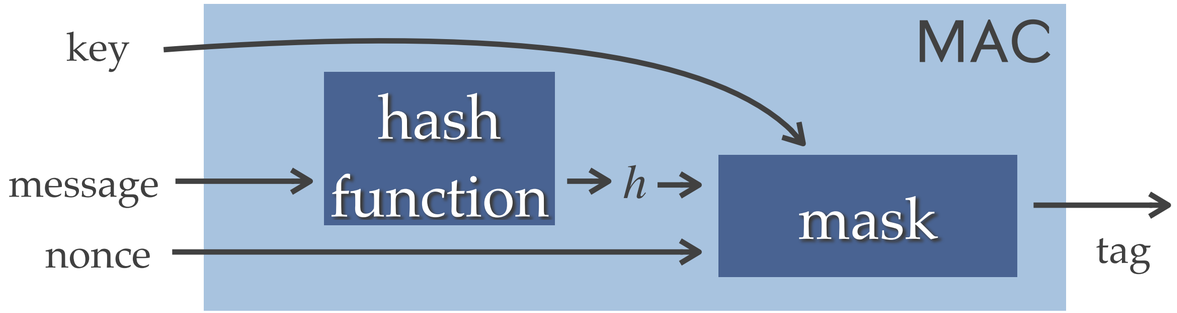
The reduction is tight



Edit operations must be translating

The Carter Wegman Paradigm [WC81]

The Carter Wegman paradigm is used to build message authentication schemes



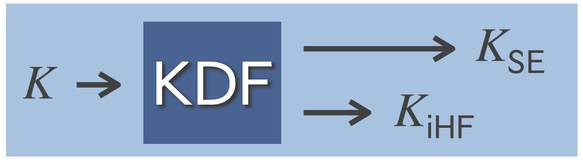
UMAC [BHKKR99], GMAC [MV04], and VMAC [KD07] are some examples of popular message authentication schemes that are based on the Carter Wegman paradigm

incremental-Hash-then-Encrypt (**iHtE**)

This is our extension of the Carter Wegman paradigm to the incremental setting

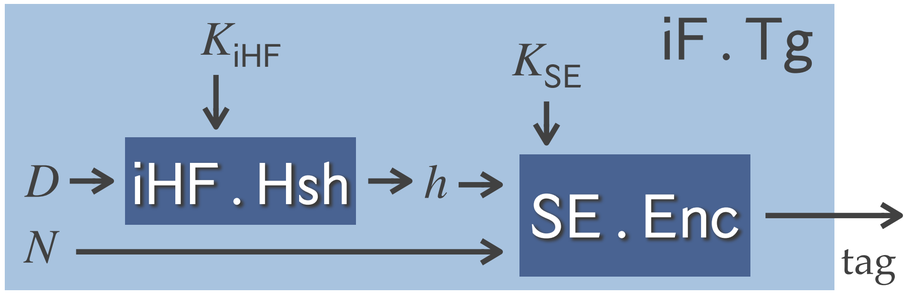
Let iF denote the iFF constructed by the transform using an incremental hash function iHF, a symmetric encryption scheme SE, and KDF, a key distribution function.

SE uses the NBE2 syntax of [BNT19]

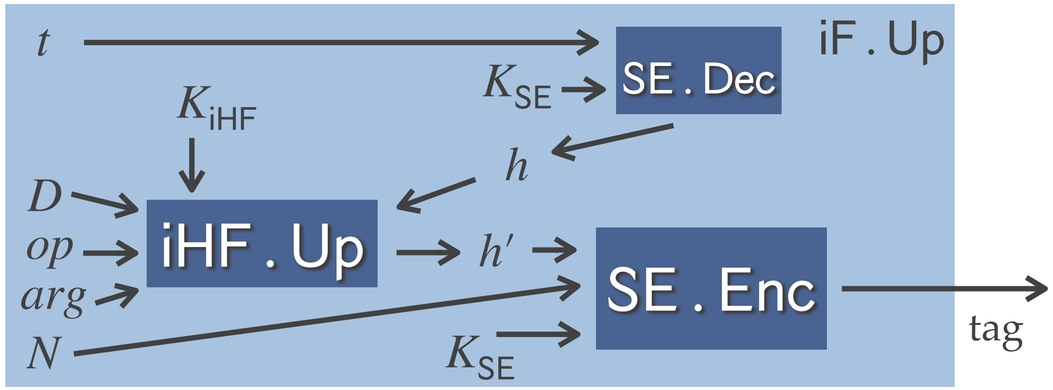


iHF is assumed to be incremental for a document editing system DE that includes the *replace* operation

We use **iHtE** to extend the incrementality of iHF to the resulting incremental function family iF and also obtain IPRF security for iF in the single-document setting.



NBE2 - nonces not needed for decryption
Necessary for performing iF updates



Instantiations

We study existing message authentication schemes and use them to construct incremental function families for the *replace* operation

For PMAC1 and PMAC, we obtain IPRF security directly

For XORMAC, GMAC, Poly1305-AES and PWC, we obtain IUF security via the natural expression in our syntax

These require the nonce to be sent with the tag, hence IPRF security does not hold

For the above schemes, we can extract an incremental hash function, and then use the **iHtE** transform to get IPRF security

The PMAC_Plus and ZMAC schemes are not incremental. We extract an incremental hash function from these schemes and then use the **iHtE** transform for IPRF security

Message Authentication Scheme (M)	**iFF**	**Security**	
		IUF	**IPRF**
PMAC1 [Rog04]	iF_{M}	Yes	Yes
PMAC [BR02]	iF_{M}	Yes	Yes

Summary

- We defined incremental function families within a nonce-based framework
- We introduced strong correctness of iFFs as a property to reduce proof complexity
- We defined notions of security (IUF and IPRF) for incremental function families
- We showed that IPRF security implies IUF security for an incremental function family
- We constructed two transforms that take a scheme secure for a single document, and return a scheme secure for multiple documents
  - This allows us to focus on building the easier, single-document schemes
- We constructed a transform that takes an incremental hash function, and return a scheme that is IPRF secure for single documents for the replace operation
  - This allows us to focus on building the incremental hash functions for this scenario
- We extract incremental hash functions from various existing message authentication schemes, and use them to build secure IPRFs for the replace operation

https://eprint.iacr.org/2020/1360