UCLouvain
Institute for Information
and Communication Technologies,
Electronics and Applied Mathematics

Ecole polytechnique de Louvain
ICTEAM Institute
UCL Crypto Group

Authentication in the presence of side-channel leakage

Francesco Berti

Thèse présentée en vue de l'obtention du grade de
docteur en sciences de l'ingénieur.

Composition du jury :
Pr. Christophe Craeye (UCLouvain) - Président du Jury
Pr. Michel Abdalla (ENS)
Pr. Alexandre Duc (HEIG-VD)
Pr. Oliver Pereira (UCLouvain) - Promoteur
Dr. Thomas Peters (UCLouvain)
Pr. François-Xavier Standaert (UCLouvain) - Promoteur

Louvain-la-Neuve, Belgique – 2020

alla mia mamma

Acknowledgments

This thesis would have not been possible without the help, support and guidance of many people. In fact I have been taught so much, so many supported me and I have passed so many nice moments. Moreover, I have spent 5 wonderful years in Belgium. Thus, I thank them all. I hope not to forget anyone (I apologize if I have forgotten anyone).

First of all, I would like to thank my supervisor, *Pr.François-Xavier Standaert* who chose me as one of his Ph.D. students 5 years ago. He is very helpful, he has given me so many good suggestions and he has helped me to grew as a researcher. I really appreciated working under his guidance. I feel that he is able to understand what I was able to give. I would also like to thank my cosupervisor *Prof. Olivier Pereira* who helped me with many corrections, who improved my scientific writing.
Moreover, I would like to thank *Dr. Thomas Peters* for all the discussions, corrections and suggestions. I would like also to thank *Proff. Michel Abdalla* and *Alexandre Duc* for having accepted to follow my thesis and to be in my jury. Their commentaries and their questions helped me to make my manuscript better. Finally, I would like to thank *Prof. Cristophe Craye* for having accepted to be the president of the jury for this thesis.

I would also thank *Proff. François-Xavier Standaert* and *Olivier Pereira* for having created a wonderful research group. Not only are there many great researcher, but also there is a wonderful atmosphere which made me really enjoy these years at work. Moreover, they were always supportive in the difficult moments (to name only an episode, I was (and still I am) really touched by the fact that *Itamar* and *Davide* came to the airport to pick me up when I was back from the funeral of my mum).
Parmi mes collègues, je voudrais commencer par remercier *Florentin* qui a partagé avec moi le bureau pour presque toute ma thèse. Il a pu ap-

précier l'expressivité et la gestualité italienne ;)[1]. Je me souviens de nombreuses taquinieries d'*Édouard*„[2], mais aussi de sa chaleur humain. Je remercie *François* pour son aide et les soirées jeux ensemble. I would like to than *Itamar* for his friendship, for the dinners together with his family. Ringrazio *Davide* che mi ha permesso infine di poter discutere di lavoro in italiano! Et puis, j'aimerais remercier tous les autres, *Alizée*, *Anthony*, *Charles*, *Chun*, *Clément*, *Dina*, *Gaëtan*, *Kashif*, *Olivier*, *Pierrick*, *Qian*, *Romain*, *Tanya*, *Sebastien*, *Tobias*, *Vincent* et *Weija* avec lesquels j'ai partagé plein d'idées. On a fait des TP ensembles, on a passé plein des pauses à la cafeteria (avec les mots fléchés...) et on a fait des beermeetings les soirs. Je remercie *Sylvie* et *Viviane* qui m'ont toujours aidé à surmonter les problèmes bureacratiques.
Enfin, j'aimerais remercier les autres membres de l'UCL (comme *Anne-Sophie*, *Jean*, *Pierre-Yves* et *Stéphanie*, pour en nommer seulement quelques-uns) qui m'ont accueilli et avec lesquels j'ai passé plein de chouettes moments.

Ringrazio la mia famiglia, *mio papà* che mi ha sempre supportato e ha sempre creduto in me. *Mio fratello Federico* con cui sono cresciuto insieme e che infine, da quando ho iniziato il dottorato, non ha più potuto "apprezzare" la mia presenza costante[3]. Ringrazio i miei nonni *Gianna* e *Alberto*, i miei *zii Andrea* e *Cesarina* ed i miei *cugini Giulio*, *Virginia* e *Saverio* che mi hanno sempre voluto bene e sostenuto.

Ringrazio i miei amici storici di matematica *Filo*, *Giulio*, *Pietro*, *Simo* (mi mancano le serate Hearts ed il ranking), *Ali*, *Giulia*, *Pengo*, *Guido*, *Ari R.*, *Chiara* (e la piccola *Alma*), *Elena* e *Mascotto* con cui è sempre bello vedersi (anche se purtroppo troppo raramente).
Ringrazio *Irvin* che conosco da così tanto tempo e con cui è sempre bello rivedersi e sciare.
Ringrazio i miei amici della montagna *Annie*, *Camilla* e *Angelo* con cui ci si rivede sempre almeno d'estate e.
Ringrazio *Monica* e *Matteo* con *Gabriele*, *Marta*, *Giacomo*, *Francesco* e *Lucia* per la compagnia tutte le estati.

Je remercie mes premiers colocataires, pour le premières deux années, *David*, *François*, *Parfait*, *Pierrot*, *Sergio* et *Vincent* et les autres qui m'ont très bien accueilli, m'ont beacoup aidé à apprendre le français

[1] Il m' a accueilli avec beaucoup de chaleur. On s'est bien amusé

[2] Par example parce qu'on n' est pas allé en Roussie en 2018, mais je te rappelle qu'à l'Euro 2016 Italie 2 -0 Belgique 0 [Giaccherini 32', Pellé 90'+2] ;)

[3] Infine ha avuto TV, PC, ecc... tutti per sè.

et avec lesquels j'ai passé plein de beaux moments.

Ringrazio *Alessandro*, *Chiara*, *Cristina*, *Floriana* e *Vincenzo* con cui abbiamo girato insieme il Belgio (e un po' d'Irlanda).
Ringrazio *Daniele* con cui abbaimo visto così tante partite insieme in Belgio.

Ringrazio *Fabio*, compagno di arrampicata [4] et sa femme *Marie*. On a passé plein de chouettes moments ensemble.

Je remercie *Adam*, *Agathe*,*Alessandra*, *Alicia*, *Antonio*, *Apolline*[5], *Astrid Carlotta*, *Charles*, *Constance*, *Donatien*[6], *Émeline*, *Fei*, *Florent*, *Florian*, *Giorgia*, *Greg*, *Guillaume Hélène*, *Jean-Aleandre*, *Laia*, *Laurence*, *Laurent*, *Louis*, *Louise*, *Manu*, *Marge Matoux*, *MG*, *MdN*, *Patrick*, *Pauline*, *Sara*, *Sarah Simon*[7],*Tanguy* et *Zbigniew*[8], qui ont été avec moi ces trois dernières années. J'ai vraiment apprecié cette période.
Je remercie *Louis* pour avoir géré un super jeu qui nous[9] pris pour 2 ans.
Je remercie *Anne-Marie* pour s'être autant attachée à moi.
Je remercie *Hugues*[10], *Letizia* et *Virginia*.
Je remercie *Charles-Hubert*, *Louis* et *Sara* pour toutes les soirées jeux.

Je remercie *Sara* avec laquelle je m'amuse toujours et qui me rend heureux.

Infine, vorrei ringraziare le persone che erano con me all'inizio di questa tesi e che oggi non ci sono piú e non possono leggere questa tesi. Il mio *Nonno Giovanni* che era così orgoglioso di suo nipote e ogni volta che lo chiamavo dal Belgio era felicissimo. La mia *nonna Elda* che quando andavo a trovarla era così felice e che mi ha sempre incitato a studiare. Per ultima, la *mia mamma*, che mi ha sempre seguito sin da bambino nel mio percorso di studi, che mi ha voluto così tanto bene e che nonostante gli mancassi molto e le facesse tristezza che non fossi più vicino a lei, era così felice che fossi in Belgio per il mio dottorato.

[4] speriamo di vedere una volta un titolo piloti per la Ferrari prima o poi
[5] Avec un p et deux l ;)
[6] Merci pour toutes les balades à vélo et en montagne, et descentes avec les skis faites ensemble!
[7] Il confinamento non è stato male, con i campionatie Champions vinte con il nostro Pescara di capitan Tonali!!
[8] et nombreuses autres
[9] *Donatien*, *Mathilde* et *Tanguy*
[10] Merci pour les partie d'échec le dimance soir

Abstract

One of the goals of modern cryptography is to prevent an adversary from making forgeries. That is, sending a message which the receiver believes valid while not sent by a genuine sender. For "black box" adversaries only able to access the inputs and outputs a cryptographic algorithm, many efficient solutions exist and provide strong mathematical security guarantees. Over the last decade, various research advances have shown that preventing black box attacks is not sufficient. For example, so-called side-channel adversaries can also access physical quantities produced during the cryptographic computations. Thanks to these physical leakages, very efficient forgery attacks can be performed, for example by extracting the long-term cryptographic keys.
In this thesis, we propose a formal solution to the problem of authenticity in the presence of side-channel leakage. For this purpose, we introduce a new theoretical framework that allows capturing security against side-channel attacks, explain what security we aim for and how we model physical leakages, and then build constructions for which the physical security can be reduced to clear assumptions thanks to rigorous proofs. In particular, our proofs indicate which part of an implementation must be strongly protected against side-channel attacks and which part can leak (sometimes in full) with limited consequences. For example, we show that it is possible to reduce the security of full fledged authentication schemes to standard black box security properties and only requiring strong protections against side-channel attacks for one execution of its underlying cryptographic primitive.

Contents

List of notations

Arithmetic

$\mathbb{N}$	Set of positive integers
$\mathbb{R}$	Set of real numbers
$\mathcal{FUNC}(\mathcal{X}, \mathcal{Y})$	Set of functions $f : \mathcal{X} \to \mathcal{Y}$
$\mathcal{PERM}(\mathcal{X})$	Set of permutations $p : \mathcal{X} \to \mathcal{X}$
$x \xleftarrow{\$} \mathcal{X}$	x is picked uniformly at random in $\mathcal{X}$

Cryptographic notations and abbreviations

NIST	National Institute of Standards and Technology
MI5	British Security Service (Military Intelligence 5)
AES	Advanced Encryption Standard
SPA	Simple Power Attack
DPA	Differential Power Attack
$\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{EE}$	Adversaries
$\mathcal{O}$	Oracle
$\mathcal{KH}$	Keyspace for hash functions
$\mathcal{HM}$	Inputspace for hash functions
$\mathcal{T}$	Outputspace for hash functions
$\mathcal{K}$	Keyspace
$\mathcal{B}$	Blockspace for block ciphers
$\mathcal{TW}$	Tweakspace
$\mathcal{ME}$	Message space for MAC and encryption schemes
$\mathcal{TAG}$	Tag space
$\{0,1\}^n$	Set of n-bits strings
$\{0,1\}^*$	Set of finite strings
$\{0,1\}^\infty$	Set of infinite strings

Black box definition - Chap. 2

H	Hash function
CR	Collision resistance
roPR	Range-oriented pre-image resistance
F	Pseudorandom function (when implemented)
BC	Block cipher
TBC	Tweakable block cipher
PRF	Pseudorandom function
PRP	Pseudorandom permutation
TPRF	Tweakable pseudorandom function
TPRP	Tweakable pseudorandom permutation
sPRP	Strong pseudorandom permutation
sTPRP	Strong tweakable pseudorandom permutation
MAC	Message authentication code
Mac	Tag-generation algorithm
Vrfy	Verification algorithm
euf	Unforgeability
suf	Strong unforgeability
AE	Authenticate Encryption
Π	AE scheme
Enc	Encryption algorithm
Dec	Decryption algorithm
pAE	Probabilistic AE scheme
INT-CTXT	Ciphertext Integrity
RO	Random Oracle
MRAE	Misuse-Resistant AE

Constructions

CCSMAC1	First MAC proposed by Pereira et al. [111]
CCSMAC2	Second MAC proposed by Pereira et al. [111]
PSV	Encryption scheme proposed by Pereira et al. [111]
HBC	Hash-then-BC1
HBC2	Hash-then-BC
HTBC	Hash-then-TBC
DTE	Digest-Tag-and-Encrypt
EDT	Encrypt-Digest-and-Encrypt
CONCRETE	Commit-Encrypt-Send-the-Key

Definition with leakage

suf-L	Strongly existentially unforgeable against chosen message and verification attacks with leakage in tag-generation
suf-L2	Strongly existentially unforgeable against chosen message and verification attacks with leakage in tag-generation and verification
CIML	Ciphertext integrity with misuse and encryption leakage
CIML2	Ciphertext integrity with misuse and (encryption & decryption) leakage
sUL	Strong unpredictability with leakage in evaluation and inversion

List of publications

This list summarizes the academic work realized during the thesis. Only bold titles of following publications are discussed in this dissertation.

Journal publications

1. Francesco Berti, Olivier Pereira, Thomas Peters and François-Xavier Standaert ***On Leakage-Resilient Authenticated Encryption with Decryption Leakages*** - IACR Transactions on Symmetric Cryptology [27].

2. Dina Kamel, François-Xavier Standaert, Alexandre Duc, Denis Flandre and Francesco Berti *Learning with Physical Noise or Errors* - IEEE Transactions on Dependable and Secure Computing 2018 [80].

3. Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters and François-Xavier Standaert TEDT*, a Leakage-Resist* AEAD *Mode for High Physical Security Applications* - IACR Transactions on Cryptographic Hardware and Embedded Systems [21] (eprint version [20]).

4. Davide Bellizia, Francesco Berti, Olivier Bronchain, Gaëtan Cassiers, Sébastien Duval, Chun Guo, Gregor Leander, Gaëtan Leurent, Itamar Levi, Charles Momin, Olivier Pereira, Thomas Peters François-Xavier Standaert, Balazs Udvarhelyi and Friedrich Wiemer *Spook: Sponge-Based Leakage-Resistant Authenticated Encryption with a Masked Tweakable Block Cipher* - IACR Transactions on Symmetric Cryptology [**?**].

International conference publications

1. Francesco Berti and François-Xavier Standaert *An Analysis of the Learning Parity with Noise Assumption Against Fault Attacks* - CARDIS 2016 [30].

2. Farzaneh Abed, Francesco Berti and Stefan Lucks *Is* RCB *a Leakage Resilient Authenticated Encryption Scheme?* - NordSec 2016 [4] (eprint version [3]).

3. Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters and François-Xavier Standaert ***Ciphertext Integrity with Misuse and Leakage: Definition and Efficient Constructions with Symmetric Primitives*** - AsiaCCS 2018 [24] (eprint version [23]).

4. Francesco Berti, Olivier Pereira and Thomas Peters *Reconsidering Generic Composition: The Tag-then-Encrypt Case* - INDOCRYPT 2018 [25] (eprint version [26]).

5. Francesco Berti, Olivier Pereira and François-Xavier Standaert ***Reducing the Cost of Authenticity with Leakages: a*** CIML2-***Secure*** AE ***Scheme with One Call to a Strongly Protected Tweakable Block Cipher*** - AFRICACRYPT 2019 [29] (eprint version [28]).

6. Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters and François-Xavier Standaert ***Strong Authenticity with Leakage under Weak and Falsifiable Physical Assumptions*** - Inscrypt 2019 (eprint version [22]).

Proposal to competitions

1. Davide Bellizia, Francesco Berti, Olivier Bronchain, Gaëtan Cassiers, Sébastien Duval, Chun Guo, Gregor Leander, Gaëtan Leurent, Itamar Levi, Charles Momin, Olivier Pereira, Thomas Peters François-Xavier Standaert, and Friedrich Wiemer *Spook: Sponge-Based Leakage-Resilient Authenticated Encryption with a Masked Tweakable Block Cipher* - Submission to NIST Lightweight Cryptography [18].

Chapter 1

Introduction

Contents

1.1 Scope and motivation

Messages, when sent, can be tampered by a malicious adversary.
Today, open source software are deployed. They often need updates which are usually downloaded from the Internet. Although the authors are not concerned about the confidentiality of these updates, since they are publicly available, the authors care that the actual update is downloaded and not a version modified by a malicious adversary.
The goal of authenticity is to prevent that anybody can send a message impersonating someone else.
The cryptographic primitive for this is a MAC (Message Authentication Code). It is a vastly deployed primitive [81].

Consider when the results of medical exams are sent by email to patients. Not only is it necessary that these results are correct, but also

that nobody sees them.
Authenticated Encryption (AE) schemes are the solution for this. For AE the authenticity property is usually called *integrity*.

As a first step to the deployment of these schemes, their security is assessed in the *blackbox scenario*: the adversaries have only access to their outputs and usually their inputs. The proofs of the security of these schemes are based on the security of underlying primitives, as, for example, block ciphers (BCs), hash functions [81].
However, these schemes must be implemented on real devices, and these implementations use time, power consumption. The observation of these physical quantities may give additional information [86, 87]. These additional sources of information are called *side-channels*. Attacks exploiting them are called *side-channel attacks*.

Authenticity may be affected by these attacks. For example, the adversary may be able to retrieve the scheme's key. Nevertheless, this is not the only threat side-channel attacks pose. There may be other intermediate values obtained during the cryptographic computation, which, if exposed, may affect authenticity.
The fact that side-channel attacks pose a threat to authenticity is a significant problem. We give two examples.
First, imagine a credit card reader that has been tampered by an adversary. This device is used to collect side-channel information about the secrets of client cards, for example, their PINs. We would like to prevent that via this "malicious reader" an adversary can create credit card payments which are considered valid.
Second, imagine a device, for example, an Internet-of-Things (IoT) sensor. This device receives updates. These updates are authenticated. Via side-channel attacks, it may be possible to create a malicious update that passes the authentication check. If this happens, an adversary can take control of that device. And, in IoT, it is often enough to be able to have control of one of the devices to be able to create significant damage to the whole net [120].
Thus, leakage is not only a theoretical threat but also poses real problems to authenticity.

To prevent this, first, it is necessary to have a theoretical comprehension of the situation. Thus, we introduce a theoretical scenario, stating the security definitions and introducing new leakage models.
These definitions give concrete security in real scenarios. Using the leakage models we present, the security proofs highlight the primitives which

must be protected. Our contribution is not only theoretical, but also practical since we provide many constructions achieving the definitions we have stated in our leakage models.

1.2 Blackbox authenticity and integrity

We assume that two parties share a key k to guarantee the authenticity of their communication. This is the *symmetric-key* (also called *private-key*) setting.
According to the Kerchoff principle, "*the cipher method must not be required to be secret, and it must be able to fall into the hands of the enemy without inconvenience*". That is, "*security rely solely on the secrecy of the key*" [81].
Message Authentication Code (MAC) is the common cryptographic primitive to provide authenticity. MAC computes a tag τ to authenticate the message m. To ensure that a message is genuine and not tampered, the tag τ attached to the message m is verified.
MAC security ensures that it is hard to produce a *forgery*, that is, a valid tag for a new message [81].
Although this solution is enough when the message contains no private information, in many situations, the message m should not be sent in clear, because it contains private information. AE schemes are the cryptographic primitive to provide confidentiality in addition to authenticity [16, 82, 88, 14].
To send a message m using an AE scheme, m is encrypted producing a ciphertext c. When this ciphertext c is received, it is decrypted in a message m, and the authenticity of c is verified.
Ciphertext-integrity security ensures that it is hard to produce a fresh ciphertext that is deemed authentic. Such a ciphertext fabricated by the adversary is called *forgery*.
To provide it, many AE schemes compute a tag [99, 107, 112, 19]. In some cases, AE schemes are built via the composition of existing MACs and encryption schemes [104].
We end noting that AE is widely used [55, 99, 107, 112, 19] (e.g., it is only the one accepted in TLS 1.3 [74]).

As a first step, the security for these schemes is proved in the *blackbox model*. In this model, adversaries have access only to the outputs and, usually, to the inputs.

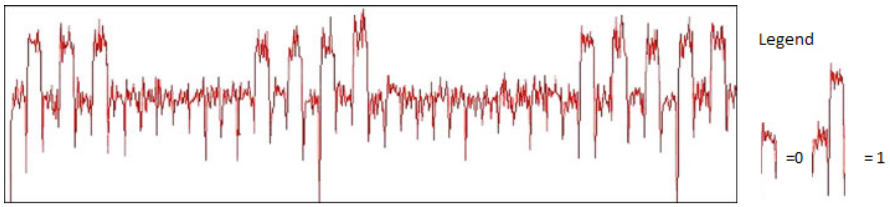


Figure 1.1: A leakage trace, from Oswald et Standaert [110]. Note that since bit keys are processed one by one (it is an exponentiation and the key is the exponent), with the legend, it is possible to recover the full key with a single leakage trace.

1.3 Leakage

Consider a malicious vendor, who owns a smart card reader, which is used to allow electronic payments. He can do much more than merely seeing the inputs and the outputs of his device. For example, he may observe the instantaneous power consumption of his device, and he may receive a trace similar to Fig. 1.1. During this computation, the key is processed bit-by-bit. Furthermore, the instantaneous power consumption is different if the key bit processed is 0 or 1, as shown by the legend of Fig. 1.1. Thus, the adversary can "easily" retrieve the key via side-channel.
Consequently, having a secure scheme in the blackbox scenario is not enough in the real world.

The previous attack is an example of *side-channel attacks*.
Side-channel attacks start from the fact that schemes must be implemented on real devices. So the adversary may obtain other information than the outputs (and the inputs). For example, to do the cryptographic operation needed by a MAC or an AE scheme, these devices consume electrical power, generate an electromagnetic field, use time, etc...
Moreover, all these physical quantities may be measured, and these observations give additional information about the inner computations performed by the device. In some cases, this is enough to recover the full key breaking the security of the scheme [6, 87, 96, 97, 93].

These attacks are particularly effective against the Internet-of-Things (IoT) nets. Sensors, being deployed in a not-secure environment, are an easy target for side-channel attacks. For example, it may be possible to mount an attack using side-channels, where an adversary can make the

sensor accept a malicious update.
Moreover, since sensors should be cheap and countermeasures against side-channel are expensive (see Sec. 1.3.1), they are often less protected.

The physic behind these attacks is very complicated. In particular, it is tough to compute a leakage trace even knowing all the values and the physical quantities involved in that cryptographic computation [124]. The only known way to obtain them is to sample by taking actual measurements from the target circuits on given inputs [124].
Thus, to theoretically model an adversary who can do side-channel attacks, he is supposed to have access to a *leakage function*, which gives him the information he may obtain via leakage. As the physic behind side-channel is hard, we observe that it is tough to model the leakage function correctly [124, 78]. There are many different models (for example, memory leakage, leakage from computation, for a complete survey, see the survey of Kalai and Reyzin [78]) for the leakage function.
Usually, the adversary has some constraints for his leakage function; for example, bounding the output size [56]. A critical drawback of this abstraction is that quantifying an "overall amount of leakage" is hard for hardware engineers and to assure in practice [124].

1.3.1 Countermeasures

We start observing that, in general, it is more difficult to mount an attack against a scheme when the adversary has access to the leakage of a single execution than with the leakage of more executions.

The countermeasures against side-channel attacks which have been introduced, work at different levels.
At the hardware level countermeasures target the reduction of the information leaked, for example adding noise [94].
At the implementation level, the previous countermeasures can be amplified reducing the side-channel leakage [116, 95, 127]. The most studied implementation level countermeasure is *masking* whose main idea is to split a sensitive data into many random shares. Masking has a strong theoretical background [76, 53, 54]. Its security depends on the number of shares, the higher, the better. On the other hand, it has considerable overheads, which are roughly quadratic in the number of shares [62, 77]. Thus, a device with a strong masking protection can be a thousand times slower.
At the mode level, *leakage-resiliency* aims to design schemes that are inherently more secure against side-channel attacks [58], for example,

updating the key [111].

To reduce the cost of countermeasures Pereira et al. [111] have proposed the concept of *leveled implementation*: that is, combining the minimal use of a primitive which is strongly protected against side-channel attacks and thus may be very slow, with a mode of operations where the bulk of the computation is carried by weakly protected (or not protected at all) primitives which are much more efficient.

Leak-free. It remains the problem of how good the strongly protected implementations should be. Pereira et al. [111] have modeled them as *leak-free*: that is, they do not leak anything. Although this model seems too demanding, it has many advantages. It reasonably models strongly protected implementations (as for example, a TBC with an high-order masking scheme). It is very straightforward to use. Additionally, this model has the advantage of showing where to put the countermeasure. In practice, leak-free implementations do not entirely exist. We can see actual strongly-protected implementations as an imperfect realization of a leak-free implementation [111].

1.4 Our contributions

We have followed two lines for our contributions. From a theoretical point of view, we try to capture the security that is needed. Moreover, we try to model leakage in a simple way, which encompasses many attacks. More precisely:

- Since we want to capture different scenarios, we provide different security definitions.
  These definitions are different because they allows to consider different scenarios with different trade offs between the security needed and efficiency.

- We propose a leakage model, easy to use: leak-free for strongly protected implementations and unbounded leakage (that is, we assume that these implementations have no security at all with leakage) for everything else.

- We create a model (strong unpredictability) to relax the leak-free assumption for strongly protected implementations.

From a more practical point of view, we build and analyze several constructions, both MACs, and AE schemes, in our leakage models.
First, the security is assessed supposing that the strongly protected primitives are leak-free, then, that they are only strongly unpredictable.
To be more efficient, all our constructions uses components whose implementation is either strongly protected or unprotected at all. This leads us to assume that the unprotected components leak everything, thus we have the *unbounded leakage*.

1.4.1 Theoretical framework

Definitions We start from the authenticity definition for MACs of Pereira et al. [111]: *strongly existentially unforgeable against chosen message and verification attacks with leakage in the tag-generation* (suf-L).
In this definition, the adversary tries to create a forgery having only access to the leakage of the tag-generation algorithm.
In particular, the adversary may not only target the key, but he can target some intermediate values which he may find useful for forgeries.
First, we extend this definition to the AE-case, supposing that only the encryption algorithm leaks (CIML).
Then, we extend these two definitions to the case when also the verification (for MACs), with suf-L2, or the decryption (for AE schemes), with CIML2, algorithm leaks.
Note that in all these definitions, we suppose that when the key is generated, the adversary has not access to the leakage since this may done during the fabrication of the device. On the other hand, we allow the adversary to choose the random coins used by the algorithms when he is interacting with them. That is, we also consider that the random coins (or the nonce) are not used correctly in tag-generation/encryption queries.
A significant result is that for both MAC and AE, there can be schemes whose security depends if the adversary has access or not to the leakage of the verification or the decryption oracle. This result may be useful for devices which are used only to send (or authenticate) messages, not to receive (or verify). In this case, the adversary may have the leakages only of the encryption (or the tag-generation).

As a side-note, we observe that for AE, integrity with leakage may also improve confidentiality with leakage. In fact, an AE scheme which provides integrity may prevent the leakage of any valuable information from the decryption of invalid ciphertexts. Such decryptions may be source of interesting information [7, 11, 33].

Leakage models We introduce a pragmatic leakage model: the *unbounded leakage model*. In this model, the adversary receives every input, output, and key used by any primitive underlying the scheme except for the key used by the strongly protected implementations of primitives (which we model as leak-free). This model, although it keeps the problem of the leak-free model for strongly protected implementations, has many advantages:
(*i*) it is easy for practitioners to identify where the efforts against side-channel attacks should be put.
(*ii*) The leakage function is straightforward to describe. In fact, it usually consists of one or two values, from which the adversary is usually able to recompute every other secret value used during the computation (except for the key of the leak-free primitive).
(*iii*) We observe that the adversary cannot obtain any other information from the weak-protected primitives.
Thus, a proof in our unbounded leakage model means that to break the authenticity, the adversary must attack the strongly protected component to win or the underlying blackbox assumption. As a side note, we observe that it is very hard for an adversary to be able to perfectly recover every secret value (apart from the key of the leak-free primitives). Consequently, when we do a proof in our leakage model, we can reduce the authenticity/integrity of the whole scheme to some black-box properties of hash functions, (tweakable) block ciphers ((T)BC) and the goodness of the strongly protected implementation even in the face of the most powerful possible side-channel adversaries.

Strong unpredictability We introduce a notion relaxing the leak-free hypothesis for (T)BC: *Strong Unpredictability in the presence of leakage* (sUL) for a (tweakable) block cipher, which is an extension of the unpredictability notion introduced by Dodis and Steinberger [51, 52]. Roughly speaking, sUL states that it is hard to find for a fresh input the output of (T)BC, even if the adversary has oracle access to it, its inverse; moreover, he receives the leakage of its direct and inverse calls.
This notion has the advantage that (*i*) it is game-based and it is not an idealized physical assumption, (*ii*) it may be verified (or falsified) by laboratories, (*iii*), it gracefully degrades when the physical assumptions are not completely respected (*iv*) it is more adapted to actual implementations which protects the key heavily, but do not assume anything about the outputs and their randomness.

1.4.2 Constructions

Our second line of contribution is providing many constructions achieving our security definitions in the unbounded leakage model.
First, we start introducing a MAC, HBC (Hash-then-BC), which is secure when only the tag-generation leaks.
Second, we introduce a new construction for AE, which we denote as the DTE scheme (for Digest, Tag, and Encrypt), which is secure when only the encryption leaks. DTE uses HBC for the Digest and Tag part. Since DTE is a probabilistic scheme, we have considered the case that the random coins it uses are not good (*misuse*). We can prove its authenticity with encryption leakage and random coins chosen by the adversary. DTE uses two calls to the strongly protected BC.
Third, we tackle the problem of verification/decryption leakage.
We introduce HBC2 and HTBC to solve the problem of verification leakage for MACs. HBC2 is a modified version of HBC, where we have modified only the verification algorithm. In HBC2 we are able to verify a tag τ without having to compute the correct tag $\tilde{\tau}$ and comparing τ with $\tilde{\tau}$. HTBC is an improved version of HBC2 using a tweakable BC instead of a BC and having whose probability of forgeries is even much smaller, since it is beyond birthday secure.
Regarding the problem of decryption leakages, we introduce DTE2, which is substantially DTE, where we have replaced HBC with HBC2. Fourth, we start from the observation that in decryption, DTE2, first decrypts then verifies if the ciphertext is valid. Although the fact that a message is retrieved in decryption, before having checked its authenticity does not affect integrity (even with leakage), it may not be the case for other security property, as confidentiality with leakage. To prevent that any message is retrieved if the ciphertext is invalid, EDT is introduced, changing the paradigm of DTE. That is, in EDT, it is the ciphertext that is digested and tagged (and not the message[1] as for DTE). EDT is CIML2-secure, uses two calls per execution to the strongly protected BC. On the other hand, it is no more misuse-resistant like DTE2.
Finally, to reduce the overall cost, we propose a construction, CONCRETE, which uses only one call to the strongly protected implementation. To obtain this, we rearrange the structure of EDT (and of its variant) entirely.

Note that in all the constructions where leakage in verification/ decryption is considered, the use of a strong (tweakable) BC (with a strongly protected implementation) is crucial, because the inverse of the (T)BC

[1] And the randomness used.

is used in verification/decryption.

In all our AE constructions, to improve efficiency, we use a *leveled implementations*, using as little as possible calls to the strongly protected implementation of the (T)BC. In particular, during the encryption part, we use take advantage of an existing leakage-resilient encryption scheme which is based on rekeying, that is, changing the key during the execution.

We end noting that we have designed both EDT and CONCRETE in a way that makes the decryption of invalid ciphertexts give no useful information to an adversary, one, EDT, blocking the decryption before it gives anything useful, the other, CONCRETE, leaking random values if the ciphertext is not authentic.

Proofs based on strong unpredictability. We can prove the suf-L2-security of both HBC2 and HTBC assuming that the (T)BC is sUL-secure (and not leak-free), but on the expense of relying on a random oracle (RO). However, the RO is only considered in the black box model and we do not need any security assumptions regarding its leakage. In fact, in the proofs, every time the RO is used, we suppose that the adversary knows its inputs and its output.
These results may be extended easily to DTE2 and EDT.

1.5 Related works

Leakage-resilient constructions. The first leakage-resilient construction was the stream-cipher proposed in 2006 [58]. There is a flourishing literature on it; see, for example, the survey of Kalai and Reyzin [78]. An attractive solution, which we always use in our AE schemes, is *rekeying*. That is, the key is changed after a few executions. Abdalla et al. [1] have proved that using a leakage-resilient scheme on top of an existing secure encryption scheme in the standard model, leads to a leakage resilient encryption scheme.
We also take advantage of the literature on leakage-resilient stream ciphers in order to reduce the use of strongly protected implementations [58, 113, 132, 59, 131, 124].
To our knowledge, the first leakage-resilient AE scheme is RCB [5], which is not secure, as proved by Abed et al. [4].

The CCS 2015 paper. [111]. Our work starts from the CCS 2015 paper by Pereira et al. [111]. In particular, we have used their definition of authenticity with leakage in tag-generation for MAC. Moreover, they have introduced a leakage-resilient encryption scheme, PSV, which is based on rekeying. We have used PSV in all our AE schemes. In fact, we use it for the Encryption part of DTE, DTE2, EDT and CONCRETE.
Differently from us, Pereira et al. [111] establish the authenticity of the MAC, assuming a weak hypothesis (simulatability) on the not strongly-protected implementation of the BC.

Works based on this thesis. Our works are the base of many others. At LatinCrypt 2019, Guo et al. [65] give a complete survey of the security definitions with leakage for AE (thus, also considering confidentiality) also establishing the relations among them. Their authenticity definitions are ours.
Moreover, they propose a variant of EDT, AEDT to treat associated data, data which need to be authenticated but not encrypted.
Finally, they introduce FEMALE, which is an improved version of EDT with the encryption part modified in order to be misuse-resistant.

At CHES 2020 [21] a scheme based on EDT has been proposed: TEDT (Tweakable EDT). This scheme, which uses only TBC, is beyond birthday secure. Moreover, a one-pass version of it, called TET1, is proposed. TET1 is more efficient, but relinquishes some security properties.

Finally, TET1Sponge, an instantiation of TET1 with a sponge for the encryption part is the NIST submission Spook [18] to the competition for Lightweight Cryptography.

Note that in all the constructions achieving integrity leakage in both encryption and decryption, in decryption, the inverse of the (T)BC is used.

Other theoretical works. Guo et al. [65] and Barwell et al. [10] have proposed two theoretical framework for AE in the presence of leakage. Regarding integrity, Guo et al. [65] incorporate our definitions in their framework. Instead, the work of Barwell et al. [10] is more focused on composition results, while we pay particular attention to efficient instances of MACs and AE schemes. As a result of this choice, a second difference is that their instantiations require all the building blocks to be well protected against side-channel analysis, while we aim to minimize the use of strongly protected implementations. In particular, in

our construction, the encryption part does not need to be as protected as theirs. Third, and more technically, we discuss what can be achieved by symmetric cryptographic building blocks, while Barwell et al. [10] use elliptic curves operations. Note the leakage-resilient-pairing-based-PRF proposed in the latter work could be an option to instantiate our leak-free component, instead of a TBC with an high-order masking scheme.

In a distinct line of work, the problem of leakages resulting from decryption failures has been investigated [33, 7, 72]. The motivation of these works is that, when decryption fails (as a result of an incorrect ciphertext), the decryption software typically reveals more information than just this failure: for example, it often happens that different error codes are sent depending on the step at which decryption fails. In this setting, leakage naturally happens when decryption fails only; that is, correct decryption operations do not leak anything since there is just no error message. However, in the context of side-channel attacks, this restriction becomes meaningless: decryption takes time, consumes power, and produces electromagnetic radiations, whether if it succeeds or fails. (And we may even expect that implementations will leak more in case of successful decryption since this is likely to be the case during which the most substantial amount of computation takes place.)
We observe that CIML2 (and suf-L2) security in the unbounded leakage model provides stronger security guarantees. In fact, knowing all the internal values computed (apart from the key of the primitives used with a the strongly-protected implementation), the adversary also has access to the value that is compared in the check that generates the invalid message.

The distinction between leakage of valid and invalid ciphertexts also puts the security notions that we propose in this work out of the unifying definition framework of Barwell et al. [11], called Subtle Authenticated Encryption (SAE). In [27], the advantages of CIML2 in the unbounded model with respect to SAE are discussed.

Note that although the unbounded leakage model does not cover the release of unverified plaintext (see [7]) in general, our schemes are always CIML2 even if unverified plaintexts are released. In particular, CONCRETE is secure even if unverified plaintexts are released (it provides both authenticity and privacy, while DTE and EDT provide only integrity).

Other leakage resilient constructions. Dobraunig et al. [46] combine a concrete instance of fresh re-keying (borrowed from [100, 48]),

with a sponge-based construction [31] giving birth to ISAP (which is also a NIST candidate). This line has also been followed by ASCON by Dobraunig et al. [47] and Xoodyak by Daemen et al. [41]. All these schemes are also based on sponges (they use the seminal work on sponges by Bertoni et al. [32]).
Their security has been studied in detail [66, 49, 42, 50]. To obtain integrity in the unbounded leakage model, they have to protect the comparison between the transmitted tag and the correct tag [50].

1.6 Structure of the thesis

In the first two chapters, we introduce the background. Chap. 2 is devoted to the blackbox case while Chap. 3 to the leakage case. We put some additional blackbox definitions in App. A. These definitions are not necessary to understand the integrity problem with leakage, but are useful to understand other properties of our schemes.
Chap. 4 is devoted to the theoretical framework, introducing the leakage model, the model for strongly protected implementations, while Chap. 5 to the constructions. We end Chap. 4 comparing our definitions with those of Barwell et al. [10], while Chap. 5 we compare our constructions with ISAP [46], ASCON [47], Xoodyak [41] and the one by Barwell et al. [10].
In Chap. 5 we do all the proofs assuming strongly protected implementation as leak-free. In the last chapter, Chap. 6, we prove again the security of some of the previous constructions assuming strongly protected implementation as strongly unpredictable.
In the thesis, we only put a sketch of our proofs, leaving the full proofs to App. B.

Chapter 2

Background

Contents

In this chapter, we recall some fundamental notions of symmetric-key cryptography. We present the blackbox definitions, and we do not consider leakage.

2.1 Notations

We usually use the callygraphic notation ($\mathcal{A}$) for sets.
Given a set $\mathcal{X}$, we denote with $|\mathcal{X}|$ its cardinality.

Given two sets $\mathcal{X}$ and $\mathcal{Y}$, the set of all functions $\mathsf{f} : \mathcal{X} \to \mathcal{Y}$ is denoted by $\mathcal{FUNC}(\mathcal{X}, \mathcal{Y})$, while the set of all permutations $\mathsf{p} : \mathcal{X} \to \mathcal{X}$ is denoted by $\mathcal{PERM}(\mathcal{X})$. Note that if $\mathcal{X}$ and $\mathcal{Y}$ are both finite, then, $|\mathcal{FUNC}(\mathcal{X}, \mathcal{Y})| = |\mathcal{Y}|^{|\mathcal{X}|}$, while $|\mathcal{PERM}(\mathcal{X})| = |\mathcal{X}|!$.
Given a non-empty finite set $\mathcal{X}$, let $x \xleftarrow{\$} \mathcal{X}$ denote the draw of an element x from $\mathcal{X}$ uniformly at random.
A *random function* $\mathsf{f} : \mathcal{X} \to \mathcal{Y}$ is a function picked uniformly at random from the set $\mathcal{FUNC}(\mathcal{X}, \mathcal{Y})$, that is, $\mathsf{f} \xleftarrow{\$} \mathcal{FUNC}(\mathcal{X}, \mathcal{Y})$. Thus, when we have to use a random function, $\forall x \in \mathcal{X}$, we may simply pick a y uniformly at random in $\mathcal{Y}$ and define $\mathsf{f}(x) := y$.
A *random permutation* $\mathsf{p} : \mathcal{X} \to \mathcal{X}$ is a permutation picked uniformly at random from the set $\mathcal{PERM}(\mathcal{X})$, that is, $\mathsf{p} \xleftarrow{\$} \mathcal{PERM}(\mathcal{X})$. Thus, when we have to use a random permutation, $\forall x \in \mathcal{X}$, we may simply pick a y uniformly at random in $\mathcal{X} \setminus \mathcal{L}$, where $\mathcal{L}$ is the list of the $y \in \mathcal{X}$ already picked as image of previous $x' \in \mathcal{X}$, and define $\mathsf{p}(x) := y$.
Two functions $f : \mathcal{X} \to \mathcal{Y}$ and $f' : \mathcal{X}' \to \mathcal{Y}'$ have the *same signature* if their domain and codomain are the same, that is, $\mathcal{X} = \mathcal{X}'$ and $\mathcal{Y} = \mathcal{Y}'$.

Definition 1. *A function $f : \mathbb{N} \to \mathbb{N}$ is* negligible *if for every polynomial $p(.)$, $\exists N \in \mathbb{N}$ s.t. $\forall n \in \mathbb{N}, n \geq N$*

$$f(n) \leq \frac{1}{|p(n)|}.$$

That is, a negligible function decreases faster than any polynomial function.

Given $a, b \in \mathbb{Z}$, we denote with $\{a, ..., b\}$ the set of all integer numbers between a and b, that is, $\{a, .. :, b\} = [a, b] \cap \mathbb{Z}$, where $[a, b]$ is the closed interval in $\mathbb{R}$.
Given $x, y \in \mathbb{N}$ with $x \leq 2^y - 1$, we denote with $[x]_y$ the binary writing of x with y digits (thus, pre-appending as many 0s as necessary).
We write $\Pr[B; A_1, A_2, ...]$ for the probability that event B happens given that the events $A_1, A_2, ...$ have happened.

Strings

We use finite binary strings. The string of x 0s is denoted by 0^x. The length of the string x is denoted by $|x|$.
The set of all finite strings is denoted by $\{0,1\}^*$, the set of all n-bit strings by $\{0,1\}^n$, the set of strings of at most n bits by $\{0,1\}^{\leq n}$, the set of strings of infinite length by $\{0,1\}^\infty$.

Given a string $x = (x_1 x_2 ... x_l)$ of l bits, we denote with $\pi_t(x)$ the string $(x_1 ... x_T)$ where $T = min(|x|, t)$, that is, the string obtained by the first t bits of the string x. Thus, we can define the function $\pi_t : \{0,1\}^n \to \{0,1\}^{\leq t}$ as the projection on the first t bits.
Given two strings, x and y, let $x \| y$ denote the string obtained by concatenating x and y, while $x \oplus y$ denotes the string obtained XORing bitwise x and y, provided that $|x| = |y|$.

Parsing. Often, we have to parse a message m in n-bit blocks. This means that the message m is divided in $(m_1, ..., m_{l-1}, m_l)$, with $|m_1| = ... = |m_{l-1}| = n$, $|m_l| \leq n$ and $m = m_1 \| ... \| m_{l-1} \| m_l$.

2.1.1 Time notation

In this section, we collect all the time notations we use in the proofs:
$t_{\mathsf{ch}(x,\mathcal{B})}$: is the time needed to pick uniformly at random x values in $\mathcal{B}$
$t_{\mathsf{chn}(x,\mathcal{B})}$: is the time needed to pick uniformly at random with no repetitions x values in $\mathcal{B}$.
t_{Alg}: the time needed to execute once the algorithm Alg
$t_{\$}$: the time needed to randomly sample a value
$t_{\mathsf{f}(y)}$: the time needed to lazy sample the random permutation (or function) f y times with $y \in \mathbb{N}$.

2.2 Adversaries and proofs

Since this thesis aims to do a further step in providing authenticity in cryptography, we want that all our schemes are provably secure. That is, given a scheme Π, we want to be able to give an upper bound ϵ to the probability that an adversary with certain resources (for us, usually, time and number of queries to their oracles) can break the property X of the scheme Π. In this case, we say that a scheme is ϵ-*secure* with respect to this property.
These results must and may only be obtained through "rigorous proofs of security" [81].

Quantitative proofs. Our proofs are *quantitative*, that is, we prove that a scheme is secure when adversaries have access to a certain amount of resources (for example, time).
Since we want to prove the security of actual schemes, we prefer this approach than the *asymptotic* ones. In the asymptotic proofs, the probability that an adversary breaks the security of a scheme is bounded by a negligible function, $\epsilon(n)$, of a security parameter, n, (for example, the

size of the key). Asymptotic proofs, although very important, do not provide security for actual schemes, since cryptographic schemes, once deployed, cannot usually (or can only very difficultly) change this security parameter.
Note that in our quantitative proofs we do not consider the time needed for communications.

To reach our goal of security, first, we need to define what adversaries are, then, what is the security we aim for, after that, what security assumptions we make and finally how we combine all this.

Adversaries. First, we define what are adversaries.

Definition 2. *A* $(q_1, ..., q_d, t)$-adversary A *against* Π *is a probabilistic algorithm* A *having oracle access to* $\mathcal{O}_1, ..., \mathcal{O}_d$*, making at most* q_i *queries to oracle* $\mathcal{O}_i$*, running in time bounded by* t*, and outputting a finite string of bits.*

In some cases, the vector $q_1, ..., q_d$ is denoted with $\underline{q}$.
We do not need to consider bounded memory since A is $(q_1, ..., q_d, t)$-bounded, thus, the maximal size that the adversary uses is necessarily bounded.
Observe that our adversaries have access to bounded resources. Otherwise, schemes which are secure against unbounded adversaries, have strong limitations, for example, having the key as long as the message and never reusing this key [81]. Thus, secure schemes against unbounded adversaries are often unusable in most setting[1]
Our adversaries are probabilistic, because the ability to toss coins may provide additional power [81] and we need that the class of adversaries to be as large as possible.[2]

Security definitions The security we aim is provided by *security definitions*. In particular, one of the aim of this thesis is to justify four (three of which are new) security definitions.
Security definitions model the security we want (which must be well

[1] For example, a scheme secure against unbounded adversaries, one-time-pad, is "rumored" to be used for the "red phone", which was the system allowing direct communication between the White House and the Kremlin during the Cold War [81].

[2] We do not consider in this thesis, *quantum* adversaries (for example see [34]), that is, adversaries who can evaluate an oracle "in superposition" (on quantum states). In fact, every adversary, in this thesis, queries his oracles on a (classical) input x, not on a superposition of inputs $\sum_x \alpha_x$.

understood) and explain when a scheme can be deployed safely. In addition, they can be compared. Moreover, precise definitions allow rigorous proofs [81].

Security assumptions To prove the security of a construction, we need to base the security on *security assumptions*. Security assumptions, which cannot be avoided, should be minimal, clear and, if possible, broadly accepted [81].

Security proofs Finally, to prove the security of schemes, we use the *reductionist approach*. We usually have theorems of the form: "Given that Assumption X is true, Construction Y is secure according to Definition C".
Our proofs typically show how an adversary breaking Construction Y can be used as a sub-routine to break Assumption X [81].

Game-playing proofs

For most of the proofs, we use a game-based approach [123, 17].
Suppose that we have to prove that a scheme Π is secure. To do this, we want to upper bound the advantage of a bounded adversary A in attacking some cryptographic constructions. The advantage is the difference between the probability that A outputs 1 in two different "worlds". World 0 is when he interacts with the scheme Π. Thus, we write a code initializing variables, showing how A can interact with Π. This is *Game* 0 and can be seen as a piece of code. Then, we write another piece of code that we call Game 1. We build Game 1 such that it is "substantially as hard as" Game 0 for the adversary A. That is, we show that the advantage that A has playing Game 0 with respect to Game 1 is little, either using a security assumption or showing that these two games are syntactically identical if a given event, whose probability we bound, happens. Next, we create Game 2 and show that it is "substantially as hard as" Game 1 for the adversary A. Iterating, we produce a chain of games ending with a *terminal game* in which it is possible to compute the probability that adversary A wins with conventional arguments (that is, either via a computational assumption or an information-theoretical argument) [17].

Game A *game* G is a program, that is, a collection of procedures, run by an adversary A, which ends in a finite (polynomial) number of steps. At the end of the game G, the adversary A outputs a value x, which may be probabilistic (and, in this case, its probability is denoted with $\Pr[\mathsf{A}^{\mathsf{G}} \Rightarrow x]$). The game processes this output to produce the outcome

y. Usually, we are interested in bounding or computing $\Pr[\mathsf{G}_{\mathsf{A}} \Rightarrow y]$, the probability that the outcome y happens. Since adversaries are probabilistic and, thus, their outputs, too, then, the outcome y of a game is probabilistic.

Usually, during the proofs there will be a sequence of games $\mathsf{G}^0, ..., \mathsf{G}^I$. In our proofs, we usually prove that every couple $(i, i+1)$ of games G^i and G^{i+1} is (q, t, ϵ)-*indistinguishable*, that is, for any (q, t)-adversary A, for any possible outcome y of these games, we bound the difference $|\Pr[\mathsf{G}^i_{\mathsf{A}} \Rightarrow y] - \Pr[\mathsf{G}^{i+1}_{\mathsf{A}} \Rightarrow y]| < \epsilon$, where ϵ is little.

The oracle $\$(\cdot)$ and $\perp(\cdot)$. In many security proofs and definitions, we have to prove or to assume that an adversary is not able to distinguish a set of oracles implemented with the real algorithms from ideal oracles. Two ideal oracles are used: $\$(\cdot)$ or $\perp(\cdot)$.
The oracle $\$(\cdot)$ outputs a random string, whose length is specified by the definition or the proof. For example, if A has to distinguish between the oracles $\mathcal{O}_A(\cdot)$ and $\$(\cdot)$, the oracle $\$(\cdot)$ on input x returns a random string of length $|\mathcal{O}_A(x)|$.
The oracle $\perp(\cdot)$ always returns $\perp$.

2.3 Hash functions

First, we introduce *hash functions*.
Hash functions are generally used to compress strings.

Definition 3. *A hash function is a function* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \mathcal{T}$*, indexed by a key selected from the key set* $\mathcal{KH}$*.*

We need that hash functions are *keyed* to be able to give security definitions for them, even if, in practice, most of them are *unkeyed*. On the other hand, the key of the hash function is supposed to be public. Thus, we will often omit to explicit it.

Hash-functions should be *collision resistant*; that is, given the key s, it should be hard to find a *collision*. A collision is a couple of different inputs x, x' s.t. $\mathsf{H}_s(x) = \mathsf{H}_s(x')$. Formally:

Definition 4 (CR)**.** *A* (t, ϵ)-collision resistant hash function $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \mathcal{T}$ *is a function such that, for every* t*-bounded adversary* A*, the probability that* $\mathsf{A}(s)$ *outputs a pair of distinct inputs*

$(m^0, m^1) \in (\mathcal{HM})^2$, such that $\mathsf{H}_s(m^0) = \mathsf{H}_s(m^1)$ and $m^0 \neq m^1$, is bounded by ϵ, where $s \xleftarrow{\$} \mathcal{KH}$ is picked uniformly at random, that is:

$$\Pr[\mathsf{A}(s) \Rightarrow (m^0, m^1) \in (\mathcal{HM})^2 \text{ s.t. } m^0 \neq m^1, \mathsf{H}_s(m^0) = \mathsf{H}_s(m^1) | s \xleftarrow{\$} \mathcal{KH}] \leq \epsilon$$

2.3.1 The birthday bound

In practice, the hash functions we use have $\mathcal{HM} = \{0,1\}^*$ and $\mathcal{T} = \{0,1\}^n$, for a certain fixed n. Thus, due to the *pigeon-hole principle* (called also *Dirichlet's box principle*), there are always many collisions. For a collision-resistant hash function, it should be hard to find one of them.
Since an adversary may go on trying different inputs until he finds a collision, we recall what is the best possible collision resistant hash function.[3] The following theorem answers this question.

Theorem 1. *Let $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \mathcal{T}$ be a random function. Then, H is (q, ϵ)-collision resistant, with*

$$\epsilon \leq \frac{q(q-1)}{2|\mathcal{T}|}.$$

Note that we do not bound the time the adversary has access to since we use a combinatorial argument and not a computational one. Thus, we only consider the number of queries he can ask.

Proof. See App.A.4 [81]. □

This bound is tight, that is, there are matching attacks [81].

A tight bound means that if $q = \sqrt{|\mathcal{T}|}$ roughly half of the times, we find a collision. In the literature, this problem is called the *Birthday problem*[4] and the bound is called *Birthday Bound* (BB). As we have proved, this bound is unavoidable for hash functions.

[3] We do not consider the case $|\mathcal{HM}| \leq |\mathcal{T}|$ because it is irrelevant in practice; moreover, in our works $|\mathcal{HM}| \gg |\mathcal{T}|$.

[4] In fact, the usual statement of the problem is: How many people (supposing that there are no twins) need to be in a room to have a 50% chance to have two people who share the same birthday? The answer, to the surprise of the most, turns out to be 23.

2.3.2 Multi-Collisions

We are also interested to bound the probability that a *multiple collision* happens, that is, asking q different evaluation of a random function f, what is the probability that we have asked s different inputs $x_1, ..., x_s$ among our q queries, such that $\mathsf{f}(x_1) = ... = \mathsf{f}(x_s)$. Suzuki et al. [126] have studied this problem.

Let $1 \leq s \leq q \leq n$. We consider the experiment where we uniformly throw q balls at random into n bins. $\mathsf{MultiColl}(n, q) \geq s$ denotes the event that at least one bin contains at least s balls. We recall a useful upper-bound on the probability of multi-collisions.

Theorem 2. *[126] Let $1 \leq s \leq q \leq n$, with $s, q, n \in \mathbb{N}$.*

$$\textit{Then, } \Pr[\mathsf{MultiColl}(n, q) \geq s] \leq \frac{1}{n^{s-1}} \binom{q}{s}.$$

2.3.3 Pre-image resistance

Informally, a hash function is pre-image resistant if it is hard to "invert", that is, given $y \in \mathcal{T}$ it is hard to find a pre-image for y, that is an element of $\mathsf{H}_s^{-1}(y)$. Although the idea is straightforward, it is difficult to formally capture pre-image resistance with a "good definition" [118, 8].
For simplicity we only mention here the definition we use.[5].

Definition 5 (roPR)**.** *A (t, ϵ)-range-oriented pre-image resistant hash function $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \mathcal{T}$ is a function such that, for every t-bounded adversary A, the probability that $\mathsf{A}(s, y)$ outputs a string m s.t $\mathsf{H}_s(m) = y$, is bounded by ϵ, where $s \xleftarrow{\$} \mathcal{KH}$, $y \xleftarrow{\$} \mathcal{T}$ are picked uniformly at random, that is:*

$$\Pr\left[\mathsf{A}(s, y) \Rightarrow m \ s.t. \ \mathsf{H}_s(m) = y \mid s \xleftarrow{\$} \mathcal{KH}, \ y \xleftarrow{\$} \mathcal{T}\right] \leq \epsilon \tag{2.1}$$

As an example of the problem of formalizing pre-image resistance, we observe that the hash function $\mathsf{H}_s(x) := s, \forall x \in \mathcal{HM}$ [6], is range-oriented pre-image resistant, although it is not a good hash function.

[5] The interested reader may find all the other definitions and all the details in the aforementioned papers [118, 8]

[6] It is necessary to assume that $\mathcal{KH} = \mathcal{T}$.

2.4 Pseudorandomness

Randomness is at the core of many cryptographic schemes. Since it is difficult and expensive to provide true randomness [81], we need to rely on an alternative source which provides something which "simulates" randomness "good enough".
In cryptography, a *pseudorandom function* is a primitive allowing randomness simulation.
Roughly speaking, a pseudo-random function is a function whose outputs are indistinguishable from random ones. Formally:

Definition 6 (PRF)**.** *A function* $\mathsf{F} : \mathcal{K} \times \mathcal{M} \to \mathcal{T}$ *is a* (q, t, ϵ)-pseudorandom function (PRF) *if for every* (q, t)*-adversary* A*, the advantage:*

$$\mathsf{Adv}^{\mathsf{PRF}}_{\mathsf{F}}(\mathsf{A}) := \left| \Pr\left[\mathsf{A}^{\mathsf{F}_k(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\mathsf{f}(\cdot)} \Rightarrow 1\right] \right|$$

is upper bounded by ϵ*, where* k *and* f *are chosen uniformly at random from their domains, namely* $\mathcal{K}$ *and* $\mathcal{FUNC}(\mathcal{M}, \mathcal{T})$*.*

Note that we can see a PRF as a family of functions which is indexed by the key.

Maximal security for PRFs. It is always possible to break a PRF trying all keys. Such an attack is called *exhaustive key search*. Thus, the keyspace should be big enough[7].

Sometimes, it is interesting to have an invertible PRF. It is possible to adapt the PRF definition to this situation:

Definition 7 (PRP)**.** *A function* $\mathsf{F} : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ *is a* (q, t, ϵ)- pseudorandom permutation (PRP) *if if for every* $k \in \mathcal{K}$*,* $\mathsf{F}_k : \mathcal{M} \to \mathcal{M}$ *is a permutation and for every* (q, t)*-adversary* A*, the advantage :*

$$\mathsf{Adv}^{\mathsf{PRP}}_{\mathsf{F}}(\mathsf{A}) := |\Pr[\,\mathsf{A}^{\mathsf{F}_k(\cdot)} \Rightarrow 1\,] - \Pr[\,\mathsf{A}^{\mathsf{f}(\cdot)} \Rightarrow 1\,]|$$

is upper bounded by ϵ*, where* k *and* f *are chosen uniformly at random from their domains, namely* $\mathcal{K}$ *and* $\mathcal{PERM}(\mathcal{M})$*.*

Usually, pseudorandom permutations are called *blockciphers* (BCs) and their input is called *block*.

When designing PRPs, we usually ask that the most efficient attack against it is the exhaustive key search.

[7] As an example, in the recent call for lightweight cryptography, the NIST imposes that the key size is at least 128 bits [12].

PRF vs PRP-security Since a random permutation is picked from the set $\mathcal{PERM}(\mathcal{M})$ which is much smaller than $\mathcal{FUNC}(\mathcal{M},\mathcal{M})$[8] it is natural to wonder if we may see a secure PRP as a secure PRF. This is is proved by the following proposition:

Proposition 1. *Let* $\mathsf{F}:\mathcal{K}\times\mathcal{M}\to\mathcal{M}$ *be a* $(q,t,\epsilon_{\mathsf{PRP}})$-PRP*, then,* F *is a* $(q,t,\epsilon_{\mathsf{PRF}})$-PRF*, with:*

$$\epsilon_{\mathsf{PRF}} = \epsilon_{\mathsf{PRP}} + \frac{q(q+1)}{2|\mathcal{M}|}.$$

Proof. A standard proof can be found in [81, 17].

The main idea of the proof is, first to replace the pseudo-random permutation with a random permutation, then, to compute the probability that the event *Output collision* (OC), that is, there is at least a collision when we evaluate a random function on q different inputs. Note that we have already studied the probability of event OC in Thm. 1 and it is precisely the birthday bound.
We end noting that a random function is indistinguishable from a random permutation if event OC does not happen. □

The previous proposition gives also the bound when we have to distinguish a random permutation from a random function:

Corollary 1. *Let* $\mathsf{p}:\mathcal{M}\to\mathcal{M}$ *be a random permutation. Then, for any q-adversary, the distinguishing advantage between* p *and the random function* f *(with the same signature as* p*) is bounded by:*

$$|\Pr[\,\mathsf{A}^{\mathsf{p}(\cdot)}\Rightarrow 1\,]-\Pr[\,\mathsf{A}^{\mathsf{f}(\cdot)}\Rightarrow 1\,]| \leq \frac{q(q+1)}{2|\mathcal{M}|}$$

Note that the proof is based on a combinatorial argument and not on a computational one. Thus, we only consider the number of queries A is allowed to ask, and we do not need to bound his running time.

Proof. See [17]. □

Thus, it has been proved that we can use a PRP as a PRF. On the other hand, we have a bound on the maximal security we can achieve using a PRP (the so-called birthday bound (BB)). Improved constructions based on PRPs can overcome this problem, for example, see [102].

[8] In fact, if $|\mathcal{M}|<\infty$, $|\mathcal{PERM}(\mathcal{M})|=|\mathcal{M}|!$, while $|\mathcal{FUNC}(\mathcal{M})|=|\mathcal{M}|^{|\mathcal{M}|}$. We remember that $n!\sim\sqrt{2\pi n}\left(\frac{n}{e}\right)^n$ due to the Stirling approximation. For infinite sets the problem is much harder, but, since we do not use infinite sets, we can omit to treat this problem.

Beyond birthday security. Therefore, the birthday bound affect many schemes. Schemes having a better security are called *beyond birthday secure* (BBB) and there is a flourishing literature about them, for example [52, 38, 112, 68] and in our work [20].

2.4.1 Tweakable pseudorandom functions

To add more flexibility, Liskov et al. [92] introduced *tweakable blockciphers* (TBC). Roughly speaking, TBCs have an additional input, the *tweak*, in addition to the key and the block. The role of the tweak is to provide "independent variability", while the key provides "uncertainty to the adversary" [92].

Similarly as for PRFs, we may require that the outputs of a TBC remain random, even if the adversary chooses the inputs and the tweaks.

Definition 8 (TPRF)**.** *A function* $\mathsf{F} : \mathcal{K} \times \mathcal{TW} \times \mathcal{M} \to \mathcal{T}$ *is a* (q, t, ϵ)-tweakable pseudorandom function (TPRF) *if for every* (q, t)*-adversary* A*, the advantage :*

$$\mathsf{Adv}^{\mathsf{TPRF}}_{\mathsf{F}}(\mathsf{A}) := |\Pr[\,\mathsf{A}^{\mathsf{F}_k(\cdot,\cdot)} \Rightarrow 1\,] - \Pr[\,\mathsf{A}^{\mathsf{f}(\cdot,\cdot)} \Rightarrow 1\,]|$$

is upper bounded by ϵ*, where* k *and* f *are chosen uniformly at random from their domains, namely* $\mathcal{K}$ *and* $\mathcal{FUNC}(\mathcal{TW} \times \mathcal{M}, \mathcal{T})$*.*

A function f taken uniformly at random from $\mathcal{FUNC}(\mathcal{TW} \times \mathcal{M}, \mathcal{T})$ is called a *random tweakable function.*

Often, to make the notation lighter, the tweak is put as superscript, i.e., $\mathsf{F}^{tw}_k(m)$ stands for $\mathsf{F}_k(m, tw)$.

As for PRFs, it is interesting to have that $\forall\ k \in \mathcal{K}$ and $\forall tw \in \mathcal{TW}$, $\mathsf{F}^{tw}_k : \mathcal{M} \to \mathcal{M}$ is a permutation. We adapt the TPRF definition to this situation:

Definition 9 (TPRP)**.** *A function* $\mathsf{F} : \mathcal{K} \times \mathcal{TW} \times \mathcal{M} \to \mathcal{M}$ *is a* (q, t, ϵ)-tweakable pseudorandom permutation (TPRP) *if for every* $k \in \mathcal{K}$ *and* $tw \in \mathcal{TW}$*,* $\mathsf{F}^{tw}_k(\cdot) : \mathcal{M} \to \mathcal{M}$ *is a permutation and for every* (q, t)*-adversary* A*, the advantage :*

$$\mathsf{Adv}^{\mathsf{TPRP}}_{\mathsf{F}}(\mathsf{A}) := |\Pr[\,\mathsf{A}^{\mathsf{F}_k(\cdot,\cdot)} \Rightarrow 1\,] - \Pr[\,\mathsf{A}^{\mathsf{f}(\cdot,\cdot)} \Rightarrow 1\,]|$$

is upper bounded by ϵ*, where* k *and* f *are chosen uniformly at random from their domains, namely* $\mathcal{K}$ *and* $\mathcal{TPERM}(\mathcal{TW}, \mathcal{M})$ *(which is the*

space of all functions $\mathsf{f} : \mathcal{TW} \times \mathcal{M} \to \mathcal{M}$ *s.t.* $\forall tw \in \mathcal{TW}$, $\mathsf{f}^{tw}(\cdot)$ *is a permutation).*

A function f taken uniformly at random from $\mathcal{TPERM}(\mathcal{TW}, \mathcal{M})$ is called a *random tweakable permutation.*

Likewise, as for PRPs, TPRPs are called *tweakable blockciphers.* As for PRFs, most TPRFs used in practice are TPRPs. In fact, a secure TPRP is also a secure TPRF. The proof is a simple extension of Prop. 1.

It is possible to build a TPRP $\tilde{\mathsf{F}}$ from the PRP F, for example Liskov et al. have proposed $\tilde{\mathsf{F}}_k(tw, m) := \mathsf{F}_k(tw \oplus \mathsf{F}_k(m))$ [92].
Improved constructions are able to have better security bounds [90, 89, 101].

2.4.2 Strong pseudorandom permutations

The advantage of PRPs and TPRPs is that they are invertible. Some constructions exploit deeply this using the inverse. Thus, to provide security, it is needed that also the inverse gives random outputs. Formally:

Definition 10 (sPRP)**.** *A function* $\mathsf{F} : \mathcal{K} \times \mathcal{M} \to \mathcal{M}$ *is a* (q, t, ϵ)-strong pseudorandom permutation (sPRP) *if for every* $k \in \mathcal{K}$ $\mathsf{F}_k : \mathcal{M} \to \mathcal{M}$ *is a permutation and for every* (q, t)*-adversary* A*, the advantage :*

$$\mathsf{Adv}^{\mathsf{sPRP}}_{\mathsf{F}}(\mathsf{A}) := |\Pr[\,\mathsf{A}^{\mathsf{F}_k(\cdot),\mathsf{F}_k^{-1}(\cdot)} \Rightarrow 1\,] - \Pr[\,\mathsf{A}^{\mathsf{f}(\cdot),\mathsf{f}^{-1}(\cdot)} \Rightarrow 1\,]|$$

is upper bounded by ϵ*, where* k *and* f *are picked uniformly at random from their domains, namely* $\mathcal{K}$ *and* $\mathcal{PERM}(\mathcal{M})$*.*
The adversary can do at most q' *queries to the first oracle and* $q - q'$ *queries to the second oracle for any* $q' \leq q$*.*

Naturally, it is possible to adapt the previous definition to TPRPs:

Definition 11 (sTPRP)**.** *A function* $\mathsf{F} : \mathcal{K} \times \mathcal{TW} \times \mathcal{M} \to \mathcal{M}$ *is a* (q, t, ϵ)-strong tweakable pseudorandom permutation (sTPRP) *if for every* (q, t)*-adversary* A*, the advantage :*

$$\mathsf{Adv}^{\mathsf{sTPRP}}_{\mathsf{F}}(\mathsf{A}) := |\Pr[\,\mathsf{A}^{\mathsf{F}_k(\cdot,\cdot),\mathsf{F}_k^{-1}(\cdot,\cdot)} \Rightarrow 1\,] - \Pr[\,\mathsf{A}^{\mathsf{f}(\cdot,\cdot),\mathsf{f}^{-1}(\cdot,\cdot)} \Rightarrow 1\,]|$$

is upper bounded by ϵ*, where* k *and* f *are chosen uniformly at random from their domains, namely* $\mathcal{K}$ *and* $\mathcal{TPERM}(\mathcal{TW}, \mathcal{M})$*.*
The adversary can do at most q' *queries to the first oracle and* $q - q'$ *queries to the second oracle for any* $q' \leq q$*.*

To make the notation more compact and to avoid the use of different oracles, we add another input, which is ± 1, where $+1$ stands for a query to the first oracle (that is, either F or f), while -1 for an inverse query, that is, a query to the second oracle (either F^{-1} or f^{-1}). That is, a query on input $(m, +1)$ (or $(tw, m, +1)$) stands for a query to the first oracle on input m (or (tw, m)), while a query on input $(m, -1)$ (or $(tw, m, -1)$) stands for a query to the inverse (second) oracle on input m (or (tw, m)).

2.5 Message Authentication Codes (MACs)

The basic cryptographic primitive allowing authenticating message is a Message Authentication Code (MAC) [81]. A MAC is composed of three algorithms. The first is designed to generate the key, which is shared between the sender and the receiver. The sender computes a tag via the tag-generation algorithm to authenticate a message; instead, the receiver verifies the tag associated with the message to check the authenticity of a message. Formally:

Definition 12 (MAC)**.** *A* Message Authentication Code (MAC) *is a triple* $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *where*

- Gen *picks a key in the keyspace* $\mathcal{K}$*.*
- *the tag-generation algorithm* Mac *is an algorithm that takes as input a couple* $(k, m) \in \mathcal{K} \times \mathcal{ME}$ *and outputs a tag* $\tau \leftarrow \mathsf{Mac}_k(m)$ *from the tag space* $\mathcal{TAG}$*.*
- *the verification algorithm* Vrfy *takes as input a triple* (k, m, τ) *in* $\mathcal{K} \times \mathcal{ME} \times \mathcal{TAG}$ *and outputs either the value "*$\top$*"* ("accept") *or "*$\bot$*"* ("reject").

We ask that the couple composed by a message m *and the tag* τ *computed for the message* m *by* Mac *is always considered valid. Formally:*

Correctness: $\forall (k, m) \in \mathcal{K} \times \mathcal{ME},\ \mathsf{Vrfy}(k, m, \mathsf{Mac}(k, m)) = \top$.

A string-input MAC strMAC *has as input space a set of strings, that is* $\mathcal{ME} \subseteq \{0,1\}^*$.

In this work, we consider only deterministic MAC, that is, MAC whose tag-generation algorithm is deterministic (there are also probabilistic MACs, *iv*-based MAC and nonce-based MAC, for example).

2.5.1 MAC security: authenticity

The usual security notion for MAC is "unforgeability". That is, to assert authenticity of a tag from a MAC we ask that it is difficult to create a valid tag τ^* for a message m^* even if the adversary can see tags of messages of his choice, where m^* is a message for which has never been authenticated yet. The couple (m^*, τ^*) is called an *existential forgery*. If we relinquish the requirement that m^* has never been authenticated, we have *strong unforgeability*. (Since when the adversary has received a valid tag τ for a message m, he can simply replay this couple message, tag, to avoid trivial victory, we ask that the couple message, tag (m^*, τ^*) is fresh). Formally:

The $\mathsf{FORGE}^{\mathsf{euf-cma}}_{\mathsf{MAC,A}}$ and $\mathsf{FORGE}^{\mathsf{suf-cma}}_{\mathsf{MAC,A}}$ experiments.	
Initialization: $k \leftarrow \mathsf{Gen}$ $\mathcal{S} \leftarrow \emptyset$ Finalization: $(m^*, \tau^*) \leftarrow \mathsf{A}^{\mathsf{Mac}_k(\cdot),\mathsf{Vrfy}(\cdot,\cdot)}$ If $\top = \mathsf{Vrfy}_k(m^*, \tau^*)$ If $m^* \notin \mathcal{S}$ [If $(m^*, \tau^*) \notin \mathcal{S}$] Return 1 Return 0	Oracle $\mathsf{Mac}_k(m)$: $\tau = \mathsf{Mac}_k(m)$ $\mathcal{S} \leftarrow \mathcal{S} \cup \{m\}$ [$\mathcal{S} \leftarrow \mathcal{S} \cup \{(m, \tau)\}$] Return τ Oracle $\mathsf{Vrfy}_k(m, \tau)$: Return $\mathsf{Vrfy}_k(m, \tau)$

Table 2.1: The $\mathsf{FORGE}^{\mathsf{euf-cma}}$ and $\mathsf{FORGE}^{\mathsf{euf-cma}}$ experiments. For the $\mathsf{FORGE}^{\mathsf{suf-cma}}$ experiment use the lines within the square brackets.

Definition 13 (euf [81])**.** *A* MAC $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *is* (q_M, q_V, t, ϵ)- existentially unforgeable-*secure (*euf*) if for every* (q, t)-*adversary* A

$$\Pr\left[\mathsf{FORGE}^{\mathsf{euf-cma}}_{\Pi,\mathsf{A}} \Rightarrow 1\right] \leq \epsilon$$

where the $\mathsf{FORGE}^{\mathsf{euf-cma}}$*-experiment is defined in Tab. 2.1.*

Definition 14 (euf [81])**.** *A* MAC $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *is* (q_M, q_V, t, ϵ)- strongly unforgeable-*secure (*suf*) if for every* (q, t)*-adversary* A

$$\Pr\left[\mathsf{FORGE}^{\mathsf{suf-cma}}_{\Pi,\mathsf{A}} \Rightarrow 1\right] \leq \epsilon$$

where the $\mathsf{FORGE}^{\mathsf{suf-cma}}$*-experiment is defined in Tab. 2.1.*

Note that the adversary has always a chance of winning: in fact, if he chooses a message m^* in the message space and he picks a candidate tag τ^* uniformly at random, he may, by pure chance, have bumped

into the correct tag. Thus, no scheme can be $(q_M, q_V, t, 0)$-unforgeable. Moreover, we can give a lower bound of the unforgeability security: $\epsilon \geq |\mathcal{TAG}|^{-1}$.

In the rest of the thesis we are interested in strongly unforgeable MAC, thus, when we talk about an unforgeable MAC we mean a strongly unforgeable MAC.

Multiple verification query. In our previous definition, the adversary A has oracle access also to the verification oracle. It may seem that having access to a verification oracle is not so useful. In fact, the only information an adversary can receive from the verification oracle is if the couple (m, τ) he is querying on is valid or not. If it is valid, he can simply reuse it as a forgery; otherwise, it seems that he has learned nothing more and he must go on from scratch. The previous intuition is true and is formalized by the following:

Proposition 2. *Let* MAC $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *be* $(q_M, 0, t, \epsilon)$*-unforgeable. Then, it is* $(q_M, q_V, t, (q_V + 1)\epsilon)$*-unforgeable.*

Note that the bound is $(q_V + 1)\epsilon$ because the adversary may win either asking a valid verification query and using this as his output or trying with a new query as his output.

Proof. The proof is straightforward and can be found in [25]. □

Secure MAC from random functions. A natural construction for a MAC is to create the tag τ applying a pseudo-random function F_k to the message m. Thus, the adversary has to guess a fresh output of a PRF to create a valid forgery. Since a pseudo-random function has outputs indistinguishable from random ones, this should be hard. This is the case and it is formalized by the following

Proposition 3. *Let* $\mathsf{F} : \mathcal{K} \times \mathcal{ME} \to \mathcal{TAG}$ *be a* $(q_M + q_V + 1, t)$-PRF. *Let* MAC $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *defined as follow:*
Gen *picks a random key* k *in* $\mathcal{K}$*, that is,* $k \xleftarrow{\$} \mathcal{K}$
Mac $\mathsf{Mac} : \mathcal{K} \times \mathcal{ME} \to \mathcal{TAG}$*, defined by* $\mathsf{Mac}_k(m) := \mathsf{F}_k(m)$,
Vrfy $\mathsf{Vrfy}_k(m, \tau) = \top$ *iff* $\mathsf{F}_k(m) = \tau$*; otherwise* $\bot$.
Then, MAC *is* $(q_M, q_V, t, \epsilon + (q_V + 1)|\mathcal{T}|^{-1})$*-unforgeable.*

Proof. See Theorem 4.4 [81]. □

Some authors, for example [104], when they define MAC security, they ask the tag-generation function Mac to be a PRF.

Secure MACs need not to provide random tags. Observe that there is no need that the tag is random to have a secure MAC. In fact, let $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ be a (q_M, q_V, t, ϵ)-unforgeable MAC, then, $\Pi' = (\mathsf{Gen}', \mathsf{Mac}', \mathsf{Vrfy}')$ defined by

- $\mathsf{Gen}' = \mathsf{Gen}$,
- $\mathsf{Mac}'_k(m) := m||\mathsf{Mac}_k(m)$,
- $\mathsf{Vrfy}'_k(m, \tau') := \top$ iff $\tau' = m\|\tau$ and $\mathsf{Vrfy}_k(m, \tau) = \top$.

Then Π' is (q_M, q_V, t, ϵ)-unforgeable. In fact, from a forgery (m, τ') against Π', it is possible to create a forgery against Π simply removing the message prefixed in τ'.
On the other hand, Mac' does not output a random value.

This previous secure MAC shows that a tag does not hide the authenticated message. Thus, authenticity does not provide confidentiality

2.6 Authenticated Encryption (AE)

When, both authenticity and confidentiality are needed, the cryptographic tool to use is Authenticated Encryption (AE).
Like a MAC, an authenticated encryption scheme is composed by three algorithms. The first is designed to generate the key, which is shared between the sender and the receiver. The second is a procedure to encrypt a message, obtaining a ciphertext; the third is a procedure for decrypting a ciphertext and verifying the authenticity of the ciphertext. Formally:

Definition 15 (pAE)**.** *A scheme for* probabilistic Authenticated Encryption *(*pAE*) is a triple* $\Pi := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$*, where*

- Gen *picks a key* k *in the keyspace* $\mathcal{K}$*, which is not empty.*
- *the encryption algorithm* Enc *is a probabilistic algorithm which takes as input the couple* $(k, m) \in \mathcal{K} \times \mathcal{ME}$ *and outputs* $c \leftarrow \mathsf{Enc}_k(m)$ *called ciphertext.*
- *the decryption algorithm* Dec *is a deterministic algorithm which takes as input the tuple* $(k, c) \in \mathcal{K} \times \{0,1\}^*$ *and outputs* $m \leftarrow \mathsf{Dec}_k(c)$ *which is either a string* $m \in \mathcal{ME}$ *or the symbol "*$\perp$*" ("invalid").*

We require that the algorithm Dec *is always able to correctly decrypt the output of* Enc*, that is:*

- (Correctness) *if* $\mathsf{Enc}_k(m) = c$*, then,* $\mathsf{Dec}_k(c) = m$ *for any key* k

If $\mathsf{Dec}_k(c) = \perp$ *we say that the algorithm* "rejects" c*, otherwise it* "accepts" c *and* c *is "valid".*

The message m is also called plaintext.

In order to have a probabilistic encryption scheme, often, it is easier to use a deterministic algorithm that takes an additional input, which is randomly picked [119]. For more details, see App. A.

2.6.1 Integrity

Authenticity for AE is called *integrity*. We ask that the adversary is not able to create a new ciphertext, which is deemed valid by the decryption algorithm [14]. Formally:

Definition 16 (INT-CTXT)**.** *A probabilistic authenticated encryption scheme* pAE $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* (q_E, q_D, t, ϵ)-INT-CTXT(Ciphertext integrity)-secure *if the advantage*

$$\mathsf{Adv}^{\mathsf{INT\text{-}CTXT}}_{\Pi}(\mathsf{A}) \;:=\; \Pr\left[\perp \neq m^* \leftarrow \mathsf{Dec}_k(c^*);\; c^* \leftarrow \mathsf{A}^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)}\right]$$

is bounded by ϵ *for every* (q_E, q_D, t) *adversary* A*. The adversary* A *is not allowed to output* c^* *if he received* c^* *as* $c^* \leftarrow \mathsf{Enc}_k(m^*)$ *for a certain input* m^* *that he has asked to the first oracle.*

We can also see the previous security definition as follow:
for every (q_E, q_D, t)-adversary A

$$\Pr[\mathsf{INT\text{-}CTXT}_{\mathsf{A},\Pi} \Rightarrow 1] \leq \epsilon.$$

where the INT-CTXT experiment is described in Tab. 2.2.
Informally, the previous probability can also be stated as:

$$\Pr[\mathsf{A}^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)} \text{ forges }].$$

Access to the decryption oracle. Similarly to what done for MAC, we can observe that having access to the decryption oracle is not useful. In fact,

Proposition 4. *Let* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *be* $(q_E, 0, t, \epsilon)$-INT-CTXT *secure. Then, it is* $(q_E, q_D, t, (q_D + 1)\epsilon)$-INT-CTXT *secure.*

The INT-CTXT experiment.	
Initialization: $k \leftarrow \mathsf{Gen}$ $\mathcal{S} \leftarrow \emptyset$ Finalization: $c^* \leftarrow \mathsf{A}^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)}$ If $\perp \neq \mathsf{Dec}_k(c^*)$ If $c^* \notin \mathcal{S}$ Return 1 Return 0	Oracle $\mathsf{Enc}_k(m)$: $c = \mathsf{Enc}_k(m)$ $\mathcal{S} \leftarrow \mathcal{S} \cup \{c\}$ Return c Oracle $\mathsf{Dec}_k(c)$: Return $\mathsf{Dec}_k(c)$

Table 2.2: The INT-CTXT experiment.

2.6.2 Confidentiality and Integrity

When we ask confidentiality in addition to authenticity, we want that not only is it difficult for any adversary to create a forgery but also that no information about the message can be inferred from the ciphertext. The use of random ciphertexts is a possible solution to avoid that any information is obtained from the ciphertext. Formally:

Definition 17 (pAE-security)**.** *A probabilistic authenticated encryption scheme* (pAE) $\Pi :=$ (Gen, Enc, Dec) *is* (q_E, q_D, t, ϵ)-pAE-secure *if the advantage*

$$\mathsf{Adv}_{\Pi}^{\mathsf{pAE}}(\mathsf{A}) := \left|\Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot),\mathsf{Dec}_k(\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$(\cdot),\perp(\cdot)} \Rightarrow 1\right]\right| \qquad (2.2)$$

is bounded by ϵ *for every* (q_E, q_D, t)*-adversary* A *that respects the following condition:*

(i) If A *queried the first (encryption) oracle on input* m *and was answered* c*, then he is not allowed to query the second (decryption) oracle on input* c*;*

This notion provides privacy, since it assumes that ciphertexts are indistinguishable from random, and authenticity, since it assumes decryption queries made by the adversary on fresh ciphertexts, are not valid.

To have confidentiality, the length of the ciphertext must not give any additional information about the plaintext. Thus, we need the following property:

Definition 18. *Let* Alg *be an algorithm whose inputs are in* $\mathcal{M}_1 \times ... \times \mathcal{M}_n$ *and whose outputs are in* $\mathcal{T}$*. We say that algorithm* Alg

does not reveal, via the length of his output, any information about its inputs apart from their lengths *if there exists a deterministic function* $f : \mathbb{N}^n \longmapsto \mathbb{N}$ *s.t. for all possible inputs* $(x_1, ..., x_n) \in \mathcal{M}_1 \times ... \times \mathcal{M}_n$, $|y| = f(|x_1|, ..., |x_n|)$, *where* $y \leftarrow \mathsf{Alg}(x_1, ..., x_n)$.

This can be seen as a "public length function".

2.7 Random oracle model

When we assume that a blockcipher is a PRP or that a hash function is collision resistant (or pre-image resistant) we are making some assumptions which are in the so called *standard model*.
However, in some cases cryptographic primitives, especially hash functions, are used in an "injustified" way (already in 1993, Bellare and Rogaway [15] realised this).
In particular, in many schemes, the outputs of the hash functions are assumed to be "random". It makes sense, since for a good hash functions, its outputs should be "unpredictable" and the adversary should not be able to control them. On the other hand, since the adversary knows the key of the hash function, he may trivially distinguish the outputs of a hash function, from the ones of a random function.
Thus to give a theoretical base of these schemes, they defined the *random oracle model (ROM)*:

Definition 19 (RO [15])**.** *A* random oracle (RO) *R is a map $R : \{0,1\}^* \to \{0,1\}^n$ chosen by selecting each bit of $R(x)$ uniformly and independently, for every x.*

In practice, the random oracle is used to model hash functions.
That is, hash functions are assumed to be replaced by a random oracle with appropriate range. In this way, an adversary is not able to evaluate the result of the hash function. Instead, to evaluate the hash function, he has to call an oracle.
In this way, we can suppose that the adversary cannot pre-compute the outputs of the hash function and they are "random" for him.

The ROM "controversy". Note that proofs in this model are not completely satisfactory. In fact, a scheme secure in the random oracle model (ROM) may be broken entirely when the random oracle is instantiated with an actual hash function, see [35]. In particular, Canetti et al. [35] built a scheme which is secure in the ROM model, but, which is insecure when the RO is instantiated with *any* possible hash function.
Luckily, by now, all the schemes secure in the ROM-model but insecure

when the RO is instantiated with a hash function, are artificial. That is, they do not arise from actual schemes, but they are only built with the idea of proving a separation result between the ROM-model and the standard model [40, 73].
Considering everything, a proof in the ROM should be as much better than no proof at all, but we must be aware that such a proof might leave some security gaps. However, none of these potential gaps has led so far to a practical attack against any deployed scheme whose security has been proved in the ROM model.
ROM-based security proofs allow more flexible and efficient schemes. Moreover, they allow some cryptographic tasks, as non-interactive non-committing encryption [106].
A complete survey on the topic can be found in [83, 84].

Chapter 3

Leakage and countermeasures

Contents

In this chapter, we present the challenges due to side-channel attacks. First, we introduce what leakage is, explaining in particular why there is leakage, and how it can be exploited. Then, we introduce the countermeasures at the hardware and the implementation level, especially masking, which is the most studied. Finally, we introduce countermeasures at mode level, introducing our starting point regarding constructions; that is, the constructions presented at CCS15 by Pereira et al. [111]: 2 leakage-resilient MACs and a leakage resilient encryption scheme, PSV.

Legend for figures

In all the figures, we denote in red long-term secret, in orange ephemeral secrets, while in green inputs, outputs or values publicly computable from them. For primitives, we denote with dark grey the strongly protected implementations, while in light grey or white the weak protected or not protected implementations.

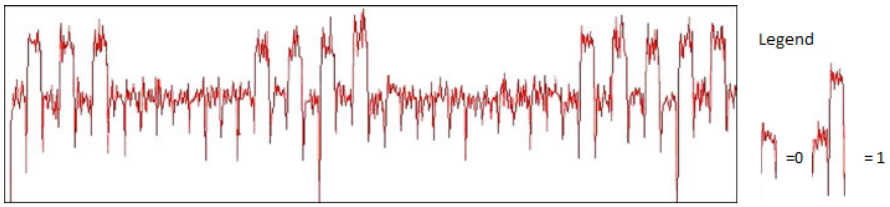


Figure 3.1: A leakage trace, from Oswald et Standaert. [110]. Note that since bit keys are processed one by one (it is an exponentiation and the key is the exponent), thus, with the legend, it is possible to recover the full key with a single leakage trace. In the x-axis we have put the time, while in the y-axis the instantaneous consummation of power.

3.1 Leakage

In the previous chapter, Chap. 2, we have seen some security definitions. Many schemes are achieving them, see, for example, [81].
Note that in all these security definitions, adversaries interact with their oracles only querying them and receiving only their answers.
On the other hand, these schemes (or functions) must be implemented in real devices, either in hardware or software. Then, these devices do all the computations involved to compute the outputs. To do this, they need time and electrical power. Moreover, the computations, moving electrons, create a variable electromagnetic field which induces an electromagnetic radiation which can be measured. Thus, an adversary may collect information about the computations carried by the scheme not only by the outputs he receives but also via these other sources of information, called *side-channel*s.
The information collected in this way may be able, for example, to reveal the key or the plaintext completely. See, for example, Fig. 3.1.

3.1.1 Sources of leakages

The first documented use of side-channel to break cryptography was made by MI5, in 1956 [130].
Kocher made the first academic paper describing a side-channel attack in 1996 [86]. In this paper, Kocher explains "how to find Diffie-Hellman exponents, factor RSA-keys carefully measuring the amount of time used to perform private key operations".
Subsequent works proved that information may be leaked via power consumption [87, 96] or via the electromagnetic radiation a device produces [60, 115], exhibiting attacks working against real devices. Inter-

estingly, even if the electronic device is powered off, it is possible to retrieve information about the secret key, via the so-called "cold boots" attacks [69]. These seminal works were followed by a very vast literature of attacks [96].

3.1.2 Simple and Differential Power Analysis

In this section, we want to present two examples of leakage attacks. In particular, we want to highlight that the number of traces collected with the same key and different inputs is a critical element in the success of a leakage attack. Thus, we can decide to use weaker countermeasures for a component whose key is used only once (or twice) with a given key in the whole history of its usage.

The first attack exploiting the consummation of electrical power used the leakage of a single execution of the device. The measurement is called a *trace* (for example, we have presented a trace in Fig. 3.1). This trace $L = L(t)$ is collected by an oscilloscope and may be seen as the sum of a deterministic part $D = D(t)$, and a noise N (which usually is assumed to be gaussian), that is:

$$L(t) = D(t) + N.$$

An attack based on the leakage of the execution of a single input is called *Simple Power Analysis* (SPA).
Such an attack can exploit conditional branching operations which are key-dependent (see, for example, the leakage trace of Fig. 3.1, where if the key bit used as the exponent is 1, an additional operation, much more power-consuming is performed).
On the other hand, there are effects correlated to data which the device manipulate [87]. Even if these variations are small, they are possible to be detected, especially using many leakage traces for the same key with different inputs. This is called a *Differential Power Analysis* (DPA).
Note that a DPA is much more effective than SPA, but it is also much more complex to do. Moreover, it requires many measurements (for example thousands [128]).
A detailed study of this can be found in [96].

3.2 Countermeasures

Apart from avoiding branching dependent on secret data (especially the key, but also the plaintext), there are many possible ways to limit the

information a leakage trace conveys. For example, there is masking (studied in Sec. 3.2.1), the most used by far, hiding, and shuffling.

Hiding. "The goal of hiding is to make the power consumption of cryptographic devices independent of the intermediate values and independent of the operations that are performed" [95].

Shuffling Already, in the seminal work about power analysis [87], the authors mentioned time randomization as a possible solution.
In the literature, this idea has been developed in many ways:

- *Random delays*, see, for example [37]
- *Shuffling*, see, for example [116].
- *Building a non-deterministic processor*, see, for example [13].

A comprehensive study may be found in [127].

3.2.1 Masking

The most popular and more used countermeasure against side-channel is *masking*. The idea is to split a sensible value x in some shares $x_1, ..., x_{d+1}$, with the constraint that $x_1 + ... + x_{d+1} = x$.[1] That is, $x_1, ..., x_d$ are picked independently, while x_{d+1} is picked according to the recombination law. This is called a *d-order masking*.
In the literature, masking may be done via Boolean masks, polynomial masks, multiplicative masks. A complete survey may be found in [63].
Moreover, every time, or at least very often, the shares are used, they are refreshed, that is, they are changed.
From the seminal works [36, 61], masking have been studied and proved secure [53, 54, 114, 125]. The physical assumptions (independent leakage of the shares) behind these proofs has been questioned, for example, see [91].
We have already implementations of masked AES, for example, [70].

Higher-order masking. To have better security, thus, higher-order masking has been studied [121]. For example, a version of AES masked with a 10-order masking scheme has been proposed [62] and even a 32-order [77] one.

Cost. Masking, and especially high-order masking, is very expensive. It may be even thousands time more expensive, as proved by Goudarzi

[1] The + denotes the operation used to recombine the share. It must not only be understood as the addition in $\mathbb{Z}_n^2$.

and Rivain [62] and Journault and Standaert [77].
For example a version of unmasked AES on 32-bit Cortex M4 needs 661.7 cycles [122], while if masked with a 10-order masking, it needs 480,942 cycles [62] and if with a 32-order [77] scheme, it needs 2,783,510 cycles.

The previous countermeasures can be seen as low level countermeasures, that is, they are either at the hardware or at the implementation level.

3.3 Leakage-resilience

A complementary approach is to design countermeasures at the mode level. That is, a scheme is designed to be inherently more secure against side-channel attacks. Against side-channel attacks, there is flourishing literature proposing schemes that are secure against such attacks. These schemes are called *leakage resilient* (a survey can be found in the work of Kalai and Reyzin [78]).
Observe that most of the papers cited in the survey [78], provide schemes secure in their leakage model.

A constant in these schemes is that they are designed not only exploiting the countermeasures against side-channel attacks, like masking, but also they try to reduce the surface exploitable for an adversary. In this way, these schemes try to be "inherently (more) secure against such physical attacks" [111].
Moreover, since the countermeasure we have against side-channel attacks may be very expensive, it may be useful to reduce the number of execution of primitives implemented with such heavy countermeasures, especially for constrained devices.

3.3.1 Rekeying

One of the most widespread technique in leakage-resilient cryptography is *rekeying* [2].
When two parties have a shared key k, instead of using k directly to encrypt the data, they produce some derived keys $k_1, k_2, ...$ which are then used.
Although rekeying initially was used to reduce the number of blocks processed with the same key (and especially for DES, this was crucial to keep the security[2]), immediately its potential to counter side-channel attacks was understood [85].

[2]Since DES is a 64-bit block cipher with a 56-bit key.

This idea was later developed by Abdalla et al. [1] and Dziembowski et al. [57]. The idea is to use the shared key and an additional input, either random or simply not repeated, to generate a session key. In this way, the master key (which is a long-term secret) can only be targeted once per execution, while if a session key is exposed and retrieved, the adversary is only able to obtain information when that particular session key is used. In particular, he cannot infer any information from an ephemeral key about the master key. In particular, as we show in the next sections (Sec. 3.3.3 and Sec. 3.3.4), it is possible to design encryption or MAC scheme where the master key is used only once per execution, while a session key is used only once (for MAC and encryption) in the whole execution of the protocol.

3.3.2 Leveled implementation

Rekeying allows to implement the primitives in a scheme with different levels of protections. This is called *leveled implementation*.
In a leveled implementation, we have a primitive implemented in a strongly protected way, and others with a weakly (or not at all) protected implementation.

3.3.3 The CCS2015 leakage-resilient MACs

Pereira et al. [111] designed two MACs which use the key only once per authentication. In this way, on each round, only a single component involved in the computation must be strongly protected since only this component can be the target of a DPA attack.

The first MAC. For the authentication of the message m, a stream of ephemeral keys is created from the key k, a random value, called IV, and the message m. The tag corresponds to the last updated key. The details can be found in Alg. 1 and in Fig. 3.2.
Note that these ephemeral keys $k_1, ..., k_l$ are used only in this query and not in subsequent tag-generation queries. Thus, the leakage of these keys is less problematic, and we have only to prevent the leakage of k, protecting only one call to the block cipher F^*.

Security of the MAC1. Pereira et al. [111] showed that it is hard to forge this MAC even if the tag-generation algorithm leaks, based on simulatability, assuming that F^* is implemented with a strongly protected implementations and F has a weakly protected implementation.

Algorithm 1 The CCS2015 leakage-resilient
$\mathsf{MAC1} = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ [111].

It uses a PRF $\mathsf{F} : \mathcal{K} \times \mathcal{B} \to \mathcal{B}$, a strongly protected PRF $\mathsf{F}^* : \mathcal{K}^* \times \mathcal{B} \to \mathcal{B}$ with $\mathcal{B} = \mathcal{K} = \{0,1\}^n$.
For simplicity, we consider only messages composed by full blocks.

- Gen:
  - $k \leftarrow \mathcal{K}^*$
- $\mathsf{Mac}_k(m)$
  - Parse $m = (m_1, ..., m_l)$ with $|m_1| = ... = |m_l| = n$
  - Pick a random $iv \xleftarrow{\$} \{0,1\}^n$
  - $k_1 = \mathsf{F}^*_k(iv)$
  - For $i = 1, ..., l$
    - $k_{i+1} = \mathsf{F}_{k_i}(m_i)$
  - $\tau = k_{l+1}$
  - Return $\overline{\tau} = (iv, \tau)$
- $\mathsf{Vrfy}^1_k(m, \overline{\tau})$:
  - Parse $\overline{\tau} = (iv, \tau)$
  - Parse $m = (m_1, ..., m_l)$ with $|m_1| = ... = |m_l| = n$
  - $k_1 = \mathsf{F}^*_k(iv)$
  - For $i = 1, ..., l$
    - $k_{i+1} = \mathsf{F}_{k_i}(m_i)$
  - $\tilde{\tau} = k_{l+1}$
  - If $\tilde{\tau} = \tau$
    - Return $\top$
  - Else Return $\bot$

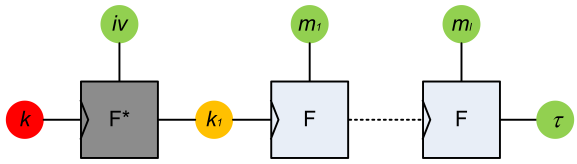


Figure 3.2: The CCS2015 leakage-resilient MAC [111].

The second MAC.

To authenticate a message m, the tag generation algorithm Mac first creates an ephemeral key k_1 from the key k and from a random value, called IV. Then, it digests the message m via a hash function, obtaining $h = \mathsf{H}_s(m)$. Finally, it authenticates the digest h , using a PRF with the ephemeral key, that is, $\tau = \mathsf{F}_{k_1}(h)$. The details can be found in Alg. 2 and in Fig. 3.3.
Note that these ephemeral key k_1 is used only in this authentication and not in subsequent tag-generation queries. Thus, the leakage of these keys is less problematic, and we have only to prevent the leakage of k, protecting only one call to the block cipher F^*.

Algorithm 2 The CCS2015 leakage-resilient MAC2 = (Gen, Mac, Vrfy) [111].

It uses a PRF $\mathsf{F} : \mathcal{K} \times \mathcal{B} \to \mathcal{B}$, a strongly protected PRF $\mathsf{F}^* : \mathcal{K}^* \times \mathcal{B} \to \mathcal{B}$ with $\mathcal{B} = \mathcal{K} = \{0,1\}^n$ and a hash function $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \mathcal{B}$.

- Gen:
  - $k \xleftarrow{\$} \mathcal{K}^*$
  - $s \xleftarrow{\$} \mathcal{KH}$ (s is a public parameter)
- $\mathsf{Mac}_k(m)$
  - Pick a random $iv \xleftarrow{\$} \{0,1\}^n$
  - $k_1 = \mathsf{F}^*_k(iv)$
  - $h = \mathsf{H}_s(m)$
  - $\tau = \mathsf{F}_{k_1}(h)$
  - Return $\overline{\tau} = (iv, \tau)$
- $\mathsf{Vrfy}^1_k(m, \overline{\tau})$:
  - $k_1 = \mathsf{F}^*_k(iv)$
  - $h = \mathsf{H}_s(m)$
  - $\tilde{\tau} = \mathsf{F}_{k_1}(h)$
  - If $\tilde{\tau} = \tau$
    - Return $\top$
  - Else Return $\bot$

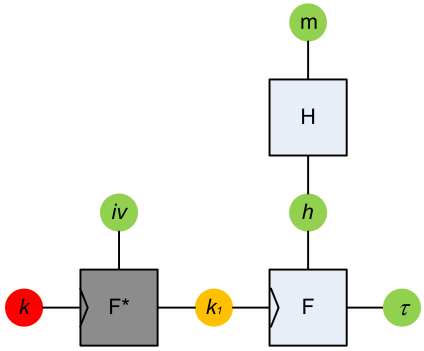


Figure 3.3: The CCS2015 leakage-resilient MAC2 [111].

Security of the MAC2. Pereira et al. [111] proved that it is hard to forge this MAC even if the tag-generation algorithm leaks, based on simulatability. In the proof, F^* is assumed to be a PRF and strongly protected against side-channel attacks, and H is assumed to be collision resistant.

3.3.4 A leakage-resilient encryption scheme

In the same paper [111], the authors put forth a leakage-resilient encryption scheme. We call this scheme PSV after the name of their authors: Pereira, Standaert, and Vivek.

PSV is based on the stream cipher introduced by Standaert et al. [124]. The stream is obtained using a block cipher, and it is based on rekeying. To obtain the first ephemeral key k_1, the master key k is used with a random iv, $k_1 = \mathsf{F}^*_k(iv)$. The iv is given as part of the output.
Given an ephemeral key k_i, it works as follow:

- a new ephemeral key k_{i+1} is produced: $k_{i+1} = \mathsf{F}_{k_i}(p_A)$
- a new stream block y_i is computed: $y_i = \mathsf{F}_{k_i}(p_B)$

where p_A and p_B are two public constants of n-bit, with $p_A \neq p_B$ (n is the block size of the BC F).
To obtain an encryption scheme, the authors simply XOR the message to this pseudo-random stream of blocks.
Note that in this way, each ephemeral key is used only twice in an encryption query and never reused for following encryption queries (here, we do not consider decryption queries). Thus, only a SPA can be made against these ephemeral keys.
The PSV scheme is described in detail in Alg. 3 and Fig.3.4.

Algorithm 3 The PSV encryption scheme [111].

It uses a strongly protected BC^* $\mathsf{F}^* : \mathcal{K}^* \times \mathcal{B}^* \to \mathcal{B}^*$ and a weakly protected BC $\mathsf{F} : \mathcal{K} \times \mathcal{B} \to \mathcal{B}$ with $\mathcal{B}^* = \mathcal{K} = \mathcal{B} = \{0,1\}^n$.

- Gen
  - $k \xleftarrow{\$} \mathcal{K}^*$
  - $p_A, p_B \leftarrow \mathcal{B}$ with $p_A \neq p_B$ $\quad$ p_A and p_B are public
- $\mathsf{Enc}_k(m)$:
  - Parse $m = (m_1, ..., m_l)$ with $|m_1| = ... = |m_{l-1}| = n$ and $|m_l| \leq n$
  - Pick $iv \xleftarrow{\$} \mathcal{B}$
  - $k_1 = \mathsf{F}_k^*(iv)$
  - For $i = 1, ...l-1$
    - $(c_i, k_{i+1}) \leftarrow \mathsf{sPSV}(k_i, m_i)$
  - $y_l = \mathsf{F}_{k_l}(p_B)$
  - $c_l = \pi_{|m_l|}(y_l) \oplus m_l$
  - Return $c = (iv, c_1, ..., c_l)$
- $\mathsf{Dec}_k(c)$:
  - Parse $c = (iv, c_1, ..., c_l)$ with $|c_1| = ... = |c_{l-1}| = n$ and $|c_l| \leq n$
  - $k_1 = \mathsf{F}_k^*(iv)$
  - For $i = 1, ...l-1$
    - $(m_i, k_{i+1}) \leftarrow \mathsf{sPSV}(k_i, c_i)$
  - $y_l = \mathsf{F}_{k_l}(p_B)$
  - $m_l = \pi_{|c_l|}(y_l) \oplus c_l$
  - Return $m = (m_1, ..., m_l)$
- $\mathsf{sPSV}(k_i, m_i)$:
  - $k_{i+1} = \mathsf{F}_{k_i}(p_A)$
  - $y_i = \mathsf{F}_{k_i}(p_B)$
  - $c_i = m_i \oplus \pi_{|m_i|}(y_i)$
  - Return (c_i, k_{i+1})

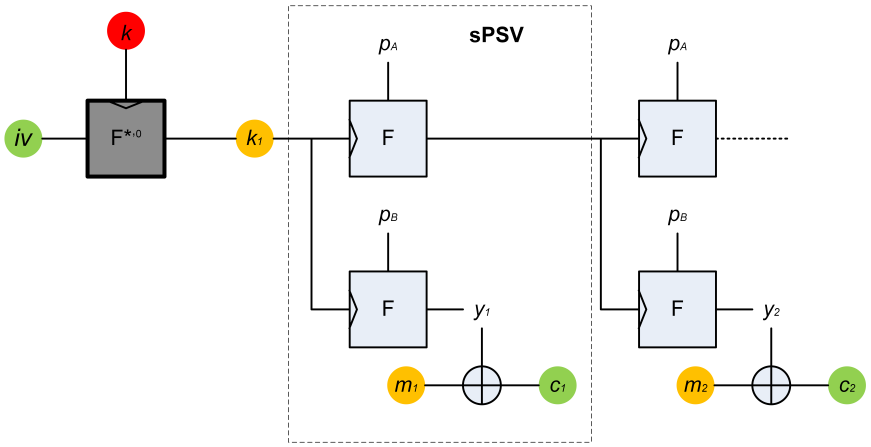


Figure 3.4: The PSV encryption scheme [111]. The single block component sPSV is highlighted.

Security. PSV is aimed to provide confidentiality with leakage. Its security, confidentiality with leakage when only the encryption algorithm leaks (CPAL), has been proved by the authors, again based on simulatability [111].

Chapter 4

Theoretical framework for authenticity with leakage

Contents

This chapter is devoted to the theoretical framework. First, we introduce four security definitions: authenticity for MACs when the tag-generation algorithm leaks and then when both the tag-generation and the verification algorithm leak; after that, we consider the AE case: authenticity for AEs when the encryption algorithm leaks and then when both the encryption and the decryption algorithm leak. Then, we model leakage. After that, we give two models for the leakage of strongly protected components. Finally, we compare our definitions with those of Barwell et al. [10].

4.1 Security definitions with leakage

We start giving the authenticity definitions for MACs, first when the adversary collects the leakage of the tag-generation algorithm, then, when he collects in addition also the leakage of the verification algorithm.
Then, we pass to integrity for AE schemes, first when the adversary collects the leakage of the encryption algorithm, then, when he coolects also the leakage of the decryption algorithm.

Modeling leakage in definitions. In our definitions, to denote that the adversary A can do side-channel attacks, we grant him oracle access to the leakage L of the algorithm $\mathsf{Alg}(\cdot)$. Thus, we use the notation $\mathsf{A}^{\mathsf{Alg}\mathsf{L}(\cdot)}$.
The leakage function of the execution of the algorithm Alg with input x and key k, it is denoted with $\mathsf{L}_{\mathsf{Alg}}(x;k)$.
Following Pereira et al. [111] to denote that the adversary can also profile the leakage, we denote it with $\mathsf{A}^{\mathsf{L}(\cdot)}$ or A^{L}. In fact, the adversary, having access to a device, where he can choose the inputs and the keys used, can learn a lot about the profile of the leakage trace and the information that can be extracted from it.

Structure of the definitions. Our definitions are modifications, for MACs, of the authenticity definition (Def. 14) and, for AE, of the integrity definition (Def. 16).
To denote that we consider leakage we add the suffix L if only tag-generation, for MACs or encryption, for AE, leaks. Instead, for MACs if also verification leaks we add the suffix L2. We do similarly for AE.

Misuse For AE, we consider that the adversary is allowed to tamper the random coins that are used in encryption. To model this, we allow the adversary to choose the randomness r used by the encryption algorithm. We denote this, adding the suffix M.

4.1.1 suf-L: Strong unforgeability with tag-generation leakage

In the unforgeability definition for MAC, Def. 14, we have asked that it is hard for an adversary to produce a forgery. We want that it remains hard even if the adversary has access to the leakage of the tag-generation oracle. Formally (this definition has been stated before from Pereira et al. [111]):

The $\mathsf{FORGEL}^{\mathsf{suf-vcma-L}}_{\mathsf{MAC},\mathsf{L}_M,\mathsf{A}^{\mathsf{L}}}$ experiment.	
Initialization: $k \leftarrow \mathsf{Gen}$ $\mathcal{S} \leftarrow \emptyset$ Finalization: $(m,\tau) \leftarrow \mathsf{A}^{\mathsf{L},\mathsf{MacL}_k(\cdot),\mathsf{Vrfy}_k(\cdot,\cdot)}$ If $(m,\tau) \in \mathcal{S}$ or $\perp = \mathsf{Vrfy}_k(m,\tau)$ Return 0 Return 1	Oracle $\mathsf{MacL}_k(m)$: $\tau = \mathsf{Mac}_k(m)$ $\mathcal{S} \leftarrow \mathcal{S} \cup \{(m,\tau)\}$ Return $(\tau, \mathsf{L}_M(m;k))$ Oracle$\mathsf{Vrfy}_k(m,\tau)$: Return $\mathsf{Vrfy}_k(m,\tau)$

Table 4.1: The $\mathsf{FORGEL}^{\mathsf{suf-vcma-L}}$ experiment against the scheme $\mathsf{MAC} = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$.

Definition 20 ((suf-L))**.** *A Message Authentication Codes* $\mathsf{MAC} = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *with tag-generation leakage function* L_M *provides* $(q_L, q_M, q_V, t, \epsilon)$-strongly existentially unforgeable against chosen message and verification attacks with leakage in the tag-generation *(*suf-L*) if for all* (q_L, q_M, q_V, t)*-adversaries* A^{L}*, we have*

$$\Pr\left[\mathsf{FORGEL}^{\mathsf{suf-vcma-L}}_{\mathsf{MAC},\mathsf{L}_M,\mathsf{A}} \Rightarrow 1\right] \leq \epsilon.$$

where the $\mathsf{FORGEL}^{\mathsf{suf-vcma-L}}$ *experiment is defined in Tab. 4.1.*

We can observe that this definition is simply the *unforgeability* definition where the adversary has oracle access to the leakage of tag-generation queries; moreover he can profile the leakage.

Multiple verification queries. We can observe that, as for unforgeability (euf), it is irrelevant if the adversary has or not oracle access to the verification oracle. This is formalized by the following proposition:

Proposition 5. *Let* $\mathsf{MAC}\ \Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *be* $(q_L, q_M, 0, t, \epsilon)$-suf-L*-unforgeable. Then, it is* $(q_L, q_M, q_V, t, (q_V+1)\epsilon)$-suf-L*-unforgeable.*

Proof. The proof is similar to the proof of Prop. 2. It is enough to adapt that proof to our syntax. □

4.1.2 suf-L2: Strong unforgeability with tag-generation and verification leakage

We extend the previous definition to the case when also verification leaks.

The $\mathsf{FORGEL2}^{\mathsf{suf-vcma-L2}}_{\mathsf{MAC},\mathsf{L}_M,\mathsf{L}_V,\mathsf{A}^{\mathsf{L}}}$ experiment	
Initialization: $k \leftarrow \mathsf{Gen}$ $\mathcal{S} \leftarrow \emptyset$ Finalization: $(m,\tau) \leftarrow \mathsf{A}^{\mathsf{L},\mathsf{MacL}_k(\cdot),\mathsf{VrfyL}_k(\cdot,\cdot)}$ If $(m,\tau) \in \mathcal{S}$ or $\perp = \mathsf{Vrfy}_k(m,\tau)$ Return 0 Return 1	Oracle $\mathsf{MacL}_k(m)$: $\tau = \mathsf{Mac}_k(m)$ $\mathcal{S} \leftarrow \mathcal{S} \cup \{(m,\tau)\}$ Return $(\tau, \mathsf{L}_M(m;k))$ Oracle $\mathsf{VrfyL}_k(m,\tau)$: Return $(\mathsf{Vrfy}_k(m,\tau), \mathsf{L}_V(m,\tau;k))$

Table 4.2: The $\mathsf{FORGEL2}^{\mathsf{suf-vcma-L2}}$ experiment against the scheme $\mathsf{MAC} = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$.

Definition 21 (suf-L2)**.** *A Message Authentication Codes* $\mathsf{MAC} = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *with tag-generation leakage function* L_M *and verification leakage function* L_V *provides* $(q_L, q_M, q_V, t, \epsilon)$-strongly existentially unforgeable against chosen message and verification attacks with leakage in the tag-generation and the verification *(*suf-L2*) if for all* (q_L, q_M, q_V, t)*-adversaries* A^{L}*, we have*

$$\Pr\left[\mathsf{FORGEL2}^{\mathsf{suf-vcma-L2}}_{\mathsf{MAC},\mathsf{L}_M,\mathsf{L}_V,\mathsf{A}} \Rightarrow 1\right] \leq \epsilon.$$

where the $\mathsf{FORGEL2}^{\mathsf{suf-vcma-L2}}$ *experiment is defined in Tab. 4.2.*

In Chap. 5.3.1, we prove that suf-L and suf-L2 are not equivalent. That is, suf-L-secure does not imply suf-L2-secure (with $q_V \neq 0$). On the other hand, suf-L2-secure implies suf-L secure. In fact, a (q_L, q_M, t)-suf-L-adversary against MAC can be seen as a $(q_L, q_M, 0, t)$-suf-L2-adversary against MAC.

Note that for the blackbox definitions the fact that the adversary has or does not have oracle access to the verification oracle is not essential, see Chap. 2.5.1.

4.1.3 CIML: Ciphertext integrity with misuse and encryption leakage

Similarly, we proceed for AE.
First, we want the adversary not being able to produce a valid ciphertext even if he receives the leakage of the encryption algorithm. Moreover,

The $\mathsf{CIML}_{\Pi,\mathsf{L}_E,\mathsf{A}^{\mathsf{L}}}$ experiment	
Initialization: $k \leftarrow \mathsf{Gen}$ $\mathcal{S} \leftarrow \emptyset$ Finalization: $c \leftarrow \mathsf{Adv}^{\mathsf{L},\mathsf{EncL}_k(\cdot,\cdot),\mathsf{Dec}_k(\cdot)}$ If $c \in \mathcal{S}$ or $\perp = \mathsf{Dec}_k(c)$ Return 0 Return 1	Oracle $\mathsf{EncL}_k(r, m)$: $c = \mathsf{Enc}_k(r, m)$ $\mathcal{S} \leftarrow \mathcal{S} \cup \{C\}$ Return $(c, \mathsf{L}_E(r, m; k))$ Oracle $\mathsf{Dec}_k(c)$: Return $\mathsf{Dec}_k(c)$

Table 4.3: The CIML experiment against the scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

we allow the adversary to control the random source used by the probabilistic algorithm Enc.
This may be seen as a generalization of *ciphertext-integrity* [14]. Starting from the INT-CTXT definition, Def. 16, we define the CIML experiment in Tab. 4.3 which leads to the following definition:

Definition 22 (CIML)**.** *An authenticated encryption* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with encryption leakage function* L_E *provides* $(q_L, q_E, q_D, t, \epsilon)$-ciphertext integrity with coin misuse and leakage on encryption *if for all* (q_L, q_E, q_D, t)-*adversaries* A^{L}*, we have*

$$\Pr\left[\mathsf{CIML}_{\Pi,\mathsf{L}_E,\mathsf{A}^{\mathsf{L}}} \Rightarrow 1\right] \leq \epsilon.$$

We can observe that this definition is simply the *ciphertext-integrity* definition (with the adversary in control of the random source) where the adversary has oracle access to the leakage of encryption queries.

To model the adversarial control of the random source, we allow the adversary to choose the randomness r used by the encryption algorithm.

This definition may also be extended to the other possible syntax for AE (those defined in App. A).

In the original paper defining CIML [24], encryption and decryption queries are not separated. Instead, only the total number of queries made to the oracles is bounded. Here, we prefer to divide encryption and decryption querier to be more coherent with the rest of definitions.

The $\mathsf{CIML2}_{\Pi,\mathsf{L}_E,\mathsf{L}_D,\mathsf{A}^{\mathsf{L}}}$ experiment	
Initialization: $k \leftarrow \mathsf{Gen}$ $\mathcal{S} \leftarrow \emptyset$ Finalization: $c \leftarrow \mathsf{A}^{\mathsf{L},\mathsf{EncL}_k(\cdot,\cdot),\mathsf{DecL}_k(\cdot)}$ If $c \in \mathcal{S}$ or $\perp = \mathsf{Dec}_k(c)$ Return 0 Return 1	Oracle $\mathsf{EncL}_k(r,m)$: $c = \mathsf{Enc}_k(r,m)$ $\mathcal{S} \leftarrow \mathcal{S} \cup \{c\}$ Return $(c, \mathsf{L}_E(r,m;k))$ Oracle $\mathsf{DecL}_k(c)$: Return $(\mathsf{Dec}_k(c), \mathsf{L}_D(c;k))$

Table 4.4: The CIML2 experiment against the scheme $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$.

4.1.4 CIML: Ciphertext integrity with misuse and leakage in both encryption and decryption

Similarly to what was done for MAC, we extend the previous definition to the case when also decryption leaks. Formally:

Definition 23 (CIML2)**.** *An authenticated encryption* $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *with encryption leakage function* L_E *and decryption leakage function* L_D *provides* $(q_L, q_E, q_D, t, \epsilon)$-ciphertext integrity with coin misuse and leakage on encryption and decryption *if for all* (q_L, q_E, q_D, t)*-adversaries* A^{L}*, we have*

$$\Pr\left[\mathsf{CIML2}_{\Pi,\mathsf{L}_E,\mathsf{L}_D,\mathsf{A}} \Rightarrow 1\right] \leq \epsilon$$

where the CIML2 *experiment is defined in Tab. 4.4.*

This definition may be also extended to the other possible syntax for AE schemes (see App. A) [27, 65].

4.2 Unbounded leakage model

There are plenty of leakage models [79].
Instead of using previous models, we introduce a new model where we have divided the implementations of hash functions, block ciphers in two classes: strongly protected and not protected. For the not protected implementations, we suppose that it leaks entirely, that is, input, outputs, and key. For the strongly protected ones, we model their leakage in Sec. 4.3 Formally:

Definition 24. *We define the* unbounded leakage model *as a model in which the leakage functions* L*, when queried, returns:*

- *for unprotected building blocks: inputs, outputs* and *keys;*
- *for strongly protected building blocks: inputs and outputs,* but not *the keys*

We call this model unbounded because the leakage of not protected implementations has no bound in the sense that having access to the key, the input, and the output, the adversary has all the information.

Advantages of the unbounded model. This model has many advantages:

Clarity: in this model, strongly protected building blocks and unprotected building blocks are distinguished. Thus, this model highlights on which building block(s) the side-channel countermeasures should be put.

Simple: it completely avoids the problem of describing a leakage function, which is consistent with the traces collected. Instead, in this model, the leakage function, as we see, is straightforward to describe. In fact, the leakage function usually consists of some (often one or two) values computed by the oracle during its execution.
Moreover, in other leakage models the security holds only when the leakage function L is in a certain class. Thus, when the implementations of a scheme change the proofs hold no more.

Strength: A security proof of a scheme in this model can be seen as a proof that even the most potent leakage adversary has to break the strongly protected implementations (or the blackbox assumption) in order to break the scheme.
We note that it is challenging for any adversary to obtain all the information he receives via leakage with the precision we give. Moreover, this approach clarifies where the protections need to be and reduces the risk that implementers create side-channel problems via an optimized implementation (explaining where they must not optimize).

Efficiency: This allows a *leveled implementation* where we can have one component, used only a few times per execution (one or two usually), very well protected against side-channel, for example, a BC with an high-order masking scheme (and thus slower, see
Chap. 3.2.1) and a second one not protected at all, and, presumably, much more efficient.

A consequence of the unbounded leakage model is that a secure scheme has to use its key only as a key of a strongly protected prim-

itive.

Unbounded model and profiling. When the unbounded model is used, the adversary do not need to profile the leakage of the scheme, but only (when it is useful) the leakage of the strongly protected building blocks. In fact, he receives already everything he needs for the not protected building blocks, so he does not gain anything from profiling them. Thus, we may omit that the adversary can profile the leakage of the scheme.

4.3 Modeling strongly protected implementations

It remains to define what the strongly protected implementations leak. First, we explain why it is not enough to protect the key. Then, we offer two models, one leak-free, which is ideal, but usable in a broader contest, another, strong unpredictability, which is more tailored to the authenticity situation.

Protecting only the key it is not enough.

The Kerchoff's laws explain that the security of a cryptosystem should rely *only* in the key which must remain secret. So, we may wonder if we may consider "good enough" an implementation of a primitive s.t. its key is hard to retrieve via side-channel attacks. This is not the case as we show with two examples:

Theoretical Consider a PRF $\mathsf{F} : \mathcal{K} \times \mathcal{B} \to \mathcal{B}$ with $\mathcal{K} = \mathcal{K}_1 \times \mathcal{K}_2$ s.t. $\mathsf{F}_{k_1,k_2}(x) = \mathsf{F}_{k_1,k_2'}(x)$ $\forall x \in \mathcal{B}$, $k_1 \in \mathcal{K}_1$ and $\forall k_2, k_2' \in \mathcal{K}_2$ [that is, we may consider the second part of the key, k_2 as not used].
Now, if the adversary is able to retrieve only the first part of the key, k_1, while the second part, k_2 remains very hard to retrieve, he has retrieved everything he needs. Thus, although the key recovery attack may succeed with probability $2^{-|\mathcal{K}_2|}$, this cannot be considered an implementation "good enough".

Feistel Consider the well known Feistel scheme, see, for example [81] based on a random function. Luby and Rackooff [**?**] proved that a 3-round Feistel construction F'_k with $k = (k_1, k_2, k_3)$ based on a PRF f is a PRP, where k_1, k_2 and k_3 are three keys for the PRF f. So, if we define $\mathsf{F}(x) := \pi_{\frac{|\mathsf{F}'_k(x)|}{2}}(\mathsf{F}'_k(x))$ (that is, F outputs the first half of $\mathsf{F}'_k(x)$), F is a

PRF.
On the other hand, it is simple to see that if the adversary is able to recover all the inputs and outputs of F' via leakage, F is no more a PRF. Moreover, he is able to predict the output for any value.
In fact, supposing that the input x is divided in two parts $x = x_l \| x_r$ of the same size, we have that F' is built as follow:

- $y_{1,l} := x_r$ and $y_{1,r} := \mathsf{f}_{k_1}(x_r) \oplus x_l$
- $y_{2,l} := y_{1,r}$ and $y_{2,r} := \mathsf{f}_{k_2}(y_{1,r}) \oplus y_{1,l}$
- $y_{3,l} := y_{2,r}$ and $y_{3,r} := \mathsf{f}_{k_3}(y_{2,r}) \oplus y_{2,l}$

Output of F': $y := y_{3,l} \| y_{3,r}$
Output of F': $y := y_{3,l}$
Now, an adversary if the adversary is able to recover all the $y_{i,r}$ and $y_{i,l}$ via side-channel, F is no more secure. Let us suppose that the adversary wants to predict the output of $x^* = x_l^* \| x_r^*$. He can proceed as follow:

- A asks $\mathsf{F}(x^1)$ with $x^1 = x_l^1 \| x_r^*$ where x_l^1 is picked uniformly at random with $x_l^1 \neq x_l^*$, so he recovers $y_{1,r}^1$ and he computes the value $z^1 := y_{1,r}^1 \oplus x_l^1$ which is equal to $\mathsf{f}_{k_1}(x_r^1) = \mathsf{f}_{k_1}(x_r^*)$.

- A asks $\mathsf{F}(x^2)$ with $x^2 = x_l^2 \| x_r^2$ where x_l^2 and x_r^2 are picked uniformly at random with $x^2 \neq x^1, x^*$, so he recovers $y_{1,r}^2$ and he computes the value $z^2 := y_{1,r}^2 \oplus x_2^1$ which is equal to $\mathsf{f}_{k_1}(x_r^2)$.

- A asks $\mathsf{F}(x^3)$ with $x^3 = x_l^3 \| x_r^3$ where $x_r^3 = x_r^2$ and $x_l^3 := z^2 \oplus z^1 \oplus x_l^*$, so he recovers $y_{2,r}^3$ and he computes the value $z_2^3 := y_{1,r}^3 \oplus y_{2,1}^3$ which is equal to $\mathsf{f}_{k_2}(z^2 \oplus z^1 \oplus x_l^* \oplus z^2) = \mathsf{f}_{k_2}(z^1 \oplus x_l^*)$.

- We observe that $\mathsf{F}_k(x^*) = z_2^3 \oplus x_r^*$.

Thus, even supposing that an adversary is not able to recover any information about the keys used by F, only the knowledge of the inputs and outputs of f is enough to make F not good for our scopes.

4.3.1 Leak-free

To model the leakage of a heavy protected component, at CCS15 [111] a new model has been introduced: *leak-free*. Roughly speaking, this model says that the adversary cannot make any efficient side-channel attack against this implementation.

Definition 25. *A* PRF $\mathsf{F}^* : \mathcal{K}^* \times \mathcal{B}^* \longmapsto \mathcal{B}^*$ *is implemented in a* leak-free way *if the adversary is not able to recover any useful information from its leakage. We model this, saying that the leakage function of this implementation is void.*

This notion can easily be adapted to PRP, TPRF, and TPRP (for the PRP and TPRP we also need to assume that the strongly protected implementation of the inverse is leak-free).

Strength and weaknesses of the leak-free model. This definition is unambiguous and easy to use. Moreover, it reasonable models for very-well protected components. Additionally, this model (used in combination with the unbounded model) explicitly show where implementers must put all the resources against side-channel.
On the other hand, the existence of such an implementation is questionable, and we may see actual implementations as an imperfect realization. Overall, we think that this model is worth to be used.

The leak-free model used with the unbounded leakage. When we use these two models together, the adversary receives the inputs and the outputs of the strongly protected implementation of primitives. Thus, he does not receive only the keys of the primitives with a strongly protected implementation.

Leak-free and profiling. Since we assume that the leakage of a strongly protected implementation gives no information, thus, profiling the leakage of a strongly protected implementation is useless. Thus, when, we use the leak-free model with the unbounded leakage model, we do not have to worry about profiling and we may omit the term q_{L}.

4.3.2 Strong unpredictability

In this subsection, we consider a new leakage model for strongly protected implementation, which assumes that the adversary can collect some information also about the secrets involved in the computation of the strongly protected implementations. We do not directly bound the information he may collect; instead, we bound what he may do with them.

Unpredictability with leakage. We start from the definition of *unpredictability with leakage* for block ciphers (BC) introduced by Dodis and Steinberger [51]. Roughly speaking, for an unpredictable BC, it is hard to find the output for a fresh input even if the adversary has access to the leakage when the BC is evaluated.

We extend this definition, for PRPs and TPRPs, also considering the leakage of the inverse.
To save some space, we directly describe this notion for TBCs.
We get the corresponding notion for BCs by removing all the tweaks in the definition below.

We denote by $\mathsf{L} = (\mathsf{L}_{\mathsf{Eval}}, \mathsf{L}_{\mathsf{Inv}})$ the leakage function pair associated to an implementation of the TBC, where $\mathsf{L}_{\mathsf{Eval}}(tw, x; k)$ (*resp.* $\mathsf{L}_{\mathsf{Inv}}(tw, z; k)$) is the leakage resulting from the computation of $\mathsf{F}_k(tw, x)$ (*resp.* $\mathsf{F}_k^{-1}(tw, z)$). We also allow the adversary A to profile the leakages and write A^{L} as before, like in [111].

Definition 26 (sUL)**.** *A tweakable block cipher* $\mathsf{F}^* : \mathcal{K}^* \times \mathcal{TW}^* \times \mathcal{M}^* \to \mathcal{T}^*$ *with leakage function pair* $\mathsf{L} = (\mathsf{L}_{\mathsf{Eval}}, \mathsf{L}_{\mathsf{Inv}})$ *is* $(q_E, q_I, q_{\mathsf{L}}, t, \epsilon)$ strongly unpredictable with leakage in evaluation and inversion *(*sUL*), or* $(q_{\mathsf{L}}, q_E, q_I, t, \epsilon)$*-*SUL2*, if for any* $(q_{\mathsf{L}}, q_E, q_I, t)$*-adversary* A*, we have*

$$\Pr[\mathsf{sUL}_{\mathsf{A},\mathsf{F}^*,\mathsf{L}} \Rightarrow 1] \leq \epsilon,$$

where the sUL *experiment is defined in Tab. 4.5, and where* A^{L} *makes at most* q_{L} *(offline) queries to* L*.*

The $\mathsf{sUL}_{\mathsf{A},\mathsf{F}^*,\mathsf{L}}$ experiment.	
Initialization: $k \xleftarrow{\$} \mathcal{K}$ $\mathcal{L} \leftarrow \emptyset$ Finalization: $(x, tw, z) \leftarrow \mathsf{A}^{\mathsf{L},\mathsf{LEval}(\cdot,\cdot),\mathsf{LInv}(\cdot,\cdot)}$ If $(x, tw, z) \in \mathcal{L}$ Return 0 If $z = \mathsf{F}_k(tw, x)$ Return 1 Return 0	Oracle $\mathsf{LEval}(tw, x)$: $z = \mathsf{F}_k(tw, x)$ $\mathsf{l_e} = \mathsf{L}_{\mathsf{Eval}}(tw, x; k)$ $\mathcal{L} \leftarrow \mathcal{L} \cup \{(x, tw, z)\}$ Return $(z, \mathsf{l_e})$ Oracle $\mathsf{LInv}(tw, z)$: $x = \mathsf{F}_k^{-1}(tw, z)$ $\mathsf{l_i} = \mathsf{L}_{\mathsf{Inv}}(tw, z; k)$ $\mathcal{L} \leftarrow \mathcal{L} \cup \{(x, tw, z)\}$ Return $(x, \mathsf{l_i})$

Table 4.5: Strong unpredictability with leakage in evaluation and inversion experiment.

Strength and weaknesses. This definition is game-based, and it is not an idealized physical assumption. In fact, an implementation may be leak-free or not. Instead, sUL is a computational assumption, where the adversary has a certain probability, which we hope is very low, to

win.
It is possible to test in laboratories if an implementation is strongly unpredictable or not. Moreover, it may be quantified the probability that the adversary wins.
With the leak-free assumption, if we have assumed that an implementation was leak-free and after a while, a new attack against it is discovered, we cannot infer anything about the security of the whole scheme. Instead, for sUL, if a new attack is discovered showing that the success of an adversary against that implementation is bigger than what was thought and we are able to quantify the security loss for sUL, we are also able to quantify the security loss for the whole scheme. That is, the security *gracefully degrades* when based on sUL.
Furthermore, actual countermeasures are usually aimed to protect the key, that is, to make a key-recover attack unfeasible. Inputs and the outputs are usually much less protected. Thus, this model assumes that the adversary has access to the inputs and outputs of the implementation he is attacking. He "only" does not fully know the key.

The sUL model used in the unbounded leakage model. When we use these models together, it means that the adversary receives the leakage due to the unbounded model. Moreover, he also receives the leakage of the evaluation of the strongly protected implementation and the leakage of the strongly protected implementation of its inverse evaluation.

Profiling in the unbounded leakage model. We have already explained that if we assume that the leakage of not protected building blocks is unbounded, an adversary has only to profile the leakage of the strongly protected building blocks. When these implementations are modelled as sUL, we consider the q_{L} profiling queries at the mode level as the queries used to profile the leakage of the strongly protected building blocks.

4.4 The Barwell et al. authenticity definition

We finish this chapter presenting the integrity definition with leakage of Barwell et al. [10] presented at ASIACRYPT2017.

Definition 27. *Let* $\mathsf{MAC} = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$ *be a* MAC *with tag generation leakage function* L_M *and verification leakage function* L_V*. The* MAC *is* $(q_M, q_V, q_{l_V}, t, \epsilon)$*-*strongly existentially unforgeable under an adaptive

chosen message with leakage attack ($\mathsf{sEUF-CMLA}$) *if for every* (q_M, q_V, q_{l_V}, t)*-adversary* A *the following advantage*

$$\mathsf{Adv}^{\mathsf{sEUF-CMLA}} := \Big|\Pr\left[\mathsf{A}^{\mathsf{Vrfy}_k(\cdot,\cdot),\mathsf{MacL}_k(\cdot),\mathsf{VrfyL}_k(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\perp(\cdot,\cdot),\mathsf{MacL}_k(\cdot),\mathsf{VrfyL}_k(\cdot,\cdot)} \Rightarrow 1\right]\Big|$$

is bounded by ϵ*. The adversary has access to* q_V *queries to the first oracle (either* $\mathsf{Vrfy}_k(\cdot,\cdot)$ *or* $\perp(\cdot,\cdot)$*),* q_M *to the second oracle and* q_{l_V} *to the third oracle. If the adversary* A *has received* $(\tau,\cdot)$ *as the answer to a query to the second oracle on input* m*, he cannot query his first oracle on input* (m,τ)*.*

We have rephrased here the definition of Barwell et al. [10] to make it consistent with our notations and in a quantitative way. In particular, there, for the authors, a scheme and its implementation are different. Thus, they consider the leakage of that implementation, and then they assess the security for that implementation of a scheme.
A similar definition can be given for AE schemes.

The LAE security of Barwell et al. The previous definition is then used to prove a unique security definition for authenticated encryption (LAE):

$$\mathsf{Adv}^{\mathsf{LAE}} := \Big|\Pr\left[\mathsf{A}^{\mathsf{Enc}_k,\mathsf{Dec}_k,\mathsf{EncL}_k,\mathsf{DecL}_k} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$,\perp,\mathsf{EncL}_k,\mathsf{DecL}_k} \Rightarrow 1\right]\Big|$$

Note that the adversary receives no leakage for the queries which are treated differently in the two situations he has to distinguish.

Difference with our $\mathsf{suf\text{-}L2}$ definition. We start observing that their definition is given via a distinguishing game, while ours is via a computational game. That is, "the adversary must forge a tag". This change is not relevant since the two approaches are equivalent.
Thus, their definition and ours are very similar since in both definitions the adversary has oracle access to $\mathsf{MacL}_k(\cdot)$ and $\mathsf{VrfyL}_k(\cdot,\cdot)$ The only difference is that, for us, the adversary can profile the leakage and we have modeled this with A^{L}.
On the other hand, defining with a distinguishing game is more consistent with their goal: that is first, give a unique security definition for AE with leakage (as it is often done in the blackbox case), second to prove the security of a generic construction via composable security definitions.

Thus, they have to use the same leakage model for both confidentiality and authenticity.
Instead, with our approach (followed by Guo et al. [65]) we want to prove the authenticity with leakage separately from the confidentiality with leakage. Thus, we can use a different leakage model in each situation.

Chapter 5

Constructions

Contents

This chapter is devoted to present the MAC constructions achieving suf-L and suf-L2 and the AE schemes achieving CIML and CIML2.
First we present HBC which is suf-L. It is an amelioration of the CCS MAC2 (see 3.3.3, in particular Fig. 3.3 and Alg. 2).
Then, we use this MAC as a subroutine in AE-encryption scheme, DTE, which is CIML.
After that, we discuss the problem of verification and decryption leakage. Then, we argue why the solution to the previous problem is not trivial. After that, we provide a solution for MACs, building a suf-L2-secure scheme. We obtain the solution changing only the verification algorithm of HBC, from recomputing the tag $\tilde{\tau}$ and comparing it to the actual value τ provided by the adversary, to computing a certain value $\tilde{h}$ from the tag and checking it, which, if the tag is not good is "random", thus, it gives no information to the adversary even if it is leaked.
In addition, we provide a variant which is BBB secure.
This solution is applied to DTE, obtaining DTE2, which is CIML2.
From DTE2, we propose another scheme EDT (Encrypt-Digest-then-Tag) which is CIML2-secure. With respect to DTE2 it is no more misuse-resistant, but since the ciphertext is verified before the plaintext is computed, the decryption of invalid ciphertexts does not give any useful information to the adversary (in particular, it is CCA-secure with leakage.
Finally, we propose a scheme, CONCRETE, which, differently from all other constructions, is CIML2-secure using a single call to the strongly protected component, while, both DTE2 and EDT needs two calls to the strongly protected component.

The leakage model used. As in all this thesis, we use the unbounded leakage model (see Def. 24). Note that in this chapter, we model strongly protected implementations as *leak-free* (see Def. 25).

5.1 HBC: a suf-L MAC.

We start by observing that the CCS MAC2 (see Chap. 3.3.3, Fig. 3.3 and Alg. 2) is not suf-L in the unbounded leakage model. Then, we propose

a new MAC, HBC, which is a modification of the CCSMAC2 which is suf-L.

The CCS MAC2 in the unbounded leakage model. Although Pereira et al. [111] proved the security of MAC2 when there is leakage, MAC2 is not secure in the unbounded leakage model. This because their proof assumes that F^* is leak-free and F is 2-simulatable, while, here, in the unbounded leakage model, F is assumed to leak everything.
Infact, if k_1 is leaked during the computation of the tag (iv^1, τ^1) for a message m^1, a forgery can be easily computed. It is enough to compute $h^2 = \mathsf{H}(m^2)$ for a different message m^2. Simply, compute $\tau^2 = \mathsf{F}_{k_1}(h^2)$ and the forgery is (m^2, iv, τ^2).

We can do a similar attack for the CCS MAC1.

We stress again the fact that the security or insecurity of MAC2 depends on the leakage model used in the proof. We do not want to make any criticism of the work of Pereira et al. [111].

The simplest way to have a leakage-resilient MAC is to compose a hash function with a blockcipher implemented with strong countermeasures. (This MAC has been implicitly introduced in the ASIACCS2018 paper [24] and its eprint version [23]. It has also been mentioned in the work of Dodis and Steinbergerer [51]).

HBC from the CCS MAC2. In the CCS MAC2, first, an ephemeral key is created, then, this ephemeral key is used to authenticate the hash.
It is easier and more efficient to use directly the strongly protected PRF F^* to authenticate the hash.

Considering a hash function $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \{0,1\}^n$ and a PRF $\mathsf{F}^* : \mathcal{K}^* \times \{0,1\}^n \to \{0,1\}^n$ with a strongly protected implementation, with $\mathcal{HM} = \{0,1\}^*$, we can build a MAC, HBC, which is suf-L-secure in the unbounded model, as described in Alg. 4 and Fig. 5.1. Note that for this MAC, $\mathcal{ME} = \{0,1\}^*$ and $\mathcal{TAG} = \mathcal{T}^* = \{0,1\}^n$.

5.1.1 Security of HBC

Before proving the security of HBC, we have to describe its leakage in the unbounded model:

Algorithm 4 HBC based on the Digest and Tag parts of DTE - Full description.

- Gen:
  - $k \xleftarrow{\$} \mathcal{K}^*$
  - $s \xleftarrow{\$} \mathcal{KH}$ (s is a public parameter)
- $\mathsf{MAC}_k(m)$:
  - $h = \mathsf{H}_s(m)$ // digest
  - $\tau = \mathsf{F}^*_k(h)$ // tag
  - Return τ
- $\mathsf{Vrfy}_k(m, \tau)$:
  - $h = \mathsf{H}_s(m)$
  - $\tilde{\tau} = \mathsf{F}^*_k(h)$
  - If $\tau == \tilde{\tau}$ Return $\top$, Else Return $\perp$.

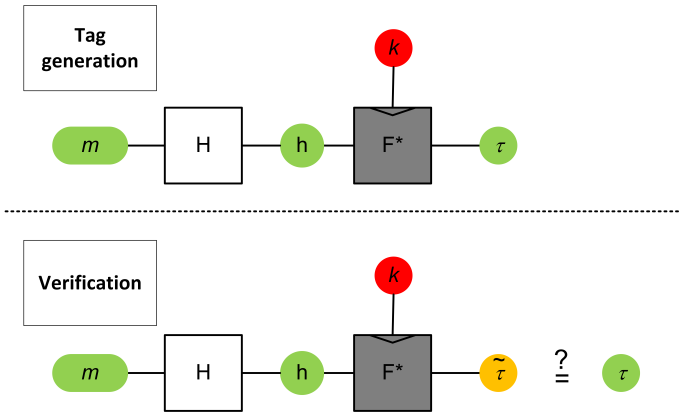


Figure 5.1: The leakage resilient MAC HBC. Leakage reveals the orange value.

Leakage: $\mathsf{L}_M(m; k) := \emptyset$, that is, there is no leakage, since h can be computed by anyone and τ is the output.

Then, we can prove the authenticity with the leakage of HBC:

Theorem 3. *Let* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \{0,1\}^n$ *be a* $(t_2, \epsilon_{\mathsf{CR}})$*-collision resistant hash function. Let* $\mathsf{F}^* : \mathcal{K}^* \times \{0,1\}^n \to \{0,1\}^n$ *be a* $(q, t_1, \epsilon_{\mathsf{PRF}})$*-pseudorandom function (*PRF*). Let* $\mathcal{HM} = \{0,1\}^*$.
Then, HBC *is* (q_M, q_V, t, ϵ)-suf-L*-secure in the unbounded leakage model with*

$$\epsilon \leq \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{CR}} + (q_V + 1)2^{-n}$$

with $q = q_M + q_V + 1$, $t_1 = t + t_{\mathsf{ch}(1,\mathcal{KH})} + q' t_{\mathsf{H}}$ *and* $t_2 = t + q t_{\mathsf{H}} + t_{\mathsf{f}^*(q')}$.

Idea of the proof. Here we present only a sketch of the proof. The full proof may be found in App. B.1.
Note that the actual flow of the proof is different; instead, our sketch aims to give only the main ideas.

Sketch. First, we replace the PRF $\mathsf{F}^*_k(\cdot)$ with a random function f^* with the same signature.
Then, we want to prove that every verification query (m^i, τ^i) is invalid. We may have two different events with respect to the value $h^i = \mathsf{H}_s(m^i)$ obtained during the ith verification query:

F_1: $h^i = h^j$ with $h^i = \mathsf{H}_s(m^i)$ and $h^j = \mathsf{H}_s(m^j)$, where m^i and m^j are respectively the ith and the jth tag-generation query.
Since this implies a collision for the hash function, the probability that this happens is bounded easily by ϵ_{CR}.

F_2: Event F_1 does not happen, thus $\mathsf{f}^*(h^i)$ has never been computed in tag-generation queries.
Since f^* is a random function, the probability that $\mathsf{f}^*(h^i) = \tau^i$ is exactly 2^{-n}.

□

5.2 DTE, Digest-Tag-and-Encrypt, a CIML-secure AE-scheme

This section is inspired by the ASIACCS18 paper by Berti et al. [24] and its eprint version [23].
First, we introduce the idea to compose a leakage-resilient MAC and a leakage-resilient encryption scheme. Then, we introduce DTE and finally we prove its integrity with leakage.

5.2.1 The double IV composition

We want to build a CIML-secure AE scheme. The first idea is to combine the previous MAC, HBC with the LR-encryption scheme PSV. Among the various possibilities, we choose the *Double* IV (DIV) approach:

- DIV is based on the MAC-then-Enc paradigm, composing a MAC and an encryption scheme. MAC and Enc has two different keys, respectively k_E and k_M.
- first a randomness r is picked uniformly at random and the MAC is used to authenticate both the randomness r and the message m

obtaining the tag $\tau = \mathsf{Mac}_{k_M}(r, m)$[1].

- The tag τ of the MAC is asked to be (pseudo)random.
- The encryption scheme Enc uses as his own randomness r_E the tag τ and encrypts both the randomness r and the message m, that is $C = \mathsf{Enc}_{k_E}(r_E, r\|m)$ with $r_E = \tau$.
- The ciphertext is $c = (\tau, C)$. To decrypt, first both the randomness r and the message m are retrieved via $(r, m) = \mathsf{Dec}_{k_E}(r_{\mathsf{Enc}}, C)$, with $r_{\mathsf{Enc}} = \tau$, then, it is checked if the tag τ is correct.

For more details, see Alg. 5.

As explained in [24] this idea of encrypting the $\mathsf{IV}_{\mathsf{Mac}}$, gives stronger confidentiality with leakage.
Note that, if we do not consider leakage, the randomness r need only not be repeated and may be seen as a *nonce* [119]. In this case, the DIV composition is misuse-resistant (MRAE, see Def. 31) as long as the encryption scheme is good.

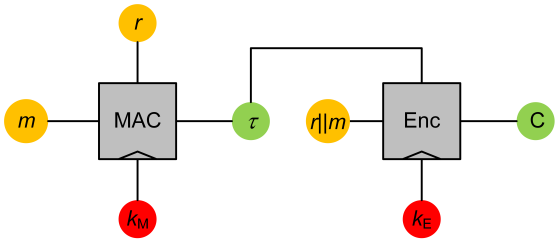


Figure 5.2: The DIV composition.

5.2.2 The DTE construction

We would like to use the DIV composition using the previous MAC, HBC and the encryption scheme PSV (see Chap. 3.3.4, Fig. 3.4 and Alg. 3). If possible, we would like that a single key is used.
Our first proposal is DTE, Digest-Tag-and-Encrypt:
Digest: $h \leftarrow \mathsf{H}(r||m)$ with r random
that is, first a randomness r is picked; then, the randomness r and the message m are digested via an hash function

[1] Often, in the literature an encryption scheme using in this way the randomness is called IV-based encryption scheme and the randomness is called Initialization Vector (IV) [104].

Algorithm 5 The DIV (Double IV) composition [24]

The DIV composition, based on a PRF-secure
IV-MAC $= (\mathsf{Gen}_M, \mathsf{Mac}, \mathsf{Vrfy})$ and an IV-based-encryption scheme
$\Pi = (\mathsf{Gen}_E, \mathsf{Enc}, \mathsf{Dec})$, obtaining the pAE scheme
$\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$:

- Gen:
  - $(k_M, k_E) \leftarrow (\mathsf{Gen}_M, \mathsf{Gen}_E)$
- $\mathsf{Enc}_{k_M,k_E}(m)$:
  - Pick r uniformly at random
  - $\tau \leftarrow \mathsf{Mac}_{k_M}(r, m)$
  - $C \leftarrow \mathsf{Enc}_{k_E}(\tau, r\|m)$
  - Return (τ, C)
- $\mathsf{Dec}_{k_M,k_E}(c)$:
  - Parse c in (τ, C)
  - $(r, m) \leftarrow \mathsf{Dec}_{k_E}(\tau, C)$
  - If $\mathsf{Vrfy}_{k_M}(\tau, (r, m)) = \top$ Return m
  - Else Return $\perp$

Tag: $\tau = \mathsf{F}^*_k(h)$
that is, we authenticate the digest
Encrypt: $C \leftarrow \mathsf{PSVEnc}_k(\tau, r\|m)$
that is, we encrypt using PSV, where the first ephemeral key, $k_0 = \mathsf{F}^*_k(\tau)$. [2]

- the output is $c = (\tau, C)$.

Omitting r, HBC corresponds to the Digest and Tag part of DTE. For decryption, first, r and m are retrieved from $c = (\tau, C)$ via $\mathsf{PSVDec}_k(\tau, C)$, then, the authenticity of c is checked, that is, we check if $\tau = \mathsf{F}_k(h)$ with $h = \mathsf{H}_s(r\|m)$.

For more details, see Alg. 6 and Fig. 5.3.

Security of DTE. DTE achieves many security goals:

- it is a secure AE-scheme (nonce-based, where the randomness r is seen as a nonce, and the adversary provides it)
- it is misuse-resistant (blackbox) (r is again seen as a nonce and provided by the adversary)
- it is CIML-secure

[2] Note that when we have described PSV, in Alg. 3 the first ephemeral key is denoted with k_1, here we denote it with k_0 since, in this way k_i will be used to create the stream block used to encrypt the ith message block.

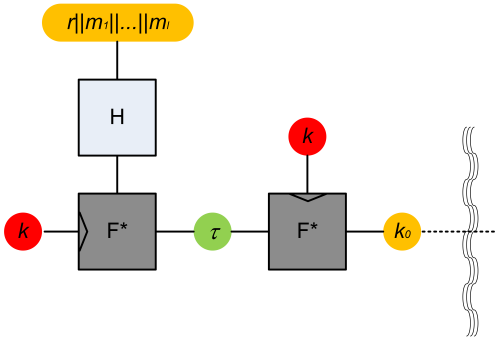


Figure 5.3: DTE leakage-resilient AE (part I). Part II can be found in Fig. 5.4.

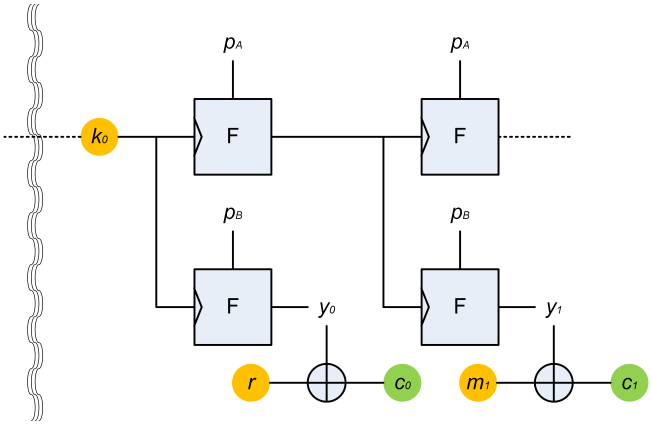


Figure 5.4: Part II of DTE.

- it provides confidentiality in the presence of leakage in encryption[3]

All the other results being a sideline to the scope, here, we prove only the CIML-security. For the other proofs, see the ASIACCS18 paper [24] or its eprint version [23].

5.2.3 The CIML-security of DTE

Before proving the CIML security of DTE, we have to consider what misuse means and what is its leakage in the unbounded model:

Misuse: Since misuse models the situation where the adversary has taken control of the random source, the adversary during encryption queries provides the randomness r alongside with the message. He has no constraint on the choice of r (other than $r \in \mathcal{B}$).

[3]The leakage model used in this proof is 2-simulatability

Algorithm 6 DTE - Full description.

DTE

- Gen:
  - $k \xleftarrow{\$} \mathcal{K}$
  - $s \xleftarrow{\$} \mathcal{KH}$
  - $p_A, p_B \xleftarrow{\$} \mathcal{B}$ (s, p_A, p_B are public parameters)

- $\mathsf{Enc}_k(m)$, where $m = (m_1, m_2, \ldots, m_l)$:
  - $r \xleftarrow{\$} \{0,1\}^n$
  - $\tau \leftarrow \mathsf{Tag}_k(r, m)$:
    - $h = \mathsf{H}_s(r||m)$ // digest
    - $\tau = \mathsf{F}^*_k(h)$ // tag
  - $C \leftarrow \mathsf{PSVEnc}_k(\tau, (r, m))$: // ...and encrypt
    - $k_0 = \mathsf{F}^*_k(\tau)$
    - $y_0 = \mathsf{F}_{k_0}(p_B)$
    - $c_0 = y_0 \oplus r$
    - For $i = 1, ..., l$
      - $k_i = \mathsf{F}_{k_{i-1}}(p_A)$
      - $y_i = \mathsf{F}_{k_i}(p_B)$
      - $c_i = \pi_{|m_i|}(y_i) \oplus m_i$
    - $C = (c_0, c_1, ..., c_l)$
  - Return $c \leftarrow (\tau, C)$
- $\mathsf{Dec}_k(c)$, where $c = (\tau, c_0, c_1, c_2, \ldots, c_l)$:
  - $(r, m) \leftarrow \mathsf{PSVDec}_k(\tau, C)$:
    - $k_0 = \mathsf{F}^*_k(\tau)$
    - $y_0 = \mathsf{F}_{k_0}(p_B)$
    - $r = y_0 \oplus c_0$
    - For $i = 1, ..., l$
      - $k_i = \mathsf{F}_{k_{i-1}}(p_A)$,
      - $y_i = \mathsf{F}_{k_i}(p_B)$
      - $m_i = \pi_{|c_i|}(y_i) \oplus c_i$
  - $\mathsf{TagVrfy}_k(\tau, (r, m))$:
    - $h = \mathsf{H}_s(r||m)$
    - $\tilde{\tau} = \mathsf{F}^*_k(h)$
    - If $\tau = \tilde{\tau}$ Return $(m_1, ..., m_l)$, Else Return $\perp$.

Leakage: Since in the unbounded model the adversary should be able to receive all the inputs and outputs of every primitive and all the keys not used by the strongly protected implementation of the primitive, we need h, all the ephemeral keys $k_0, ..., k_l$ and the $y_0, ..., y_l$ to be given by the leakage function.
Note that if we set $\mathsf{L}_E(r, m; k) := k_0$ the adversary has all the information he wants.
In fact, he can recompute h from r and m knowing the key s of the hash function; moreover, from k_0 all the $k_1, ..., k_l$ and $y_0, ..., y_l$ may be recomputed.[4]

Now, we can prove the integrity with leakage of DTE:

Theorem 4. *Let* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \{0,1\}^n$ *be a* $(t_2, \epsilon_{\mathsf{CR}})$*-collision resistant and* $(t_2, \epsilon_{\mathsf{roPR}})$*-range-oriented preimage resistant hash function. Let* $\mathsf{F}^* : \mathcal{K}^* \times \{0,1\}^n \to \{0,1\}^n$ *be a* $(2q, t_1, \epsilon_{\mathsf{PRF}})$*-pseudorandom function. Let* $\mathsf{F} : \mathcal{K} \times \mathcal{B} \to \mathcal{B}$*. Let* $\mathcal{HM} = \{0,1\}^*$*. Let* L *be the maximal number of blocks for a message.*
Then, DTE *is* (q_E, q_D, t, ϵ)-CIML*-secure in the unbounded leakage model with*

$$\epsilon \leq \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{CR}} + q\epsilon_{\mathsf{roPR}} + (q_D + 1)2^{-n}$$

with $q = q_E + q_D + 1$ *and*

$$t_1 = t + t_{\mathsf{ch}(1,\mathcal{KH})} + t_{\mathsf{chn}(2,\mathcal{B})} + (q+1)(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}), \text{ and}$$

$$t_2 = t + t_{\mathsf{chn}(2,\mathcal{B})} + (q+1)(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}) + t_{\mathsf{f}^*(2(q+1))}.$$

Note that we ask nothing about the security of F. We only need that it has the good syntax, that is, its input space, output space, and keyspace are all equal to $\{0,1\}^n$.
This result is particularly interesting and surprising.[5]

Here we present only a sketch of the proof. The full proof may be found in App. B.2.
Note that the actual flow of the proof is different from obtaining better bounds; instead, our sketch aims to give only the main ideas.

Sketch. First, we replace the PRF $\mathsf{F}^*_k(\cdot)$ with a random function f^* with the same signature.

[4] This example shows the power of the unbounded leakage model as the description of the leakage function is very simple in this case.

[5] Clearly, F must be a PRF to prove the AE-security, MRAE-security and confidentiality with leakage. This result should not be intended as in DTE it is not necessary to instantiate F with a PRF. Simply, we do not need this hypothesis for CIML.

Then, we want to prove that every decryption query (τ^i, c^i) is invalid. We may have three different events with respect to the value $h^i = \mathsf{H}_s(r^i \| m^i)$ obtained during the ith decryption query:

F_1: $h^i = h^j$ with $h^i = \mathsf{H}_s(r^i || m^i)$ and $\mathsf{H}_s(r^j || m^j)$ and where (r^i, m^i) and (r^j, m^j) are respectively the ith and the jth encryption query

F_2: Event F_1 does not happen and $h^i = \tau^j$ where τ^j is the tag of the jth encryption query on input (r^j, m^j)

F_3: Neither event F_1 nor event F_2 happens.

Let E_i be the event that the ith decryption query is fresh and valid.
We bound now $\Pr[E_i \cap F_{i'}]$ for $i = 1, ..., q_D + 1$ and $i' = 1, 2, 3$:

$E_i \cap F_1$: it means that we have found a collision for the hash function H_s

$E_i \cap F_2$: it means that we have found a pre-image for the hash function for the value τ^j. Since the tags are picked uniformly at random (they are the outputs of f^*), the probability that this happens is bounded by ϵ_{roPR}. Moreover, there at most $q_E + q_D$ possible different targets.

$E_i \cap F_3$: since $\mathsf{f}^*(h^i)$ has never been computed and f^* is a random function, the probability that $\mathsf{f}^*(h^i) = \tau^i$ is exactly 2^{-n}.

□

Remark on the proof. When we consider event $E_i \cap F_2$, there are q_E possible target due the τ^j computed during encryption queries. The q_D additional targets come from the proof. In fact, the adversary may have chosen a couple randomness-message (r, m), have computed his hash $h = \mathsf{H}_s(r \| m)$ and then he may first, ask a decryption query $c = (\tau, C)$ with $\tau = h$ and, then, an encryption query on input (r, m). Now, when the roPR-game is simulated, the tag the adversary receives after this encryption query is no longer random, since it has already been computed before.

In the complete proof, we use a different path to avoid to have q_{D+1} as a global factor for the advantage.

5.3 The problem of decryption and verification leakage for authenticity

Now, it is natural to wonder what happens if also verification or decryption leaks. This section is devoted to proving that such additional leakage provides a new and dangerous threat to authenticity. In fact, we prove that HBC is suf-L and not suf-L2 secure, and DTE is CIML but

no CIML2 secure, exhibiting attacks.
Thus, we have proved that both suf-L $\not\Rightarrow$ suf-L2 and CIML $\not\Rightarrow$ CIML2.

We start with the attack against HBC; then, we introduce two attacks against DTE. The first attack is due to the structure of DTE, and will be prevented with a patch, DTE′, while the second is based on the attack against HBC.
We end discussing why adding more components with a strongly protected implementation does not help.

This section covers the ToSC17 paper [27] (one attack is also present in the ASIACCS18 paper [24]).

Unbounded decryption leakage. First, we have to define both the leakage function of verification for HBC and of decryption for DTE:
HBC $\mathsf{L}_V(m, \tau; k) := \tilde{\tau}$
DTE $\mathsf{L}_D(c; k) := (k_0, \tilde{\tau})$
since, from these values, the adversary is able to recompute all intermediate values and all keys used by the algorithms, apart from the key used in the strongly protected implementations. For both tag-generation and encryption, we still use the same leakage functions, that is $\mathsf{L}_M(m; \tau) = 0$ for HBC and $\mathsf{L}_E(r, m; k) := k_0$ for DTE.

5.3.1 HBC is not suf-L2

We start with HBC. In this attack, we exploit the fact that the correct tag, $\tilde{\tau}$, is recomputed during the verification query. Since $\tilde{\tau}$ is recomputed, it may be leaked. The computation of $\tilde{\tau}$ poses a problem which we can exploit as follow:

- Ask the verification of (m, τ) for any message m and a random tag τ. Obtain via leakage $\tilde{\tau}$.
- The forgery is $(m, \tilde{\tau})$.

Note that the fact that a verification query on input (m, τ) has been asked does not invalidate the forgery.

This attack is due to the structure of the verification algorithm, which recomputes the correct tag, before comparing it. The vast majority of MACs have a verification algorithm recomputing the correct tag [81]. To the best of our knowledge, the only verification or decryption algorithm not recomputing the correct tag is the one designed by Barwell et al. [10] in a competitive work.

5.3.2 DTE is not CIML2: a first attack

Now, we move to DTE. The first attack exploits the fact that both the tag τ and the first ephemeral key k_0 are computed from the block cipher F_k^* (which has a strongly protected implementation) which uses in the computation the *same* key. Thus, in this attack in a decryption query, the computation of the first ephemeral key k_0 is used to obtain information about a correct tag. In detail:

- Pick some randomness r^* and a message $m^* = m_1^*, ..., m_l^*$ and compute $h^* = \mathsf{H}_s(r^* \| m^*)$.
- Ask for the decryption of ciphertext $c^1 = (\tau^1, C^1)$ with $\tau^1 = h^*$ and any $C^1 = (c_0^1, ..., c_l^1)$. Recover the first ephemeral key $k_0^1 = \mathsf{F}_k^*(h^*)$ via decryption leakage.
- Ask for the decryption of ciphertext $c^2 = (\tau^2, C^2)$ with $\tau^2 = k_0^1$ and any $C^2 = (c_0^2, ..., c_l^2)$. Recover the first ephemeral key k_0^2 via leakage.
- From k_0^2, compute the ciphertext C^3 for PSV in this way:
  - $k_0^3 := k_0^2$,
  - $c_0^3 = \mathsf{F}_{k_0^3}(p_B) \oplus r^*$,
  - For $i = 1, ..., l$
    - $k_i^3 = \mathsf{F}_{k_{i-1}^3}(p_A)$,
    - $c_i^3 = \mathsf{F}_{k_i^3}(p_B) \oplus m_i^*$
  - The PSV-ciphertext $C^3 = (c_0^3, ..., c_l^3)$

  and output (τ^3, C^3) with $\tau^3 = \tau^2$

This ciphertext is valid. In fact, the first ephemeral key is $k_0^3 = k_0^2$ (we have obtained already via leakage $\mathsf{F}_k^*(\tau^2)$ during the second decryption query). Thus, the randomness and the message retrieved by PSVDec is (r^*, m^*). Moreover, $\mathsf{F}_k^*(h^*) = \tau^2$, with $h^* = \mathsf{H}_s(r^* \| m^*)$ since we have already obtained via leakage this value (during the first decryption query).

Roughly speaking, we use twice the leakage of $\mathsf{F}_k^*(\cdot)$ in decryption. The first time to obtain via leakage $\mathsf{F}_k^*(h^*)$ (which is the tag τ^*) associated to h^* (=$\mathsf{H}_s(r^* \| m^*)$), the second time to obtain $\mathsf{F}_k^*(\tau^*)$ which is the first ephemeral key k_0^* associated to τ^*. With the knowledge of a correct triple (hash, tag, and first ephemeral key), it is easy to create the forgery.

Decryption leakage vs encryption leakage. The fundamental difference between encryption leakages and decryption leakages lies in the way the adversary can influence the generation of the ephemeral key k_0. In the CIML case, the adversary can only obtain the k_0's associated with unpredictable random tags while, in the latter case, the adversary

can obtain the k_0's from arbitrarily selected tags, which therefore allows forgeries of valid ciphertexts.

5.3.3 DTE′ - the first patch

Note that in the previous attack, we have used twice the computation of the first ephemeral key in decryption to obtain information about both the first ephemeral key *and* the right tag to use in the forgery.

A simple and efficient way to prevent the previous attack on DTE is to use a tweakable strongly-protected PRF to distinguish these two computations, leading to DTE' (see details in Alg. 7, the where we highlight the changes in gray, and Fig. 5.5). The only difference with respect to the DTE encryption is in the computation of $\tau = \mathsf{F}_k^{*,0}(h)$ and $k_0 = \mathsf{F}_k^{*,1}(\tau)$, that is we tweak F_k^* with one bit to distinguish the tag-computation from the first ephemeral key-computation. In this way, we can see $\mathsf{F}_k^{*,0}$, and $\mathsf{F}_k^{*,1}$ as independent pseudorandom functions. As a result, the adversary is no more able to use a decryption query to obtain the correct tag associated to a certain h, via the leakage of the first ephemeral key k_0. We show next that this patch, although it prevents the previous attack, it is not enough to provide CIML2-security, exhibiting a more powerful forgery attack:

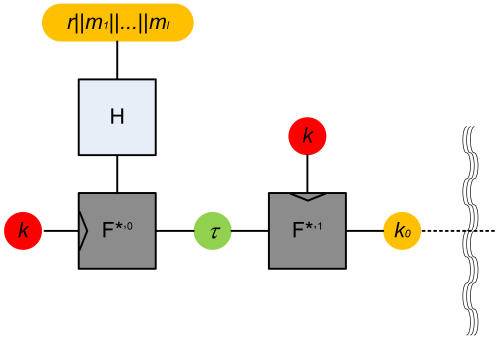


Figure 5.5: DTE' leakage-resilient AE (part I). Part II is identical to Fig. 5.4.

Security of DTE′. It can be easily proved that DTE′ keeps the same security properties as DTE: AE-security, MRAE-security, CIML-security and confidentiality with leakage.
In fact, instead of replacing a BC, a TBC must be replaced in all the security proofs. Apart from this, they are identical.

Algorithm 7 DTE' - The changes from DTE are highlighted.

- Gen:
  - $k \xleftarrow{\$} \mathcal{K}$
  - $s \xleftarrow{\$} \mathcal{KH}$
  - $p_A, p_B \xleftarrow{\$} \mathcal{B}$ (s, p_A, p_B are public parameters)
- $\mathsf{Enc}_k(m)$:
  - Parse $m = (m_1, m_2, \ldots, m_l)$
  - $r \xleftarrow{\$} \{0,1\}^n$
  - $\boxed{\tau \leftarrow \mathsf{Tag}^0_k(r, m)}$:
    - $h = \mathsf{H}_s(r \| m)$ // digest
    - $\boxed{\tau = \mathsf{F}^{*,0}_k(h)}$ // tag
  - $\boxed{C \leftarrow \mathsf{PSVEnc}^1_k(\tau, (r, m))}$: // ...and encrypt
    - $\boxed{k_0 = \mathsf{F}^{*,1}_k(\tau)}$
    - $y_0 = \mathsf{F}_{k_0}(p_B)$
    - $c_0 = y_0 \oplus r$
    - For $i = 1, .., l$
      - $k_i = \mathsf{F}_{k_{i-1}}(p_A)$
      - $y_i = \mathsf{F}_{k_i}(p_B)$
      - $c_i = \pi_{|m_i|}(y_i) \oplus m_i$
    - $C = (c_0, c_1, \ldots, c_l)$
  - Return $c = (\tau, C)$
- $\mathsf{Dec}_k(c, \tau)$:
  - Parse $C = (\tau, C)$
  - Parse $C = (\tau, c_0, c_1, c_2, \ldots, c_\ell)$
  - $\boxed{(r, m) \leftarrow \mathsf{PSVDec}^1_k(\tau, C)}$:
    - $\boxed{k_0 = \mathsf{F}^{*,1}_k(\tau)}$
    - $y_0 = \mathsf{F}_{k_0}(p_B)$
    - $r = y_0 \oplus c_0$
    - For $i = 1, .., l$
      - $k_i = \mathsf{F}_{k_{i-1}}(p_A)$
      - $y_i = \mathsf{F}_{k_i}(p_B)$
      - $m_i = \pi_{|c_i|}(y_i) \oplus c_i$
    - $(r, m) = (r, (m_1, ..., m_l)$
  - $h = \mathsf{H}_s(r \| m)$
  - $\boxed{\mathsf{TagVrfy}^0_k(\tau, (r, m))}$:
    - $\boxed{\tilde{\tau} = \mathsf{F}^{*,0}_k(h)}$ // check tag
    - If $\tau = \tilde{\tau}$ Return m; Else Return $\perp$.

5.3.4 The second attack

Although the previous attack does not work anymore against 'DTE', on the other hand, DTE' is still not CIML2-secure. In fact, it may be attacked with a variant of the attack against HBC (see Sec. 5.3.1:

- Ask for the decryption of $c^1 = (\tau^1, c_0^1, ..., c_l^1)$ for random τ^1 and $C^1 = (c_0^1, ..., c_l^1)$. During the decryption a randomness r^1 and a message $m^1 = m_1^1, ...m_l^1$ are retrieved. Then $h^1 = \mathsf{H}_s(r^1 \| m^1)$ is computed and it is verified if $\tilde{\tau}^1 = \mathsf{F}_k^{*,0}(h^1)$ is equal to τ^1. Recover via leakage r^1, m^1 and $\tilde{\tau}^1$.
- Ask for the decryption of $(\tau^2, c_0^2, ..., c_l^2)$ for $\tau^2 = \tilde{\tau}^1$ and random $C^2 = (c_0^2, ..., c_l^2)$ in order to get $k_0^2 = \mathsf{F}_k^{*,1}(\tau^2)$ via leakage.
- From τ^2 and k_0^2 it is possible to compute the valid encryption of (r^1, m^1). In fact, consider the PSV-ciphertext C^3 built as follow:
  - $k_0^3 = k_0^2$,
  - $c_0^3 := \mathsf{F}_{k_0^3}(p_B) \oplus r^1$,
  - For $i = 1, ..., l$
    - $k_i^3 = \mathsf{F}_{k_{i-1}^3}(p_A)$,
    - $c_i^3 = \mathsf{F}_{k_i^3}(p_B) \oplus m_i^1$
  - The PSV ciphertext $C^3 = (c_0^3, ..., c_l^3)$

  and outputs $c^3 = (\tau^3, C^3)$ with $\tau^3 = \tau^2$.

In fact, in decryption, r^1 and m^1 are retrieved since $k_0^3 = \mathsf{F}_k^{*,1}(\tau^3)$ and $\tilde{\tau}^3 = \tau^3$ since it is the correct tag for (r^1, m^1) (since $\tau^3 = \tilde{\tau}^1$).
As a result, DTE$'$ is not CIML2.

Roughly speaking in this attack, first, we obtain via leakage the good tag $\tilde{\tau}^1$ for a random couple randomness message (r^1, m^1). Then, via leakage, the correct first ephemeral key k_0^2 associated with the tag $\tilde{\tau^1}$ is obtained. Knowing $\tilde{\tau^1}$ and k_0^2 we are able to correctly forge.

Forgery with a given plaintext. In the previous attack, the forgery is the encryption of a random plaintext (in fact, if DTE$'$ is implemented with F a PRF, then, r^1 and m^1 are random, as it may be easily proved). Instead, if the goal is to provide a forgery where the randomness used is r^* and the message encrypted is m^*, it is possible to modify the previous attack slightly, as follow:

- Ask the decryption of a random ciphertext $c^0 = (\tau^0, C^0)$ with τ^0 and C^0 are random. Via leakage, obtain the first ephemeral key k_0^0.
- Compute the PSV-ciphertext C^1 as follow:
  - $k_0^1 = k_0^0$,

  - $c_0^1 := \mathsf{F}_{k_0^1}(p_B) \oplus r^*$,
  - For $i = 1, ..., l$
    - $k_i^1 = \mathsf{F}_{k_{i-1}^3}(p_A)$,
    - $c_i^1 = \mathsf{F}_{k_i^1}(p_B) \oplus m_i^*$
  - The PSV ciphertext $C^1 = (c_0^1, ..., c_l^1)$

  and ask the decryption of $c^1 = (\tau^1, C^1)$ with $\tau^1 = \tau^0$. Obtain via leakage $\tilde{\tau}^1$ (Note that $\tilde{\tau}^1 = \mathsf{F}_k^{*,0}(h^1)$ with $h^1 = \mathsf{H}_s(r^* \| m^*)$ since during the decryption, the couple randomness-message retrieved is (r^*, m^*).
- Then, proceeds as in the previous attack. (Note that here $(r^*, m^*) = (r^1, m^1)$)

5.4 More leak-free components do not help.

We may wonder if this attack may be solved using more primitives with a strongly protected implementations. We give a negative answer both for HBC and DTE.

5.4.1 For HBC

We argue that adding leak-free does not improve security by visual inspection. That is, say, we add more leak-free components after each execution of an F^* in Fig. 5.1. We obtain a composition of strongly-protected PRFs, which can be viewed as a single (less efficient) PRF from the adversary's point of view. Indeed, even if we modify $\mathsf{Mac}_k(m)$ of HBC to return $\tau' = \mathsf{F}_i^* \circ \ldots \mathsf{F}_2^* \circ \mathsf{F}_k^{*,0}(h)$ with $h = \mathsf{H}_s(m)$, for any non negative integers i, j, l, we simply have $\tau' = \mathsf{F}_0^*(h)$ for some F_0^*. This does not prevent the aforementioned forgery attacks.

5.4.2 For DTE

Since tweaking DTE does not allow avoiding forgery attacks, another natural option to consider is the use of more strongly-protected components. Before arguing that such an approach is also unlikely to be effective, we first remind that the goal of DTE is to leverage the good security properties offered by leakage-resilient stream ciphers. This goal implies that for the encryption part of Fig. 5.4, we do not want to use leak-free components on all the blocks (or the interest of the whole construction vanishes). This goal leaves us with two main options, which we discuss next: the addition of leak-free calls in the authentication part of Fig. 5.3 and after the encryption part of Fig. 5.4.

More leak-free components in the authentication part of DTE. It is similar to Sec. 5.4.1.
We argue that such a variation does not improve security by visual inspection. That is, say we add more leak-free components after each execution of an F^* in Fig. 5.3. Independently of whether we operate this change for the first or the second execution of F^*, we obtain a composition of strongly-protected PRFs, which can be viewed as a single (less efficient) PRF from the adversary's point of view. Indeed, even if we modify $\mathsf{Tag}_k^{tw}(h)$ of DTE' to return $\tau' = \mathsf{F}_i^* \circ \ldots \mathsf{F}_2^* \circ \mathsf{F}_k^{*,0}(h)$ and $\mathsf{PSVEnc}_k^{tw}(\tau', \cdot)$ to compute the ephemeral key $k'_0 = \mathsf{F}_l^* \circ \ldots \mathsf{F}_{j+1}^* \circ \mathsf{F}_k^{*,1} \circ \mathsf{F}_j^* \circ \ldots \mathsf{F}_{i+1}^*(\tau')$, for any non negative integers i, j, l, we simply have $\tau' = \mathsf{F}_0^*(h)$ and $k'_0 = \mathsf{F}_1^*(\tau')$ for some F_0^* and F_1^*. This does not prevent the aforementioned forgery attacks.

More leak-free components after the encryption part of DTE. Starting from any $\Pi = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, we define
$\Pi' = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ such that (1) Gen' returns the output of Gen and a (one-more) strongly-protected PRF G^*, (2) Enc' returns the ciphertext c output by Enc and $\tau' = \mathsf{G}^*(c)$, (3) Dec' first checks whether $\tau' = \mathsf{G}^*(c)$ and halts if it does not hold, otherwise it outputs $\mathsf{Dec}(c)$.[6]

Assuming there is a (q, t)-bounded adversary A against Π in CIML2, we show how to build a $(2q + 1, t)$-bounded adversary A' against Π in CIML2 with the same advantage. The reduction is straightforward.

- A queries an encryption of m with randomness r to Enc: A' queries an encryption on (r, m) to Enc' and gets back (c, τ') along with $\mathsf{L}'_E(r, m; k)$. Since τ' is given in the ciphertext, we simply have $\mathsf{L}'_E(r, m; k) = \mathsf{L}_E(r, m; k)$. Then, A' hands A with c and $\mathsf{L}_E(r, m; k)$.
- A queries a decryption of c to Dec: A' conducts two steps,
  1. A' picks a dummy tag τ'_0 and queries a decryption of (c, τ'_0) to Dec'. The check $\tau'_0 = \tilde{\tau}'$, where $\tilde{\tau}' = G^*(c)$, may fail but leaks the right tag $\tilde{\tau}'$ to A';
  2. A' queries a decryption of $(c, \tilde{\tau}')$ to Dec'. Since the check passes, now A' learns $\mathsf{Dec}(c)$ and $\mathsf{L}_D(c; k)$ and forwards them to A.

Eventually A outputs a forgery c^* with some probability. Then, A' runs the subroutine in item 1 on c^* to get the right tag $\tilde{\tau}'^*$ and finally outputs $(c^*, \tilde{\tau}'^*)$, which is a forgery with the same probability.

Despite heuristic, the latter observations suggest that adding a leak-

[6] We omit the use of obvious inputs such as keys, messages and so for readability. Note also that one could additionally hash the ciphertext in Step (2) which would not affect our argument.

free-PRF anywhere inside a CIML2-insecure authenticated encryption scheme will not directly help to prevent forgeries. We leave the proof of a more formal statement (e.g., based on induction of the leak-free calls) as an interesting scope for further research, and for now, we use these observations to motivate the positive result in the next section.

5.5 HBC2 - the solution for MACs

This section is mainly inspired by the ToSC paper [27]. We present two MACs, which are suf-L2. In both, we use the inverse in decryption as a crucial element to solving the problem of verification leakage. The last MAC, HTBC, is implicitly defined by the TEDT [21] and Spook constructions [18].

5.5.1 HBC2: a suf-L2 MAC.

It turns out that using the inverse of the BC in verification provides a powerful ingredient. The main problem was that in HBC the adversary was able to retrieve, via verification leakage, the right tag $\tilde{\tau}$ for a given message m. In fact, $\tilde{\tau} = \mathsf{F}_k^*(h)$ with $h = \mathsf{H}_s(m)$.
But, being able to use the inverse of the BC, it is not necessary to recompute the good tag in verification. In HBC2 instead of computing $\tilde{\tau}$ and comparing it with the tag provided by the adversary, we compute $\tilde{h} = \mathsf{F}_k^{*,-1}(\tau)$ and compare it with $h = \mathsf{H}_s(m)$.
That is, the verification algorithm instead of saying *"your tag τ is wrong because the correct tag is $\tilde{\tau}$" and leaking $\tilde{\tau}$* tells *" your tag τ is wrong because τ is the good tag for this hash value $\tilde{h}$ which is different from your digest h" and leaking $\tilde{h}$*. We highlight this change of approach in Fig. 5.6.
Now, if it is difficult to find a pre-image of this hash value $\tilde{h}$, the scheme may be suf-L2. This modification *only* in the verification algorithm gives birth to HBC2 (see details in Alg. 8 and Fig. 5.6, the changes from HBC are highlighted in gray.)

5.5.2 Security of HBC2

As usual, before proving the security of HBC2, we have to describe its leakage in the unbounded model:
Leakage: $\mathsf{L}_M(m;k) := \emptyset$ as for HBC.

$\mathsf{L}_V(m,\tau;k) := \tilde{h}$. In fact, the adversary, from the inputs, can compute all the other values.

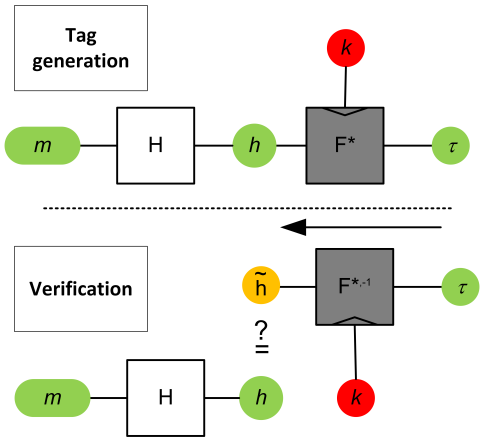


Figure 5.6: HBC2.

Theorem 5. *Let* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \mathcal{B}^*$ *be a* $(t_2, \epsilon_{\mathsf{CR}})$*-collision resistant and* $(t_3, \epsilon_{\mathsf{roPR}})$*-range-oriented pre-image resistant hash function. Let* $\mathsf{F}^* : \mathcal{K}^* \times \mathcal{B}^* \to \mathcal{B}^*$ *be a* $(q_M + q_V + 1, t_1, \epsilon_{\mathsf{sPRP}})$*-strong pseudorandom permutation with a strongly protected implementation. Let* $\mathcal{HM} = \{0,1\}^*$ *and* $\mathcal{B}^* = \mathcal{TAG} = \{0,1\}^n$.
Then, HBC2 *is* (q_M, q_V, t, ϵ)-suf-L2*-secure in the unbounded leakage model with*

$$\epsilon_{\mathsf{sTPRP}} + \epsilon_{\mathsf{CR}} + q_V \epsilon_{\mathsf{roPR}} + (q_V + 1)2^{-n} + \frac{q(q-1)}{2^{n+1}} - \frac{q_M(q_M-1)}{2^{n+1}}$$

with $q = q_M + q_V + 1$, $t + t_{\mathsf{ch}(1,\mathcal{KH})} + qt_{\mathsf{H}} \leq t_1$, $t + qt_{\mathsf{H}} + t_{\mathsf{f}^*(q))} \leq t_2$ *and* $t + qt_{\mathsf{H}} + t_{\mathsf{f}^*(q-1)} \leq t_3$.

The terms of the bound.

- $\epsilon_{\mathsf{sTPRP}}$ due to the use of a sTPRP and not a TPRP
- ϵ_{CR} since having found a collision for the hash function, it is easy to make a forgery
- $q_V \epsilon_{\mathsf{roPR}}$ since the adversary has q_V $\tilde{h}$ (which are randomly picked). If he finds a pre-image for one of them, he can make a forgery.
- $\frac{q(q-1)}{2^{n+1}} - \frac{q_M(q_M-1)}{2^{n+1}}$ because $\tilde{h}$ are not picked uniformly at random (as required for range-oriented pre-image resistance), but using a permutation (so, there are some values which cannot be picked).

Idea of the proof. Here we present only a sketch of the proof. The full proof may be found in App. B.1.
Note that the actual flow of the proof is different; instead, our sketch aims to give only the main ideas.

Algorithm 8 HBC2, a suf-L-secure MAC - Full description.
The changes from HBC are highlighted.

- Gen:
  - $k \xleftarrow{\$} \mathcal{K}$
  - $s \xleftarrow{\$} \mathcal{KH}$ (s is a public parameter)
- $\mathsf{MAC}_k(m)$:
  - $h = \mathsf{H}_s(m)$ // digest
  - $\tau = \mathsf{F}^*_k(h)$ // tag
  - Return τ
- $\boxed{\mathsf{Vrfy}_k(m, \tau)}$:
  - $h = \mathsf{H}_s(m)$
  - $\boxed{\tilde{h} = \mathsf{F}^{*,-1}_k(\tau)}$
  - $\boxed{\text{If } h = \tilde{h} \text{ Return } \top, \text{ Else Return } \bot.}$

Sketch. First, we replace the PRF $\mathsf{F}^*_k(\cdot)$ with a random permutation f^* with the same signature.
Then, we want to prove that every verification query (m^i, τ^i) is invalid. We may have two different events with respect to the value $h^i = \mathsf{H}_s(m^i)$ obtained during the ith verification query:

F_1: $h^i = h^j$ with $h^i = \mathsf{H}_s(m^i)$ and $h^j = \mathsf{H}_s(m^j)$ where m^i and m^j are respectively the ith and the jth tag-generation query.
Since this implies a collision, the probability that this happens is bounded easily by ϵ_{CR}.

F_2: Event F_1 does not happen and $h^i = \tilde{h}^j$ for a previous verification, that is $\tilde{h}^j$ is computed during the jth verification query, with $j < i$.
Since this implies having found a pre-image for a random target, the probability that this happens is bounded easily by ϵ_{roPR}.

F_3: Event F_1 and F_2 do not happen, thus $\mathsf{f}^{*,-1}(\tau^i)$ has never been computed before.
Since f^* is a random permutation, the probability that $\mathsf{f}^*(h^i) = \tau^i$ can be bounded by $\frac{1}{2^n - q_E - i + 1}$ (where $q_E + i - 1$ is the maximal number of previous evaluation of f^*).

□

5.5.3 HTBC: a BBB variant

The main problem of HBC2 is that there is a birthday-bound, due to the collision resistance of H. Thus, it may not achieve security beyond $n/2$-bits (thus, it achieves security up to half of the bit size of the blocks

of the BC).
Although in many case it is enough, we can have a better result: HTBC using a TBC instead of a BC achieves security up to the bit size of the blocks of the BC.
We obtain this doubling the size of the output the hash function. Thus, it doubles the birthday-bound for the collision-resistance (from $\frac{q(q+1)}{2^{n+1}}$, implying a security of $\frac{n}{2}$-bits to $\frac{q(q+1)}{2^{2n+1}}$),thus, now, the ϵ_{CR}-term of the bound implies a security of n-bits, which is equal to the bit size of the blocks. In this way, we have obtained a beyond birthday (BBB) security with respect to the bit size of the blocks of the TBC.

Considering a hash function $\mathsf{H} : \mathcal{KH} \times \{0,1\}^{2n} \to \mathcal{B}_1 \times \mathcal{B}_2$ and a strongly protected tweakable block cipher $\mathsf{F}^* : \mathcal{K} \times \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$, with $\mathcal{HM} = \{0,1\}^*$, simply using half of the digest as a tweak, we can build a MAC, HTBC, which is suf-L2-in the unbounded model, as described in Alg. 9 and Fig. 5.7. Note that for this MAC, $\mathcal{ME} = \{0,1\}^*$ and $\mathcal{TAG} = \{0,1\}^n$.

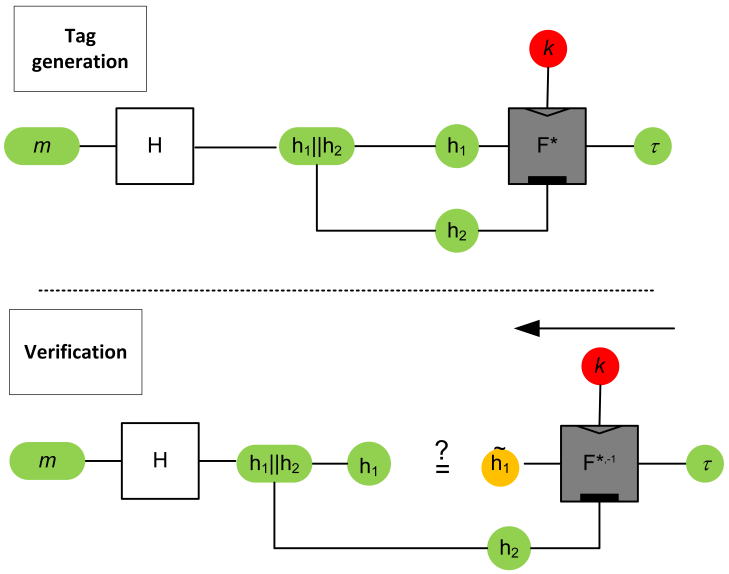


Figure 5.7: HTBC leakage-resilient MAC.

5.5.4 Security of HTBC

As usual, before proving the security of HTBC, we have to describe its leakage in the unbounded model:

Algorithm 9 HTBC based on the Digest and Tag parts of TEDT - Full description.

- Gen:
  - $k \xleftarrow{\$} \mathcal{K}^*$
  - $s \xleftarrow{\$} \mathcal{KH}$ (s is a public parameter)
- $\mathsf{MAC}_k(m)$:
  - $h_1 \| h_2 = \mathsf{H}_s(m)$ // digest
  - $\tau = \mathsf{F}^*_k(h_2, h_1)$ // tag
  - Return τ
- $\mathsf{Vrfy}_k(m, \tau)$:
  - $h_1 \| h_2 = \mathsf{H}_s(m)$
  - $\tilde{h_1} = \mathsf{F}^{*,-1}_k(h_2, \tau)$
  - If $h_1 == \tilde{h_1}$ Return $\top$, Else Return $\bot$.

Leakage: $\mathsf{L}_M(m; k) := \emptyset$ since h can be computed by anyone and τ is the output.
$\mathsf{L}_V(m, \tau; k) := \tilde{h_1}$ since all others values can be computed from anyone.

Now, to state the security of HTBC, we have to define a variant of the range-oriented pre-image resistance (Def. 5). In fact, we cannot apply straightforwardly in the proof the range-oriented pre-image resistance since the first half of the target $\tilde{h_1}$ is picked uniformly at random after the adversary has chosen the second half of the target h_2 as he wishes. This, is due to the fact that h_2 is chosen by the adversary, since he chooses the message m during verification queries (remember that in verification query $h = h_1 \| h_2 = \mathsf{H}_s(m)$).
Thus, we modify the definition of range-oriented pre-image resistance to obtain the following:

Definition 28 (roPR')**.** *A (t, ϵ)-range-oriented pre-image resistant hash function with half of the input chosen by the adversary $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \mathcal{T}_1 \times \mathcal{T}_2$ is a function such that, for every t-bounded adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$, the probability that $\mathsf{A}_2(s, y_1, \mathsf{st})$ outputs $m \in \mathcal{HM}$ s.t $\mathsf{H}_s(m) = y_1 \| y_2$, with $(y_2, \mathsf{st}) \leftarrow \mathsf{A}_1$, is bounded by ϵ, where $s \xleftarrow{\$} \mathcal{KH}$, $y_1 \xleftarrow{\$} \mathcal{T}_1$ are picked uniformly at random, that is:*

$$\Pr[\mathsf{A}_2(s, y_1, \mathsf{st}) \Rightarrow m \ s.t. \ \mathsf{H}_s(m) = y_1 \| y_2 \tag{5.1}$$

$$s.t. \ s \xleftarrow{\$} \mathcal{KH}, \ y_1 \xleftarrow{\$} \mathcal{T}_1, \ (y_2, \mathsf{st}) \leftarrow \mathsf{A}_1(s)] \leq \epsilon \tag{5.2}$$

We think that this definition is meaningful since we think that for a good hash function, it should be difficult to find a pre-image even if half of the target is chosen by the adversary, while the remaining half is picked uniformly at random.

This allows us to

Theorem 6. *Let* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \mathcal{B}_1 \times \mathcal{B}_2$ *be a* $(t_2, \epsilon_{\mathsf{CR}})$*-collision resistant and* $(t_2, \epsilon_{\mathsf{roPR}'})$*-range-oriented pre-image resistant hash function with half of the input chosen by the adversary. Let* $\mathsf{F}^* : \mathcal{K}^* \times \mathcal{TW}^* \times \mathcal{B}^* \to \mathcal{B}^*$ *be a* $(q_M + q_V + 1, t_1, \epsilon_{\mathsf{sTPRP}})$*-strong-pseudorandom tweakable permutation with a strongly protected implementation. Let* $\mathcal{B}_1 = \mathcal{B}^*$ *and* $\mathcal{B}_2 = \mathcal{TW}$*. Let* $\mathcal{HM} = \{0,1\}^*$ *and* $\mathcal{B}^* = \mathcal{TW} = \mathcal{TAG} = \{0,1\}^n$*.*
Then, HTBC *is* (q_M, q_V, t, ϵ)-suf-L2*-secure in the unbounded leakage model with*

$$\epsilon_{\mathsf{sTPRP}} + \epsilon_{\mathsf{CR}} + q_V \epsilon_{\mathsf{roPR}'} + (q_V + 1)2^{-n} + \frac{q(q-1)}{2^{n+1}} - \frac{q_M(q_M-1)}{2^{n+1}}$$

with $q = q_M + q_V + 1$*,* $t + t_{\mathsf{ch}(1,\mathcal{KH})} + qt_{\mathsf{H}} \leq t_1$*,* $t + qt_{\mathsf{H}} + t_{\mathsf{f}^*(q)} \leq t_2$ *and* $t + qt_{\mathsf{H}} + t_{\mathsf{f}^*(q-1)} \leq t_3$*.*

Note that every term of the bound is tight, having a matching attack, except the term $\frac{q(q+1)}{2^{n+1}}$, which is not beyond birthday. In fact, we have a matching attack only up to $\frac{q_V+1}{2^{n+1}}$ (merely asking the verification of the same message with different tags). To have a matching attack up $\frac{q(q+1)}{2^{n+1}}$ we need to be able to find $q_E + q_V + 1$ partial hash collision on h_2 (that is, we force to use in every query the same tweak h_2, and the messages used in tag-generation queries and the message used in verification queries (which is always the same) are all different).

Idea of the proof. The proof is very similar to the proof of the security of HBC2. Simply, it is necessary to treat the tweak correctly. Here we present only a sketch of the proof. The full proof may be found in App. B.4.
Note that the actual flow of the proof is different; instead, our sketch aims to give only the main ideas.

Sketch. First, we replace the sTPRP $\mathsf{F}^*_k(\cdot,\cdot)$ with a random tweakable permutation f^* with the same signature.
Then, we want to prove that every verification query (m^i, τ^i) is invalid. We may have two different events with respect to the value $h^i = \mathsf{H}_s(m^i)$ obtained during the ith verification query:

F_1: $h^i = h^j$ with $h^i = \mathsf{H}_s(m^i)$ and $h^j = \mathsf{H}_s(m^j)$ where m^i and m^j are respectively the ith and the jth tag-generation query.
Since this implies a collision, the probability that this happens is bounded easily by ϵ_{CR}.

F_2: Event F_1 does not happen and the couple $(\tilde{h_1}^i, h_2^i)$ has already been obtained in a previous verification, that is $\tilde{h_1}^j$ is computed during the jth verification query (m^j, τ^j) with $j < i$, and $h^j = h_1^j \| h_2^j = \mathsf{H}_s(m^j)$ is such that
$h_2^j = h_2^i$.
Since this implies having found a half pre-image for a random target $\tilde{h_1}^j$, after having chosen h_2^j the probability that this happens is bounded easily by $\epsilon_{\mathsf{roPR}'}$.

F_3: Event F_1 and F_2 do not happen, thus $\mathsf{f}^{*,-1}(h_2^i, \tau^i)$ has never been computed before.
Since f^* is a random tweakable permutation, the probability that $\mathsf{f}^{*,-1}(h_2^i, \tau^i) = h_1^i$ can be bounded by $\frac{1}{2^n - \#\mathsf{Eval}(h_2^i)}$ with $\#\mathsf{Eval}(h_2^i)$ is the number of times we have already lazy sampled $\mathsf{f}^*(h_2^i, \cdot)$ (that is, f^* with that given tweak) which is easily bounded by $\frac{1}{2^n - q_E + i - 1}$ (where $q_E + i - 1$ is the maximal number of previous evaluation of f^*).

□

Other solutions. Barwell et al. [10] have proposed a solution based on pairing. The security of this solution has been proved in the generic group model, for more details, see Chap. 5.10.

5.6 DTE2 - a solution for AE

This section is mainly inspired by the ToSC paper [27].

It presents a CIML2-secure scheme DTE2 obtained from DTE′ replacing the TPRF with a sTPRP and using its inverse in decryption. Simply, we replace HBC with HBC2.

DTE2 It turns out that using the inverse of the TBC in verification provides a powerful ingredient to avoid degrading AE-security when Dec leaks during checking ciphertext validity. The main problem was that in DTE′ the adversary was able to retrieve, via decryption leakage, the right tag $\tilde{\tau}$ for a given couple randomness-message (r, m). In fact, $\tilde{\tau} = \mathsf{F}_k^{*,0}(h)$ with $h = \mathsf{H}_s(r, m)$.

But, being able to use the inverse of the TBC, it is not necessary to recompute the good tag to have a correct and tidy[7] decryption algorithm. In DTE2 instead of computing $\tilde{\tau}$ and comparing it with the tag provided by the adversary, we compute $\tilde{h} = \mathsf{F}_k^{*,0,-1}(\tau)$ and compare it with $h = \mathsf{H}_s(r\|m)$.
That is, the decryption algorithm instead of saying *"your tag τ is wrong because the correct tag is $\tilde{\tau}$" and leaking $\tilde{\tau}$* , tells *" your tag τ is wrong because τ is the good tag for this hash value $\tilde{h}$ which is different from your digest h" and leaking $\tilde{h}$*. This change of approach is highlighted in Fig. 5.9.
Now, if it is difficult to find a pre-image of this hash value $\tilde{h}$ the scheme may be CIML2. This modification *only* in the decryption algorithm gives birth to DTE2 (see details in Alg. 10 and Fig. 5.8, the changes from DTE′ are highlighted in gray.)

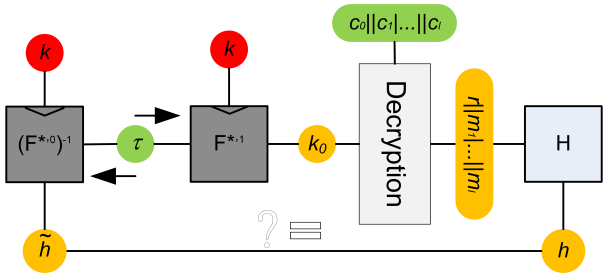


Figure 5.8: DTE2 - decryption. The encryption is equal to DTE', see Fig. 5.5.

Security of DTE2. DTE2 achieves many security goals:

- it is a secure AE-scheme
- it is misuse-resistant
- it is CIML2-secure
- it provides confidentiality in the presence of leakage in encryption[8].

All the other results being a sideline to the scope, here we only prove the CIML2-security. For the other proofs, see the ToSC17 paper [27].

5.6.1 The CIML2-security of DTE2

Before proving the CIML2 security of DTE2, we have to consider what means misuse and what is the leakage in the unbounded model:

[7] An AE-scheme is *tidy* if the decryption algorithm accepts as valid only ciphertexts which are the possible outputs of the encryption algorithm.
[8] The leakage model used in this proof is 2-simulatability

Algorithm 10 DTE2 - The changes from DTE are highlighted.

- Gen:
  - $k \xleftarrow{\$} \mathcal{K}$
  - $s \xleftarrow{\$} \mathcal{KH}$
  - $p_A, p_B \xleftarrow{\$} \mathcal{B}$ (s, p_A, p_B are public parameters)

- $\mathsf{Enc}_k(m)$:
  - Parse $m = (m_1, m_2, \ldots, m_l)$ in n-bit blocks
  - $r \xleftarrow{\$} \{0,1\}^n$
  - $\tau \leftarrow \mathsf{Tag}_k^0(r, m)$:
    - $h = \mathsf{H}_s(r \| m)$ // digest
    - $\tau = \mathsf{F}_k^{*,0}(h)$ // tag
  - $C \leftarrow \mathsf{PSVEnc}_k^1(\tau, (r, m))$: // ...and encrypt
    - $k_0 = \mathsf{F}_k^{*,1}(\tau)$
    - $y_0 = \mathsf{F}_{k_0}(p_B)$
    - $c_0 = y_0 \oplus r$
    - For $i = 1, .., l$
      - $k_i = \mathsf{F}_{k_{i-1}}(p_A)$
      - $y_i = \mathsf{F}_{k_i}(p_B)$
      - $c_i = \pi_{|m_i|}(y_i) \oplus m_i$
    - $C = (c_0, c_1, \ldots, c_l)$
  - Return $c = (\tau, C)$
- $\mathsf{Dec}_k(c)$:
  - Parse $c = (\tau, C)$
  - Parse $C = (c_0, c_1, c_2, \ldots, c_\ell)$ in n-bit blocks
  - $(r, m) \leftarrow \mathsf{PSVDec}_k^1(\tau, C)$:
    - $k_0 = \mathsf{F}_k^{*,1}(\tau)$
    - $y_0 = \mathsf{F}_{k_0}(p_B)$
    - $r = y_0 \oplus c_0$
    - For $i = 1, .., l$
      - $k_i = \mathsf{F}_{k_{i-1}}(p_A)$
      - $y_i = \mathsf{F}_{k_i}(p_B)$
      - $m_i = \pi_{|c_i|}(y_i \oplus c_i$
    - $(r, m) = (r, (m_1, ..., m_l))$
  - $h = \mathsf{H}_s(r \| m)$
  - $\boxed{\mathsf{TagVrfy}_k^0(\tau, (r, m)):}$
    - $\boxed{\tilde{h} = \mathsf{F}_k^{*,-1,0}(\tau)}$ // check tag
    - If $h = \tilde{h}$ Return m; Else Return $\perp$.

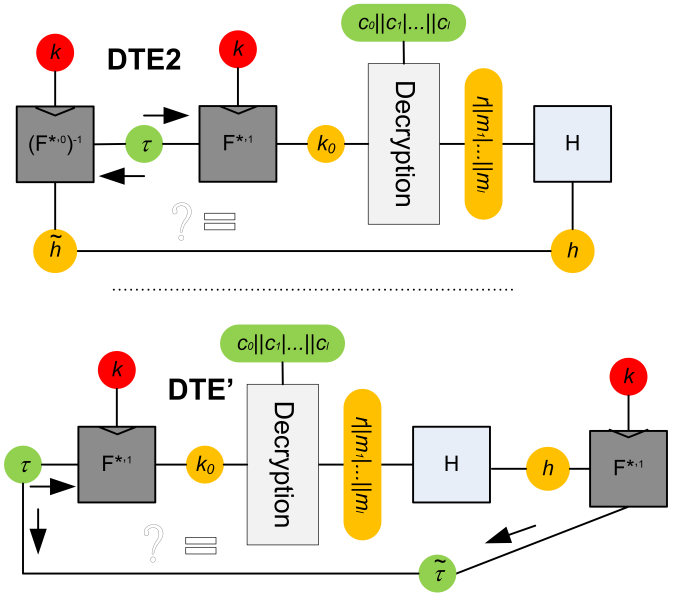


Figure 5.9: The difference between the decryption of DTE2 from that of DTE′.

Misuse: As for DTE we assume that the adversary during encryption queries provides the randomness r alongside with the message.

Leakage: As for DTE, the encryption leakage is $\mathsf{L}_E(r, m; k) := k_0$. Instead for decryption leakage, we set $\mathsf{L}_D(c; k) := (k_0, \tilde{h})$ because from these values the adversary has all the information he wants. In fact, he can recompute r and m (and thus $h = \mathsf{H}_s(r\|m)$) from k_0 because, from k_0 all the $k_1, ..., k_l$ and $y_0, ..., y_l$ may be recomputed.

Theorem 7. *Let* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \mathcal{B}'$ *be a* $(t_2, \epsilon_{\mathsf{CR}})$*-collision resistant and* $(t_2, \epsilon_{\mathsf{roPR}})$*-range-oriented preimage resistant hash function. Let* $\mathsf{F}^* : \mathcal{K}^* \times \mathcal{B}^* \times \mathcal{TW} \to \mathcal{B}^*$ *be a* $(2q, t_1, \epsilon_{\mathsf{sTPRP}})$*-strong tweakable pseudorandom permutation with a strongly protected implementation. Let* $\mathsf{F} : \mathcal{K} \times \mathcal{B} \to \mathcal{B}$*. Let* $\mathcal{HM} = \{0,1\}^*$*,* $\mathcal{TW} = \{0,1\}$ *and* $\mathcal{B}' = \mathcal{B}^* = \mathcal{K} = \mathcal{B} = \{0,1\}^n$*. Then,* DTE2 *which encrypts at most* L*-block messages is* (q_E, q_D, t, ϵ)*-*CIML2*-secure in the unbounded leakage model with*

$$\epsilon \leq \epsilon_{\mathsf{sTPRP}} + \epsilon_{\mathsf{CR}} + q_D\epsilon_{\mathsf{roPR}} + (q_D + 1)2^{-n} + \frac{q(q-1)}{2^{n+1}} - \frac{q_E(q_E-1)}{2^{n+1}}$$

where $q = q_E + q_D + 1$*,* $t + t_{\mathsf{ch}(1,\mathcal{KH})} + t_{\mathsf{chn}(2,\mathcal{B})} + q(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}) \leq t_1$*,* $t + t_{\mathsf{chn}(2,\mathcal{B})} + q(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}) + t_{\mathsf{f}^*(2q)} \leq t_2$ *and* $t + t_{\mathsf{chn}(2,\mathcal{B})} + q(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}) + t_{\mathsf{f}^*(2q-1)} \leq t_3$*.*

Note that, as for DTE, we ask nothing about the security of F. We only need that it has the right syntax.

The terms of the bound.

- $\epsilon_{\mathsf{sTPRP}}$ since we are using a sTPRP and not a TPRP
- ϵ_{CR} since having found a collision for the hash function, it is easy to make a forgery
- $q_D\epsilon_{\mathsf{roPR}}$ since the adversary has q_D $\tilde{h}$ (which are randomly picked). If he finds a pre-image for one of them, he can make a forgery.
- $\frac{q(q-1)}{2^{n+1}} - \frac{q_E(q_E-1)}{2^{n+1}}$ because $\tilde{h}$ are not picked uniformly at random (as required for range-oriented pre-image resistance), but using a permutation (so, there are some values which cannot be picked).

Here we present only a sketch of the proof. The full proof may be found in App. B.5.
Note that the actual flow of the proof is different since we may obtain better bounds (in particular we may avoid to have $q_V + 1$ as a factor of the whole bound); instead, our sketch aims to give only the main ideas of the proof.

Sketch. First, we replace the sTPRP $\mathsf{F}^*_k(\cdot,\cdot)$ with a random tweakable permutation f^* with the same signature.
Then, we want to prove that every decryption query $c^i(\tau^i, C^i)$ is invalid. We may have three different events with respect to the value $h^i = \mathsf{H}_s(r^i\|m^i)$ obtained during the ith decryption query:

F_1: $h^i = h^j$ with $h^i = \mathsf{H}_s(r^i||m^i)$ and $h^j = \mathsf{H}_s(r^j||m^j)$ where (r^i, m^i) and (r^j, m^j) are respectively the ith and the jth encryption query (the jth encryption query has happened before the ith decryption query)

F_2: Event F_1 does not happen and $h^i = \tilde{h}^j$ where $\tilde{h}^j$ is the check hash value obtained during the jth decryption query ($j < i$)

F_3: Neither event F_1 nor event F_2 happens.

Let E_i be the event that the ith decryption query is fresh and valid.
We bound now $\Pr[E_i \cap F_{i'}]$ for $i = 1, ..., q_D + 1$ and $i' = 1, 2, 3$:

$E_i \cap F_1$: it means that we have found a collision for the hash function H_s

$E_i \cap F_2$: now $\mathsf{f}^{*,0,-1}(\tau^i)$ has already been computed in the history of the CIML2-game. It means that a pre-image for the hash function for the value $\tilde{h}^j$. Since these values are picked uniformly at random (they are the outputs of $\mathsf{f}^{*,-1}$), the probability that this happens is bounded by ϵ_{roPR}. Moreover, there at most $i - 1 \leq q_D$ possible different targets.

$E_i \cap F_3$: since $\mathsf{f}^{*,0,-1}(\tau^i)$ has never been computed and f^* is a random tweakable permutation, the probability that $\mathsf{f}^{*,0,-1}(\tau^i) = h^i$ is exactly $\frac{1}{2^{-n}-q} \leq \frac{1}{2^{-n}-q_E-q_D}$ where q is the number of queries already done to $\mathsf{f}^{*,0}$ (since f is a random permutation its injectivity gives

some information).

□

5.7 EDT, Encrypt-Digest-then-Tag, a CIML2-secure AE-scheme

This section is inspired by the ToSC paper [27].

DTE2 solves the problem of authenticity with leakage in both encryption and decryption with additional nice properties: confidentiality with misuse and leakage in encryption and misuse resistance. On the other hand, when decryption leaks it is easy to mount a DPA which is able to retrieve the message encrypted, as we show here:

DPA in decryption. The idea of this attack is to mount a DPA on the XORs used in decryption. It exploits the fact that the whole psuedorandom stream used to encrypt a message depends only on τ. Suppose that an adversary has received the ciphertext $c^* = (\tau^*, C^*)$ and he wants to recover the message. He may proceed as follow:

- Ask the decryption of $C^1, ..., C^q$ with $c^i = (\tau^*, C^i)$ where C^i is random and $|C^i| = |C^*|$.
- During each decryption target with a DPA the XORs $m^i_j = y^i_j \oplus c^i_j$ and retrieve the y^j_i which are equal to $\mathsf{F}_{k^i_j}(p_B)$. Note, that $\forall i, i'$ and $\forall j$, $k^i_j = k^{i'}_j$, thus $y^i_j = y^{i'}_j$ and we denote them simply with y_j (omitting the ^{i}).
- After having recovered all the y_j, the message encrypted by C^* is m^* where $m^*_j = c^*_j \oplus y_j$

This situation may happen, for example, for secure boot loading or bitstream decryption, when the user is only supposed to be able to run the code without having access to the sources (e.g., multimedia content or video games).

Thus, it may be interesting to provide confidentiality when the adversary also has access to leakage in decryption (denoted with CCAmL. This has been formalized by the notion of eavesdropper security with leakage in decryption (see the ToSC17 paper [27]).

To achieve this security, keeping AE-security, CIML2-security and confidentiality with leakage and using only two calls to a highly-protected primitive we have to relinquish the misuse-resistance (for a detailed study

of this problem, see Guo et al. [65]) since to protect against this attack there are two possible paths, for example:

- first, to verify the ciphertext before to decrypt it. The problem is that in many misuse-resistant AE schemes the tag depends on all the plaintext. Thus, we should decrypt before verifying the authenticity of the ciphertext, as it is done in DTE2. A possible solution is to rencrypt the ciphertext and then, to authentify the new ciphertext (but, this solution is less efficient).

- otherwise, we would need that if we change a bit of the ciphertext, the whole message obtained during the decryption, would be changed. This implies a double-pass decryption. If it is combined with a double pass encryption (needed for misuse-resistance), the scheme must be more complex, and, thus, less efficient.

DTE cannot fulfill this previous notion without relying on a strong assumption since even if the tag is not correct, we decrypt. In particular, note that the first ephemeral key computed in decryption, k_0, which is part of the output of the leakage function, $\mathsf{L}_D(\cdot,\cdot;\cdot)$, is computed only from the tag. Thus, if we want to have information about the message encrypted by the ciphertext $c^* = (C^*, \tau^*)$ we have only to ask the decryption of a ciphertext $c^1 = (C', \tau^*)$ for any PSV-ciphertext C' to obtain the right ephemeral key k_0^*, from which it is possible to compute the PSV-decryption of C^*.
Note that although c^1 is not a valid ciphertext, k_0^1 is computed during decryption and $k_0^1 = k_0^*$. Thus, we would like to have a scheme which first checks the authenticity of the ciphertext, then, starts decrypting.
To obtain this security we propose EDT, Encrypt-Digest-and-Tag:

- A nonce r is provided along with the message m[9]
- $C \leftarrow \mathsf{PSVEnc}_k(r, m)$
  that is, we encrypt using PSV, where the first ephemeral key, $k_1 = \mathsf{F}_k^{*,0}(r)$.
- $h \leftarrow \mathsf{H}_s(r \| C)$
  that is, the nonce r and the PSV-ciphertext C are digested via an hash function
- $\tau = \mathsf{F}_k^{*,1}(h)$
  that is, we authenticate the hash
- the output is $c = (C, \tau)$[10].

[9] The nonce r may also be picked as a randomness, thus $r \xleftarrow{\$} \{0,1\}^n$

[10] If r is picked by the algorithm, r must be part of the output. Instead, if it is a nonce, it is already known.

For decryption, we use again the inverse trick to have CIML2. First the digest is recomputed via $\tilde{h} = \mathsf{F}_k^{*,1,-1}(\tau)$ and it is checked whether it matches the actual digest $h = \mathsf{H}_s(r\|C)$. If it is not the case, the ciphertext is deemed invalid; otherwise, the usual PSV-decryption takes place. Thus, invalid ciphertexts never trigger the PSVDec.

Note that EDT is a secure nonce-based encryption scheme which means that as long as r is not repeated during encryption queries, we keep the AE-security (and confidentiality with leakage).
On the other hand, we relinquish the misuse-resistance since PSV is not a misuse-resistant encryption scheme. In fact, the first ephemeral key is computed directly from the nonce r. Thus, the PSV ciphertext blocks c_i does not depend on all the message, which is a necessary condition to reach misuse-resistance.

For more details on EDT, see Alg. 11 and Fig. 5.10.

Security of EDT. EDT achieves many security goals:

- it is a secure AE-scheme
- it provides nonce misuse-resilience (see App. A.2)
- it is CIML2-secure
- it provides confidentiality in the presence of leakage in encryption[11].
- it has eavesdropper security with differential leakage (EavDL) [a notion introduced in the ToSC17 paper [27]]

All the other results being a sideline to the scope, here we prove only the CIML2-security. For the other proofs, see the ToSC17 paper [27].

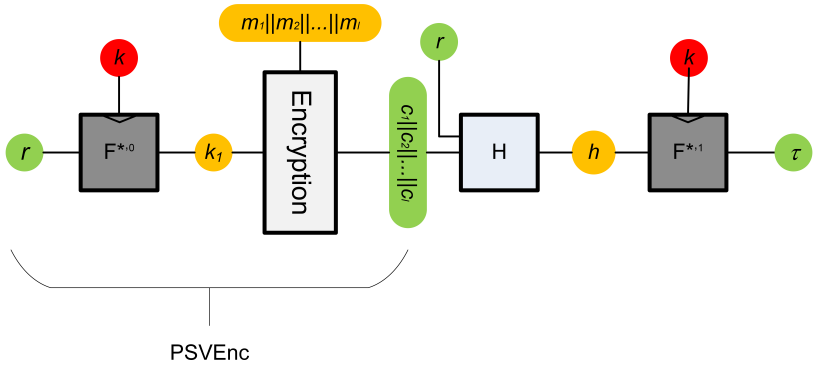


Figure 5.10: EDTencryption.

[11] The leakage model used in this proof is 2-simulatability

Algorithm 11 EDT.

- Gen:
  - $k \xleftarrow{\$} \mathcal{K}$
  - $s \xleftarrow{\$} \mathcal{KH}$
  - $p_A, p_B \xleftarrow{\$} \mathcal{B}$ (s, p_A, p_B are public parameters)

- $\mathsf{Enc}_k(r, m)$:
  - Parse $m = (m_1, m_2, \ldots, m_l)$
  - $C \leftarrow \mathsf{PSVEnc}^0_k(r, m)$ // encrypt
    - $k_1 = \mathsf{F}^{*,0}_k(r)$
    - $y_1 = \mathsf{F}_{k_1}(p_B)$
    - $c_1 = \pi_{|m_1|}(y_1) \oplus m_1$
    - For $i = 2, .., l$
      - $k_i = \mathsf{F}_{k_{i-1}}(p_A)$
      - $y_i = \mathsf{F}_{k_i}(p_B)$
      - $c_i = \pi_{|m_i|}(y_i) \oplus m_i$
    - $C = (c_1, c_2, \ldots, c_l)$
  - $\tau \leftarrow \mathsf{Tag}^1_k(C)$:
    - $h = \mathsf{H}_s(r \| C)$ // digest
    - $\tau = \mathsf{F}^{*,1}_k(h)$ // ... and tag
  - Return $c = (C, \tau)$
- $\mathsf{Dec}_k(r, c)$:
  - Parse $c = (C, \tau)$
  - Parse $C = (c_1, c_2, \ldots, c_\ell)$
  - $\mathsf{TagVrfy}^1_k(C, \tau)$:
    - $h = \mathsf{H}_s(r \| C)$
    - $\tilde{h} = \mathsf{F}^{*,1,-1}_k(\tau)$ // check tag
    - If $h \neq \tilde{h}$ Return $\perp$.
  - $m \leftarrow \mathsf{PSVDec}^0_k(r, C)$:
    - $k_1 = \mathsf{F}^{*,0}_k(r)$
    - $y_1 = \mathsf{F}_{k_1}(p_B)$
    - $m_1 = y_1 \oplus c_1$
    - For $i = 2, .., l$
      - $k_i = \mathsf{F}_{k_{i-1}}(p_A)$
      - $y_i = \mathsf{F}_{k_i}(p_B)$
      - $m_i = \pi_{|c_i|}(y_i) \oplus c_i$
    - $m = (m_1, ..., m_l)$
  - Return m

5.7.1 The CIML2-security of EDT

Before proving the CIML2 security of EDT, we have to consider what means misuse and what is the leakage in the unbounded model:

Misuse: For EDT we assume that the adversary during encryption queries provides the randomness r alongside with the message.

Leakage: As for DTE, the encryption leakage is $\mathsf{L}_E(r, m; k) := k_1$, that is, the first ephemeral key.
Instead for decryption leakage, we set $\mathsf{L}_D(c; k) := \tilde{h}$ for invalid decryption queries; k_1 for valid decryption queries (since $\tilde{h} = h$ for valid decryption queries). In fact, from these values, the adversary has all the information he wants.

Theorem 8. *Let* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \{0,1\}^n$ *be a* $(t_1, \epsilon_{\mathsf{CR}})$*-collision resistant and* $(t_1, \epsilon_{\mathsf{roPR}})$*-range-oriented preimage resistant hash function. Let* $\mathsf{F}^* : \mathcal{K}^* \times \{0,1\}^n \times \mathcal{TW} \to \{0,1\}^n$ *be a* $(2q-1, t_1, \epsilon_{\mathsf{sTPRP}})$*-strong tweakable pseudorandom permutation with a strongly-protected implementations. Let* $\mathsf{F} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$. *Let* $\mathcal{HM} = \{0,1\}^*$ *and* $\mathcal{TW} = \{0,1\}$.
Then, EDT *which encrypts at most* L*-block messages is* (q_E, q_D, t, ϵ)*-*CIML2*-secure in the unbounded leakage model with*

$$\epsilon_{\mathsf{sTPRP}} + \epsilon_{\mathsf{CR}} + q_D\epsilon_{\mathsf{roPR}} + (q_D+1)2^{-n} + \frac{q(q-1)}{2^{n+1}} - \frac{q_E(q_E-1)}{2^{n+1}}$$

with $q = q_E + q_D + 1$, $t + t_{\mathsf{ch}(1,\mathcal{KH})} + t_{\mathsf{chn}(2,\mathcal{B})} + qt_{\mathsf{H}} + (q-1)(2L-1)t_{\mathsf{F}}) \le t_1$,
$t + t_{\mathsf{chn}(2,\mathcal{B})} + qt_{\mathsf{H}} + (q-1)(2L-1)t_{\mathsf{F}} + t_{\mathsf{f}^*(2q-1))} \le t_2$
and $t + t_{\mathsf{chn}(2,\mathcal{B})} + qt_{\mathsf{H}} + (q-1)(2L-1)t_{\mathsf{F}} + t_{\mathsf{f}^*(2q-2)} \le t_3$.

Note that, as for DTE and DTE2, we ask nothing about the security of F. We only need that it has the right syntax.

Here we present only a sketch of the proof. The full proof may be found in App. B.6.
Note that the actual flow of the proof is different; instead, our sketch aims to give only the main ideas of the proof.

Difference with DTE2 proof. The proof is inspired by that of DTE2. We have only to consider that now $h = \mathsf{H}_s(r\|C)$ (and no more $h = \mathsf{H}_s(r\|m)$ and adjust everything consistently).

Sketch. First, we replace the sTPRP $\mathsf{F}^*_k(\cdot,\cdot)$ with a random tweakable permutation f^* with the same signature.
Then, we want to prove that every decryption query $c^i = (C^i, \tau^i)$ is

invalid. We may have three different events with respect to the value $h^i = \mathsf{H}_s(r^i \| C^i)$ obtained during the ith decryption query:

- F_1: $h^i = h^j$ with $h^i = \mathsf{H}_s(r^i \| C^i)$ and $h^j = \mathsf{H}_s(r^j \| C^j)$ where C^i and C^j are respectively the outputs of PSVEnc during the ith and the jth encryption query (on input (r^j, m^j))
- F_2: Event F_1 does not happen and $h^i = \tilde{h}^j$ where $\tilde{h}^j$ is the check hash value obtained during the jth decryption query ($j < i$)
- F_3: Neither event F_1 nor event F_2 happens.

Let E_i be the event that the ith decryption query is fresh and valid.
We bound now $\Pr[E_i \cap F_{i'}]$ for $i = 1, ..., q_D + 1$ and $i' = 1, 2, 3$:

- $E_i \cap F_1$: it means that we have found a collision for the hash function H_s
- $E_i \cap F_2$: now $\mathsf{f}^{*,1,-1}(\tau^i)$ has already been computed in the history of the CIML2-game. It means that we have found a pre-image for the hash function for the value $\tilde{h}^j$. Since these values are picked uniformly at random (they are the outputs of f^*), the probability that this happens is bounded by ϵ_{roPR}. Moreover, there at most $i - 1 \leq q_D$ possible different targets.
- $E_i \cap F_3$: since $\mathsf{f}^{*,1,-1}(\tau^i)$ has never been computed and f^* is a random tweakable permutation, the probability that $\mathsf{f}^{*,1,-1}(\tau^i) = h^i$ is exactly $\frac{1}{2^{-n}-q} \leq \frac{1}{2^{-n}-q_E-q_D}$ where q is the number of queries already done to $\mathsf{f}^{*,0}$ (since f is a random permutation its injectivity gives some information).

□

FEMALE. If we want a scheme which has the advantages of EDT, with respect to DTE2 with, in addition, misuse-resistance, as DTE2, Guo et al. [65] have provided FEMALE. On the other hand, it is much more expensive, since instead of PSV, a double pass encryption scheme is used. Moreover, it uses our solution, the inversion of the TBC to have the CIML2-security.

5.8 CONCRETE, reducing the number of strongly protected TBC

In this section, we exhibit a CIML2-secure AE scheme using only one call to the strongly protected TBC. This scheme is CONCRETE, which stands for COmmit-eNCRypt-sEnd-The-kEy.
This section is based on the AFRICACRYPT19 paper [29] and its extended version on eprint [28].

The reason to do this is that the strongly protected building block is very expensive. Thus, if we want to have a more efficient scheme, the first solution is to reduce the number of times this building block is used. We want to explore what we can achieve with a single call to the strongly protected building block. The mode we present is more efficient, but at the price of shifting the nonce-based AE approach to a probabilistic AE approach.

Characteristic of previous designs. We observe that we can divide every of the previous CIML2 scheme in three parts

a Generation of the first ephemeral key (which is either called k_0 or k_1)
b Encryption,[12] starting from the first ephemeral key
c Authentication

Note that these steps must not be done in this order. For example DTE (see Sec. 5.6) and DTE2 are c-a-b, while EDT is a-b-c.
The calls to the strongly protected component are usually used in Step a and c, and the verification should be done in a clever way, not simply a basic recheck of the authentication part.

Design goals. The design and security constraints of the scheme are the following:

- AE secure in the black-box model (we can use either the nonce-based AE approach or the pAE)
- CIML2 secure in the unbounded model
- CPAL and CCAL secure with some hypothesis about the leakage
- only one call to the strongly protected implementation per execution
- the key k of the scheme used only as a key of the leak-free
- F changes keys as much as possible (*rekeying*), as in DTE and EDT

The hypothesis on k is because the key of the strongly-protected F^* is the only internal secret not leaked in the unbounded model.
To reduce the possibility of leakage, we should not use the PRF F with more than two different plaintexts for any key it uses in any security game. Thus, we would like to use a scheme based on rekeying, as PSV.

Design rationale of the Commit-Encrypt-Send-the-Key, CONCRETE scheme. We present here the ideas of the mode

[12] usually using PSV or one of its variant or a sponge, as done in other works not covered in this thesis.

CONCRETE (Fig. 5.11). In particular, how we treated each of the three different parts of a leakage-resilient AE scheme:

- **Derivation of the first ephemeral key** We pick a randomly k_0 as first ephemeral key. We use a round of the PRG of Standaert et al. [124] to obtain a *commit* of k_0 (called c_0), which consists in the encryption of o^n with k_0, and a new fresh key (k_1). That is, using the public constants p_A and p_B (two strings of n bits, with $p_A \neq p_B$) we obtain $k_1 = \mathsf{F}_{k_0}(p_A)$ and $c_0 = \mathsf{E}_{k_0}(p_B)$.
- **Encryption** We use here the strongly protected building block which is a TBC F^* with $\mathcal{TW} = \{0,1\}^n$. From k_1 we can use the PSV [111] (see Fig. 3.4) encryption algorithm to encrypt m (which is parsed in $m = (m_1, ..., m_l)$), wherw k_1 is the first ephemeral key, using the constants p_A and p_B, obtaining $c_1, ..., c_l$. We denote the algorithm in this part Enc.
- **Authentication** Since k_0 is picked uniformly at random, it must be recomputed by the decryption algorithm Dec from the ciphertext c. Thus, to send it, we hash the commitment c_0 and the output of the encryption part $c_1, ..., c_l$ obtaining $h = \mathsf{H}_s(c_0\|c_1\|...\|c_l)$ and we encrypt k_0 with the long term key k obtaining $c_{l+1} = \mathsf{F}^*_k(h, k_0)$. The ciphertext is $c := (c_0, c_1, ..., c_l, c_{l+1})$.
- **Decryption** First k_0 is retrieved, with $k_0 = \mathsf{F}^{*,-1}_k(h, c_{l+1})$ with $h = \mathsf{H}_s(c_0\|...\|c_l)$, then $\tilde{c_0} = \mathsf{F}_{k_0}(p_B)$ is computed. If $c_0 = \tilde{c_0}$, the ciphertext is deemed valid and decryption proceeds in the natural way; otherwise, the ciphertext is deemed invalid.
- **Ciphertext expansion** The ciphertext has an expansion of two blocks, that is, given $c \leftarrow \mathsf{Enc}_k(m)$, $|c| = |m| + 2n$. On the other hand, since the nonce should be sent anyway, if the nonce is a n-bit string, the size of ciphertext plus the nonce sent is the same for CONCRETE, EDT and DTE.
- **Cost** If the hash function is implemented with the Hirose construction [71], thus, its cost is $2c_\mathsf{F}$ for each block processed, then, the cost of CONCRETE to process l block message is $c_{\mathsf{F}^*} + 4(l+1)c_\mathsf{F}$, with c_{F^*} and c_F the cost, respectively, of one call to F^* and F.

A detailed description of the scheme can be found in Alg. 12. Note that $c_0, ..., c_l$ can be seen as $\mathsf{PSV}_{k_0}(0^n\|m)$.

Replacing PSV in the encryption part. It is possible, as discussed in the proofs, to replace PSV with other encryption scheme based on rekeying. The security constraints of such replacements are discussed after the proof in the eprint version [28].

Although CONCRETE is AE and CIML secure and it uses only one leak

Algorithm 12 The leakage resilient pAE-scheme $\mathsf{CONCRETE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ - Full description.

- Gen:
  - $k \xleftarrow{\$} \mathcal{K}^*$
  - $s \xleftarrow{\$} \mathcal{KH}$
  - Public parameters:
    - $p_A, p_B \in \{0,1\}^n,\ p_A \neq p_B$
    - s
- $\mathsf{Enc}_k(m)$:
  - Parse $m = (m_1, ..., m_l)$ with $|m_1| = ... = |m_{l-1}| = n$ and $|m_l| \leq n$
  - $k_0 := r \xleftarrow{\$} \mathcal{K}$
  - $c_0 = \mathsf{F}_{k_0}(p_B)$ // Commit
  - For $i = 1, ..., l$: // Encrypt
    - $k_i = \mathsf{F}_{k_{i-1}}(p_A)$
    - $y_i = \mathsf{F}_{k_i}(p_B)$
    - $c_i = \pi_{|m_i|}(y_i) \oplus m_i$
  - $h = \mathsf{H}_s(c_0 \| c_1 \| ... \| c_l)$
  - $c_{l+1} = \mathsf{F}_k^{*,h}(k_0)$ // Send the key
  - Return $c = (c_0, ..., c_l, c_{l+1})$
- $\mathsf{Dec}_k(c)$:
  - Parse $c = (c_0, ..., c_{l+1})$ with $|c_0| = ... = |c_{l-1}| = |c_{l+1}| = n$ and $|c_l| \leq n$
  - $h = \mathsf{H}_s = (c_0 \| ... \| c_{l+1})$
  - $k_0 = \mathsf{F}_k^{*,-1,h}(c_{l+1})$
  - $\tilde{c}_0 = \mathsf{F}_{k_0}(p_B)$
  - If $c_0 \neq \tilde{c}_0$
    - Return $\perp$
  - For $i = 1, ..., l$
    - $k_i = \mathsf{F}_{k_{i-1}}(p_A)$
    - $y_i = \mathsf{F}_{k_i}(p_B)$
    - $c_i = \pi_{|c_i|}(y_i) \oplus m_i$
  - Return $m = (m_1, ..., m_l)$

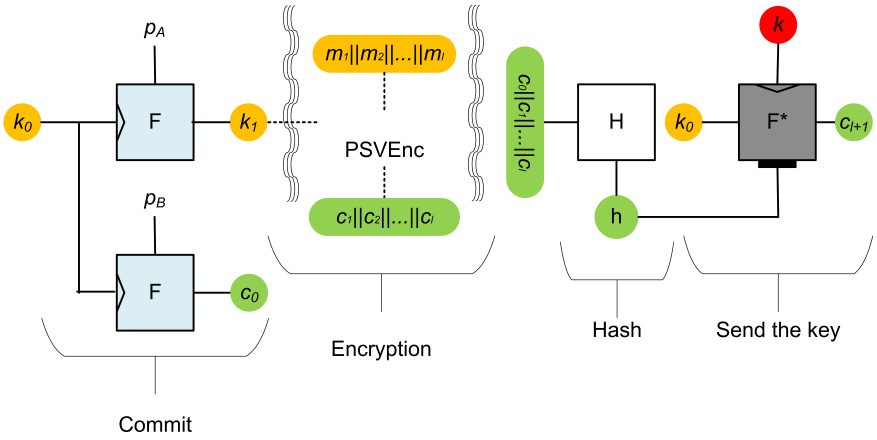


Figure 5.11: The leakage-resilient pAE-scheme CONCRETE [29].

free call, it is neither nonce-misuse resistance (as DTE [24] and FEMALE [65]) nor secure beyond birthday and in the multi-user case (as TEDT [21]).

History of the design The main idea is avoiding to use a leak-free component to generate the first ephemeral key k_1, but picking it uniformly at random, see Fig. 5.12 a.
But this imposes to send the ephemeral key k_1 with the ciphertext, to allow the receiver to decrypt.
Next, we have the problem of sending k_1. To do this we use the sTPRP F^* to generate $c_{l+1} = \mathsf{F}^*_k(tw, k_1)$, because to send k_1 there is no other possibility than to use the master key k, see Fig. 5.12 b. We have to decide what to put as tweak tw. Since we want to have authenticity, c_{l+1} must depend on all the other blocks. This can be done using $tw = h' = \mathsf{H}_s(c_1\|...\|c_l)$, see Fig. 5.12 c. Unfortunately this solution gives no authenticity, since every ciphertext would be valid[13].
Thus, we add, in the ciphertext, a commitment c_0 of k_1, as $c_0 = \mathsf{F}_{k_1}(p_C)$ for a certain constant p_C (p_C must be different from p_B [and p_A], otherwise, the first block of plaintext would be leaked [or k_2]). Thus, $h = \mathsf{H}_s(c_0\|...\|c_l)$ (see Fig. 5.12 d). This scheme is CIML secure, when we recompute $\tilde{c_0}$ and check it.
But in this last scheme, k_1 is used three times with three different plaintexts as key of F. Thus, to avoid this, we pick randomly k_0, we compute its commit $c_0 = \mathsf{F}_{k_0}(p_B)$ and we do a rekeying to obtain $k_1 = \mathsf{F}_{k_0}(p_A)$

[13]It could be argued that such a scheme would be CCA-secure, since it could be proved that every decryption query made by an adversary, which is an not answer from a previous encryption query, would result in a random answer.

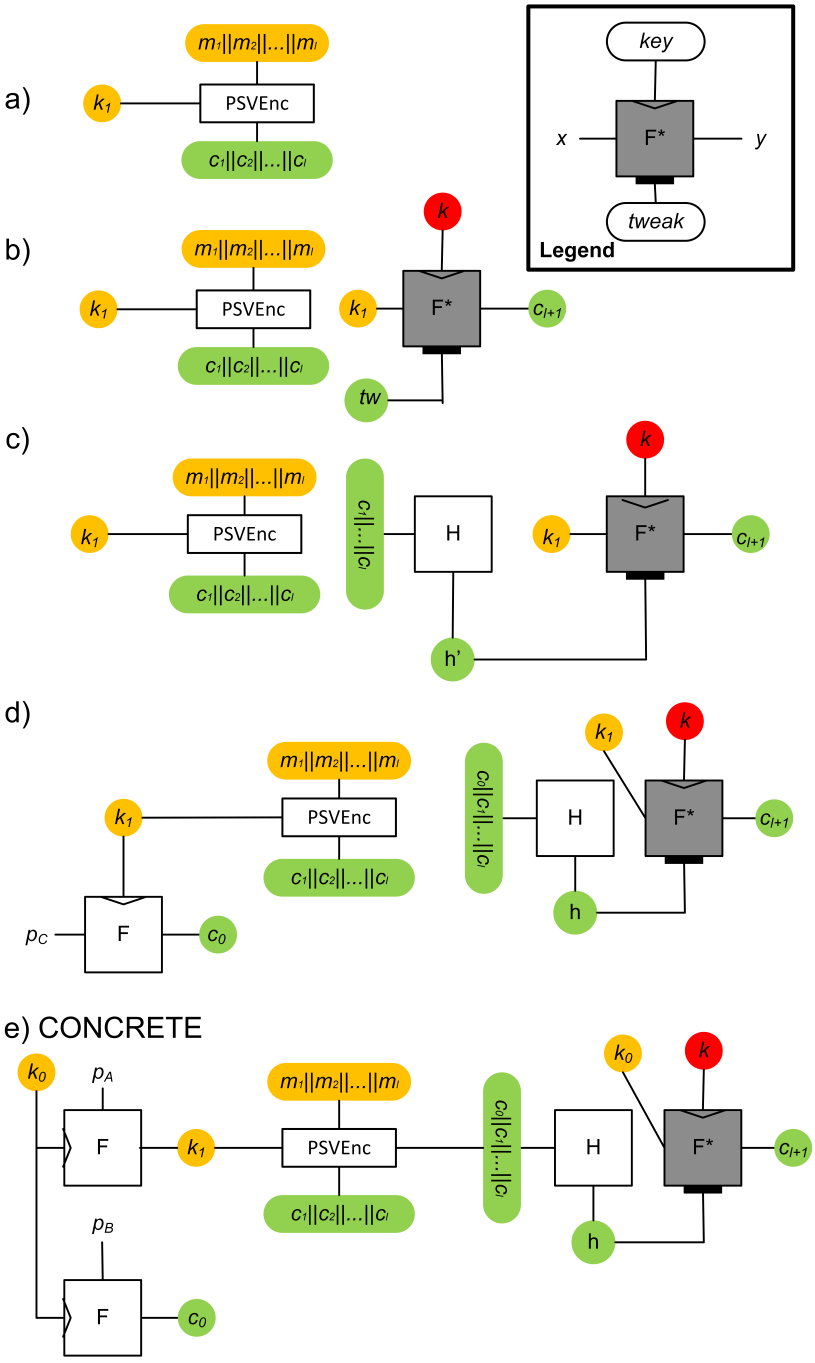


Figure 5.12: How we designe CONCRETE.

(see Fig. 5.12 e).

We note that our construction could benefit from implementations of the strongly protected component based on randomized countermeasures (such as masking, shuffling), in which case the PRG needed for both could be shared. However, this may not be systematic since the quality of the random numbers used in side-channel countermeasures may be weaker than for cryptographic keys.

Security of CONCRETE CONCRETE achieves these security goals:

- it is a secure pAE-scheme
- it is CIML2-secure
- it provides confidentiality in the presence of leakage in encryption[14].
- it is RUP-AE-secure, that is, it is secure when unverified plaintexts are released[15]

All the other results being a sideline to the scope, here we prove only the CIML2-security. For the other proofs, see the eprint paper [28].

5.8.1 The CIML2 security of CONCRETE

Before proving the CIML2 security for CONCRETE, we have, as usual, to define what means misuse and its leakage functions in the unbounded leakage model.

Misuse: we suppose that the adversary can control the PRG providing the randomness, thus, he provides r (thus, he chooses k_0)

Leakage in the unbounded model, when there is misuse, we can assume that there is no leakage since all the ephemeral keys and inner values (apart from the key of F^*) can be recomputed from the randomness r and the ciphertext c.
On the other hand, $\mathsf{L}_D(c;k) := k_0$, because from k_0, the adversary can recompute all values used in the decryption apart from k.

It is interesting that when there is randomness misuse (when the adversary provides k_0), there is no useful information in the encryption leakage for CIML2 security.

Theorem 9. *Let* $\mathsf{F}^* : \mathcal{K}^* \times \mathcal{B}^* \times \mathcal{TW}^* \to \mathcal{B}^*$ *be a leak free* $(q+1, \epsilon_{\mathsf{sTPRP}})$*-strong tweakable pseudorandom permutation (*sTPRP*), let* $\mathsf{F} : \mathcal{K} \times \mathcal{B} \to \mathcal{B}$ *be a* $(2, \epsilon_{\mathsf{PRF}})$*-pseudorandom function (*PRF*) and let* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \mathcal{B}'$

[14] The leakage model used in this proof is 2-simulatability

[15] For the RUP-AE-security, we request that the plaintext m retrieved during a decryption query, if it comes from an invalid ciphertext, is indistinguishable from random.

be a ϵ_{CR}-collision resistant hash function. Let $\mathcal{B}^* = \mathcal{K} = \mathcal{B}$ and $\mathcal{TW}^* = \mathcal{B}'$. Let $\mathcal{B} = \{0,1\}^n$.
Then, the mode CONCRETE*, which encrypts messages which are at most L-block long, is (q_E, q_D, ϵ)-*CIML2 *secure in the unbounded leakage model with*

$$\epsilon \leq \epsilon_{\mathsf{sTPRP}} + \frac{(q_E+q_D)(q_E+q_D-1)}{2^{n+1}} + \epsilon_{\mathsf{CR}} + \frac{(q_D+1)(L+1)(q_D+2q_E)}{2^{n+1}} + \frac{q_D+1}{2^n} + (q_D+1)\epsilon_{\mathsf{PRF}}.$$

with $q = q_E + q_D$ and

$$t_1 = t_{\mathsf{ch}'} + (q+1)(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}})$$

$$t_2 = t_{\mathsf{ch}'} + (q+1)(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}) + t_{\mathsf{f}(Q+1)}.$$

Observation on the bound. We want to discuss some terms of the bound:

- $\epsilon_{\mathsf{sTPRP}} + \frac{(q_E+q_D)(q_E+q_D-1)}{2^{n+1}}$ because F^* is a sTPRP and not a PRF.
- ϵ_{CR} because, if there is a collision, the mode is trivially broken: given $c_0 = \mathsf{F}_{k_0}(p_B)$ if there is a collision $((c_0, c_1, ..., c_l), (c_0, c'_1, ..., c'_l))$ we observe that $c_{l+1} = c'_{l+1}$ if they both encrypt k_0,
- $(q_D+1)\epsilon_{\mathsf{PRF}}$, because we do not check k_0, but $F_{k_0}(p_B)$. [16]
- $\frac{(q_D+1)(L+1)(q_D+2q_E)}{2^{n+1}}$ because we need that, in every decryption query, k_0 must have never been used before as ephemeral key; otherwise, c_0 would not be random anymore. It may be improved to $\frac{(q_D+1)(q_D+2q_E)}{2^{n+1}}$ if F is not used in PSV, [for example, we may use a different PRF, but this choice would require one more primitive to implement].

Sketch. In the proof, first, we replace F^* with a random tweakable permutation, then, we suppose that all the hash outputs are different (provided that their inputs are different). For fresh decryption queries, on input $c = (c_0, c_1, ..., c_l, c_{l+1})$ we observe the following:

1. If the partial ciphertext $(c_0, ..., c_l)$ is fresh, then, its hash h is fresh. Thus, k_0 is random. Consequently, the probability that $c_0 = \mathsf{F}_{k_0}(p_B)$ is bounded by ϵ_{PRF}.

2. If the partial ciphertext $(c_0, ..., c_l)$ is not fresh and comes from an encryption query, then, the couple $((c_0, ..., c_l), c_{l+1})$ must be fresh; otherwise, either the decryption query is not fresh [not possible by

[16] If we had checked k_0 and put it into the ciphertext, i.e., $c_0 = k_0$, we would have obtained a better CIML2 bound, but no pAE security.

hypothesis] or it is the repetition of a previous decryption query [so, its validity has already been established]. Thus, $k_0 = \mathsf{F}_k^{*,-1}(h, c_{l+1})$ is still random. Then, again, the probability that $c_0 = \mathsf{F}_{k_0}(p_B)$ is bounded by ϵ_{PRF}.

To prove that $\Pr[c_0 = \mathsf{F}_{k_0}(p_B)]$ is negligible, we use that F is a PRF, thus, we must assume that k_0 is fresh. □

The complete proof can be found in App. B.7.

Interestingly, we use the fact that F is a PRF in the proof (to prove that the probability that $c_0 = \mathsf{E}_{k_0}(p_B)$ is negligible). This is a difference with respect to all previous proofs, Thm. 4, Thm.'7 and 8, where no hypothesis on F has been used in the CIML and $\mathsf{CIML2}$ proofs.

5.9 Other constructions

In many constructions a new primitive is used: sponges.

Sponges. With the victory of Keccak [32] for the competition to build the new standard hash function SHA-3, a new cryptographic primitive has become popular: the *sponge* [32].
A sponge is based on a permutation $\mathsf{p} : \{0,1\}^n \to \{0,1\}^n$ and has two steps: absorbing and squeezing.
The state of the permutation is divided in two parts, the *rate* part of r bits, and the *capacity*.
First, the sponge is initialized choosing its initial state s_0.
When, a sponge absorbs a message $m = (m_1, ..., m_l)$, with $|m_i| \leq r$, it simply updates its state with $s_{i+1} \leftarrow \mathsf{p}(s_i \oplus m_i \| 0^*)$, that is, m_i is XORed to the first r bits of the state s_i, then, applying the permutation p to the results of the previous computation, a new state state s_{i+1} is obtained.
At the end, the output is obtained squeezing the sponge, that is the outputs of the sponge is the first r bits of the states $s_{l+1}, s_{l+2}, ...$ with $s_{i+1} \leftarrow s_i$. That is, we takes the first r bits of s_{l+1}, then, we update the state with the permutation and we iterate until we have a string long enough.
The security of sponges is based on the fact that the capacity part of the state is not under adversarial control.

Sponges are often used in the duplex construction:

The duplex construction. It is based on a sponge. The idea is not to wait for the end of the absorption phase to start outputting bits and using them as a stream cipher. Instead, after initialization, the sponge is used at the same time to absorb the message and to encrypt. After every time the permutation is called, a part of the updated state is output, and it is used as stream cipher block; that is, a part of the state is XORed with a block of message forming a ciphertext block. Then, the message is absorbed, that is XORed with the state, and then, the state is updated doing a new round of the permutation on it.
In this way, every block of the ciphertext depends on all previous blocks. Thus, part of the final state can be used as a tag.
In this way, a sponge can be used as a one-pass AE scheme (Fig. 5.14 depicts an example of a duplex construction).

Advantages of the duplex construction. The duplex construction seems very interesting from a leakage-resilient point of view since if the capacity (that is, the part of the state which is not output) is not fully leaked, then, the new state (after having applied the permutation to the previous state) should be "enough" random.

Taxonomy of other constructions. We divide these constructions according to the security they offer:

- Schemes which offer CIL1 (Ciphertext Integrity with leakage in encryption) and CCAL1 (CCA [17] with leakage in encryption)
- Schemes which offer CIML2 and CCAmL1 (CCA with misuse-resilience and leakage in encryption)
- Schemes which offer CIML2 and CCAmL2 (CCA with misuse-resilience and leakage in both encryption and decryption)

We conclude with the construction of Barwell et al. [10] which have different security goals.

5.9.1 Inner-keyed sponges: CIL1 and CCAL1-secure.

Here, we propose a scheme which is based on the duplex construction: PHOTON-BEETLE [?], see Fig. 5.13.
Note that the key is only used at the start of the encryption to initialize

[17] CCA stands for Chosen Ciphertext attacks security, that is, we suppose that the adversary is not able to distinguish the encryption of two plaintexts with the same length even if he has oracle access to Enc_k and Dec_k.

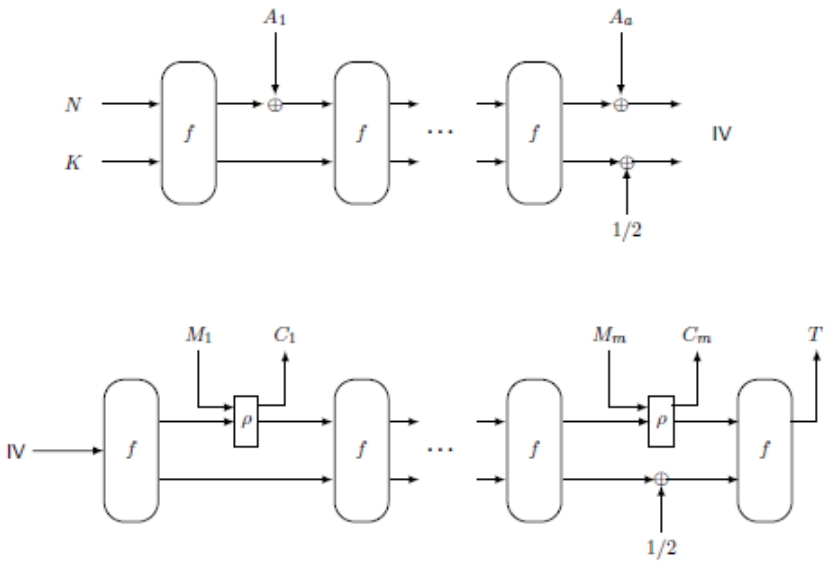


Figure 5.13: The PHOTON-BEETLE AE mode. The figure is taken from [**?**].

the sponge (Similar designs have the NIST candidates Gimli, Xoodyak [41], and Oribatida).
These schemes are called *inner-keyed* because the key is put in the capacity of the initial state of the sponge.

Confront with our modes. The idea is that during encryption if the full state of the sponge is not completely leaked, the adversary is not able to make a forgery.
On the other hand, if a nonce is reused or if the adversary has access to decryption leakages, he can mount a DPA attack on the state. Thus, recovering it, he may create any forgery.
Instead, in our modes, we use the key also when we authenticate the hash of the ciphertext (the final state of the sponge can be seen as the hash of the ciphertext). With our solution, we can cover the case of misuse and, with the inversion, decryption leakages.

5.9.2 ASCON and Spook: CIML2 and CCAmL1 secure

ASCON and Spook are candidates to the NIST competition for lightweight AE. They are based on the duplex construction.

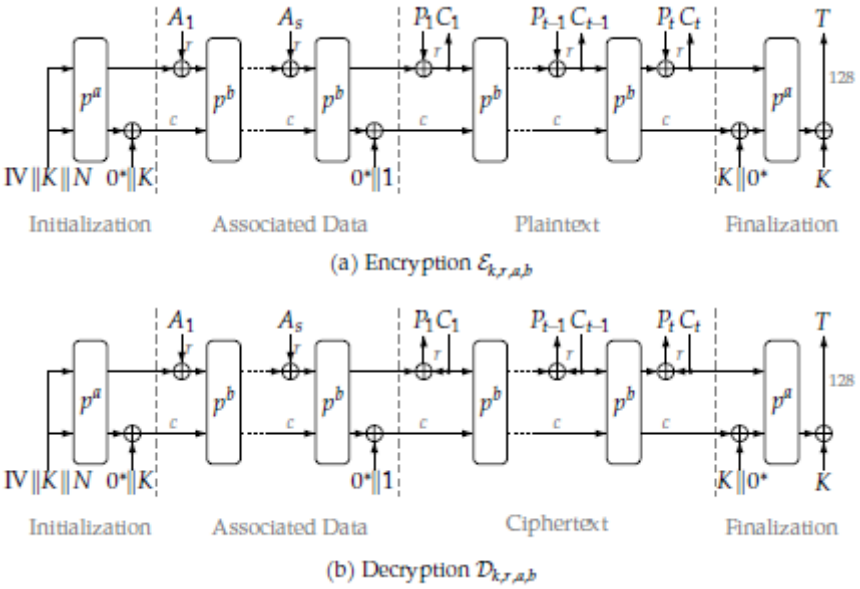


Figure 5.14: The ASCON AE mode. The figure is taken from [44]

Differently from the schemes presented in the previous subsection, in these modes the key is also used when the ciphertext is authenticated.

ASCON. ASCON [44] (see Fig. 5.14) is based on the duplex construction. The key of the scheme is used at the start as inner-keyed sponges, than, after having done the first round of the sponge, it is XORed again to the state.
Moreover, it is used again before the final permutation (which produces the "digest" of the whole ciphertext) and it is XORed again to the output in order to obtain the tag.

Spook. Spook [?] can be divided in 4 phases:

- Using a strongly protected TBC (and the key), starting from the nonce (and, using the public key P as tweak) we produce an ephemeral key
- From this ephemeral key, we use a sponge to absorb the associated data.
- We use a duplex construction to encrypt and absorb the ciphertext

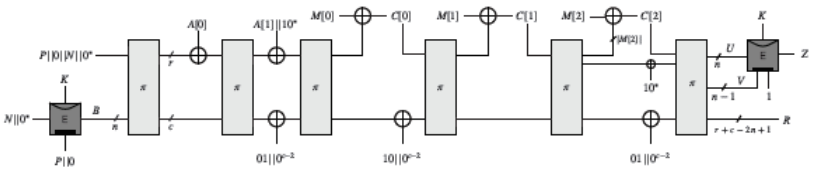


Figure 5.15: The Spook AE mode. The figure is taken from [?]

- We authenticate with the strongly protected component (and the key), the final state of the sponge[18]

- In decryption, to verify we use the inverse of the TBC, as in DTE2 and in EDT.

Spook is our proposal to the NIST competition. It can be seen in Fig. 5.15. It has been studied in many works [?, ?, ?, 67, ?].
Note that Spook achieves multi-user security.

Comparison with our modes. We start observing we can see that both schemes are following the paradigm of EDT, with the encryption and the digest done in the same phase. In particular, the key is used in both modes as for EDT, during the initialization, which depends on the nonce, and during the tag computation.
Spook allows a CIML2 security using the inversion of the TBC during decryption, while for ASCON the comparison needs to be done in a "leak-free" way, if we study it in the unbounded-leakage model.
Regarding confidentiality they do not offer security if a nonce is reused, but only for reused nonce (misuse-resilience). Moreover, if decryption leaks, a DPA attack can be easily done (similarly to what presented for DTE2, see Chap. 5.7 since before verification, the plaintext is recomputed.
On the other hand, with respect to EDT, these schemes are one-pass, thus, more efficient.

5.9.3 ISAP and TEDT: CIML2 and CCAmL2-secure

Here, we propose two schemes which are CIML2 and CCAmL2-secure: ISAP and TEDT.

[18] Note that the last bit of the tweak is one in this call, while it was forced to be 0 in the other ones to prevent the attack described in Sec. 5.3.2.

ISAP. ISAP [43] is a 2-pass encryption mode based on sponges. It is described in Fig. 5.16.
It is based on four permutations: p_K, p_B, p_E and p_H. These four permutations are modeled as being random.
During encryption, first the state is initialized with the key k and a fixed IV, and then, it is updated with a new round of the permutation p_K. Then, the nonce N is absorbed (that is, first it is XORed with part of the state, then, the new state is updated doing a new round of the permutation p_B). This absorption is done bit by bit to avoid DPA.
Then, the ciphertext is obtained using the permutation p_E as a stream cipher (that is, after every round of the permutation, r_H bits of the state are extracted and XORed with the ith block of message M_i to obtain the ith block of ciphertext C_i).
After that, a second pass is done, using the sponge as a hash function: the sponge is initialized with the nonce N and a fixed IV, IV_A, then, it absorbs the associated data and the ciphertext to obtain a value Y [19]. Finally, Y is absorbed as the nonce, bit by bit, to give K_A^*, and the tag is computed replacing the first $|k|$ bits of the state with K_A^*, doing another round of the permutation and taking the first τ bits of the output.
This mode is very well designed. It prevents DPA absorbing bit by bit the value which may be chosen by the adversary.
The security of ISAP in the presence of leakage has been studied by many works [42, 67, 45, 49, 50].

Comparison between ISAP with our modes. We can see, as in our modes, that there are parts of the mode which are more protected than others. In fact, we can see the absorption bit by bit of the nonce N and of Y as a strongly protected implementation of the sponge, while for the rest of the mode a faster, but not DPA-secure, sponge is used.
We can see that ISAP, as EDT is based on a Encryption, Digest and Tag paradigm.
On the other hand, in verification the tag must be recomputed and verified. Thus, to obtain authenticity when also decryption leaks, it is necessary to have a DPA-secure comparison between the tag T provided with the ciphertext and the correct tag $\tilde{T}$ (the authors propose to do an additional round of the permutation, that is, to compare the output of $\mathsf{p}(T\|0^*)$ with those of $\mathsf{p}(\tilde{T}\|0)$).
In particular, if we study the CIML2-security of ISAP in the unbounded leakage model, also the comparison must be assumed as "leak-free", while

[19] Note that when all the associated data are absorbed, the last bit of the state, which belongs to the capacity part of the state, is XORed with 1 to distinguish the two situations.

in our modes, the adversary receives the values that are compared.
Our approach and their are very interesting, with their pros and cons.
With respect to DTE and DTE2 which are two pass-modes, we observe that our modes offer misuse-resistance in addition.

TEDT. TEDT [21], see Fig. 5.17 is a 2-pass mode based on TBC. TEDT is based on EDT. It can be divided in 4 phases:

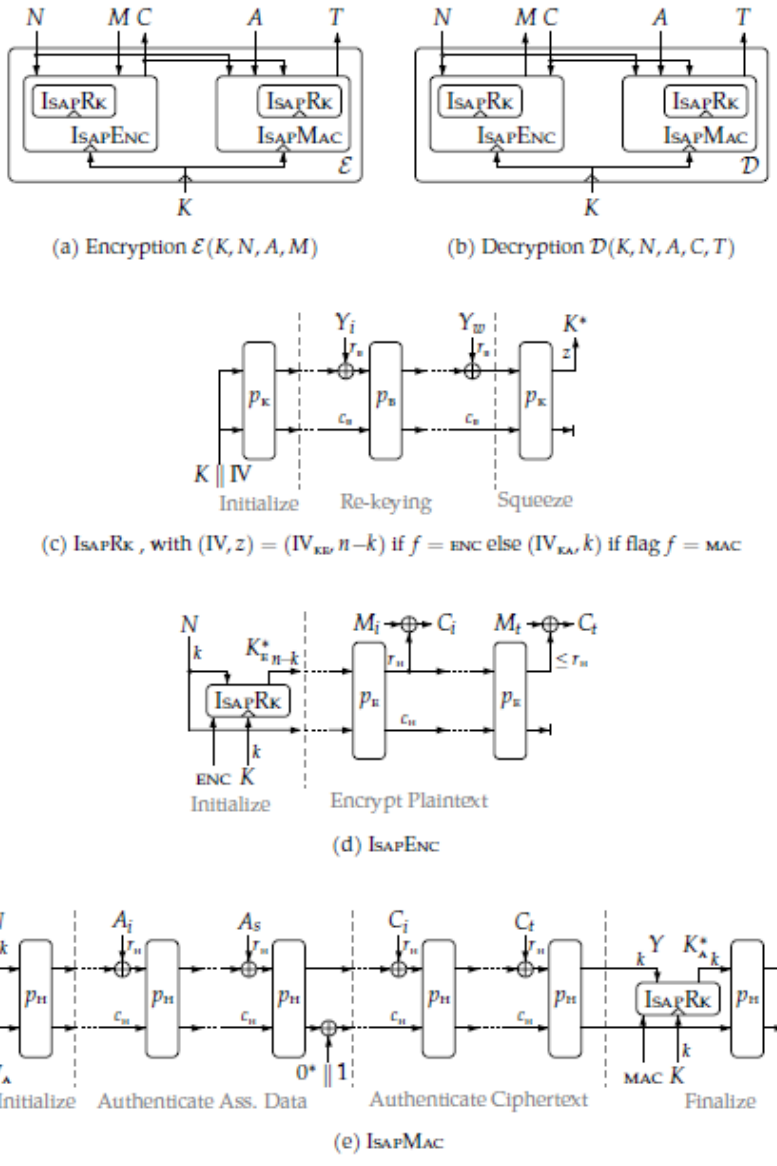
- An ephemeral key is created using the strongly protected TBC, from the nonce, the key and the public key T.
- From this ephemeral key, we use a variant of PSV to compute the ciphertext (the public key is used as a tweak for the TBC).
- We use the Hirose construction to hash the associated data, the nonce, the ciphertext and the public key.
- We authenticate with the strongly protected component (and the key), this hash with HTBC[20]
- In decryption, to verify we use the inverse of the TBC, as in DTE2 and EDT.

Comparison between TEDT. TEDT achieves CIML2 as EDT in the same way. On the other hand, it achieves BBB security (using the public key, and doubling the outputs of the key). Moreover, the use of T in the encryption part allows to avoid that if a collision appears on the ephemeral keys, confidentiality is doomed.
Finally, it achieves multi-user security.
We can see TEDT as an improvement of our schemes.

5.10 The construction of Barwell et al.

Barwell et al. [10] started proving that Encrypt-then-MAC is secure under leakage. Then, they aim to have a mode that offers both misuse and security with leakage. So they propose SIVAT (Synthetic IV and Tag). Their idea is to use an iv-based encryption scheme. They obtain the iv via a leakage resilient PRF, whose inputs are nonce, the message, and the associated data (AD). Then, the tag is computed via a leakage-resilient MAC from the ciphertext, the iv and the AD.
To instantiate the PRF, they take advantage of the existing leakage-resilient literature.
For the MAC, to avoid that the leakage of the verification may cause

[20] Note that the last bit of the tweak is one in this call, while it was forced to be 0 in the other ones to prevent the attack described in Sec. 5.3.2.



(a) Encryption $\mathcal{E}(K, N, A, M)$

(b) Decryption $\mathcal{D}(K, N, A, C, T)$

(c) IsapRk , with $(\mathrm{IV}, z) = (\mathrm{IV}_{\mathrm{KE}}, n-k)$ if $f = \textsc{enc}$ else $(\mathrm{IV}_{\mathrm{KA}}, k)$ if flag $f = \textsc{mac}$

(d) IsapEnc

(e) IsapMac

Figure 5.16: The ISAP AE mode. The figure is taken from [43]

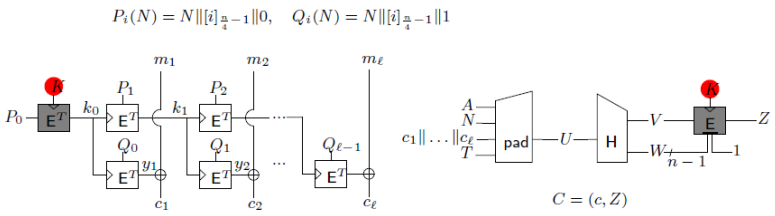


Figure 5.17: The TEDT AE mode. The figure is taken from [21].

problems, they propose to use the MAC of Martin et al. [98]. This MAC uses pairing, and it is proved secure in the generic group model.

Comparison with our modes. We can see that SIVAT has the same structure as DTE (in fact, both modes aim to obtain misuse-resistance). On the other hand, we observe that their goal is to find a good composition scheme for leakage-resilient encryption and MAC, while our schemes are designed considering leveled implementation. In fact, for their security proof everything is assumed to be well protected against leakage, while in ours, only the strongly protected component.
Moreover, this scheme aims to leakage *resilience* (that is, the leakage of other encryption or decryption queries does not affect the confidentiality of a ciphertext if the adversary has not received the leakage when it is computed); instead, we aim to have leakage-resistance, that is, to have confidentiality even if the adversary has access to the leakage of the computation of the challenge ciphertext.
Finally, we observe that our solution to have CIML2 is better because it is more efficient (since pairings have bigger keys), moreover, our proofs are done using standard assumptions (instead in theirs, they use the generic group model[21]).

[21] We observe that this latter assumption is questionable when the adversary also has access to the leakage of the MAC.

Chapter 6

Authenticity from unpredictability

Contents

This chapter is devoted to the proofs of the suf-L2-security (and the CIML2-security) in the unbounded leakage model when we model the strongly protected implementations as strongly unpredictable with leakage (sUL).
It turns out that the MACs and many of the AE schemes remain suf-L2 and CIML2 respectively although we have weakened the security leakage assumption on the implementation (from leak-free to sUL).
Here we do a trade-off, we have milder hypothesis for the strongly protected building blocks, but we have stronger hypothesis for the hash functions since they are modelled as Random Oracles. Moreover, consider that there is no key to protect for the hash function.

In this chapter, we prove the suf-L2-security of HBC2 (defined in Chap. 5.5.1) based on a sUL BC. After that, we we prove the suf-L2-security of HTBC (defined in Chap. 5.5.3) based on a sUL TBC. Then, we mention how we can extend these results to the CIML2-security of the AE schemes we have introduced in the previous chapter. Finally, we discuss the usage of the Random Oracle.

This chapter is based on the Inscrypt paper and its eprint version [22].

6.1 The suf-L2-security of HBC2 based on sUL

In the unbounded leakage model, the adversary receives all the ephemeral values computed during the tag generation and the verification. Only the key of the BC, which is the key of the MAC, remains hidden as implicitly defined by the leakage function pair of its implementation $\mathsf{L} = (\mathsf{L}_{\mathsf{Eval}}, \mathsf{L}_{\mathsf{Inv}})$. More precisely, the unbounded leakage function pair $\mathsf{L}^* = (\mathsf{L}^*_{\mathsf{Mac}}, \mathsf{L}^*_{\mathsf{Vrfy}})$ of HBC2 is thus:

$\mathsf{L}_M(m;k)$**:** return $h = \mathsf{H}(m)$ and $\mathsf{L}_{\mathsf{Eval}}(h;k)$;

$\mathsf{L}_V(m,\tau;k)$**:** return $h = \mathsf{H}(m)$ and $\tilde{h} = \mathsf{F}_k^{-1}(\tau)$ as well as $\mathsf{L}_{\mathsf{Inv}}(\tau;k)$.

Despite H is a public function, we explicitly include its outputs in the leakage. It can be considered as redundant but, as we rely on a random oracle to prove the security of HBC2, we prefer making them fully available to avoid any confusion.

Theorem 10. *Let* $\mathsf{F}^* : \mathcal{K}^* \times \mathcal{B}^* \longmapsto \mathcal{B}^*$ *be a* $(q_M, q_V, q_L, t, \epsilon_{\mathsf{sUL}})$*-strongly unpredictable block cipher in the presence of leakage, and* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \longmapsto \mathcal{B}'$ *be a hash function modeled as a random oracle that is queried at most* q_H *times. Let* $\mathcal{B}^* = \mathcal{B}' = \{0,1\}^n$ *and* $\mathcal{HM} = \{0,1\}^*$.
Then, HBC2 *is a* $(q_M, q_V, q_{\mathsf{L}}, t, \epsilon)$*-strongly unforgeable MAC in the unbounded leakage setting, with* $\mathsf{L}^* = (\mathsf{L}_M, \mathsf{L}_V)$ *defined above, where*

$$\epsilon \leq (q_H + q_V + 1)(q_V + 1)\epsilon_{\mathsf{sUL}} + (q_H + q_M + q_V + 1)^2/2^n,$$

and $t_H(q_H + q_M + q_V + 1) + (q_M + q_L - q)t_{\mathsf{F}} + (q_V + q)t_{\mathsf{F}^{-1}} \leq t$ *for any* $q \leq q_L$*, and where we assume that all the* H*-query involved in the* q_L *queries are already among the* q_H *queries, and if* $q_V \leq q_H$ *(which can be artificially fulfilled at the end of the experiment).*

The advantage is bounded by $2q_Hq_V\epsilon_{\mathsf{sUL}} + 4q_H^2 2^{-n}$ under the natural assumption $q_M + q_V + 1 \leq q_H$ (since q_M and q_V correspond to online queries while q_H corresponds to offline queries, it is expected to hold comfortably). The leading term is $2q_Hq_V\epsilon_{\mathsf{sUL}}$: for $\epsilon_{\mathsf{sUL}} = 2^{-128}$, it implies that security holds up to $q_H = 2^{64}$ and $q_V = 2^{63}$ (i.e., slightly below the birthday bound), if implemented with a 128-bit blockcipher (as AES). For a more realistic $\epsilon_{\mathsf{sUL}} = 2^{-96}$, it only holds with a stronger limit on the number of verification queries (e.g., $q_H \approx 2^{64}$ and $q_V = 2^{31}$). Note that the factor q_Hq_V may be due to the reduction proof technique, as it relates to a case where the adversary is likely to produce a forgery early in the experiment. Therefore, much of the computational power of the reduction seems useless to the adversary. Hence, it might be possible

to obtain tighter bounds using a different reduction approach. It would also be interesting to explore the possibility of making a proof based on standard assumptions on the hash function. Such assumptions would require to exclude damaging and implausible interactions between the hash function and the PRF and would be an interesting area for future research.

Idea of the proof. Assuming that an adversary A succeeds in the $\mathsf{FORGEL2}^{\mathsf{suf-vcma-L2}}_{\mathsf{A},\mathsf{HBC2},\mathsf{L}^*}$ experiment by making a total of q_M leaking tag queries and q_V leaking verification queries, let (m, τ) be the forgery, i.e., the couple returned by A in the finalization phase. To bound this winning probability, we partition this event into sub-events:

1 The tag τ appears in the answer to a leaking tag query (and thus, as an output of F_k);
2 The tag τ never appears in answer to a leaking tag query (and thus, τ can only be involved as an input of F_k^{-1}) and:
   a m appears as an input of H before $\mathsf{F}_k^{-1}(\tau)$ was ever computed in the experiment;
   b τ appears as an input of F_k^{-1} before $\mathsf{H}(m)$ was ever computed in the experiment.

We cover all the cases since when both m and τ are fresh in a verification query, we always compute (or ask the computation of) $\mathsf{H}(m)$ first so that we can say that m appears "before" (the computation of F_k^{-1} on input) τ, and since we view the last verification in the finalization as the (q_V+1)-th verification query.

The goal of the proof is to show that the collision resistance of H ensures that winning in case 6.1 is negligible, the unpredictability of F ensures that winning in case 6.1 is negligible and that preimage resistance of H ensures that winning in case 6.1 is negligible as well.

1 there is a tag query on m' which defines $\tau = \mathsf{F}_k(\mathsf{H}(m'))$. Since (m, τ) is a forgery we have $\mathsf{F}_k(\mathsf{H}(m)) = \tau$ with $m \neq m'$. Then, m and m' produce a collision as F_k is a permutation: $\mathsf{H}(m) = \mathsf{F}_k^{-1}(\tau) = \mathsf{H}(m')$.
2a we assume that F is SUL2-secure. Since m appears before τ, as a challenger, we have to use the value $h = \mathsf{H}(m)$, and we "wait" for the good tag in a verification query to win the sUL game. We do not have to consider the messages for which A makes a tag query. However, we cannot know in advance what will be the right tag, and we cannot wait until the finalization of the unforgeability experiment because even if A's output (m, τ) is the right pair, τ may have been already used in a previous verification query. If so, the challenger should have already made a leaking inverse query of

the block cipher with input τ to get $\tilde{h} = \mathsf{F}_k^{-1}(\tau)$ and $\mathsf{I}_\mathsf{i} \leftarrow \mathsf{LInv}(m;k)$ to simulate $\mathsf{L}_V(m,\tau;k)$, and it could no more win the game against F with τ. Therefore, for all possible m involved in a H-query or a verification query, we have to guess what will be the right τ in verification. Then, we need to consider all the possible such pairs, and we thus have to make at most $(q_\mathsf{H}+q_V+1)(q_V+1)$ reductions.

2b the reduction can generate the key k itself and then evaluate F and its inverse by itself. The first time τ appears (in a leaking verification query) we define the H-target $\tilde{h} = \mathsf{F}_k^{-1}(\tau)$ since the winning corresponding m is still not hashed by assumption. Note that we even do not care whether $\tilde{h}$ already appeared or not in response to a H-query since fresh queries will result in independent hashes. Therefore, the validity of (m,τ) means that m is a preimage of $\tilde{h}$, as $\mathsf{H}(m) = \mathsf{F}_k^{-1}(\tau)$, while $\mathsf{H}(m)$ is random.

The complete proof can be found in App. B.8.

In the next section, we show that HTBC has better security bound, which is beyond birthday.

6.2 The suf-L2-security of HTBC based on sUL

As usual, we start considering the unbounded leakage function pair $\mathsf{L}^* = (\mathsf{L}^*_{\mathsf{Mac}}, \mathsf{L}^*_{\mathsf{Vrfy}})$ of HTBC. We define L^* as follow:

$\mathsf{L}_M(m;k)$**:** return $h_1 \| h_2 = \mathsf{H}(m)$ and $\mathsf{L}_{\mathsf{Eval}}(h_2, h_1; k)$;

$\mathsf{L}_V(m,\tau;k)$**:** return $h_1 \| h_2 = \mathsf{H}(m)$ and $\tilde{h}_1 = \mathsf{F}^{-1}_{k,h_2}(\tau)$ as well as $\mathsf{L}_{\mathsf{Inv}}(h_2, \tau; k)$.

As we rely on the random oracle model to prove the security of HTBC, we include the digests in the leakage to capture the fact that H is a public function.

Theorem 11. *Let* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \longmapsto \mathcal{B}'$ *be a hash function modeled as a random oracle, and* $\mathsf{F}^* : \mathcal{K} \times \mathcal{TW} \times \mathcal{B} \longmapsto \mathcal{B}$ *be a* $(q_T, q_V, q_\mathsf{L}, t, \epsilon_{\mathsf{sUL}})$*-strongly unpredictable tweakable block cipher with leakage* $\mathsf{L} = (\mathsf{L}_{\mathsf{Eval}}, \mathsf{L}_{\mathsf{Inv}})$*. Let* $\mathcal{HM} = \{0,1\}^*$*,* $\mathcal{B}' = \mathcal{TW} \times \mathcal{B}$ *and* $\mathcal{B} = \{0,1\}^n$*.*
Then, HTBC *is a* $(q_T, q_V, q_\mathsf{H}, q_\mathsf{L}, t, \epsilon)$-$\mathsf{suf}$-$\mathsf{L2}$ *strongly unforgeable* MAC *with unbounded leakage function pair* $\mathsf{L}^* = (\mathsf{L}_M, \mathsf{L}_V)$ *as defined above, where*

$$\epsilon \leq \frac{(q_H + q_T + q_V)^2}{2^{2n}} + (q_V+1)\cdot\epsilon_{\mathsf{sUL}} + \frac{q_H^2 q_V}{2^n}\cdot\epsilon_{\mathsf{sUL}} + \frac{q_V(q_H+q_V)}{2^{2n}},$$

and $t_H(q_H + q_T + q_V + 1) + (q_T + q_L - q)t_{\mathsf{F}} + (q_V + q)t_{\mathsf{F}^{-1}} \leq t$ *for any* $q \leq q_L$*, and where we assume that all the* H*-query involved in the* q_L *queries are already among the* q_H *queries, as long as* $4 \leq q_H + q_T + q_V$*,* $4q_V \leq q_H$ *and* $10q_H \leq 2^n$*.*

The leading term in the security bound is $\epsilon_{\mathsf{sUL}} \cdot q_H^2 q_V 2^{-n}$. This time, for $n = 128$ and $\epsilon_{\mathsf{sUL}} \approx 2^{-96}$, security holds up to $q_H \approx 2^{80}$ and $q_V = 2^{64}$. As for Theorem 5, we are not aware of a realistic matching attack (i.e., if a reasonable hash function and TBC are used in the construction). Investigating whether the additional $q_H 2^{-n}$ factor that we gain compared to the BC-based construction can get closer to 2^{-n} is an interesting open problem.

The structure of the proof for HTBC is different from that of HBC2. The main reason is that the collision resistance of H does not cover all the winning cases when the adversary's τ of the forgery appears in an LMac query. Indeed, we might have $\mathsf{H}(m) = h_1 \| h_2 \neq h_1' \| h_2' = \mathsf{H}(m')$ such that $\mathsf{F}_k(h_2, h_1) = \tau = \mathsf{F}_k(h_2', h_1')$ with m' in an LMac query. That is because F_{k,h_2} and $\mathsf{F}_{k,h_2'}$ can be seen as two different permutations given that $h_2 \neq h_2'$: an output τ defines many possible tweak-input pairs. As we will see the distribution and the freshness of (h_2, τ) will play an important role in the proof.

Idea of the proof. Let (m, τ) be a forgery and write $\mathsf{H}(m) = h_1 \| h_2$. If no triple of the form $(\star, h_2, \tau)$ appears during the computation of all the evaluations and inversions of F, (h_1, h_2, τ) is a valid fresh triple for F which breaks the unpredictability of the TBC. However, if it is not the case, a triple $(\star, h_2, \tau)$ appears either in the evaluation of F during an LMac query or only in the inversion of F in an LVrfy query. In the former case, as the answer to an LMac query is necessarily valid, the triple $(\star, h_2, \tau)$ must actually be $(\mathsf{F}_k^{-1}(h_2, \tau), h_2, \tau)$, i.e. (h_1, h_2, τ). Of course, if the adversary has made an LMac query on m, it cannot win. If the adversary is successful, it means that it managed to request the computation of a hash value which collides on $\mathsf{H}(m)$, which only occurs with a beyond-birthday probability in $n = |k| = |\tau|$. We can thus focus on the latter case where a triple $(\star, h_2, \tau)$ only appears when answering an LVrfy query, i.e., in an inversion of F.

We split the remaining winning conditions into:

2 m appears as an input of H before ever computing a TBC inversion on input (h_2, τ) when answering a leaking verification query;

1 m appears strictly after the first computation of a TBC inversion on input (h_2, τ) when answering a leaking verification query;

no matter whether τ appears first in a LMac answer or in a LVrfy query. We note that we no more need to consider the H computation in the LMac queries as we already dealt with H-collisions. In a nutshell, the first case means the adversary chooses τ depending on the view of the hash value $h_1\|h_2$, and hence it relates to the unpredictability of F. In the second case, the target $h_1\|h_2$ is fixed in the leaking verification query while the output of $\mathsf{H}(m)$ remains uniformly random and independent of the view at that time. By convention, if m and τ first appear for the first time together in a LVrfy query, we first compute $\mathsf{H}(m)$ so that we always consider that m appears "before" τ. Besides, we consider the forgery as the (q_V+1)-th LVrfy query.

Case 1: $\mathsf{H}(m) = h_1\|h_2$ appears before the computation of a TBCtriple $(\star, h_2, \tau)$ which will first be run when answering a leaking verification query. We want to build an adversary B against F which ends by sending (h_1, h_2, τ). To make B successful, we have to prevent B from making an LInv query on input (h_2, τ) earlier, since otherwise (h_1, h_2, τ) is not a winning triple at the end. Such a query can happen only if A manages to make an LVrfy query on some (m', τ) such that $\mathsf{H}(m') = h'_1\|h'_2$ with $h'_2 = h_2$. Of course, this happens if $m = m'$ and, indeed, A can win if (m, τ) appears in an LVrfy query before the (q_V+1)-th one. However, it can also happen if $m' \neq m$, but then $h'_1 \neq h_1$. Fortunately, if the first time (h_2, τ) appears in an LVrfy query is with $m' \neq m$, we know that m appeared in a hash computation earlier for the first time (and it cannot be in a LMac query). To sum up, we cannot build a single B but by considering all the hash computations in the H queries and the LVrfy queries to combine with all the tags in the LVrfy queries, we have at most $(q_H + q_V + 1)(q_V + 1)$ reductions to build to cover all the possibilities. Fortunately, we only have to consider the messages m' with $\mathsf{H}(m') = h'_1\|h'_2$ such that h'_2 appears in a subsequent LVrfy query. Furthermore, we can see when this happens before having to invert the TBCwith tweak h'_2. We will see in the proof that the probability of multi-collision on the h^i_2-value involved in the ith LVrfy query will decrease the probability by a factor of roughly $q_H/2^n$, which gives us a beyond birthday term eventually.

Case 2: the adversary outputs a forgery (m, τ) while (h_2, τ) appears in a leaking verification query before the first computation of $\mathsf{H}(m)$. Here, we simply pick the key of the TBC to simulate the forgery experiment. If (h_2, τ) already appears in an LVrfy query the valid triple $(\tilde{h}_1, h_2, \tau)$ is already fixed in answer to that query (necessarily invalid). Therefore $\mathsf{H}(m)$ which is still uniformly random and independent of the view at that time will have to match the tar-

get $(\tilde{h}_1, h_2)$. This match thus happens with probability $1/2^{2n}$ for each future hash evaluation in a H-query or a next LVrfy query. Of course we do not know what will be the right (h_2, τ) until the adversary output its forgery in the finalization phase. So, if (h_2^i, τ^i) denotes the input of the inversion of F in the i-th leaking verification query, we actually defines q_V targets $(\tilde{h}_1^i, h_2^i, \tau^i)$, since $i < q_V + 1$ here. Therefore, the probability that this case occurs is upper-bounded by $q_V(q_H + q_V)/2^{2n}$.

The complete proof can be found in App. B.9.

6.3 Application to CIML2

These results can be extended also to CIML2. For DTE2 and EDT the extension is straightforward from Thm. 5.

CONCRETE. For CONCRETE, a straightforward adaptation of our proofs is not enough. We can only say that the retrieved key k_0 is unpredictable. This fact is not enough to say that the probability that $\tilde{c}_0 = c_0$ is negligible. Such a situation can happen due to "pathological" block ciphers. (Imagine a block cipher which does not use half of its key. This block cipher may still be a PRF. Imagine that we are only able to predict the half of the key, which is used. In this case, CONCRETE is not secure). We conjecture that assuming that F is an Ideal Cipher[1], the CIML2-security of CONCRETE can be extended to the hypothesis that F^* has a sUL-implementation.

6.4 About the usage of the Random Oracle

We would like to remove the ROM hypothesis. We do not require any form of programmability or other conveniences that come with the random oracle model). This model fits well with many applications, for instance, when we build the hash function from a sponge, which we also traditionally model as an ideal permutation. Still, our analysis does not suggest any reason why an ideal object would be needed, and it would, therefore, be interesting to investigate whether and how this assumption could be relaxed.

[1] An *ideal tweakable blockcipher* is a function $\mathsf{F} : \mathcal{K} \times \mathcal{TW} \times \mathcal{B} \longmapsto \mathcal{B}$ such that $\forall k \in \mathcal{K},\ tw \in \mathcal{TW},\ \mathsf{F}_k(tw, \cdot) : \mathcal{B} \times \mathcal{B}$ is a permutation, picked uniformly at random from the set of functions with the same signature. It is a very strong hypothesis, which is related to the Random Oracle Model (see Sec. 2.7) [39, 40, 73].

Chapter 7

Conclusion

Contents

7.1 Summary

This thesis studies the problem of authenticity with leakage. Our work focuses on two different directions:

1. Find the good security definitions and identify meaningful leakage models based on which we can build security proofs.
2. Find constructions providing integrity with respect to our definitions in our leakage models

We briefly summarize the results in this chapter. Then, we develop some open problems.

7.1.1 Definitions and leakage models

Definitions. As we have explained in Chap. 2.2 definitions must be precise and express the security we want. Our four definitions (CIML, suf-L, CIML2, and suf-L2) are all defined via games and express integrity goals corresponding to real situations. Some of our definitions have been integrated into the framework proposed by Guo et al. [65] focusing on authenticated encryption.

The unbounded leakage model. We have introduced the unbounded leakage model, in this model implementations of primitives are divided into two classes, unprotected, for which we give to the adversary inputs, outputs, and keys, and strongly protected for which we give to the adversary inputs and outputs, but no keys. Thus, our definitional framework allows reducing the security of the whole scheme to some basic black box security properties and a leakage property of the implementation of a single primitive (which may be more easily verified than the leakage of the implementation of the whole scheme).

Strong unpredictability. Finally, we tackle the problem of how to model strongly protected implementations, introducing strongly unpredictability with leakage in both evaluation and inversion. This new definition is game-based, and it is falsifiable by evaluation laboratories. Moreover, a scheme whose security is established based on a strongly unpredictable implementation has a gracefully degrading security.

7.1.2 Constructions

In this thesis, we have provided many constructions achieving our security definitions. They are resumed in Tab. 7.1 for MACs and Tab. 7.2 for unpredictability.

Name	Security	Use inverse	# calls to SP (T)BC	# calls to NP BC	# calls to H	BBB secure
CCS MAC1 [111]	suf-L not unb.	No	1	l	0	No
CCS MAC2 [111]	suf-L not unb.	No	1	1	1	No
HBC [24]	suf-L unb.	No	1	0	1	No
HBC2 [27]	suf-L unb.	Yes	1	0	1	No
HTBC [21]	suf-L2 unb.	Yes	1T	0	1	Yes

Table 7.1: The MAC discussed in this thesis. We suppose that they are used to authenticate a l block message. SP stands for strongly protected implementation, while NP to not protected implementation. With T we denote if a TBC is used instead of a BC. BBB stands for beyond birthday secure.

Name	Integ. with leak.	Use inv.	# calls to BC SP	# calls to BC WP	# calls to H	BBB secure integr.	Conf with leak.	Mis.
PSV [111]	No	No	1	$2l$	0	No	CPAL	m
DTE [24]	CIML	No	2	$2l$	1	No	CPAL	M
DTE2 [27]	CIML2	Yes	2T	$2l$	1	No	CPAL	M
EDT [27]	CIML2	Yes	2T	$2l$	1	No	CPAL EavDL	m
CONC. [29]	CIML2	Yes	1T	$2l$+2	1	No	CPAL EavDL	No
DCE [24]	CIML	No	1	$2l$	1 (RO)	No	CPAL	No
FEMALE [65]	CIML2	Yes	2T	$4l$	1	No	CCAmL2 EavDL	M
TEDT [21]	CIML2	Yes	2T	$4l$T	No	Yes	CCAmL2 EavDL	m
TET1 [21]	CIML2	Yes	2T	(2l+2)T	No	Yes	CPAL	m
Spook [18]	CIML2	Yes	2T	Sp.	No	Yes	CPAL	m

Table 7.2: The AE schemes discussed in this thesis and in other works [65, 21, 18]. We suppose that we use them for a l block message. Integ. stands for integrity, inv. for inverse, conf. for confidentiality and Mis. for misuse resistance. SP stands for strongly protected implementation, while WP for weakly protected implementation. CONC. stands for CONCRETE. With T, we denote if a TBC is used instead of a BC. BBB stands for beyond birthday security. M stands for misuse resistant, while m for misuse resilience. We take the definitions of confidentiality with leakage from the zoo of Guo et al. [65]. The last five constructions have not been treated in this thesis and are there only for comparison.

Authenticity from unpredictability. Using only the strong unpredictability hypothesis for leakage, we can prove the security of some schemes (even if we have to model the hash function as a Random Oracle (RO)).

7.2 Prospects

We now highlight some future perspectives.

Real evaluations of sUL. We know that leak-free is an ideal assumption (although, it reasonably models strongly protected building blocks). On the other hand, sUL is not an ideal assumption.
Thus, since we have based the security on the sUL hypothesis, we need a

real evaluation of the sUL of a strongly protected implementation to be able to give a concrete security bound for our schemes.
Thus, finding a technique to establish the sUL security of the implementation of a BC (or of a TBC) would be a complementary result of this thesis. We suspect that this bound is similar to the bound computed for a key-retrieval attack using side-channel. We do not see any reason why, for a reasonable PRP (not a pathological one[1]), an adversary should be able to predict the output not knowing the key.
Finally, a standardization for protected implementations can be done.

Using sUL for confidentiality. We focused on integrity but it will be worth investigating confidentiality.
We have proved that it is possible to provide CIML2 and suf-L2 using a sUL (T)BC. The natural follow up is if it is possible to provide confidentiality using a sUL (T)BC. Using sponges to encrypt, as for Spook, it should be easier, because if the input is unpredictable, the output should be random. Even using a rekeying-based scheme, as PSV, as we do for EDT, for example, it may be possible to prove the confidentiality with leakage making some additional hypothesis on the BCs used in the encryption (perhaps the Ideal Cipher one?[2]). If we succeed, we are able to base security on a verifiable assumption, which allows a gracefully degradation of security.
Moreover, the interpretation of this unpredictability assumption is an interesting topic.

Extending our results to the asymmetric case. Since asymmetric cryptography is also broadly used, we think it is interesting to extend our theoretical framework to it.
The first natural step is to extend our results to signature schemes. Since everyone can check the signature, due to its asymmetric nature, we do not have anymore the problem of verification leakage.
Then, it interesting to study the problem of integrity with leakage when an asymmetric encryption scheme is used to send a key which is then used to encrypt the message via a symmetric scheme. We think CONCRETE is particularly adapted to this situation, due to its structure. In fact, since the first ephemeral key, k_0 must be sent, we might take advantage of the structure of CONCRETE to take advantage of the asymmetric primitive to send the key.
We think it is interesting to adapt our framework to the signcryption

[1] Imagine a BC which does not use a part of its key in the computations, example.
[2] Note that also for sponges, it is needed to assume that the permutation is ideal. Thus, we need idealized assumption in both cases.

primitive, which does encryption and signature in a single logical step, can be seen as the correspondent of AE in the asymmetric context.

Necessity of a (T)BC for suf-L2 and CIML2. In all our constructions achieving integrity with leakage also in decryption or verification, the use of a (T)BC is crucial. We wonder if it is possible to achieve these definitions in our model without having to use the inverse. As a first direction, one can investigate other primitives.

Get rid of the Random Oracle hypothesis. A nice problem is the necessity of the Random Oracle hypothesis in the proof of Thm. 5 and Thm. 11.
Although this hypothesis should not create any problem in practice, we do not see any fundamental reason why it is necessary. We had to use it as an artifact of the proof.
Perhaps, it is possible, via another proof technique to be able to avoid it.
At least, it would be nice to find some MACs, which are suf-L2 based on a sUL (T)BC without having to use the RO for the hash function.

7.3 Concluding remarks

We think it is fundamental to use schemes which are proved to be secure. Although there may be bugs in proofs (see, for example, the attack against the well-known OCB2 [75]), if proofs are clear and carefully inspected by many experts, the probability that this happens is low. Moreover, it is much better to have proofs (and publicly available and, thus, verifiable) than if the security is obtained by obscurity. In fact, in the latter case, it is difficult, if not impossible, to have an extensive study of the scheme; moreover, reverse engineering of the code (or espionage) may pose a threat [81]. The advantage of "open cryptographic design" has been proved many times [81].
With our definitions and our leakage models, we conform to this direction. We think that the unbounded leakage model is a very powerful threat model. Anyway, it is appealing since it allows to reduce the security of the whole scheme to the security of the implementation of a single primitive, which should be strongly protected.
With sUL, we have modelled the security of the implementation of the strongly protected primitive, which we need. This assumption can be tested. We think that it is possible to build a sUL implementation of a (T)BC.

Finally, we have proposed only constructions for which we can prove their security. We have tried to do our proofs as correct, as complete and as precise as possible.

Bibliography

[1] Michel Abdalla, Sonia Belaïd, and Pierre-Alain Fouque. Leakage-resilient symmetric encryption via re-keying. In Guido Bertoni and Jean-Sébastien Coron, editors, *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, volume 8086 of *Lecture Notes in Computer Science*, pages 471–488. Springer, 2013.

[2] Michel Abdalla and Mihir Bellare. Increasing the lifetime of a key: A comparative analysis of the security of re-keying techniques. In Okamoto [108], pages 546–559.

[3] Farzaneh Abed, Francesco Berti, and Stefan Lucks. Insecurity of RCB: leakage-resilient authenticated encryption. *IACR Cryptology ePrint Archive*, 2016:1121, 2016.

[4] Farzaneh Abed, Francesco Berti, and Stefan Lucks. Is RCB a leakage resilient authenticated encryption scheme? In Helger Lipmaa, Aikaterini Mitrokotsa, and Raimundas Matulevicius, editors, *Secure IT Systems - 22nd Nordic Conference, NordSec 2017, Tartu, Estonia, November 8-10, 2017, Proceedings*, volume 10674 of *Lecture Notes in Computer Science*, pages 39–52. Springer, 2017.

[5] Megha Agrawal, Tarun Kumar Bansal, Donghoon Chang, Amit Kumar Chauhan, Seokhie Hong, Jinkeon Kang, and Somitra Kumar Sanadhya. Rcb: leakage-resilient authenticated encryption via re-keying. *The Journal of Supercomputing*, pages 1–26, 2016.

[6] Martin R. Albrecht and Kenneth G. Paterson. Lucky microseconds: A timing attack on amazon's s2n implementation of TLS. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference*

on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I, volume 9665 of *Lecture Notes in Computer Science*, pages 622–643. Springer, 2016.

[7] Elena Andreeva, Andrey Bogdanov, Atul Luykx, Bart Mennink, Nicky Mouha, and Kan Yasuda. How to securely release unverified plaintext in authenticated encryption. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014. Proceedings, Part I*, volume 8873 of *Lecture Notes in Computer Science*, pages 105–125. Springer, 2014.

[8] Elena Andreeva and Martijn Stam. The symbiosis between collision and preimage resistance. In Liqun Chen, editor, *Cryptography and Coding - 13th IMA International Conference, IMACC2011, Oxford, UK, December 12-15, 2011. Proceedings*, volume 7089 of *Lecture Notes in Computer Science*, pages 152–171. Springer, 2011.

[9] Tomer Ashur, Orr Dunkelman, and Atul Luykx. Boosting authenticated encryption robustness with minimal modifications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 3–33. Springer, 2017.

[10] Guy Barwell, Daniel P. Martin, Elisabeth Oswald, and Martijn Stam. Authenticated encryption in the face of protocol and side channel leakage. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology - ASIACRYPT 2017 - 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I*, volume 10624 of *Lecture Notes in Computer Science*, pages 693–723. Springer, 2017.

[11] Guy Barwell, Daniel Page, and Martijn Stam. Rogue decryption failures: Reconciling AE robustness notions. In Groth [64], pages 94–111.

[12] Lawrence Bassham, Çağdaş Çalık, Kerry McKay, and Meltem Sönmez Turan. Submission requirements and evaluation criteria for the lightweight cryptography standardization process. Technical

report, Technical report, US National Institute of Standards and Technology, 2018.

[13] Ali Galip Bayrak, Nikola Velickovic, Paolo Ienne, and Wayne Burleson. An architecture-independent instruction shuffler to protect against side-channel attacks. *TACO*, 8(4):20:1–20:19, 2012.

[14] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In Okamoto [108], pages 531–545.

[15] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993.*, pages 62–73. ACM, 1993.

[16] Mihir Bellare and Phillip Rogaway. Encode-then-encipher encryption: How to exploit nonces or redundancy in plaintexts for efficient cryptography. In Okamoto [108], pages 317–330.

[17] Mihir Bellare and Phillip Rogaway. The security of triple encryption and a framework for code-based game-playing proofs. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, volume 4004 of *Lecture Notes in Computer Science*, pages 409–426. Springer, 2006.

[18] Davide Bellizia, Francesco Berti, Olivier Bronchain, Gaëtan Cassiers, Sébastien Duval, Chun Guo, Gregor Leander, Gaëtan Leurent, Itamar Levi, Charles Momin, Olivier Pereira, Thomas Peters, François-Xavier Standaert, and Friedrich Wiemer. Spook: Sponge-based leakage-resilient authenticated encryption with a masked tweakable block cipher. *Submission to NIST Lightweight Cryptography*, 2019.

[19] Dan J Bernstein. Caesar call for submissions, final, january 27 2014.

[20] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Tedt, a leakage-resilient AEAD mode for high (physical) security applications. *IACR Cryptology ePrint Archive*, 2019:137, 2019.

[21] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Tedt, a leakage-resist AEAD mode for high physical security applications. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2020(1):256–320, 2020.

[22] Francesco Berti, Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Strong authenticity with leakage under weak and falsifiable physical assumptions. Cryptology ePrint Archive, Report 2019/1413, 2019. https://eprint.iacr.org/2019/1413.

[23] Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Leakage-resilient and misuse-resistant authenticated encryption. *IACR Cryptology ePrint Archive*, 2016:996, 2016.

[24] Francesco Berti, François Koeune, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Ciphertext integrity with misuse and leakage: Definition and efficient constructions with symmetric primitives. In Jong Kim, Gail-Joon Ahn, Seungjoo Kim, Yongdae Kim, Javier López, and Taesoo Kim, editors, *Proceedings of the 2018 on Asia Conference on Computer and Communications Security, AsiaCCS 2018, Incheon, Republic of Korea, June 04-08, 2018*, pages 37–50. ACM, 2018.

[25] Francesco Berti, Olivier Pereira, and Thomas Peters. Reconsidering generic composition: The tag-then-encrypt case. In Debrup Chakraborty and Tetsu Iwata, editors, *Progress in Cryptology - INDOCRYPT 2018 - 19th International Conference on Cryptology in India, New Delhi, India, December 9-12, 2018, Proceedings*, volume 11356 of *Lecture Notes in Computer Science*, pages 70–90. Springer, 2018.

[26] Francesco Berti, Olivier Pereira, and Thomas Peters. Reconsidering generic composition: the tag-then-encrypt case. *IACR Cryptology ePrint Archive*, 2018:991, 2018.

[27] Francesco Berti, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. On leakage-resilient authenticated encryption with decryption leakages. *IACR Trans. Symmetric Cryptol.*, 2017(3):271–293, 2017.

[28] Francesco Berti, Olivier Pereira, and François-Xavier Standaert. Reducing the cost of authenticity with leakages: a ciml2-secure

AE scheme with one call to a strongly protected tweakable block cipher. *IACR Cryptology ePrint Archive*, 2019:451, 2019.

[29] Francesco Berti, Olivier Pereira, and François-Xavier Standaert. Reducing the cost of authenticity with leakages: a \mathsf CIML2 -secure \mathsf AE scheme with one call to a strongly protected tweakable block cipher. In Johannes Buchmann, Abderrahmane Nitaj, and Tajje-eddine Rachidi, editors, *Progress in Cryptology - AFRICACRYPT 2019 - 11th International Conference on Cryptology in Africa, Rabat, Morocco, July 9-11, 2019, Proceedings*, volume 11627 of *Lecture Notes in Computer Science*, pages 229–249. Springer, 2019.

[30] Francesco Berti and François-Xavier Standaert. An analysis of the learning parity with noise assumption against fault attacks. In Kerstin Lemke-Rust and Michael Tunstall, editors, *Smart Card Research and Advanced Applications - 15th International Conference, CARDIS 2016, Cannes, France, November 7-9, 2016, Revised Selected Papers*, volume 10146 of *Lecture Notes in Computer Science*, pages 245–264. Springer, 2016.

[31] Guido Bertoni, Joan Daemen, Michaël Peeters, and Gilles Van Assche. On the indifferentiability of the sponge construction. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008, 27th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Istanbul, Turkey, April 13-17, 2008. Proceedings*, volume 4965 of *Lecture Notes in Computer Science*, pages 181–197. Springer, 2008.

[32] Guido Bertoni, Joan Daemen, Michaël Peters, Gilles Van Assche, and Ronny Van Keer. Caesar submission: Ketje v2. Technical report, 2016. https://keccak.team/files/Ketjev2-doc2.0.pdf.

[33] Alexandra Boldyreva, Jean Paul Degabriele, Kenneth G. Paterson, and Martijn Stam. On symmetric encryption with distinguishable decryption failures. In Moriai [103], pages 367–390.

[34] Dan Boneh, Özgür Dagdelen, Marc Fischlin, Anja Lehmann, Christian Schaffner, and Mark Zhandry. Random oracles in a quantum world. In Dong Hoon Lee and Xiaoyun Wang, editors, *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, volume 7073 of *Lecture Notes in Computer Science*, pages 41–69. Springer, 2011.

[35] Ran Canetti, Oded Goldreich, and Shai Halevi. The random oracle methodology, revisited. *J. ACM*, 51(4):557–594, 2004.

[36] Suresh Chari, Charanjit S. Jutla, Josyula R. Rao, and Pankaj Rohatgi. Towards sound approaches to counteract power-analysis attacks. In Wiener [129], pages 398–412.

[37] Christophe Clavier, Jean-Sébastien Coron, and Nora Dabbous. Differential power analysis in the presence of hardware countermeasures. In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2000, Second International Workshop, Worcester, MA, USA, August 17-18, 2000, Proceedings*, volume 1965 of *Lecture Notes in Computer Science*, pages 252–263. Springer, 2000.

[38] Benoît Cogliati and Yannick Seurin. EWCDM: an efficient, beyond-birthday secure, nonce-misuse resistant MAC. In Robshaw and Katz [117], pages 121–149.

[39] Jean-Sébastien Coron, Yevgeniy Dodis, Cécile Malinaud, and Prashant Puniya. Merkle-damgård revisited: How to construct a hash function. In Victor Shoup, editor, *Advances in Cryptology - CRYPTO 2005: 25th Annual International Cryptology Conference, Santa Barbara, California, USA, August 14-18, 2005, Proceedings*, volume 3621 of *Lecture Notes in Computer Science*, pages 430–448. Springer, 2005.

[40] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In David A. Wagner, editor, *Advances in Cryptology - CRYPTO 2008, 28th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2008. Proceedings*, volume 5157 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2008.

[41] Joan Daemen, Seth Hoffert, Michaël Peeters, Gilles Van Assche, and Ronny Van Keer. Xoodyak, a lightweight cryptographic scheme. *IACR Transactions on Symmetric Cryptology*, pages 60–87, 2020.

[42] Jean Paul Degabriele, Christian Janson, and Patrick Struck. Sponges resist leakage: The case of authenticated encryption. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on*

the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part II, volume 11922 of *Lecture Notes in Computer Science*, pages 209–240. Springer, 2019.

[43] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. Isap v2.0. *Submission to the NIST LWC Competition*, 2019.

[44] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, Bart Mennink, Robert Primas, and Thomas Unterluggauer. Submission to nist. *Submission to the NIST LWC Competition*, 2019.

[45] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP - authenticated encryption inherently secure against passive side-channel attacks. *IACR Cryptol. ePrint Arch.*, 2016:952, 2016.

[46] Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel, and Thomas Unterluggauer. ISAP - towards side-channel secure authenticated encryption. *IACR Trans. Symmetric Cryptol.*, 2017(1):80–105, 2017.

[47] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1. 2. *Submission to the CAESAR Competition*, 2016.

[48] Christoph Dobraunig, François Koeune, Stefan Mangard, Florian Mendel, and François-Xavier Standaert. Towards fresh and hybrid re-keying schemes with beyond birthday security. In Naofumi Homma and Marcel Medwed, editors, *Smart Card Research and Advanced Applications - 14th International Conference, CARDIS 2015, Bochum, Germany, November 4-6, 2015. Revised Selected Papers*, volume 9514 of *Lecture Notes in Computer Science*, pages 225–241. Springer, 2015.

[49] Christoph Dobraunig and Bart Mennink. Leakage resilience of the duplex construction. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, volume 11923 of *Lecture Notes in Computer Science*, pages 225–255. Springer, 2019.

[50] Christoph Dobraunig and Bart Mennink. Security of the suffix keyed sponge. *IACR Trans. Symmetric Cryptol.*, 2019(4):223–248, 2019.

[51] Yevgeniy Dodis and John P. Steinberger. Message authentication codes from unpredictable block ciphers. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009, 29th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings*, volume 5677 of *Lecture Notes in Computer Science*, pages 267–285. Springer, 2009.

[52] Yevgeniy Dodis and John P. Steinberger. Domain extension for macs beyond the birthday barrier. In Kenneth G. Paterson, editor, *Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, volume 6632 of *Lecture Notes in Computer Science*, pages 323–342. Springer, 2011.

[53] Alexandre Duc, Stefan Dziembowski, and Sebastian Faust. Unifying leakage models: From probing attacks to noisy leakage. In Nguyen and Oswald [105], pages 423–440.

[54] Alexandre Duc, Sebastian Faust, and François-Xavier Standaert. Making masking security proofs concrete - or how to evaluate the security of any leaking device. In Oswald and Fischlin [109], pages 401–429.

[55] Morris J Dworkin. Recommendation for block cipher modes of operation: Galois/counter mode (gcm) and gmac. Technical report, 2007.

[56] Stefan Dziembowski. Intrusion-resilience via the bounded-storage model. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography, Third Theory of Cryptography Conference, TCC 2006, New York, NY, USA, March 4-7, 2006, Proceedings*, volume 3876 of *Lecture Notes in Computer Science*, pages 207–224. Springer, 2006.

[57] Stefan Dziembowski, Sebastian Faust, Gottfried Herold, Anthony Journault, Daniel Masny, and François-Xavier Standaert. Towards sound fresh re-keying with hard (physical) learning problems. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology*

Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II, volume 9815 of *Lecture Notes in Computer Science*, pages 272–301. Springer, 2016.

[58] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 293–302. IEEE Computer Society, 2008.

[59] Sebastian Faust, Krzysztof Pietrzak, and Joachim Schipper. Practical leakage-resilient symmetric cryptography. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 213–232. Springer, 2012.

[60] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2001, Third International Workshop, Paris, France, May 14-16, 2001, Proceedings*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

[61] Louis Goubin and Jacques Patarin. DES and differential power analysis (the "duplication" method). In Çetin Kaya Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems, First International Workshop, CHES'99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, volume 1717 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 1999.

[62] Dahmun Goudarzi and Matthieu Rivain. How fast can higher-order masking be in software? In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part I*, volume 10210 of *Lecture Notes in Computer Science*, pages 567–597, 2017.

[63] Vincent Grosso. *Towards side-channel secure block ciphers.* PhD thesis, Catholic University of Louvain, Louvain-la-Neuve, Belgium, 2015.

[64] Jens Groth, editor. *Cryptography and Coding - 15th IMA International Conference, IMACC 2015, Oxford, UK, December 15-17,*

2015. Proceedings, volume 9496 of *Lecture Notes in Computer Science*. Springer, 2015.

[65] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Authenticated encryption with nonce misuse and physical leakage: Definitions, separation results and first construction - (extended abstract). In Peter Schwabe and Nicolas Thériault, editors, *Progress in Cryptology - LATINCRYPT 2019 - 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2-4, 2019, Proceedings*, volume 11774 of *Lecture Notes in Computer Science*, pages 150–172. Springer, 2019.

[66] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Towards lightweight side-channel security and the leakage-resilience of the duplex sponge. *IACR Cryptology ePrint Archive*, 2019:193, 2019.

[67] Chun Guo, Olivier Pereira, Thomas Peters, and François-Xavier Standaert. Towards low-energy leakage-resistant authenticated encryption from the duplex sponge construction. *IACR Trans. Symmetric Cryptol.*, 2020(1):6–42, 2020.

[68] Chun Guo, Yaobin Shen, Lei Wang, and Dawu Gu. Beyond-birthday secure domain-preserving prfs from a single permutation. *Des. Codes Cryptogr.*, 87(6):1297–1322, 2019.

[69] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. In Paul C. van Oorschot, editor, *Proceedings of the 17th USENIX Security Symposium, July 28-August 1, 2008, San Jose, CA, USA*, pages 45–60. USENIX Association, 2008.

[70] Christoph Herbst, Elisabeth Oswald, and Stefan Mangard. An AES smart card implementation resistant to power analysis attacks. In Jianying Zhou, Moti Yung, and Feng Bao, editors, *Applied Cryptography and Network Security, 4th International Conference, ACNS 2006, Singapore, June 6-9, 2006, Proceedings*, volume 3989 of *Lecture Notes in Computer Science*, pages 239–252, 2006.

[71] Shoichi Hirose. Some plausible constructions of double-block-length hash functions. In Matthew J. B. Robshaw, editor, *Fast*

Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers, volume 4047 of *Lecture Notes in Computer Science*, pages 210–225. Springer, 2006.

[72] Viet Tung Hoang, Ted Krovetz, and Phillip Rogaway. Robust authenticated-encryption AEZ and the problem that it solves. In Oswald and Fischlin [109], pages 15–44.

[73] Thomas Holenstein, Robin Künzler, and Stefano Tessaro. The equivalence of the random oracle model and the ideal cipher model, revisited. In Lance Fortnow and Salil P. Vadhan, editors, *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, San Jose, CA, USA, 6-8 June 2011*, pages 89–98. ACM, 2011.

[74] IETF. The transport layer security (tls) protocol version 1.3 draft-ietf-tls-tls13-28. Technical report, 2018. https://tools.ietf.org/html/draft-ietf-tls-tls13-28.

[75] Akiko Inoue, Tetsu Iwata, Kazuhiko Minematsu, and Bertram Poettering. Cryptanalysis of OCB2: attacks on authenticity and confidentiality. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 3–31. Springer, 2019.

[76] Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 2003, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[77] Anthony Journault and François-Xavier Standaert. Very high order masking: Efficient implementation and security evaluation. In Wieland Fischer and Naofumi Homma, editors, *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, volume 10529 of *Lecture Notes in Computer Science*, pages 623–643. Springer, 2017.

[78] Yael Tauman Kalai and Leonid Reyzin. A survey of leakage-resilient cryptography. *IACR Cryptology ePrint Archive*, 2019:302, 2019.

[79] Yael Tauman Kalai and Leonid Reyzin. A survey of leakage-resilient cryptography. In Oded Goldreich, editor, *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 727–794. ACM, 2019.

[80] Dina Kamel, François-Xavier Standaert, Alexandre Duc, Denis Flandre, and Francesco Berti. Learning with physical noise or errors. *IEEE Transactions on Dependable and Secure Computing*, 2018.

[81] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition.* CRC Press, 2014.

[82] Jonathan Katz and Moti Yung. Unforgeable encryption and chosen ciphertext secure modes of operation. In Bruce Schneier, editor, *Fast Software Encryption, 7th International Workshop, FSE 2000, New York, NY, USA, April 10-12, 2000, Proceedings*, volume 1978 of *Lecture Notes in Computer Science*, pages 284–299. Springer, 2000.

[83] Neal Koblitz and Alfred Menezes. Critical perspectives on provable security: Fifteen years of "another look" papers. *Adv. in Math. of Comm.*, 13(4):517–558, 2019.

[84] Neal Koblitz and Alfred J. Menezes. The random oracle model: a twenty-year retrospective. *Des. Codes Cryptogr.*, 77(2-3):587–610, 2015.

[85] Paul C Kocher. Leak-resistant cryptographic indexed key update, march 25 2003. *United States Patent*, 6.

[86] Paul C. Kocher. Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[87] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Wiener [129], pages 388–397.

[88] Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is ssl?). In Joe Kilian, editor, *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, volume 2139 of *Lecture Notes in Computer Science*, pages 310–331. Springer, 2001.

[89] Rodolphe Lampe and Yannick Seurin. Tweakable blockciphers with asymptotically optimal security. In Moriai [103], pages 133–151.

[90] Will Landecker, Thomas Shrimpton, and R. Seth Terashima. Tweakable blockciphers with beyond birthday-bound security. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 14–30. Springer, 2012.

[91] Itamar Levi, Davide Bellizia, and François-Xavier Standaert. Reducing a masked implementation's effective security order with setup manipulations and an explanation based on externally-amplified couplings. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2019(2):293–317, 2019.

[92] Moses Liskov, Ronald L. Rivest, and David A. Wagner. Tweakable block ciphers. *J. Cryptology*, 24(3):588–613, 2011.

[93] Jake Longo, Elke De Mulder, Dan Page, and Michael Tunstall. Soc it to EM: electromagnetic side-channel attacks on a complex system-on-chip. In Tim Güneysu and Helena Handschuh, editors, *Cryptographic Hardware and Embedded Systems - CHES 2015 - 17th International Workshop, Saint-Malo, France, September 13-16, 2015, Proceedings*, volume 9293 of *Lecture Notes in Computer Science*, pages 620–640. Springer, 2015.

[94] Stefan Mangard. Hardware countermeasures against DPA ? A statistical analysis of their effectiveness. In Tatsuaki Okamoto, editor, *Topics in Cryptology - CT-RSA 2004, The Cryptographers' Track at the RSA Conference 2004, San Francisco, CA, USA, February 23-27, 2004, Proceedings*, volume 2964 of *Lecture Notes in Computer Science*, pages 222–235. Springer, 2004.

[95] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Hiding*, pages 167–199. In [96], 2007.

[96] Stefan Mangard, Elisabeth Oswald, and Thomas Popp. *Power analysis attacks - revealing the secrets of smart cards*. Springer, 2007.

[97] Stefan Mangard, Elisabeth Oswald, and François-Xavier Standaert. One for all - all for one: unifying standard differential power analysis attacks. *IET Information Security*, 5(2):100–110, 2011.

[98] Daniel P. Martin, Elisabeth Oswald, Martijn Stam, and Marcin Wójcik. A leakage resilient MAC. In Groth [64], pages 295–310.

[99] David A. McGrew. An interface and algorithms for authenticated encryption. *RFC*, 5116:1–22, 2008.

[100] Marcel Medwed, François-Xavier Standaert, Johann Großschädl, and Francesco Regazzoni. Fresh re-keying: Security against side-channel and fault attacks for low-cost devices. In Daniel J. Bernstein and Tanja Lange, editors, *Progress in Cryptology - AFRICACRYPT 2010, Third International Conference on Cryptology in Africa, Stellenbosch, South Africa, May 3-6, 2010. Proceedings*, volume 6055 of *Lecture Notes in Computer Science*, pages 279–296. Springer, 2010.

[101] Bart Mennink. Optimally secure tweakable blockciphers. In Gregor Leander, editor, *Fast Software Encryption - 22nd International Workshop, FSE 2015, Istanbul, Turkey, March 8-11, 2015, Revised Selected Papers*, volume 9054 of *Lecture Notes in Computer Science*, pages 428–448. Springer, 2015.

[102] Bart Mennink and Samuel Neves. Optimal prfs from blockcipher designs. *IACR Trans. Symmetric Cryptol.*, 2017(3):228–252, 2017.

[103] Shiho Moriai, editor. *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*. Springer, 2014.

[104] Chanathip Namprempre, Phillip Rogaway, and Thomas Shrimpton. Reconsidering generic composition. In Nguyen and Oswald [105], pages 257–274.

[105] Phong Q. Nguyen and Elisabeth Oswald, editors. *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*. Springer, 2014.

[106] Jesper Buus Nielsen. Separating random oracle proofs from complexity theoretic proofs: The non-committing encryption case. In Moti Yung, editor, *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126. Springer, 2002.

[107] Yoav Nir and Adam Langley. Chacha20 and poly1305 for IETF protocols. *RFC*, 7539:1–45, 2015.

[108] Tatsuaki Okamoto, editor. *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, volume 1976 of *Lecture Notes in Computer Science*. Springer, 2000.

[109] Elisabeth Oswald and Marc Fischlin, editors. *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*. Springer, 2015.

[110] Elisabeth Oswald and François-Xavier Standaert. Side-channel analysis and its relevance to fault attacks. In Marc Joye and Michael Tunstall, editors, *Fault Analysis in Cryptography*, Information Security and Cryptography, pages 3–15. Springer, 2012.

[111] Olivier Pereira, François-Xavier Standaert, and Srinivas Vivek. Leakage-resilient authentication and encryption from symmetric cryptographic primitives. In Indrajit Ray, Ninghui Li, and Christopher Kruegel, editors, *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, CO, USA, October 12-16, 2015*, pages 96–108. ACM, 2015.

[112] Thomas Peyrin and Yannick Seurin. Counter-in-tweak: Authenticated encryption modes for tweakable block ciphers. In Robshaw and Katz [117], pages 33–63.

[113] Krzysztof Pietrzak. A leakage-resilient mode of operation. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*, pages 462–482. Springer, 2009.

[114] Emmanuel Prouff and Matthieu Rivain. Masking against side-channel attacks: A formal security proof. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology - EUROCRYPT 2013, 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings*, volume 7881 of *Lecture Notes in Computer Science*, pages 142–159. Springer, 2013.

[115] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): measures and counter-measures for smart cards. In Isabelle Attali and Thomas P. Jensen, editors, *Smart Card Programming and Security, International Conference on Research in Smart Cards, E-smart 2001, Cannes, France, September 19-21, 2001, Proceedings*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.

[116] Matthieu Rivain, Emmanuel Prouff, and Julien Doget. Higher-order masking and shuffling for software implementations of block ciphers. In Christophe Clavier and Kris Gaj, editors, *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, volume 5747 of *Lecture Notes in Computer Science*, pages 171–188. Springer, 2009.

[117] Matthew Robshaw and Jonathan Katz, editors. *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I*, volume 9814 of *Lecture Notes in Computer Science*. Springer, 2016.

[118] Phillip Rogaway and Thomas Shrimpton. Cryptographic hash-function basics: Definitions, implications, and separations for preimage resistance, second-preimage resistance, and collision resistance. In Bimal K. Roy and Willi Meier, editors, *Fast Software Encryption, 11th International Workshop, FSE 2004, Delhi, India, February 5-7, 2004, Revised Papers*, volume 3017 of *Lecture Notes in Computer Science*, pages 371–388. Springer, 2004.

[119] Phillip Rogaway and Thomas Shrimpton. Deterministic authenticated-encryption: A provable-security treatment of the key-wrap problem. *IACR Cryptology ePrint Archive*, 2006:221, 2006.

[120] Eyal Ronen, Adi Shamir, Achi-Or Weingarten, and Colin O'Flynn. Iot goes nuclear: Creating a zigbee chain reaction. *IEEE Security & Privacy*, 16(1):54–62, 2018.

[121] Kai Schramm and Christof Paar. Higher order masking of the AES. In David Pointcheval, editor, *Topics in Cryptology - CT-RSA 2006, The Cryptographers' Track at the RSA Conference 2006, San Jose, CA, USA, February 13-17, 2006, Proceedings*, volume 3860 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2006.

[122] Peter Schwabe and Ko Stoffelen. All the AES you need on cortex-m3 and M4. In Roberto Avanzi and Howard M. Heys, editors, *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*, volume 10532 of *Lecture Notes in Computer Science*, pages 180–194. Springer, 2016.

[123] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.

[124] François-Xavier Standaert, Olivier Pereira, and Yu Yu. Leakage-resilient symmetric cryptography under empirically verifiable assumptions. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 335–352. Springer, 2013.

[125] François-Xavier Standaert, Nicolas Veyrat-Charvillon, Elisabeth Oswald, Benedikt Gierlichs, Marcel Medwed, Markus Kasper, and Stefan Mangard. The world is not enough: Another look on second-order DPA. In Masayuki Abe, editor, *Advances in Cryptology - ASIACRYPT 2010 - 16th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 5-9, 2010. Proceedings*, volume 6477 of *Lecture Notes in Computer Science*, pages 112–129. Springer, 2010.

[126] Kazuhiro Suzuki, Dongvu Tonien, Kaoru Kurosawa, and Koji Toyota. Birthday paradox for multi-collisions. In Min Surp Rhee and Byoungcheon Lee, editors, *Information Security and Cryptology - ICISC 2006, 9th International Conference, Busan, Korea, November 30 - December 1, 2006, Proceedings*, volume 4296 of *Lecture Notes in Computer Science*, pages 29–40. Springer, 2006.

[127] Nicolas Veyrat-Charvillon, Marcel Medwed, Stéphanie Kerckhof, and François-Xavier Standaert. Shuffling against side-channel attacks: A comprehensive study with cautionary note. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 740–757. Springer, 2012.

[128] Weijia Wang, Yu Yu, François-Xavier Standaert, Dawu Gu, Sen Xu, and Chi Zhang. Ridge-based profiled differential power analysis. In Helena Handschuh, editor, *Topics in Cryptology - CT-RSA 2017 - The Cryptographers' Track at the RSA Conference 2017, San Francisco, CA, USA, February 14-17, 2017, Proceedings*, volume 10159 of *Lecture Notes in Computer Science*, pages 347–362. Springer, 2017.

[129] Michael J. Wiener, editor. *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*. Springer, 1999.

[130] Peter H. Wright and Paul Greengrass. *Spycatcher: The Candid Autobiography of a Senior Intelligence Officer*. Heinemann, 1987.

[131] Yu Yu and François-Xavier Standaert. Practical leakage-resilient pseudorandom objects with minimum public randomness. In Ed Dawson, editor, *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings*, volume 7779 of *Lecture Notes in Computer Science*, pages 223–238. Springer, 2013.

[132] Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, Chicago, Illinois, USA, October 4-8, 2010*, pages 141–151. ACM, 2010.

Chapter A
Additional definitions

In this appendix, we study how a probabilistic encryption algorithm can be built from a deterministic one, using an additional input. Then, we study what security we can achieve if this additional input is not correctly used.

A.1 Syntactic definitions for (AE)

As we have explained in Chap. 2.6, it is necessary to have a probabilistic encryption scheme. However, it is not easy to design a probabilistic algorithm.

IV-based encryption schemes. A standard solution is to use a deterministic algorithm that uses an additional input called the Initialization Vector (IV). This IV is supposed to be random and is externally provided.
When we state security definitions for IV-based encryption schemes, we ask that the IV are be picked uniformly at random from the IV-space, during encryption queries (for example, see [119]).

Nonce-based encryption schemes In many cases, it is an overkill to suppose that the IV is picked uniformly at random. In fact, often, it is possible to prove the security of many schemes, only supposing that the IV is not repeated in different encryption queries. In this case, the IV is called *nonce*. We can formalize the syntax for *nonce-based Authenticated Encryption scheme* as follow:

Definition 29. *A scheme for* nonce-based Authenticated Encryption *(*nAE*) is a triple* $\Pi := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *of algorithms, where*

- Gen *picks a key* k *in the keyspace* $\mathcal{K}$*, which is not empty.*
- *The encryption algorithm* Enc *is a deterministic algorithm which takes as input the triple* $(k, N, m) \in \mathcal{K} \times \mathcal{N} \times \mathcal{ME}$ *and outputs a string* $c \leftarrow \mathsf{Enc}_k^N(m)$ *called ciphertext.*
- *The decryption algorithm* Dec *is a deterministic algorithm which takes as input the tuple* $(k, N, c) \in \mathcal{K} \times \mathcal{N} \times \{0,1\}^*$ *and outputs* $m \leftarrow \mathsf{Dec}_k^N(c)$ *which is either a string* $m \in \mathcal{ME}$ *or the symbol "*$\perp$*" ("invalid").*

We require that the algorithms Enc *and* Dec *are the inverse of each other, that is:*

- (Correctness) *if* $\mathsf{Enc}_k(N, m) = c$*, then,* $\mathsf{Dec}_k(N, c) = m$
- (Tidiness) *if* $\mathsf{Dec}_k(N, c) = m \neq \perp$*, then,* $\mathsf{Enc}_k(N, m) = c$.

If $\mathsf{Dec}_k(N, c) = \perp$ *we say that the algorithm* "rejects" c*, otherwise it* "accepts" c.

In the literature there are some schemes which do no respect the tidiness condition. They are called sloppy schemes.

For the security, we have to simply adapt Def. 17 to the nAE syntax and the constraint on nonces:

Definition 30. *A nonce-based authenticated encryption scheme* (nAE) $\Pi := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* (q_E, q_D, t, ϵ)-nAE-secure *if the advantage*

$$\mathsf{Adv}_{\Pi}^{\mathsf{nAE}}(\mathsf{A}) := \left| \Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot,\cdot),\mathsf{Dec}_k(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1\right] \right| \quad \text{(A.1)}$$

is bounded by ϵ *for every* (q_E, q_D, t)*-adversary* A *that respects the following two conditions:*

(i) If A *queried the first (encryption) oracle on input* (N, m) *and was answered* c*, then he is not allowed to query the second (decryption) oracle on input* (N, c)*;*

(i) A *is not allowed to repeat the first component (the nonce) on different first oracle queries (encryption queries).*

This notion provides privacy, since it assumes that ciphertexts are indistinguishable from random, and authenticity, since it assumes decryption queries made by the adversary are not valid.

A.2 Misuse-resistance

The previous security definition does not say anything about the case when a nonce is repeated. In some case, it is enough to repeat once a nonce to completely break the security of the scheme [9].
If a nonce N is repeated with the same message m in two different encryption query, the resulting ciphertext will be the same.
The following definition assumes that an adversary cannot give any additional information from the repetition of the nonces is if they are repeated with the same message m.

Definition 31. *[119] A nonce-based authenticated encryption scheme* (nAE) $\Pi := (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ *is* (q_E, q_D, t, ϵ)-MRAE-secure (misuse resistant)*, if the advantage*

$$\mathsf{Adv}^{\mathsf{MRAE}}_{\Pi}(\mathsf{A}) := \left|\Pr\left[\mathsf{A}^{\mathsf{Enc}_k(\cdot,\cdot),\mathsf{Dec}_k(\cdot,\cdot)} \Rightarrow 1\right] - \Pr\left[\mathsf{A}^{\$(\cdot,\cdot),\perp(\cdot,\cdot)} \Rightarrow 1\right]\right|$$

is bounded by ϵ *for every* (q_E, q_D, t)*-adversary* A *.*

Note that this is Def.30 (nAE-security) where we have omitted the constraint that nonces are not repeated in encrytption queries (queries to the first oracle).

There is a flourishing literature about them, for example, [119, 112, 65]. In our works, we studied this in [24, 27].

Misuse-resilience There is a weaker notion of security due to Ashur et al. [9], which states the if a nonce is repeated, we have security problems only for that nonce, not for nonces never used or used only once.

Chapter B
Detailed proofs

Contents

B.1 Proof of the suf-L security of HBC

Theorem 3. *Let* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \{0,1\}^n$ *be a* $(t_2, \epsilon_{\mathsf{CR}})$*-collision resistant hash function. Let* $\mathsf{F}^* : \mathcal{K}^* \times \{0,1\}^n \to \{0,1\}^n$ *be a* $(q, t_1, \epsilon_{\mathsf{PRF}})$*-pseudorandom function (*PRF*). Let* $\mathcal{HM} = \{0,1\}^*$.
Then, HBC *is* (q_M, q_V, t, ϵ)-suf-L*-secure in the unbounded leakage model with*

$$\epsilon \leq \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{CR}} + (q_V + 1)2^{-n}$$

with $q = q_M + q_V + 1$, $t_1 = t + t_{\mathsf{ch}(1,\mathcal{KH})} + q' t_{\mathsf{H}}$ *and* $t_2 = t + q t_{\mathsf{H}} + t_{\mathsf{f}^*(q')}$.

Proof. As usual, we use a sequence of games.
Game 0. Let Game 0 be the suf-L-game where the (q_M, q_V, t)-adversary A^{L} tries to forge '$MACB$. Let E_0 be the event that the adversary wins this game, i.e., that the output of the game is 1.

Game 1. Let Game 1 be Game 0, where we have replaced the PRF F^* with a random function, named f^*. Let E_1 be the event that the adversary wins this game.

Transition between Game 0 and Game 1. To bound the difference $\Pr[E_0] - \Pr[E_1]$ we build a $(q+1, t_1)$-PRF adversary B against F^* based on A.

The $(q+1, t_1)$-PRF adversary B. B has to distinguish if he is interacting against an oracle implemented with the PRF F^* or with the random function f^*. To do this, he uses the suf-L-adversary A.
At the start of the game, B picks a key s for the hash function H uniformly at random in $\mathcal{KH}$. Then, he relays it to A^{L}. Moreover, he picks a list $\mathcal{S}$, which is empty at the start. This takes time $t_{\mathsf{ch}(1,\mathcal{KH})}$.
When A does a tag-generation query on input m^i, B (1) simply computes $h^i = \mathsf{H}_s(m^i)$, then, (2) he calls his oracle on input h^i, receiving τ^i, Finally (3), he answers A τ^i and he adds $\{(m^i, \tau^i)\}$ to $\mathcal{S}$.
This takes time t_{H}. Moreover, 1 oracle query is needed.
When A asks a verification query on input (m^i, τ^i), B first (1), he computes $h^i = \mathsf{H}_s(m^i)$ and he calls his oracle on input h^i, obtaining $\tilde{\tau}^i$. Finally (2), if $\tau^i = \tilde{\tau}^i$, B answers $\top$; otherwise $\bot$. This takes time t_{H}. Moreover, one oracle query is needed.
When A asks outputs its forgery $(m^{q_D+1}, \tau^{q_D+1})$, B first (1), he computes $h^{q_V+1} = \mathsf{H}_s(m^{q_V+1})$ and he calls his oracle on input h^{q_V+1}, obtaining $\tilde{\tau}^{q_V+1}$. Finally (2), if $\tau^{q_V+1} = \tilde{\tau}^{q_V+1}$ and $(m^{q_V+1}, \tau^{q_V+1)} \notin \mathcal{S}$, B outputs 1; otherwise 0.
This takes time t_{H}. Moreover, one oracle queries are needed.
Thus, B does at most $q_M + q_V + 1 = q + 1$ queries to his oracle. He needs time $t + t_{\mathsf{ch}(1,\mathcal{KH})} + (q+1)t_{\mathsf{H}} \leq t_1$.

Bounding $|\Pr[E_0] - \Pr[E_1]|$. Clearly if the oracle B faces is implemented with $\mathsf{F}^*_k(\cdot)$, he correctly simulates Game 0 for A otherwise Game 1. Thus,

$$|\Pr[E_0] - \Pr[E_1]| = |\Pr[\mathsf{B}^{\mathsf{F}^*_k(\cdot)} \Rightarrow 1] - \Pr[\mathsf{B}^{\mathsf{f}^*(\cdot)} \Rightarrow 1]| \leq \epsilon_{\mathsf{PRF}}$$

where the last inequality is due to the fact that B is a $(q+1, t_1)$-adversary and F^* is a $(q+1, t_1, \epsilon_{\mathsf{PRF}})$-PRF.

Game 2. Let Game 2 be Game 1, where we suppose that there are no collisions for the hash function. Let E_2 be the event that A wins this game.

Transition between Game 1 and Game 2. Clearly, Game 1 and Game 2 are identical if the following event HC (*Hash collision*) does not happen:

$$HC := \left\{ \begin{array}{c} \exists i, j \in \{1, ..., q_E\} \cup \{1, ..., q_V + 1\} \text{ with } i \overset{\%}{=} j \\ \text{s.t. } h^i = h^j \text{ and } r^i \| m^i \neq r^j \| m^j \end{array} \right\}.$$ [1]

[1] $i \overset{\%}{=} j$ means that if i comes from a tag-generation query and j from a verification query, or viceversa, then, they are considered differently.

To compute this event, we build a t_2-CR-adversary C.

The t_2-CR-adversary C. C has to find a collision for the hash function H_s and he is based on A^{L}. At the start of the game, he is provided a key s for the hash function, which he relays to A^{L}. Moreover, he has a list $\mathcal{H}$ which is empty at the start.
When A does a tag-generation query on input m^i, C (1) simply computes $h^i = \mathsf{H}_s(m^i)$, then, (2) he lazy samples $\tau^i = \mathsf{f}^*(h^i)$, receiving τ^i. Finally (3), he answers τ^i to A and he adds $\{(m^i, h^i)\}$ to $\mathcal{H}$.
This takes time t_{H} and the time needed to lazy sample f^* once.
When A asks a verification query on input (m^i, τ^i), C first (1), he computes $h^i = \mathsf{H}_s(m^i)$, adds $\{(m^i, h^i)\}$ to $\mathcal{H}$, and he lazy samples $\tilde{\tau}^i = \mathsf{f}^*(h^i)$. Finally (2), if $\tau^i = \tilde{\tau}^i$, C answers $\top$; otherwise $\bot$. This takes time t_{H} and the time needed to lazy sample f^* once.
When A asks outputs its forgery $(m^{q_V+1}, \tau^{q_D+1})$, C first (1), he computes $h^{q_V+1} = \mathsf{H}_s(m^{q_V+1})$, adds $\{(m^{q_V+1}, h^{q_V+1})\}$ to $\mathcal{H}$, and he lazy samples $\tilde{\tau}^{q_V+1} = \mathsf{f}^*(h^{q_V+1})$.
This takes time t_{H} and the time needed to lazy sample f^* once.
At the end of the game, he looks up the list $\mathcal{S}$ to find a collision. If he finds it, he outputs it; otherwise, 0^n and 1^n.
Thus, in total, he needs to lazy sample f^* $(q+1)$ times. Moreover, he needs time $t + (q+1)t_{\mathsf{H}} + t_{\mathsf{f}^*(q+1)} \leq t_2$.

Bounding $|\Pr[E_1] - \Pr[E_2]|$. Observe that if event HC happens, C wins. Since C is a t_2-adversaries and H is a $(t_2, \epsilon_{\mathsf{CR}})$-collision resistant hash function, we can bound:

$$|\Pr[E_1] - \Pr[E_2]| \leq \epsilon_{\mathsf{CR}}.$$

Game 3. In Game 3 we suppose that all verification queries are deemed invalid (if the couple (m^i, h^i) is fresh).

Transition between Game 2 and 3. We build a sequence of $q_V + 2$ games Game 2^i $i = 0, ..., q_V + 1$. In Game 2^i, for the first i decryption queries, if h^i is fresh, then, they are invalid. That is, $\tau^i \neq \mathsf{f}^*(h^i)$. Thus, all the first i verification queries are invalid if fresh. Let E_2^i be the event that the adversary wins in Game 3^i.

Bounding $|\Pr[E_2^{i-1}] - \Pr[E_2^i]|$. Note that the difference between Game 2^{i-1} and Game 2^i are equal if the ith verification query is not both fresh and valid. Thus:

$$|\Pr[E_2^{i-1}] - \Pr[E_2^i]| = \Pr[(m^i, \tau^i) \text{ fresh and valid}].$$

Since we are in Game 2, event CR has not happened. Thus, the only possibility is that $\mathsf{f}^*(h^i)$ has never been computed with $h^i = \mathsf{H}_s(|m^i)$. Thus

$$|\Pr[E_2^{i-1}] - \Pr[E_2^i]| \leq 2^{-n}$$

since $\mathsf{f}^*(h^i)$ is picked uniformly at random. Thus, the probability that $\mathsf{f}^*(h^i) = \tau^i$ is 2^{-n}.

Bounding $|\Pr[E_2] - \Pr[E_3]|$. Since Game 2 is Game 2^0 and Game 3 is Game 2^{q_V+1} we can bound

$$|\Pr[E_2] - \Pr[E_3]| \leq \sum_{i=1}^{q_V+1} |\Pr[E_2^{i-1}] - \Pr[E_2^i]| \leq (q_V+1)2^{-n}.$$

Thus, putting everything together, we can conclude:
Bounding $\Pr[E_0]$. Since $\Pr[E_3] = 0$ (since, in Game 3, all verification queries are invalid (if fresh)), we can conclude:

$$\Pr[E_0] = \Pr[E_0] - \Pr[E_3] = \sum_{i=1}^{3} |\Pr[E_{i-1}] - \Pr[E_i]| \leq \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{CR}} + (q_V+1)2^{-n}.$$

□

B.2 Proof of the CIML-security of DTE

Theorem 4. *Let $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \{0,1\}^n$ be a $(t_2, \epsilon_{\mathsf{CR}})$-collision resistant and $(t_2, \epsilon_{\mathsf{roPR}})$-range-oriented preimage resistant hash function. Let $\mathsf{F}^* : \mathcal{K}^* \times \{0,1\}^n \to \{0,1\}^n$ be a $(2q, t_1, \epsilon_{\mathsf{PRF}})$-pseudorandom function. Let $\mathsf{F} : \mathcal{K} \times \mathcal{B} \to \mathcal{B}$. Let $\mathcal{HM} = \{0,1\}^*$. Let L be the maximal number of blocks for a message.*
Then, DTE is (q_E, q_D, t, ϵ)-CIML-secure in the unbounded leakage model with

$$\epsilon \leq \epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{CR}} + q\epsilon_{\mathsf{roPR}} + (q_D+1)2^{-n}$$

with $q = q_E + q_D + 1$ and

$$t_1 = t + t_{\mathsf{ch}(1,\mathcal{KH})} + t_{\mathsf{chn}(2,\mathcal{B})} + (q+1)(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}), \text{ and}$$

$$t_2 = t + t_{\mathsf{chn}(2,\mathcal{B})} + (q+1)(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}) + t_{\mathsf{f}^*(2(q+1))}.$$

Proof. As usual, we use a sequence of games.
Game 0. Let Game 0 be the CIML-game where the (q_E, q_D, t)-adversary A^{L} tries to produce a valid and fresh ciphertext when he plays against DTE. Let E_0 be the event that the adversary wins this game, i.e., that

the output of the game is 1.

Game 1. Let Game 1 be Game 0, where we have replaced the PRF F^* with a random function, named f^*. Let E_1 be the event that the adversary wins this game.

Transition between Game 0 and Game 1. To bound the difference $\Pr[E_0] - \Pr[E_1]$ we build a $(2(q+1), t_1)$-PRF adversary B against F^* based on A.

The $(2(q+1), t_1)$-PRF adversary B. B has to distinguish if he is interacting against an oracle implemented with the PRF F^* or with the random function f^*. To do this, he uses the CIML-adversary A.
At the start of the game, B picks two random values p_A, p_B uniformly at random in $\mathcal{B}$ with $p_A \neq p_B$ and a key s for the hash function H uniformly at random in $\mathcal{KH}$. Then, he relays them to A^{L}. Moreover, he picks a list $\mathcal{S}$, which is empty at the start. This takes time $t_{\mathsf{ch}(1,\mathcal{KH})} + t_{\mathsf{chn}(2,\mathcal{B})}$.
When A does an encryption query on input (r^i, m^i), B (1) simply computes $h^i = \mathsf{H}_s(r^i \| m^i)$ and he parses $m^i = (m^i_1, ..., m^i_{l^i})$, then, (2) he calls his oracle on input h^i, receiving τ^i, and (3) he calls again his oracle on input τ^i obtaining k^i_0. From k^i_0, he (4) computes $c^i_0 = \mathsf{F}_{k^i_0}(p_B) \oplus r^i$, after that, for every $i' \in [1, l^i]$, he computes $k^i_{i'} = \mathsf{F}_{k^i_{i'-1}}(p_A)$, $y_{i'} = \mathsf{F}_{k^i_{i'}}(p_B)$ and $c^i_{i'} = y^i_{i'} \oplus m^i_{i'}$ [2]. Finally (5), he answers A $c^i = (\tau^i, C^i)$ with $C^i = (c^i_0, c^i_1, ..., c^i_{l^i})$, and the leakage k^i_0 and he adds $\{c^i\}$ to $\mathcal{S}$.
This takes time $t_{\mathsf{H}} + (2l^i + 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}$. Moreover, 2 oracle queries are needed.
When A asks a decryption query on input $c^i = (\tau^i, C^i)$, B first (1) queries his oracle on input τ^i obtaining k^i_0 and he parses
$C^i = (c_0 `i, c^i_1, ..., c^i_{l^i})$. From k^i_0, he (2) computes $r^i = \mathsf{F}_{k^i_0}(p_B) \oplus c^i_0$, after that, for every $i' \in [1, l^i]$, he computes $k^i_{i'} = \mathsf{F}_{k^i_{i'-1}}(p_A)$, $y^i_{i'} = \mathsf{F}_{k^i_{i'}}(p_B)$ and $m^i_{i'} = y^i_{i'} \oplus c^i_{i'}$ [3]. Then (3), he computes $h^i = \mathsf{H}_s(r^i \| m^i)$ and he calls his oracle on input h^i, obtaining $\tilde{\tau}^i$. Finally (4), if $\tau^i = \tilde{\tau}^i$, B answers $m^i = (m^i_1, ..., m^i_{l^i})$ to A; otherwise, $\perp$. This takes time $t_{\mathsf{H}} + (2l^i + 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}$. Moreover, 2 oracle queries are needed.
When A asks outputs its forgery $c^{q_V+1} = (\tau^{q_V+1}, C^{q_V+1})$, B first (1) queries his oracle on input τ^{q_V+1} obtaining $k^{q_V+1}_0$ and he parses
$C^{q_V+1} = (c^{q_V+1}_0, c^{q_V+1}_1, ..., c^{q_V+1}_{l^{q_V+1}})$. From $k^{q_V+1}_0$, he (2) computes
$r^{q_V+1} = \mathsf{F}_{k^{q_V+1}_0}(p_B) \oplus c^{q_V+1}_0$, after that, for every $i' \in [1, l^i]$, he computes

[2] For $i' = l^i$, $c^i_{i'} = \pi_{|m^i_{i'}|}(y_{i'})^i \oplus m^i_{i'}$
[3] For $i' = l^i$, $m^i_{i'} = \pi_{|m^i_{i'}|}(y_{i'})^i \oplus c^i_{i'}$

$k_{i'}^{q_V+1} = \mathsf{F}_{k_{i'-1}^{q_V+1}}(p_A)$, $y_{i'}^{q_V+1} = \mathsf{F}_{k_{i'}^{q_V+1}}(p_B)$ and $m_{i'}^{q_V+1} = y_{i'}^{q_V+1} \oplus c_{i'}^{q_V+1}$ [4]. Then (3), he computes $h^{q_D+1} = \mathsf{H}_s(r^{q_D+1} \| m^{q_D+1})$ and he calls his oracle on input h^{q_D+1}, obtaining $\tilde{\tau}^{q_D+1}$. Finally (4), if $\tau^{q_D+1} = \tilde{\tau}^{q_D+1}$ and $c^{q_D+1} \notin \mathcal{S}$, B outputs 1; otherwise 0.
This takes time $t_{\mathsf{H}} + (2l^i + 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}$. Moreover, 2 oracle queries are needed.
Thus, B does at most $2(q_E + q_D + 1) = 2(q+1)$ queries to his oracle. He needs time $t + t_{\mathsf{chn}(1,\mathcal{KH})} + t_{\mathsf{chn}(2,\mathcal{B})} + (q+1)(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}) \leq t_1$.

Bounding $|\Pr[E_0] - \Pr[E_1]|$**.** Clearly if the oracle B faces is implemented with $\mathsf{F}_k^*(\cdot)$, he correctly simulates Game 0 for A otherwise Game 1. Thus,

$$|\Pr[E_0] - \Pr[E_1]| = |\Pr[\mathsf{B}^{\mathsf{F}_k^*(\cdot)} \Rightarrow 1] - \Pr[\mathsf{B}^{\mathsf{f}^*(\cdot)} \Rightarrow 1]| \leq \epsilon_{\mathsf{PRF}}$$

where the last inequality is due to the fact that B is a $(2(q+1), t_1)$-adversary and F^* is a $(2(q+1), t_1, \epsilon_{\mathsf{PRF}})$-$\mathsf{PRF}$.

Game 2. Let Game 2 be Game 1, where we suppose that there are no collisions for the hash function. Let E_2 be the event that A wins this game.

Transition between Game 1 and Game 2. Clearly, Game 1 and Game 2 are identical if the following event HC (*Hash collision*) does not happen:

$$HC := \left\{ \begin{array}{c} \exists i, j \in \{1, ..., q_E\} \cup \{1, ..., q_D + 1\} \text{ with } i \overset{\%}{=} j \\ \text{s.t. } h^i = h^j \text{ and } r^i \| m^i \neq r^j \| m^j \end{array} \right\}.$$ [5]

To compute this event, we build a t_2-CR-adversary C.

The t_2-CR-adversary C. C has to find a collision for the hash function H_s and he is based on A^{L}. At the start of the game, he is provided a key s for the hash function. Moreover, he picks two values $p_A, p_B \in \mathcal{B}$ with $p_A \neq p_B$. Then, he relays s, p_A and p_B to A^{L}. Moreover, he has a list $\mathcal{H}$, which is empty at the start. This takes time $t_{\mathsf{chn}(2,\mathcal{B})}$.
When A does an encryption query on input (r^i, m^i), C (1) simply computes $h^i = \mathsf{H}_s(r^i \| m^i)$ and he parses $m^i = (m_1^i, ..., m_{l^i}^i)$, then, (2) he lazy samples $\tau^i = \mathsf{f}^*(h^i)$, receiving τ^i, and (3) he lazy samples $k_0^i = \mathsf{f}^*(\tau^i)$. From k_0^i, he (4) computes $c_0^i = \mathsf{F}_{k_0^i}(p_B) \oplus r^i$, after that, for every

[4] For $i' = l^{q_D+1}$, $m_{i'}^{q_V+1} = \pi_{|m_{i'}^{q_V+1}|}(y_{i'})^{q_V+1} \oplus c_{i'}^{q_D+1}$

[5] $i \overset{\%}{=} j$ means that if i comes from an encryption query and j from a decryption query, or viceversa, then, they are considered differently.

$i' \in [1, l^i]$, he computes $k^i_{i'} = \mathsf{F}_{k^i_{i'-1}}(p_A)$, $y_{i'} = \mathsf{F}_{k^i_{i'}}(p_B)$ and $c^i_{i'} = y^i_{i'} \oplus m^i_{i'}$ [6]. Finally (5), he answers to A $c^i = (\tau^i, C^i)$ with $C^i = (c^i_0, c^i_1, ..., c^i_{l^i})$, and the leakage k^i_0 and he adds $\{(r^i \| m^i, h^i)\}$ to $\mathcal{H}$.
This takes time $t_{\mathsf{H}} + (2l^i + 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L + 1)t_{\mathsf{F}}$ and the time needed to lazy sample f^* twice.
When A asks a decryption query on input $c^i = (\tau^i, C^i)$, C first (1) lazy samples $k^i_0 = \mathsf{f}^*(\tau^i)$ and he parses $c^i = (c^i_0, c^i_1, ..., c^i_{l^i})$. From k^i_0, he (2) computes $r^i = \mathsf{F}_{k^i_0}(p_B) \oplus c^i_0$, after that, for every $i' \in [1, l^i]$, he computes $k^i_{i'} = \mathsf{F}_{k^i_{i'-1}}(p_A)$, $y^i_{i'} = \mathsf{F}_{k^i_{i'}}(p_B)$ and $m^i_{i'} = y^i_{i'} \oplus c^i_{i'}$ [7]. Then (3), he computes $h^i = \mathsf{H}_s(r^i \| m^i)$, adds $\{(r^i \| m^i, h^i)\}$ to $\mathcal{H}$, and he lazy samples $\tilde{\tau}^i = \mathsf{f}^*(h^i)$. Finally (4), if $\tau^i = \tilde{\tau}^i$, C answers $m^i = (m^i_1, ..., m^i_{l^i})$ to A; otherwise, $\perp$. This takes time $t_{\mathsf{H}} + (2l^i + 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L + 1)t_{\mathsf{F}}$ and the time needed to lazy sample f^* twice.
When A asks outputs its forgery $c^{q_D+1} = (\tau^{q_D+1}, C^{q_D+1})$, C first (1) lazy samples $k^{q_D+1}_0 = \mathsf{f}^*(\tau^{q_D+1})$ and he parses
$c^{q_D+1} = (c^{q_D+1}_0, c^{q_D+1}_1, ..., c^{q_D+1}_{l^{q_D+1}})$. From $k^{q_D+1}_0$, he (2) computes $r^{q_D+1} = \mathsf{F}_{k^{q_D+1}_0}(p_B) \oplus c^{q_D+1}_0$, after that, for every $i' \in [1, l^i]$, he computes $k^{q_D+1}_{i'} = \mathsf{F}_{k^{q_D+1}_{i'-1}}(p_A)$, $y^{q_D+1}_{i'} = \mathsf{F}_{k^{q_D+1}_{i'}}(p_B)$ and $m^{q_D+1}_{i'} = y^{q_D+1}_{i'} \oplus c^{q_D+1}_{i'}$ [8]. Then (3), he computes $h^{q_D+1} = \mathsf{H}_s(r^{q_D+1} \| m^{q_D+1})$, adds
$\{(r^{q_D+1} \| m^{q_D+1}, h^{q_D+1})\}$ to $\mathcal{H}$, and he lazy samples $\tilde{\tau}^{q_D+1} = \mathsf{f}^*(h^{q_D+1})$.
This takes time $t_{\mathsf{H}} + (2l^i + 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L + 1)t_{\mathsf{F}}$ and the time needed to lazy sample f^* twice.
At the end of the game, he looks up the list $\mathcal{S}$ to find a collision. If he finds it, he outputs it; otherwise, 0^n and 1^n.
Thus, in total, he needs to lazy sample f^* $2(q + 1)$ times. Moreover, he needs time $t + t_{\mathsf{chn}(2,\mathcal{B})} + (q + 1)(t_{\mathsf{H}} + (2L + 1)t_{\mathsf{F}}) + t_{\mathsf{f}^*(2(q+1))} \leq t_2$.

Bounding $|\Pr[E_1] - \Pr[E_2]|$**.** Observe that if event HC happens, C wins. Since C is a t_2-adversaries and H is a $(t_2, \epsilon_{\mathsf{CR}})$-collision resistant hash function, we can bound:

$$|\Pr[E_1] - \Pr[E_2]| \leq \epsilon_{\mathsf{CR}}.$$

Game 3. Let Game 3 be Game 2, where we suppose that for every τ^j obtained in an encryption query, that is, the answer of the jth encryption query is c^j with $c^j = (\tau^j, C^J)$ the adversary is not able to ask a valid decryption query $c^{j'}$ such that $\tau^{j'} = \tau^j$. That is, for every τ^j obtained

[6] For $i' = l^i$, $c^i_{i'} = \pi_{|m^i_{i'}|}(y_{i'})^i \oplus m^i_{i'}$
[7] For $i' = l^i$, $m^i_{i'} = \pi_{|m^i_{i'}|}(y_{i'})^i \oplus c^i_{i'}$
[8] For $i' = l^{q_D+1}$, $m^{q_D+1}_{i'} = \pi_{|m^{q_D+1}_{i'}|}(y_{i'})^{q_D+1} \oplus c^{q_D+1}_{i'}$

as a tag in an encryption query, the adversary is not able to find a hash pre-image for it.
To make the reduction work we also need to assume that the adversary is not able to find a pre-image for all k_0^J obtained in decryption queries if fresh (that is, $\mathsf{f}^*(\tau^J)$ has never been computed, where the Jth decryption query is $c^J = (\tau^J, C^J)$), the adversary A^{L} is not able to ask a valid decryption query $c^{j'}$ such that $\tau^{j'} = \tau^j$ or $\tau^{j'} = k_0^J$ as we will show. Let E_3 be the event that the adversary wins this game.

Transition between Game 2 and 3. We build a sequence of $q+1$ games Game $3^{i'}$ $i' = 0, ..., q$. In Game $3^{i'}$, for the first i' encryption and decryption queries (that is, we have a counter ctr for the sum of the number of encryption queries and decryption queries and i' is compared with this counter [9]. To simplify we write $ctr = ctr_E + ctr_D$, that is, we split ctr in the two parts. We suppose that the adversary has never found a value x s.t. $\mathsf{H}_s(x) = \tau^j$ where, $\forall j = 1, ..., ctr_E$, where τ^j is the tag computed during the jth encryption query, or s.t. x s.t. $\mathsf{H}_s(x) = k_0^J$ where, $\forall J = 1, ..., ctr_D$, where k_0^J is the firs ephemeral key computed during the Jth decryption query, with $ctr_E + ctr_D = i'$. Let E_2^i be the event that the adversary wins in Game 2^i.

Transition between Game 2^{I-1} and Game 2^I. Clearly, Game 2^{I-1} and Game 2^I are equal if the following event

$$PR^I := \left\{ \begin{array}{c} \text{if the } I\text{th query is the } J\text{th encryption query} \\ \exists j \in \{1, ..., q_D + 1\} \text{ s.t. } \mathsf{Dec}_k(c^j) \neq \perp \wedge \tau^j = \tau^J \\ \text{with } c^J = (\tau^J, C^J) = \mathsf{Enc}_k(r^J, m^J) \\ \text{else if the } I\text{th query is the } J'\text{th decryption query} \\ \exists j \in \{1, ..., q_D + 1\} \text{ s.t. } \mathsf{Dec}_k(c^j) \neq \perp \wedge \tau^j = k_0^{J'} \\ \text{with } c^{J'} = (\tau^{J'}, C^{J'}) \text{ and } k_0^{J'} = \mathsf{f}^*(\tau^{J'}) \end{array} \right\}$$

does not happen. We build a t_3-roPR adversary D_I to bound the probability that event PR^I happens.

The D_I adversary. The adversary D_I receives a random key s for the hash function and a random value x in the target space of H, and he has to find a hash pre-image for it. He is based on A^{L}.
To obtain a hash pre-image for x, the idea is that D_I sets $\tau^I := x$ and uses a forgery c^j made by A^{L} where $h^j = \tau^I$ with $h^j = \mathsf{H}_s(r^j \| m^j)$ where (r^j, m^j) is the couple randomness-message retrieved during the jth decryption query. Formally:

[9] We do not have to consider the $q_D + 1$th decryption query because $k_0^{q_D+1}$ can never be used as a target

At the start of the game, D_I is provided a key s for the hash function and a target value x. Moreover, he picks two values $p_A, p_B \in \mathcal{B}$ with $p_A \neq p_B$. Then, he relays s, p_A and p_B to A^{L}. Moreover, he has two lists $\mathcal{S}$ and $\mathcal{H}$, which are empty at the start. This takes time $t_{\mathsf{chn}(2,\mathcal{B})}$.
When A does an encryption query on input (r^i, m^i), D_I (1) simply computes $h^i = \mathsf{H}_s(r^i \| m^i)$, he adds $\}(r^i \| m^i, h^i)\{$ to $\mathcal{H}$, and he parses $m^i = (m^i_1, ..., m^i_{l^i})$, then, (2), if (a) this is not the Ith query (among encryption and decryption queries) he lazy samples $\tau^i = \mathsf{f}^*(h^i)$; otherwise, if $\mathsf{f}^*(h^i)$ is fresh, (b) he sets $x = \tau^i := \mathsf{f}^*(h^i)$ (otherwise, he computes correctly $\mathsf{f}^*(h^i)$) [10] [11], and (3) he lazy samples $k^i_0 = \mathsf{f}^*(\tau^i)$. From k^i_0, he (4) computes $c^i_0 = \mathsf{F}_{k^i_0}(p_B) \oplus r^i$, after that, for every $i' \in [1, l^i]$, he computes $k^i_{i'} = \mathsf{F}_{k^i_{i'-1}}(p_A)$, $y_{i'} = \mathsf{F}_{k^i_{i'}}(p_B)$ and $c^i_{i'} = y^i_{i'} \oplus m^i_{i'}$ [12]. Finally (5), he answers A $c^i = (\tau^i, C^i)$ with $C^i = (c^i_0, c^i_1, ..., c^i_{l^i})$, and the leakage k^i_0. Moreover, D_I adds $\{(c^i, \top)\}$ to $\mathcal{S}$.
This takes time $t_{\mathsf{H}} + (2l^i + 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}$ and the time needed to lazy sample f^* twice, if $i \neq I$, once if $i = I$. Moreover, D_I adds $\{(c^i, \bot)\}$ to $\mathcal{S}$
When A asks a decryption query on input $c^i = (\tau^i, C^i)$, D_I first (1), if (a) this is not the Ith query (among encryption and decryption queries) lazy samples $k^i_0 = \mathsf{f}^*(\tau^i)$; otherwise, if $\mathsf{f}^*(\tau^i)$ is fresh, (b) he sets $x = k^i_0 := \mathsf{f}^*(\tau^i)$ (otherwise, he computes correctly $\mathsf{f}^*(h^i)$). He parses $c^i = (c^i_0, c^i_1, ..., c^i_{l^i})$. From k^i_0, he (2) computes $r^i = \mathsf{F}_{k^i_0}(p_B) \oplus c^i_0$, after that, for every $i' \in [1, l^i]$, he computes $k^i_{i'} = \mathsf{F}_{k^i_{i'-1}}(p_A)$, $y^i_{i'} = \mathsf{F}_{k^i_{i'}}(p_B)$ and $m^i_{i'} = y^i_{i'} \oplus c^i_{i'}$ [13]. Then (3), he computes $h^i = \mathsf{H}_s(r^i \| m^i)$, he adds $\}(r^i \| m^i, h^i)\{$ to $\mathcal{H}$, and he lazy samples $\tilde{\tau}^i = \mathsf{f}^*(h^i)$. Finally (4), if $\tau^i = \tilde{\tau}^i$, D_I answers $m^i = (m^i_1, ..., m^i_{l^i})$ to A; otherwise, $\bot$. Moreover, D_I adds $\{(c^i, \bot)\}$ to $\mathcal{S}$ if D_I has answered $\bot$ to A^{L}; otherwise $\{(c^i, \top)\}$.
This takes time $t_{\mathsf{H}} + (2l^i + 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}$ and the time needed to lazy sample f^* twice, if $i \neq I$, once if $i = I$.
When A asks outputs its forgery $c^{q_D+1} = (\tau^{q_D+1}, C^{q_D+1})$, D_I first (1)

[10] Since, by hypothesis there are no collision of the hash functions, then, $h^i \neq h^j$ (for both previous encryption and decryption queries); moreover, if $h^I = \tau^j$, for a previous encryption query, we have already obtained a pre-image for τ^j for $j \leq i-1$, which is prevented by the fact that in both Games 2^{i-1} and 2^i, events $PR^1, ..., PR^{i-1}$ do not happen.

[11] This is why we need to consider also x as k^J_0 for decryption queries. In fact, we need to lazy sample f^* correctly. If the adversary has asked before the decryption of a ciphertext $c^j = (\tau^J, C^J)$ where $\tau^j = h' := \mathsf{H}_s(r' \| m')$ for a couple (randomness,message) of his choice and then, he asks the encryption of (r', m'), $\mathsf{f}^*(h')$ has already been sampled and cannot be picked as x.

[12] For $i' = l^i$, $c^i_{i'} = \pi_{|m^i_{i'}|}(y_{i'})^i \oplus m^i_{i'}$

[13] For $i' = l^i$, $m^i_{i'} = \pi_{|m^i_{i'}|}(y_{i'})^i \oplus c^i_{i'}$

lazy samples $k_0^{q_D+1} = \mathsf{f}^*(\tau^{q_D+1})$ and he parses $c^{q_D+1} = (c_0^{q_D+1}, c_1^{q_D+1}, ..., c_{l^{q_D+1}}^{q_D+1})$. From $k_0^{q_D+1}$, he (2) computes $r^{q_D+1} = \mathsf{F}_{k_0^{q_D+1}}(p_B) \oplus c_0^{q_D+1}$, after that, for every $i' \in [1, l^i]$, he computes $k_{i'}^{q_D+1} = \mathsf{F}_{k_{i'-1}^{q_D+1}}(p_A)$, $y_{i'}^{q_D+1} = \mathsf{F}_{k_{i'}^{q_D+1}}(p_B)$ and $m_{i'}^{q_D+1} = y_{i'}^{q_D+1} \oplus c_{i'}^{q_D+1}$ [14]. Then (3), he computes $h^{q_D+1} = \mathsf{H}_s(r^{q_D+1} \| m^{q_D+1})$, he adds $\{(r^{q_D+1} \| m^{q_D+1}, h^{q_D+1})\}$ to $\mathcal{H}$, and he lazy samples $\tilde{\tau}^{q_D+1} = \mathsf{f}^*(h^{q_D+1})$. Finally (4), D_I adds $\{(c^i, \perp)\}$ to $\mathcal{S}$ if $\tau^{q_D+1} = \tilde{\tau}^{q_D+1}$ (or if c^{q_D+1} is not fresh); otherwise $\{(c^{q_D+1}, \top)\}$. This takes time $t_{\mathsf{H}} + (2l^i + 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L + 1)t_{\mathsf{F}}$ and the time needed to lazy sample f^* twice[15].
At the end of the game, he looks up the list $\mathcal{H}$ if there is an entry $(c^j, \top)$ such that $c^j = (\tau^j, C^j)$ with $\tau^j = x$. If he finds it, he outputs the corresponding entry $r^j \| m^j$ of $\mathcal{H}$; otherwise, 0^n.
Thus, in total, he needs to lazy sample f^* $2q + 1$ times [16]. Moreover, he needs time $t + t_{\mathsf{chn}(2,\mathcal{B})} + (q + 1)(t_{\mathsf{H}} + (2L + 1)t_{\mathsf{F}}) + t_{\mathsf{f}^*(2(q+1))} \leq t_2$.

Bounding $|\Pr[E_2^{I-1}] - \Pr[E_2^I]|$**.** Observe that if event PR^I happens, D_I wins. Since D_I is a t_2-adversaries and H is a $(t_2, \epsilon_{\mathsf{roPR}})$-range-oriented pre-image resistant hash function, we can bound:

$$|\Pr[E_2^{I-1}] - \Pr[E_2^I]| \leq \Pr[PR^I] \leq \epsilon_{\mathsf{roPR}}$$

Bounding $|\Pr[E_2] - \Pr[E_3]|$**.** Since Game 2 is Game 2^0 and Game 3 is Game 2^q we can bound

$$|\Pr[E_2] - \Pr[E_3]| \leq \sum_{I=1}^{q} |\Pr[E_2^{I-1}] - \Pr[E_2^I]| \leq q\epsilon_{\mathsf{roPR}}.$$

Game 4. In Game 4 we suppose that all decryption queries are deemed invalid (if the ciphertext c^i is fresh).

Transition between Game 3 and 4. We build a sequence of $q_D + 2$ games Game 3^i $i = 0, ..., q_D + 1$. In Game 3^i, for the first i decryption queries, if h^i is fresh, then, they are invalid. That is, $\tau^i \neq \mathsf{f}^*(h^i)$. Thus, all the first i decryption queries are invalid if fresh. Let E_3^i be the event that the adversary wins in Game 3^i.

Bounding $|\Pr[E_3^{i-1}] - \Pr[E_3^i]|$**.** Note that the difference between Game 3^{i-1} and Game 3^i are equal if the ith decryption query is not both fresh

[14] For $i' = l^{q_D+1}$, $m_{i'}^{q_D+1} = \pi_{|m_{i'}^{q_D+1}|}(y_{i'})^{q_D+1} \oplus c_{i'}^{q_D+1}$

[15] We note that $k_0^{q_D+1}$ cannot be the target since it cannot be reused in future decryption queries.

[16] Note that in the ith query, we have lazy sampled f^* only once.

and valid. Thus:

$$|\Pr[E_3^{i-1}] - \Pr[E_3^i]| = \Pr[c^i \text{ fresh and valid}].$$

Since we are in Game 3 nor event CR nor events $PR^1, ..., PR^q$ have happened. Thus, the only possibility is that $\mathsf{f}^*(h^i)$ has never been computed with $h^i = \mathsf{H}_s(r^i \| m^i)$, where r^i and m^i are the couple randomness and message is the couple retrieved during the decryption of the ith decryption query on input c^i. Thus

$$|\Pr[E_3^{i-1}] - \Pr[E_3^i]| \leq 2^{-n}$$

since $\mathsf{f}^*(h^i)$ is picked uniformly at random. Thus, the probability that $\mathsf{f}^*(h^i) = \tau^i$ is 2^{-n}.

Bounding $|\Pr[E_3] - \Pr[E_4]|$. Since Game 3 is Game 3^0 and Game 4 is Game 3^{q_D+1} we can bound

$$|\Pr[E_3] - \Pr[E_4]| \leq \sum_{i=1}^{q_D+1} |\Pr[E_3^{i-1}] - \Pr[E_3^i]| \leq (q_D + 1)2^{-n}.$$

Thus, putting everything together, we can conclude:
Bounding $\Pr[E_0]$. Since $\Pr[E_4] = 0$ (since, in Game 4, all decryption queries are invalid (if fresh)), we can conclude:

$$\Pr[E_0] = \Pr[E_0] - \Pr[E_4] = \sum_{i=1}^{4} |\Pr[E_{i-1}] - \Pr[E_i]| \leq$$

$$\epsilon_{\mathsf{PRF}} + \epsilon_{\mathsf{CR}} + q\epsilon_{\mathsf{roPR}} + (q_D + 1)2^{-n}.$$

□

Note on the bound. Observe that we have a matching attack for all the term of the bound with one exception: we have a matching attack only for the term $q_E\epsilon_{\mathsf{roPR}}$ and not for $q\epsilon_{\mathsf{roPR}}$.
In fact, it is due to an artifice of the proof to be able to simulate correctly and to avoid the situation where the adversary has asked before the decryption of a ciphertext $c^j = (\tau^J, C^J)$ where $\tau^j = h' := \mathsf{H}_s(r' \| m')$ for a couple (randomness, message) of his choice and then, he asks the encryption of (r', m'). Thus, $\mathsf{f}^*(h')$ has already been computed.
However, doing this, the adversary should not get any advantage.

B.3 Proof of the suf-L2 security of HBC2

Theorem 5. *Let* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \mathcal{B}^*$ *be a* $(t_2, \epsilon_{\mathsf{CR}})$*-collision resistant and* $(t_3, \epsilon_{\mathsf{roPR}})$*-range-oriented pre-image resistant hash function. Let* $\mathsf{F}^* : \mathcal{K}^* \times \mathcal{B}^* \to \mathcal{B}^*$ *be a* $(q_M + q_V + 1, t_1, \epsilon_{\mathsf{sPRP}})$*-strong pseudorandom permutation with a strongly protected implementation. Let* $\mathcal{HM} = \{0,1\}^*$ *and* $\mathcal{B}^* = \mathcal{TAG} = \{0,1\}^n$*.*
Then, HBC2 *is* (q_M, q_V, t, ϵ)-suf-L2*-secure in the unbounded leakage model with*

$$\epsilon_{\mathsf{sTPRP}} + \epsilon_{\mathsf{CR}} + q_V \epsilon_{\mathsf{roPR}} + (q_V + 1)2^{-n} + \frac{q(q-1)}{2^{n+1}} - \frac{q_M(q_M - 1)}{2^{n+1}}$$

with $q = q_M + q_V + 1$*,* $t + t_{\mathsf{ch}(1,\mathcal{KH})} + qt_{\mathsf{H}} \leq t_1$*,* $t + qt_{\mathsf{H}} + t_{\mathsf{f}^*(q))} \leq t_2$ *and* $t + qt_{\mathsf{H}} + t_{\mathsf{f}^*(q-1)} \leq t_3$*.*

Proof. As usual, we use a sequence of games.
Game 0. Let Game 0 be the suf-L2-game where the (q_M, q_V, t)-adversary A^{L} tries to produce a forgery when he plays against HBC2. Let E_0 be the event that the adversary wins this game, i.e., that the output of the game is 1.

Game 1. Let Game 1 be Game 0, where we have replaced the sPRP F^* with a random permutation, named f^*. Let E_1 be the event that the adversary wins this game.

Transition between Game 0 and Game 1. To bound the difference $\Pr[E_0] - \Pr[E_1]$ we build a (q, t_1)-sPRP adversary B against F^* based on A.

The (q, t_1)**-sPRP adversary** B**.** B has to distinguish if he is interacting against an oracle implemented with the sPRP F^* or with the random permutation f^*. To do this, he uses the
suf-L2-adversary A.
At the start of the game, B picks a key s for the hash function H uniformly at random in $\mathcal{KH}$. Then, he relays them to A. Moreover, he picks a list $\mathcal{S}$, which is empty at the start. This takes time $t_{\mathsf{ch}(1,\mathcal{KH})}$.
When A does an tag-generation query on input m^i, B (1) simply computes $h^i = \mathsf{H}_s(m^i)$, then, (2) he calls his oracle on input $(h^i, +1)$, receiving τ^i. Finally (3), he answers A τ^i and he adds $\{(m^i, \tau^i)\}$ to $\mathcal{S}$.
This takes time t_{H}. Moreover, one oracle query is needed.
When A asks a verification query on input (m^i, τ^i), B first (1), he computes $h^i = \mathsf{H}_s(m^i)$, then (2), he calls his oracle on input $(\tau^i, -1)$, obtaining $\tilde{h}^i$. Finally (3), if $h^i = \tilde{h}^i$, B answers $\top$ to A; otherwise, $\bot$.

This takes time t_{H}. Moreover, one oracle query is needed.
When A outputs its forgery $(m^{q_V+1}, \tau^{q_V+1})$, B first (1), he computes $h^{q_V+1} = \mathsf{H}_s(m^{q_V+1})$, then, (2) he calls his oracle on input $(\tau^{q_V+1}, -1)$, obtaining $\tilde{h}^{q_V+1}$. Finally (4), if $h^{q_V+1} = \tilde{h}^{q_V+1}$ and $(m^{q_V+1}, \tau^{q_V+1}) \notin \mathcal{S}$, B outputs 1; otherwise 0.
This takes time t_{H}. Moreover, one oracle query is needed.
Thus, B does at most q queries to his oracle. He needs time $t + t_{\mathsf{ch}(1,\mathcal{KH})} + qt_{\mathsf{H}} \leq t_1$.

Bounding $|\Pr[E_0] - \Pr[E_1]|$**.** Clearly if the oracle B faces is implemented with $\mathsf{F}^*_k(\cdot)$, he correctly simulates Game 0 for A otherwise Game 1. Thus,

$$|\Pr[E_0] - \Pr[E_1]| = |\Pr[\mathsf{B}^{\mathsf{F}^*_k(\cdot)} \Rightarrow 1] - \Pr[\mathsf{B}^{\mathsf{f}^*(\cdot)} \Rightarrow 1]| \leq \epsilon_{\mathsf{sPRP}}$$

where the last inequality is due to the fact that B is a (q, t_1)-adversary and F^* is a $(q, t_1, \epsilon_{\mathsf{sPRP}})$-sPRP.

Game 2. Let Game 2 be Game 1, where we suppose that there are no collisions for the hash function. Let E_2 be the event that A wins this game.

Transition between Game 1 and Game 2. Clearly, Game 1 and Game 2 are identical if the following event HC (*Hash collision*) does not happen:

$$HC := \{\exists i, j \in \{1, ..., q_M\} \cup \{1, ..., q_V + 1\} \text{ with } i \overset{\%}{=} j \text{ s.t. } h^i = h^j\}.^{17}$$

To compute this event, we build a t_2-CR-adversary C.

The t_2**-CR-adversary** C**.** C has to find a collision for the hash function H_s. At the start of the game, he is provided a key s for the hash function, which he relays to A^{L}. Moreover, he has a list $\mathcal{H}$ which is empty at the start.
When A does a tag-generation query on input m^i, C (1) simply computes $h^i = \mathsf{H}_s(m^i)$, then, (2) he lazy samples $\tau^i = \mathsf{f}^*(h^i, +1)$, receiving τ^i, Finally (3), he answers A τ^i and he adds $\{(m^i, h^i)\}$ to $\mathcal{H}$.
This takes time t_{H} and the time needed to lazy sample f^* once.
When A asks a verification query on input (m^i, τ^i), C first, (1) he computes $h^i = \mathsf{H}_s(m^i)$, adds $\{(m^i, h^i)\}$ to $\mathcal{H}$, then (2), he lazy samples $\tilde{h}^i = \mathsf{f}^*(\tau^i, -1)$. Finally (3), if $h^i = \tilde{h}^i$, C answers $\top$ to A; otherwise, $\bot$.

[17] $i \overset{\%}{=} j$ means that if i comes from a tag-generation query and j from a verification query, or viceversa, then, they are considered differently.

This takes time t_{H} and the time needed to lazy sample f^* once.
When A asks outputs its forgery $(m^{q_V+1}, \tau^{q_V+1})$, C first, (1) he computes $h^{q_V+1} = \mathsf{H}_s(m^{q_V+1})$, adds $\{(m^{q_V+1}, h^{q_V+1})\}$ to $\mathcal{H}$, then, (2) he lazy samples $\tilde{h}^{q_V+1} = \mathsf{f}^*(\tau^{q_V+1}, -1)$.
This takes time t_{H} and the time needed to lazy sample f^* once.
At the end of the game, he looks up the list $\mathcal{H}$ to find a collision. If he finds it, he outputs it; otherwise, 0^n and 1^n.
Thus, in total, he needs to lazy sample f^* at most q times. Moreover, he needs time $t + qt_{\mathsf{H}} + t_{\mathsf{f}^*(q))} \leq t_2$.

Bounding $|\Pr[E_1] - \Pr[E_2]|$. Observe that if event HC happens, C wins. Since C is a t_2-adversaries and H is a $(t_2, \epsilon_{\mathsf{CR}})$-collision resistant hash function, we can bound:

$$|\Pr[E_1] - \Pr[E_2]| \leq \epsilon_{\mathsf{CR}}.$$

Game 3. Let Game 3 be Game 2, where we suppose that for every $\tilde{h^j}$ obtained in an invalid verification query, the adversary is not able to ask a valid verification query $(m^{j'}, \tau^{j'})$ such that $h^{j'} = \tilde{h}^j$. Let E_3 be the event that the adversary wins this game.

Transition between Game 2 and 3. We build a sequence of $q_V + 1$ games Game 3^I $I = 0, ..., q_E$ [18]. In Game 3^I, for the first I verification queries, if they are invalid, the adversary has never found a value x s.t. $\mathsf{H}_s(x) = \tilde{h}^j$ where, $\forall j = 1, ..., I$, where $\tilde{h}^j$ is the check hash computed during the jth verification query, $\forall j = 1, ..., I$. Let E_2^I be the event that the adversary wins in Game 2^I.

Transition between Game $2^{i'-1}$ and Game 2^I. Clearly, Game 2^{I-1} and Game 2^I are equal if the following event

$$PR^I := \left\{ \begin{array}{l} \exists j \in \{1, ..., q_V + 1\} \text{ s.t. } \mathsf{Vrfy}_k(c^j) \neq \perp \wedge h^j = \tilde{h}^I \\ \text{with } c^j \text{ the } j\text{th verification query which is fresh.} \end{array} \right\}$$

does not happen. We build a t_3-roPR adversary D_I to bound the probability that event PR^I happens.

The D_I adversary. The adversary D_I receives a random value x, and he has to find a hash pre-image for it. He is based on A^{L}.
To obtain a hash pre-image for x, the idea is that D_I sets $\tilde{h}^I := x$ and uses a valid forgery (m^j, τ^j) made by A^{L} where $h^j = \tilde{h}^I$ with $h^j = \mathsf{H}_s(m^j)$.

[18] $\tilde{h}^{q_V+1}$ cannot be used as a target in subsequent verification query

Formally:
At the start of the game, D_I is provided a key s for the hash function and a target value x. Then, he relays s to A^{L}. Moreover, he has two lists $\mathcal{H}$ and $\mathcal{S}$, which are empty at the start. When A does an tag-generation query on input m^i, D_I (1) simply computes $h^i = \mathsf{H}_s(m^i)$ and he adds $\{(m^i, h^i)\{$ to $\mathcal{H}$, then, (2), he lazy samples $\tau^i = \mathsf{f}^*(h^i, +1)$. Finally (3), he answers τ^i to A.
This takes time t_{H} and the time needed to lazy sample f^* once. Moreover, D_I adds $\{(c^i, \perp)\}$ to $\mathcal{S}$
When A asks a verification query on input (m^i, τ^i), D_I first (1), he computes $h^i = \mathsf{H}_s(m^i)$, he adds $\{(m^i, h^i)\}$ to $\mathcal{H}$. After that (2), if (a) it is the Ith verification query and $\mathsf{f}^*(\tau^I, -1)$ has never been samples, he sets $x := \tilde{h}^I = \mathsf{f}^*(\tau^I, -1)$ [19]; otherwise, (b) he lazy samples $\tilde{h}^i = \mathsf{f}^*(\tau^i, -1)$. Finally (3), if $h^i = \tilde{h}^i$, D_I answers $\top$ to A; otherwise, $\perp$.
This takes time t_{H} and the time needed to lazy sample f^* once, if $i \neq I$; otherwise no lazy sampling is needed.
When A asks outputs its forgery $(m^{q_V+1}, \tau^{q_V+1})$, D_I first, (1) he computes $h^{q_V+1} = \mathsf{H}_s(m^{q_V+1})$, he adds
$\{(m^{q_V+1}, h^{q_V+1})\}$ to $\mathcal{H}$, then, (2) he lazy samples $\tilde{h}^{q_V+1} = \mathsf{f}^*(\tau^{q_V+1}, -1)$. Finally (3), D_I answers $\top$ to A if $h^{q_V+1} = \tilde{h}^{q_V+1}$; otherwise, $\perp$.
This takes time t_{H} and the time needed to lazy sample f^* once.
At the end of the game, D_I looks into his list $\mathcal{H}$ if he finds a pre-image for x. If he finds it, he outputs the corresponding entry, that is, if there is an entry (m^j, h^j) in $\mathcal{H}$ s.t. $h^j = x$, he outputs m^j; otherwise, 0^n.
Thus, in total, he needs to lazy sample f^* $q-1$ times[20]. Moreover, he needs time $t + qt_{\mathsf{H}} + t_{\mathsf{f}^*(q-1)} \leq t_3$.

[19] Note that it may be the case that it is not possible to set $x = \mathsf{f}^*(\tau^I, -1)$ since f^* is a tweakable permutation. But, this may happen only if
a either $f^*(\cdot, +1)$ has been sampled on input x
b or if sampling $f^*(\cdot, -1)$ has given x as an output.
This is not a problem. In fact:
a it means that we have asked to sample $f^*(x, +1)$. This situation may happen only in a tag-generation query. Moreover, when it happens, the input is the hash of the randomness and the message, that is, $x = \mathsf{H}_s(m^j)$ for the previous jth tag-generation query, thus, D_I proceeds normally, lazy sampling $f^*(0, \cdot, +1)$. Additionally, he has already found a pre-image for x, which is precisely m^j.
b it means that we have sampled $x = \tilde{h}^j = f^*(\tau^j, -1)$. This situation may happen only in a verification query. We call this verification query the jth with $J < I$. But, since in both games Game 2^{I-} and Game 2^I, event PR^j does not happen, so it is impossible that D_I wins finding a pre-image for x.

[20] D_I does not have to lazy sample $\mathsf{f}^*(\tau^I, -1)$ because it is equal to x.

Bounding $|\Pr[E_2^{I-1}] - \Pr[E_2^I]|$**.** Observe that if event PR^I happens, D_I wins. Since D_I is a t_3-adversaries and H is a $(t_3, \epsilon_{\mathsf{roPR}})$-range-oriented pre-image resistant hash function, we can bound:

$$|\Pr[E_2^{I-1}] - \Pr[E_2^I]| \leq \Pr[PR^I] \leq \epsilon_{\mathsf{roPR}}$$

Bounding $|\Pr[E_2] - \Pr[E_3]|$**.** Since Game 2 is Game 2^0 and Game 3 is Game 2^q we can bound

$$|\Pr[E_2] - \Pr[E_3]| \leq \sum_{I=1}^{q_V} |\Pr[E_2^{I-1}] - \Pr[E_2^I]| \leq q_V \epsilon_{\mathsf{roPR}}.$$

Game 4. In Game 4 we suppose that all verification queries are deemed invalid (if the ciphertext c^i is fresh).

Transition between Game 3 and 4. We build a sequence of $q_V + 2$ games Game 3^i $i = 0, ..., q_V + 1$. In Game 3^i, for the first i verification queries, if h^i is fresh, then, they are invalid. That is, $\tilde{h}^i \neq \mathsf{f}^*(\tau^i, -1)$. Thus, all the first i verification queries are invalid if fresh. Let E_3^i be the event that the adversary wins in Game 3^i.

Bounding $|\Pr[E_3^{i-1}] - \Pr[E_3^i]|$**.** Note that the difference between Game 3^{i-1} and Game 3^i are equal if the ith verification query is not both fresh and valid. Thus:

$$|\Pr[E_3^{i-1}] - \Pr[E_3^i]| = \Pr[c^i \text{ fresh and valid}].$$

Since we are in Game 3 nor event CR nor events $PR^1, ..., PR^q$ have happened. Thus, the only possibility is that $\mathsf{f}^*(\tau^i, -1)$ has never been computed where the ith verification query is (m^i, τ^i). Thus

$$|\Pr[E_3^{i-1}] - \Pr[E_3^i]| \leq \frac{1}{2^n - q_M - i + 1}$$

since $\mathsf{f}^*(\tau^i, -1)$ is picked uniformly at random with the constraint that $\mathsf{f}^*(0, \cdot)$ remains a permutation, thus, there at most $q_M + i - 1$ values that cannot be picked in $\mathcal{B}^*$. Thus, the probability that $\mathsf{f}^*(\tau^i, -1) = h^i$ is $\frac{1}{2^n - q_M - i + 1}$.

Bounding $|\Pr[E_3] - \Pr[E_4]|$**.** Since Game 3 is Game 3^0 and Game 4 is Game 3^{q_V+1} we can bound

$$|\Pr[E_3] - \Pr[E_4]| \leq \sum_{i=1}^{q_V+1} |\Pr[E_3^{i-1}] - \Pr[E_3^i]| \leq \sum_{i=1}^{q_V+1} \frac{1}{2^n - q_M - i + 1} \leq$$
$$(q_V + 1)2^{-n} + \frac{q(q-1)}{2^{n+1}} - \frac{q_M(q_M - 1)}{2^{n+1}}.$$ [21]

Thus, putting everything together, we can conclude:
Bounding $\Pr[E_0]$**.** Since $\Pr[E_4] = 0$ (since, in Game 4, all verification queries are invalid (if fresh)), we can conclude:

$$\Pr[E_0] = \Pr[E_0] - \Pr[E_4] = \sum_{i=1}^{4} |\Pr[E_{i-1}] - \Pr[E_i]| \leq$$

$$\epsilon_{\mathsf{sTPRP}} + \epsilon_{\mathsf{CR}} + q_V \epsilon_{\mathsf{roPR}} + (q_V + 1)2^{-n} + \frac{q(q-1)}{2^{n+1}} - \frac{q_M(q_M-1)}{2^{n+1}}.$$

□

B.4 Proof of the suf-L2 security of HTBC

Theorem 6. *Let* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \mathcal{B}_1 \times \mathcal{B}_2$ *be a* $(t_2, \epsilon_{\mathsf{CR}})$*-collision resistant and* $(t_2, \epsilon_{\mathsf{roPR}'})$*-range-oriented pre-image resistant hash function with half of the input chosen by the adversary. Let* $\mathsf{F}^* : \mathcal{K}^* \times \mathcal{TW}^* \times \mathcal{B}^* \to \mathcal{B}^*$ *be a* $(q_M + q_V + 1, t_1, \epsilon_{\mathsf{sTPRP}})$*-strong-pseudorandom tweakable permutation with a strongly protected implementation. Let* $\mathcal{B}_1 = \mathcal{B}^*$ *and* $\mathcal{B}_2 = \mathcal{TW}$*. Let* $\mathcal{HM} = \{0,1\}^*$ *and* $\mathcal{B}^* = \mathcal{TW} = \mathcal{TAG} = \{0,1\}^n$*.*
Then, HTBC *is* (q_M, q_V, t, ϵ)-suf-L2*-secure in the unbounded leakage model with*

$$\epsilon_{\mathsf{sTPRP}} + \epsilon_{\mathsf{CR}} + q_V \epsilon_{\mathsf{roPR}'} + (q_V + 1)2^{-n} + \frac{q(q-1)}{2^{n+1}} - \frac{q_M(q_M-1)}{2^{n+1}}$$

with $q = q_M + q_V + 1$*,* $t + t_{\mathsf{ch}(1,\mathcal{KH})} + qt_{\mathsf{H}} \leq t_1$*,* $t + qt_{\mathsf{H}} + t_{\mathsf{f}^*(q)} \leq t_2$ *and* $t + qt_{\mathsf{H}} + t_{\mathsf{f}^*(q-1)} \leq t_3$*.*

Proof. As usual, we use a sequence of games.
Game 0. Let Game 0 be the suf-L2-game where the (q_M, q_V, t)-adversary A^{L} tries to produce a forgery when he plays against HTBC. Let E_0 be the event that the adversary wins this game, i.e., that the output of the game is 1.

Game 1. Let Game 1 be Game 0 where the sTPRP F^* has been replaced by a tweakable random permutation, named f^*. Let E_1 be the event that the adversary wins this game.

Transition between Game 0 and Game 1. To bound the difference $\Pr[E_0] - \Pr[E_1]$ we build a (q, t_1)-sTPRP adversary B against F^* based on A.

The (q, t_1)-sPRP adversary B. B has to distinguish if he is interacting against an oracle implemented with the sTPRP F^* or with the random tweakable permutation f^*. To do this, he uses the
suf-L2-adversary A.
At the start of the game, B picks a key s for the hash function H uniformly at random in $\mathcal{KH}$. Then, he relays them to A. Moreover, he picks a list $\mathcal{S}$, which is empty at the start. This takes time $t_{\mathsf{ch}(1,\mathcal{KH})}$.
When A does an tag-generation query on input m^i, B first, (1) he simply computes $h^i = \mathsf{H}_s(m^i)$ and he parses it in $h^i = h_1^i \| h_2^i$, then, (2) he calls his oracle on input $(h_2^i, h_1^i, +1)$, receiving τ^i. Finally (3), he answers A τ^i and he adds $\{(m^i, \tau^i)\}$ to $\mathcal{S}$.
This takes time t_{H}. Moreover, one oracle query is needed.
When A asks a verification query on input (m^i, τ^i), B first (1), he computes $h^i = \mathsf{H}_s(m^i)$ and he parses it in $h^i = h_1^i \| h_2^i$, then (2), he calls his oracle on input $(h_2^i, \tau^i, -1)$, obtaining $\tilde{h}_1^i$. Finally (3), if $h_1^i = \tilde{h}_1^i$, B answers $\top$ to A; otherwise, $\bot$.
This takes time t_{H}. Moreover, one oracle query is needed.
When A outputs its forgery $(m^{q_V+1}, \tau^{q_V+1})$, B first (1), he computes $h^{q_V+1} = \mathsf{H}_s(m^{q_V+1})$ and he parses it in $h^{q_V+1} = h_1^{q_V+1} \| h_2^{q_V+1}$, then, (2) he calls his oracle on input $(h_2^{q_V+1}, \tau^{q_V+1}, -1)$, obtaining $\tilde{h}^{q_V+1}$. Finally (4), if $h^{q_V+1} = \tilde{h}^{q_V+1}$ and $(m^{q_V+1}, \tau^{q_V+1}) \notin \mathcal{S}$, B outputs 1; otherwise 0.
This takes time t_{H}. Moreover, one oracle query is needed.
Thus, B does at most q queries to his oracle. He needs time $t + t_{\mathsf{ch}(1,\mathcal{KH})} + qt_{\mathsf{H}} \leq t_1$.

Bounding $|\Pr[E_0] - \Pr[E_1]|$. Clearly if the oracle B faces is implemented with $\mathsf{F}_k^*(\cdot)$, he correctly simulates Game 0 for A otherwise Game 1. Thus,

$$|\Pr[E_0] - \Pr[E_1]| = |\Pr[\mathsf{B}^{\mathsf{F}_k^*(\cdot)} \Rightarrow 1] - \Pr[\mathsf{B}^{\mathsf{f}^*(\cdot)} \Rightarrow 1]| \leq \epsilon_{\mathsf{sTPRP}}$$

where the last inequality is due to the fact that B is a (q, t_1)-adversary and F^* is a $(q, t_1, \epsilon_{\mathsf{sTPRP}})$-sTPRP.

Game 2. Let Game 2 be Game 1, where we suppose that there are no collisions for the hash function. Let E_2 be the event that A wins this game.

Transition between Game 1 and Game 2. Clearly, Game 1 and Game 2 are identical if the following event *HC* (*Hash collision*) does not

happen:

$$HC := \{\exists i, j \in \{1, ..., q_M\} \cup \{1, ..., q_V + 1\} \text{ with } i \overset{\%}{=} j \text{ s.t. } h^i = h^j\}.$$[22]

To compute this event, we build a t_2-CR-adversary C.

The t_2-CR-adversary C. C has to find a collision for the hash function H_s. At the start of the game, he is provided a key s for the hash function, which he relays to A^{L}. Moreover, he has a list $\mathcal{H}$ which is empty at the start.
When A does a tag-generation query on input m^i, C (1) simply computes $h^i = \mathsf{H}_s(m^i)$ and he parses it in $h^i = h_1^i \| h_2^i$, then, (2) he lazy samples $\tau^i = \mathsf{f}^*(h_2^i, h_1^i, +1)$, receiving τ^i, Finally (3), he answers A τ^i and he adds $\{(m^i, h^i)\}$ to $\mathcal{H}$.
This takes time t_{H} and the time needed to lazy sample f^* once.
When A asks a verification query on input (m^i, τ^i), C first, (1) he computes $h^i = \mathsf{H}_s(m^i)$, adds $\{(m^i, h^i)\}$ to $\mathcal{H}$ and he parses it in $h^i = h_1^i \| h_2^i$, then (2), he lazy samples $\tilde{h}_1^i = \mathsf{f}^*(h_2^i, \tau^i, -1)$. Finally (3), if $h_1^i = \tilde{h}_1^i$, C answers $\top$ to A; otherwise, $\bot$.
This takes time t_{H} and the time needed to lazy sample f^* once.
When A asks outputs its forgery $(m^{q_V+1}, \tau^{q_V+1})$, C first, (1) he computes $h^{q_V+1} = \mathsf{H}_s(m^{q_V+1})$, adds $\{(m^{q_V+1}, h^{q_V+1})\}$ to $\mathcal{H}$ and he parses it in $h^{q_V+1} = h_1^{q_V+1} \| h_2^{q_V+1}$, then, (2) he lazy samples
$\tilde{h}_1^{q_V+1} = \mathsf{f}^*(h_2^{q_V+1}, \tau^{q_V+1}, -1)$.
This takes time t_{H} and the time needed to lazy sample f^* once.
At the end of the game, he looks up the list $\mathcal{H}$ to find a collision. If he finds it, he outputs it; otherwise, 0^n and 1^n.
Thus, in total, he needs to lazy sample f^* at most q times. Moreover, he needs time $t + qt_{\mathsf{H}} + t_{\mathsf{f}^*(q))} \leq t_2$.

Bounding $|\Pr[E_1] - \Pr[E_2]|$. Observe that if event HC happens, C wins. Since C is a t_2-adversaries and H is a $(t_2, \epsilon_{\mathsf{CR}})$-collision resistant hash function, we can bound:

$$|\Pr[E_1] - \Pr[E_2]| \leq \epsilon_{\mathsf{CR}}.$$

Game 3. Let Game 3 be Game 2, where we suppose that for every $\tilde{h}^j$ obtained in an invalid verification query, the adversary is not able to ask a valid verification query $(m^{j'}, \tau^{j'})$ such that $h_1^{j'} = \tilde{h}_1^j$ and $h_2^{j'} = h_2^j$. Let E_3 be the event that the adversary wins this game.

[22] $i \overset{\%}{=} j$ means that if i comes from a tag-generation query and j from a verification query, or viceversa, then, they are considered differently.

Transition between Game 2 and 3. We build a sequence of $q_V + 1$ games Game 3^I $I = 0, ..., q_E$ [23]. In Game 3^I, for the first I verification queries, if they are invalid, the adversary has never found a value x s.t. $\mathsf{H}_s(x) = \tilde{h}_1^j \| h_2^j$ where, $\forall j = 1, ..., I$, where $\tilde{h}_1^j$ is the check hash computed during the jth verification query, $\forall j = 1, ..., I$. Let E_2^I be the event that the adversary wins in Game 2^I.

Transition between Game $2^{i'-1}$ and Game 2^I. Clearly, Game 2^{I-1} and Game 2^I are equal if the following event

$$PR^I := \left\{ \begin{array}{c} \exists j \in \{1, ..., q_V + 1\} \text{ s.t. } \mathsf{Vrfy}_k(c^j) \neq \perp \\ \wedge h_1^j = \tilde{h}_1^I \text{ and} h_2^j = h_2^I \text{ where } c^j \text{ is the} \\ j\text{th verification query which is fresh.} \end{array} \right\}$$

does not happen. We build a t_3-roPR adversary D_I to bound the probability that event PR^I happens.

The D_I adversary. The adversary $\mathsf{D}_I = (\mathsf{D}_{I,1}, \mathsf{D}_{I,2})$ is composed by two adversaries, the first $\mathsf{D}_{I,1}$ knows the hash key and outputs a value y and the information st he wants to send to $\mathsf{D}_{I,2}$; $\mathsf{D}_{I,2}$ receives a random value x and he has to find an hash pre-image for $x \| y$. He is based on A^{L}. To obtain a hash pre-image for x, the idea is that D_I sets $y = h_2^I$, $\tilde{h}_1^I := x$ and uses a valid forgery (m^j, τ^j) made by A^{L} where $h_1^j = \tilde{h}_1^I$ and $h_2^j = h_2^I$ with $h^j = h_1^j \| h_2^j = \mathsf{H}_s(m^j)$. Formally:
At the start of the game, D_I is provided a key s. Then, he relays s to A^{L}. Moreover, he has two lists $\mathcal{H}$ and $\mathcal{S}$, which are empty at the start. When A does an tag-generation query on input m^i, $\mathsf{D}_{I,1}$ or $\mathsf{D}_{I,2}$ (1) simply computes $h^i = h_1^i \| h_2^i = \mathsf{H}_s(m^i)$ and he adds $\{(m^i, h^i)\{$ to $\mathcal{H}$, then, (2), he lazy samples $\tau^i = \mathsf{f}^*(h_2^i, h_1^i, +1)$. Finally (3), he answers τ^i to A. This takes time t_{H} and the time needed to lazy sample f^* once. Moreover, $\mathsf{D}_{I,1}$ (or $\mathsf{D}_{I,2}$ adds $\{(m^i, \perp)\}$ to $\mathcal{S}$
When A asks a verification query on input (m^i, τ^i), $\mathsf{D}_{I,1}$ or $\mathsf{D}_{I,2}$ first (1), he computes $h^i = h_1^i \| h_1^i = \mathsf{H}_s(m^i)$, he adds $\{(m^i, h^i)\}$ to $\mathcal{H}$. After that (2), if (a) it is the Ith verification query and $\mathsf{f}^*(h_2^I, \tau^I, -1)$ has never been samples, $\mathsf{D}_{I,1}$ outputs $h_2^I : y$ and the information $\mathsf{st} = (s, \mathcal{H})$, then, $\mathsf{D}_{I,2}$ receives x and he sets $x := \tilde{h}_1^I = \mathsf{f}^*(h_2^i, \tau^I, -1)$ [24]; otherwise, (b) he

[23] $\tilde{h}^{q_V+1}$ cannot be used as a target in subsequent verification query

[24] Note that it may be the case that it is not possible to set $x = \mathsf{f}^*(h_2^I, \tau^I, -1)$ since f^* is a tweakable permutation. But, this may happen only if
a either $f^*(\cdot, \cdot, +1)$ has been sampled on input x
b or if sampling $f^*(\cdot, \cdot, -1)$ has given x as an output.
This is not a problem. In fact:
a it means that we have asked to sample $f^*(y, x, +1)$. This situation may hap-

lazy samples $\tilde{h}_1^i = \mathsf{f}^*(h_2^i, tau^i, -1)$. Finally (3), if $h_1^i = \tilde{h}_1^i$, D_I answers $\top$ to A; otherwise, $\bot$.
This takes time t_{H} and the time needed to lazy sample f^* once, if $i \neq I$; otherwise no lazy sampling is needed.
When A outputs its forgery $(m^{q_V+1}, \tau^{q_V+1})$, $\mathsf{D}_{I,2}$ first, (1) he computes $h^{q_V+1} = h_1^{q_V+1} \| h_2^{q_V+1} = \mathsf{H}_s(m^{q_V+1})$, he adds $\{(m^{q_V+1}, h^{q_V+1})\}$ to $\mathcal{H}$, then, (2) he lazy samples $\tilde{h}_1^{q_V+1} = \mathsf{f}^*(h_2^{q_V+1}, \tau^{q_V+1}, -1)$. Finally (3), D_I answers $\top$ to A if $h_1^{q_V+1} = \tilde{h}_1^{q_V+1}$; otherwise, $\bot$.
This takes time t_{H} and the time needed to lazy sample f^* once.
At the end of the game, D_I looks into his list $\mathcal{H}$ if he finds a pre-image for $x\|y$. If he finds it, he outputs the corresponding entry, that is, if there is an entry (m^j, h^j) in $\mathcal{H}$ s.t. $h^j = x\|y$, he outputs m^j; otherwise, 0^n.
Thus, in total, he needs to lazy sample f^* $q-1$ times[25]. Moreover, he needs time $t + qt_{\mathsf{H}} + t_{\mathsf{f}^*(q-1)} \leq t_3$.

Bounding $|\Pr[E_2^{I-1}] - \Pr[E_2^I]|$**.** Observe that if event PR^I happens, D_I wins. Since D_I is a t_3-adversaries and H is a $(t_3, \epsilon_{\mathsf{roPR}'})$-range-oriented pre-image resistant' hash function, we can bound:

$$|\Pr[E_2^{I-1}] - \Pr[E_2^I]| \leq \Pr[PR^I] \leq \epsilon_{\mathsf{roPR}'}$$

Bounding $|\Pr[E_2] - \Pr[E_3]|$**.** Since Game 2 is Game 2^0 and Game 3 is Game 2^q we can bound

$$|\Pr[E_2] - \Pr[E_3]| \leq \sum_{I=1}^{q_V} |\Pr[E_2^{I-1}] - \Pr[E_2^I]| \leq q_V \epsilon_{\mathsf{roPR}'}.$$

Game 4. In Game 4 we suppose that all verification queries are deemed invalid (if the ciphertext c^i is fresh).

Transition between Game 3 and 4. We build a sequence of $q_V + 2$ games Game 3^i $i = 0, ..., q_V + 1$. In Game 3^i, for the first i verification

pen only in a tag-generation query. Moreover, when it happens, the input is the hash of the randomness and the message, that is, $x\|y = \mathsf{H}_s(m^j)$ for the previous jth tag-generation query, thus, D_I proceeds normally, lazy sampling $f^*(0, \cdot, +1)$. Additionally, he has already found a pre-image for $x\|y$, which is precisely m^j.

b it means that we have sampled $x = \tilde{h}^j = f^*(h_2^j, \tau^j, -1)$. This situation may happen only in a verification query. We call this verification query the jth with $J < I$. But, since in both games Game 2^{I-} and Game 2^I, event PR^j does not happen, so it is impossible that D_I wins finding a pre-image for $x\|y$.

[25] D_I does not have to lazy sample $\mathsf{f}^*(h_2^I, \tau^I, -1)$ because it is equal to x.

queries, if h^i is fresh, then, they are invalid. That is, $\tilde{h}_1^i \neq \mathsf{f}^*(h_2^i, \tau^i, -1)$. Thus, all the first i verification queries are invalid if fresh. Let E_3^i be the event that the adversary wins in Game 3^i.

Bounding $|\Pr[E_3^{i-1}] - \Pr[E_3^i]|$**.** Note that the difference between Game 3^{i-1} and Game 3^i are equal if the ith verification query is not both fresh and valid. Thus:

$$|\Pr[E_3^{i-1}] - \Pr[E_3^i]| = \Pr[c^i \text{ fresh and valid}].$$

Since we are in Game 3 nor event CR nor events $PR^1, ..., PR^q$ have happened. Thus, the only possibility is that $\mathsf{f}^*(h_2^i, \tau^i, -1)$ has never been computed where the ith verification query is (m^i, τ^i). Thus

$$|\Pr[E_3^{i-1}] - \Pr[E_3^i]| \leq \frac{1}{2^n - q_M - i + 1}$$

since $\mathsf{f}^*(h_2^i, \tau^i, -1)$ is picked uniformly at random with the constraint that $\mathsf{f}^*(0, \cdot)$ remains a permutation, thus, there at most $q_M + i - 1$ values that cannot be picked in $\mathcal{B}^*$. Thus, the probability that $\mathsf{f}^*(h_2^i, \tau^i, -1) = h^i$ is $\frac{1}{2^n - q_M - i + 1}$.

Bounding $|\Pr[E_3] - \Pr[E_4]|$**.** Since Game 3 is Game 3^0 and Game 4 is Game 3^{q_V+1} we can bound

$$|\Pr[E_3] - \Pr[E_4]| \leq \sum_{i=1}^{q_V+1} |\Pr[E_3^{i-1}] - \Pr[E_3^i]| \leq \sum_{i=1}^{q_V+1} \frac{1}{2^n - q_M - i + 1} \leq$$
$$(q_V + 1)2^{-n} + \frac{q(q-1)}{2^{n+1}} - \frac{q_M(q_M - 1)}{2^{n+1}}.$$ [26]

Thus, putting everything together, we can conclude:
Bounding $\Pr[E_0]$**.** Since $\Pr[E_4] = 0$ (since, in Game 4, all verification queries are invalid (if fresh)), we can conclude:

$$\Pr[E_0] = \Pr[E_0] - \Pr[E_4] = \sum_{i=1}^{4} |\Pr[E_{i-1}] - \Pr[E_i]| \leq$$

$$\epsilon_{\mathsf{sTPRP}} + \epsilon_{\mathsf{CR}} + q_V \epsilon_{\mathsf{roPR}'} + (q_V + 1)2^{-n} + \frac{q(q-1)}{2^{n+1}} - \frac{q_M(q_M - 1)}{2^{n+1}}.$$

□

B.5 Proof of the CIML2-security of DTE2

Theorem 7. *Let* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \mathcal{B}'$ *be a* $(t_2, \epsilon_{\mathsf{CR}})$*-collision resistant and* $(t_2, \epsilon_{\mathsf{roPR}})$*-range-oriented preimage resistant hash function. Let* $\mathsf{F}^* : \mathcal{K}^* \times \mathcal{B}^* \times \mathcal{TW} \to \mathcal{B}^*$ *be a* $(2q, t_1, \epsilon_{\mathsf{sTPRP}})$*-strong tweakable pseudorandom permutation with a strongly protected implementation. Let* $\mathsf{F} : \mathcal{K} \times \mathcal{B} \to \mathcal{B}$*. Let* $\mathcal{HM} = \{0,1\}^*$*,* $\mathcal{TW} = \{0,1\}$ *and* $\mathcal{B}' = \mathcal{B}^* = \mathcal{K} = \mathcal{B} = \{0,1\}^n$*. Then,* DTE2 *which encrypts at most* L*-block messages is* (q_E, q_D, t, ϵ)*-*CIML2*-secure in the unbounded leakage model with*

$$\epsilon \leq \epsilon_{\mathsf{sTPRP}} + \epsilon_{\mathsf{CR}} + q_D\epsilon_{\mathsf{roPR}} + (q_D+1)2^{-n} + \frac{q(q-1)}{2^{n+1}} - \frac{q_E(q_E-1)}{2^{n+1}}$$

where $q = q_E + q_D + 1$*,* $t + t_{\mathsf{ch}(1,\mathcal{KH})} + t_{\mathsf{chn}(2,\mathcal{B})} + q(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}) \leq t_1$*,* $t + t_{\mathsf{chn}(2,\mathcal{B})} + q(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}) + t_{\mathsf{f}^*(2q))} \leq t_2$ *and* $t + t_{\mathsf{chn}(2,\mathcal{B})} + q(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}) + t_{\mathsf{f}^*(2q-1))} \leq t_3$*.*

Proof. As usual, we use a sequence of games.
Game 0. Let Game 0 be the CIML2-game where the (q_E, q_D, t)-adversary A^{L} tries to produce a valid and fresh ciphertext when he plays against DTE2. Let E_0 be the event that the adversary wins this game, i.e., that the output of the game is 1.

Game 1. Let Game 1 be Game 0, where we have replaced the sTPRP F^* with a tweakable random permutation, named f^*. Let E_1 be the event that the adversary wins this game.

Transition between Game 0 and Game 1. To bound the difference $\Pr[E_0] - \Pr[E_1]$ we build a $(2q, t_1)$-sTPRP adversary B against F^* based on A.

The $(2q, t_1)$**-sTPRP adversary** B**.** B has to distinguish if he is interacting against an oracle implemented with the sTPRP F^* or with the random tweakable permutation f^*. To do this, he uses the CIML2-adversary A.
At the start of the game, B picks two random values p_A, p_B uniformly at random in $\mathcal{B}$ with $p_A \neq p_B$ and a key s for the hash function H uniformly at random in $\mathcal{KH}$. Then, he relays them to A. Moreover, he picks a list $\mathcal{S}$, which is empty at the start. This takes time $t_{\mathsf{ch}(1,\mathcal{KH})} + t_{\mathsf{chn}(2,\mathcal{B})}$.
When A does an encryption query on input (r^i, m^i), B (1) simply computes $h^i = \mathsf{H}_s(r^i \| m^i)$ and he parses $m^i = (m^i_1, ..., m^i_{l^i})$, then, (2) he calls his oracle on input $(0, h^i, +1)$, receiving τ^i, and (3)he calls again his oracle on input $(1, \tau^i, +1)$ obtaining k^i_0. From k^i_0, he (4) computes

$c_0^i = \mathsf{F}_{k_0^i}(p_B) \oplus r^i$, after that, for every $i' \in [1, l^i]$, he computes $k_{i'}^i = \mathsf{F}_{k_{i'-1}^i}(p_A)$, $y_{i'} = \mathsf{F}_{k_{i'}^i}(p_B)$ and $c_{i'}^i = y_{i'}^i \oplus m_{i'}^i$ [27]. Finally (5), he answers A $c^i = (\tau^i, C^i)$ with $C^i = (c_0^i, c_1^i, ..., c_{l^i}^i)$, and the leakage k_0^i and he adds $\{c^i\}$ to $\mathcal{S}$.
This takes time $t_{\mathsf{H}} + (2l^i + 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L + 1)t_{\mathsf{F}}$. Moreover, 2 oracle queries are needed.
When A asks a decryption query on input $c^i = (\tau^i, C^i)$, B first (1) queries his oracle on input $(1, \tau^i, +1)$ obtaining k_0^i and he parses $C^i = (c_0^i, c_1^i, ..., c_{l^i}^i)$. From k_0^i, he (2) computes $r^i = \mathsf{F}_{k_0^i}(p_B) \oplus c_0^i$, after that, for every $i' \in [1, l^i]$, he computes $k_{i'}^i = \mathsf{F}_{k_{i'-1}^i}(p_A)$, $y_{i'}^i = \mathsf{F}_{k_{i'}^i}(p_B)$ and $m_{i'}^i = y_{i'}^i \oplus c_{i'}^i$ [28]. Then (3), he computes $h^i = \mathsf{H}_s(r^i \| m^i)$ and he calls his oracle on input $(0, \tau^i, -1)$, obtaining $\tilde{h}^i$. Finally (4), if $h^i = \tilde{h}^i$, B answers $m^i = (m_1^i, ..., m_{l^i}^i)$ to A; otherwise, $\perp$.
This takes time $t_{\mathsf{H}} + (2l^i + 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L + 1)t_{\mathsf{F}}$. Moreover, 2 oracle queries are needed.
When A outputs its forgery $c^{q_D+1} = (\tau^{q_D+1}, C^{q_D+1})$, B first (1) queries his oracle on input $(1, \tau^{q_D+1}, +1)$ obtaining $k_0^{q_D+1}$ and he parses $C^{q_D+1} = (c_0^{q_D+1}, c_1^{q_D+1}, ..., c_{l^{q_D+1}}^{q_D+1})$. From $k_0^{q_D+1}$, he (2) computes $r^{q_D+1} = \mathsf{F}_{k_0^{q_D+1}}(p_B) \oplus c_0^{q_D+1}$, after that, for every $i' \in [1, l^i]$, he computes $k_{i'}^{q_D+1} = \mathsf{F}_{k_{i'-1}^{q_D+1}}(p_A)$, $y_{i'}^{q_D+1} = \mathsf{F}_{k_{i'}^{q_D+1}}(p_B)$ and $m_{i'}^{q_D+1} = y_{i'}^{q_D+1} \oplus c_{i'}^{q_D+1}$ [29]. Then (3), he computes $h^{q_D+1} = \mathsf{H}_s(r^{q_D+1} \| m^{q_D+1})$ and he calls his oracle on input $(0, \tau^{q_D+1}, -1)$, obtaining $\tilde{h}^{q_D+1}$. Finally (4), if $h^{q_D+1} = \tilde{h}^{q_D+1}$ and $c^{q_D+1} \notin \mathcal{S}$, B outputs 1; otherwise 0.
This takes time $t_{\mathsf{H}} + (2l^{q_D+1} + 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L + 1)t_{\mathsf{F}}$. Moreover, 2 oracle queries are needed.
Thus, B does at most $2q$ queries to his oracle. He needs time $t + t_{\mathsf{ch}(1,\mathcal{KH})} + t_{\mathsf{chn}(2,\mathcal{B})} + q(t_{\mathsf{H}} + (2L + 1)t_{\mathsf{F}}) \leq t_1$.

Bounding $|\Pr[E_0] - \Pr[E_1]|$. Clearly if the oracle B faces is implemented with $\mathsf{F}_k^*(\cdot)$, he correctly simulates Game 0 for A otherwise Game 1. Thus,

$$|\Pr[E_0] - \Pr[E_1]| = |\Pr[\mathsf{B}^{\mathsf{F}_k^*(\cdot)} \Rightarrow 1] - \Pr[\mathsf{B}^{\mathsf{f}^*(\cdot)} \Rightarrow 1]| \leq \epsilon_{\mathsf{sTPRP}}$$

where the last inequality is due to the fact that B is a $(2q, t_1)$-adversary and F^* is a $(2q, t_1, \epsilon_{\mathsf{sTPRP}})$-$\mathsf{sTPRP}$.

[27] For $i' = l^i$, $c_{i'}^i = \pi_{|m_{i'}^i|}(y_{i'})^i \oplus m_{i'}^i$
[28] For $i' = l^i$, $m_{i'}^i = \pi_{|m_{i'}^i|}(y_{i'})^i \oplus c_{i'}^i$
[29] For $i' = l^{q_D+1}$, $m_{i'}^{q_D+1} = \pi_{|m_{i'}^{q_D+1}|}(y_{i'})^{q_D+1} \oplus c_{i'}^{q_D+1}$

Game 2. Let Game 2 be Game 1, where we suppose that there are no collisions for the hash function. Let E_2 be the event that A wins this game.

Transition between Game 1 and Game 2. Clearly, Game 1 and Game 2 are identical if the following event HC (*Hash collision*) does not happen:

$$HC := \{\exists i, j \in \{1, ..., q_E\} \cup \{1, ..., q_D + 1\} \text{ with } i \overset{\%}{=} j \text{ s.t. } h^i = h^j\}.^{30}$$

To compute this event, we build a t_2-CR-adversary C.

The t_2-CR-adversary C. C has to find a collision for the hash function H_s. At the start of the game, he is provided a key s for the hash function. Moreover, he picks two values $p_A, p_B \in \mathcal{B}$ with $p_A \neq p_B$. Then, he relays s, p_A and p_B to A^{L}. Moreover, he has a list $\mathcal{H}$, which is empty at the start. This takes time $t_{\mathsf{chn}(2,\mathcal{B})}$.
When A does an encryption query on input (r^i, m^i), C (1) simply computes $h^i = \mathsf{H}_s(r^i \| m^i)$, adds $\{(r^i \| m^i, h^i)\}$ to $\mathcal{H}$, and he parses $m^i = (m^i_1, ..., m^i_{l^i})$, then, (2) he lazy samples $\tau^i = \mathsf{f}^*(0, h^i, +1)$, receiving τ^i, and (3) he lazy samples $k^i_0 = \mathsf{f}^*(1, \tau^i, +1)$. From k^i_0, he (4) computes $c^i_0 = \mathsf{F}_{k^i_0}(p_B) \oplus r^i$, after that, for every $i' \in [1, l^i]$, he computes $k^i_{i'} = \mathsf{F}_{k^i_{i'-1}}(p_A)$, $y_{i'} = \mathsf{F}_{k^i_{i'}}(p_B)$ and $c^i_{i'} = y^i_{i'} \oplus m^i_{i'}$ [31]. Finally (5), he answers A $c^i = (\tau^i, C^i)$ with $C^i = (c^i_0, c^i_1, ..., c^i_{l^i})$, and the leakage k^i_0 and he adds $\{(r^i \| m^i, h^i)\}$ to $\mathcal{H}$.
This takes time $t_{\mathsf{H}} + (2l^i + 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L + 1)t_{\mathsf{F}}$ and the time needed to lazy sample f^* twice.
When A asks a decryption query on input $c^i = (\tau^i, C^i)$, C first (1) lazy samples $k^i_0 = \mathsf{f}^*(1, \tau^i, +1)$ and he parses $C^i = (c^i_0, c^i_1, ..., c^i_{l^i})$. From k^i_0, he (2) computes $r^i = \mathsf{F}_{k^i_0}(p_B) \oplus c^i_0$, after that, for every $i' \in [1, l^i]$, he computes $k^i_{i'} = \mathsf{F}_{k^i_{i'-1}}(p_A)$, $y^i_{i'} = \mathsf{F}_{k^i_{i'}}(p_B)$ and $m^i_{i'} = y^i_{i'} \oplus c^i_{i'}$ [32]. Then (3), he computes $h^i = \mathsf{H}_s(r^i \| m^i)$, adds $\{(r^i \| m^i, h^i)\}$ to $\mathcal{H}$, and he lazy samples $\tilde{h}^i = \mathsf{f}^*(0, \tau^i, -1)$. Finally (4), if $h^i = \tilde{h}^i$, C answers $m^i = (m^i_1, ..., m^i_{l^i})$ to A; otherwise, $\perp$.
This takes time $t_{\mathsf{H}} + (2l^i + 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L + 1)t_{\mathsf{F}}$ and the time needed to lazy sample f^* twice.
When A asks outputs its forgery $c^{q_D+1} = (\tau^{q_D+1}, C^{q_D+1})$, C first (1) lazy

[30] $i \overset{\%}{=} j$ means that if i comes from a encryption query and j from a decryption query, or viceversa, then, they are considered differently.
[31] For $i' = l^i$, $c^i_{i'} = \pi_{|m^i_{i'}|}(y_{i'})^i \oplus m^i_{i'}$
[32] For $i' = l$, $m^i_{i'} = \pi_{|m^i_{i'}|}(y_{i'})^i \oplus c^i_{i'}$

samples $k_0^{q_D+1} = \mathsf{f}^*(1, \tau^{q_D+1}, +1)$ and he parses $C^{q_D+1} = (c_0^{q_D+1}, c_1^{q_D+1}, ..., c_{l^{q_D+1}}^{q_D+1})$. From $k_0^{q_D+1}$, he (2) computes $r^{q_D+1} = \mathsf{F}_{k_0^{q_D+1}}(p_B) \oplus c_0^{q_D+1}$, after that, for every $i' \in [1, l^i]$, he computes $k_{i'}^{q_D+1} = \mathsf{F}_{k_{i'-1}^{q_D+1}}(p_A)$, $y_{i'}^{q_D+1} = \mathsf{F}_{k_{i'}^{q_D+1}}(p_B)$ and $m_{i'}^{q_D+1} = y_{i'}^{q_D+1} \oplus c_{i'}^{q_D+1}$ [33]. Then (3), he computes $h^{q_D+1} = \mathsf{H}_s(r^{q_D+1} \| m^{q_D+1})$, adds $\{(r^{q_D+1} \| m^{q_D+1}, h^{q_D+1})\}$ to $\mathcal{H}$, and he lazy samples $\tilde{h}^{q_D+1} = \mathsf{f}^*(0, \tau^{q_D+1}, -1)$. This takes time $t_{\mathsf{H}} + (2l^{q_D+1}+1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}$ and the time needed to lazy sample f^* twice.
At the end of the game, he looks up the list $\mathcal{H}$ to find a collision. If he finds it, he outputs it; otherwise, 0^n and 1^n.
Thus, in total, he needs to lazy sample f^* at most $2q$ times. Moreover, he needs time $t + t_{\mathsf{chn}(2,\mathcal{B})} + q(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}) + t_{\mathsf{f}^*(2q))} \leq t_2$.

Bounding $|\Pr[E_1] - \Pr[E_2]|$**.** Observe that if event HC happens, C wins. Since C is a t_2-adversaries and H is a $(t_2, \epsilon_{\mathsf{CR}})$-collision resistant hash function, we can bound:

$$|\Pr[E_1] - \Pr[E_2]| \leq \epsilon_{\mathsf{CR}}.$$

Game 3. Let Game 3 be Game 2, where we suppose that for every $\tilde{h^j}$ obtained in a not valid decryption query, the adversary is not able to ask a valid decryption query $c^{j'}$ such that $h^{j'} = \tilde{h}^j$. Let E_3 be the event that the adversary wins this game.

Transition between Game 2 and 3. We build a sequence of $q_D + 1$ games Game 3^I $I = 0, ..., q_E$ [34]. In Game 3^I, for the first I decryption queries, if they are invalid, the adversary has never found a value x s.t. $\mathsf{H}_s(x) = \tilde{h}^j$ where, $\forall j = 1, ..., I$, where $\tilde{h}^j$ is the check hash computed during the jth decryption query, $\forall j = 1, ..., I$. Let E_2^I be the event that the adversary wins in Game 2^I.

Transition between Game $2^{i'-1}$ and Game 2^I. Clearly, Game 2^{I-1} and Game 2^I are equal if the following event

$$PR^I := \left\{ \begin{array}{l} \exists j \in \{1, ..., q_D + 1\} \text{ s.t. } \mathsf{Dec}_k(c^j) \neq \perp \wedge h^j = \tilde{h}^I \\ \text{with } c^j \text{ the } j\text{th decryption query which is fresh.} \end{array} \right\}$$

does not happen. We build a t_3-roPR adversary D_I to bound the probability that event PR^I happens.

[33] For $i' = l^{q_D+1}$, $m_{i'}^{q_D+1} = \pi_{|m_{i'}^{q_D+1}|}(y_{i'})^{q_D+1} \oplus c_{i'}^{q_D+1}$

[34] $\tilde{h}^{q_D+1}$ cannot be used as a target in subsequent decryption query

The D_I adversary. The adversary D_I receives a random value x, and he has to find a hash pre-image for it. He is based on A^{L}.
To obtain a hash pre-image for x, the idea is that D_I sets $\tilde{h}^I := x$ and uses a valid forgery c^j made by A^{L} where $h^j = \tilde{h}^I$ with $h^j = \mathsf{H}_s(r^j \| m^j)$ where (r^j, m^j) is the couple randomness-message retrieved during the jth decryption query. Formally:
At the start of the game, D_I is provided a key s for the hash function and a target value x. Moreover, he picks two values $p_A, p_B \in \mathcal{B}$ with $p_A \neq p_B$. Then, he relays s, p_A and p_B to A^{L}. Moreover, he has two lists $\mathcal{H}$ and $\mathcal{S}$, which are empty at the start. This takes time $t_{\mathsf{chn}(2,\mathcal{B})}$.
When A does an encryption query on input (r^i, m^i), D_I (1) simply computes $h^i = \mathsf{H}_s(r^i \| m^i)$, he adds $\{(r^i \| m^i, h^i)\{$ to $\mathcal{H}$, and he parses $m^i = (m^i_1, ..., m^i_{l^i})$, then, (2), he lazy samples $\tau^i = \mathsf{f}^*(0, h^i, +1)$, and (3) he lazy samples $k^i_0 = \mathsf{f}^*(1, \tau^i, +1)$. From k^i_0, he (4) computes $c^i_0 = \mathsf{F}_{k^i_0}(p_B) \oplus r^i$, after that, for every $i' \in [1, l^i]$, he computes $k^i_{i'} = \mathsf{F}_{k^i_{i'-1}}(p_A)$, $y_{i'} = \mathsf{F}_{k^i_{i'}}(p_B)$ and $c^i_{i'} = y^i_{i'} \oplus m^i_{i'}$ [35]. Finally (5), he answers A $c^i = (\tau^i, C^i)$ with $C^i = (c^i_0, c^i_1, ..., c^i_{l^i})$, and the leakage k^i_0.
This takes time $t_{\mathsf{H}} + (2l^i + 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L + 1)t_{\mathsf{F}}$ and the time needed to lazy sample f^* twice. Moreover, D_I adds $\{(c^i, \perp)\}$ to $\mathcal{S}$
When A asks a decryption query on input $c^i = (\tau^i, C^i)$, D_I first (1), he lazy samples $k^i_0 = \mathsf{f}^*(1, \tau^i, +1)$ and he parses $C^i = (c^i_0, c^i_1, ..., c^i_{l^i})$. From k^i_0, he (2) computes $r^i = \mathsf{F}_{k^i_0}(p_B) \oplus c^i_0$, after that, for every $i' \in [1, l^i]$, he computes $k^i_{i'} = \mathsf{F}_{k^i_{i'-1}}(p_A)$, $y^i_{i'} = \mathsf{F}_{k^i_{i'}}(p_B)$ and $m^i_{i'} = y^i_{i'} \oplus c^i_{i'}$ [36]. Then (3), he computes $h^i = \mathsf{H}_s(r^i \| m^i)$, he adds $\{(r^i \| m^i, h^i)\}$ to $\mathcal{H}$. After that (4), if (a) it is the Ith decryption query and $\mathsf{f}^*(0, \tau^I, -1)$ has never been samples, he sets $x := \tilde{h}^I = \mathsf{f}^*(0, \tau^I, -1)$ [37]; otherwise, (b) he lazy samples

[35] For $i' = l^i$, $c^i_{i'} = \pi_{|m^i_{i'}|}(y_{i'})^i \oplus m^i_{i'}$

[36] For $i' = l^i$, $m^i_{i'} = \pi_{|m^i_{i'}|}(y_{i'})^i \oplus c^i_{i'}$

[37] Note that it may be the case that it is not possible to set $x = \mathsf{f}^*(0, \tau^I, -1)$ since f^* is a tweakable permutation. But, this may happen only if

a either $f^*(0, \cdot, +1)$ has been sampled on input x
b or if sampling $f^*(0, \cdot, -1)$ has given x as an output.

This is not a problem. In fact:

a it means that we have asked to sample $f^*(0, \cdot, +1)$. This situation may happen only in the encryption queries. Moreover, when it happens, the input is the hash of the randomness and the message, that is, $x = \mathsf{H}_s(r^j \| m^j)$ for the previous jth encryption query, thus, D_I proceeds normally, lazy sampling $f^*(0, x, +1)$. Additionally, he has already found a pre-image for x, which is precisely $r^j \| m^j$.
b it means that we have sampled $x = \tilde{h}^j = f^*(0, \tau^j, -1)$. This situation may happen only in the decryption queries. We call this decryption query the jth with $J < I$. But, since in both games Game 2^{I-} and Game 2^I, event PR^j does not happen, so it is impossible that D_I wins finding a pre-image for x.

$\tilde{h}^i = \mathsf{f}^*(0, \tau^i, -1)$. Finally (4), if $h^i = \tilde{h}^i$, D_I answers $m^i = (m^i_1, ..., m^i_{l^i})$ to A; otherwise, $\perp$.
This takes time $t_{\mathsf{H}} + (2l^i + 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L + 1)t_{\mathsf{F}}$ and the time needed to lazy sample f^* once, if $i = I$; otherwise, twice.
When A asks outputs its forgery $c^{q_D+1} = (\tau^{q_D+1}, C^{q_D+1})$, D_I first (1) lazy samples $k_0^{q_D+1} = \mathsf{f}^*(1, \tau^{q_D+1}, +1)$ and he parses
$C^{q_D+1} = (c_0^{q_D+1}, c_1^{q_D+1}, ..., c_{l^{q_D+1}}^{q_D+1})$. From $k_0^{q_D+1}$, he (2) computes $r^{q_D+1} = \mathsf{F}_{k_0^{q_D+1}}(p_B) \oplus c_0^{q_D+1}$, after that, for every $i' \in [1, l^i]$, he computes $k_{i'}^{qV+1} = \mathsf{F}_{k_{i'-1}^{q_D+1}}(p_A)$, $y_{i'}^{q_D+1} = \mathsf{F}_{k_{i'}^{q_D+1}}(p_B)$ and $m_{i'}^{q_D+1} = y_{i'}^{q_D+1} \oplus c_{i'}^{q_D+1}$ [38]. Then (3), he computes $h^{q_D+1} = \mathsf{H}_s(r^{q_D+1} \| m^{q_D+1})$, he adds $\{(r^{q_D+1} \| m^{q_D+1}, h^{q_D+1})\}$ to $\mathcal{H}$, and he lazy samples $\tilde{h}^{q_D+1} = \mathsf{f}^*(0, \tau^{q_D+1}, -1)$. Finally (4), D_I answers m^{q_D+1} is $h^{q_D+1} = \tilde{h}^{q_D+1}$; otherwise, $\perp$.
This takes time $t_{\mathsf{H}} + (2l^{q_D+1} + 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L + 1)t_{\mathsf{F}}$ and the time needed to lazy sample f^* twice.
At the end of the game, D_I looks into his list $\mathcal{H}$ if he finds a pre-image for x. If he finds it, he outputs the corresponding entry, that is, if there is an entry $(r^j \| m^j, h^j)$ in $\mathcal{H}$ s.t. $h^j = x$, he outputs $r^j \| m^j$; otherwise, 0^n.
Thus, in total, he needs to lazy sample f^* $2q - 1$ times[39]. Moreover, he needs time $t + t_{\mathsf{chn}(2,\mathcal{B})} + q(t_{\mathsf{H}} + (2L + 1)t_{\mathsf{F}}) + t_{\mathsf{f}^*(2q-1)} \leq t_3$.

Bounding $|\Pr[E_2^{I-1}] - \Pr[E_2^I]|$**.** Observe that if event PR^I happens, D_I wins. Since D_I is a t_3-adversaries and H is a $(t_3, \epsilon_{\mathsf{roPR}})$-range-oriented pre-image resistant hash function, we can bound:

$$|\Pr[E_2^{I-1}] - \Pr[E_2^I]| \leq \Pr[PR^I] \leq \epsilon_{\mathsf{roPR}}$$

Bounding $|\Pr[E_2] - \Pr[E_3]|$**.** Since Game 2 is Game 2^0 and Game 3 is Game 2^q we can bound

$$|\Pr[E_2] - \Pr[E_3]| \leq \sum_{I=1}^{q_D} |\Pr[E_2^{I-1}] - \Pr[E_2^I]| \leq q_D \epsilon_{\mathsf{roPR}}.$$

Game 4. In Game 4 we suppose that all decryption queries are deemed invalid (if the ciphertext c^i is fresh).

Transition between Game 3 and 4. We build a sequence of $q_D + 2$ games Game 3^i $i = 0, ..., q_D + 1$. In Game 3^i, for the first i decryption

[38] For $i' = l^{q_D+1}$, $m_{i'}^{q_D+1} = \pi_{|m_{i'}^{q_D+1}|}(y_{i'})^{q_D+1} \oplus c_{i'}^{q_D+1}$

[39] D_I does not have to lazy sample $\mathsf{f}^*(0, \tau^I, -1)$ because it is equal to x.

queries, if h^i is fresh, then, they are invalid. That is, $\tilde{h}^i \neq \mathsf{f}^*(0, \tau^i, -1)$. Thus, all the first i verification queries are invalid if fresh. Let E_3^i be the event that the adversary wins in Game 3^i.

Bounding $|\Pr[E_3^{i-1}] - \Pr[E_3^i]|$**.** Note that the difference between Game 3^{i-1} and Game 3^i are equal if the ith decryption query is not both fresh and valid. Thus:

$$|\Pr[E_3^{i-1}] - \Pr[E_3^i]| = \Pr[c^i \text{ fresh and valid}].$$

Since we are in Game 3 nor event CR nor events $PR^1, ..., PR^q$ have happened. Thus, the only possibility is that $\mathsf{f}^*(0, \tau^i, -1)$ has never been computed where the ith decryption query is $c^i = (\tau^i, C^i)$. Thus

$$|\Pr[E_3^{i-1}] - \Pr[E_3^i]| \leq \frac{1}{2^n - q_E - i + 1}$$

since $\mathsf{f}^*(0, \tau^i, -1)$ is picked uniformly at random with the constraint that $\mathsf{f}^*(0, \cdot)$ remains a permutation, thus, there at most $q_E + i - 1$ values that cannot be picked in $\mathcal{B}^*$. Thus, the probability that $\mathsf{f}^*(0, \tau^i, -1) = h^i$ is $\frac{1}{2^n - q_E - i + 1}$.

Bounding $|\Pr[E_3] - \Pr[E_4]|$**.** Since Game 3 is Game 3^0 and Game 4 is Game 3^{q_D+1} we can bound

$$|\Pr[E_3] - \Pr[E_4]| \leq \sum_{i=1}^{q_D+1} |\Pr[E_3^{i-1}] - \Pr[E_3^i]| \leq \sum_{i=1}^{q_D+1} \frac{1}{2^n - q_E - i + 1} \leq$$
$$(q_D + 1)2^{-n} + \frac{q(q-1)}{2^{n+1}} - \frac{q_E(q_E - 1)}{2^{n+1}}.$$ [40]

Thus, putting everything together, we can conclude:
Bounding $\Pr[E_0]$**.** Since $\Pr[E_4] = 0$ (since, in Game 4, all decryption queries are invalid (if fresh)), we can conclude:

$$\Pr[E_0] = \Pr[E_0] - \Pr[E_4] = \sum_{i=1}^{4} |\Pr[E_{i-1}] - \Pr[E_i]| \leq$$

$$\epsilon_{\mathsf{sTPRP}} + \epsilon_{\mathsf{CR}} + q_D \epsilon_{\mathsf{roPR}} + (q_D + 1)2^{-n} + \frac{q(q-1)}{2^{n+1}} - \frac{q_E(q_E - 1)}{2^{n+1}}.$$

□

B.6 Proof of the CIML2-security of EDT

Theorem 8. *Let* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \{0,1\}^n$ *be a* $(t_1, \epsilon_{\mathsf{CR}})$*-collision resistant and* $(t_1, \epsilon_{\mathsf{roPR}})$*-range-oriented preimage resistant hash function. Let* $\mathsf{F}^* : \mathcal{K}^* \times \{0,1\}^n \times \mathcal{TW} \to \{0,1\}^n$ *be a* $(2q-1, t_1, \epsilon_{\mathsf{sTPRP}})$*-strong tweakable pseudorandom permutation with a strongly-protected implementations. Let* $\mathsf{F} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$*. Let* $\mathcal{HM} = \{0,1\}^*$ *and* $\mathcal{TW} = \{0,1\}$*.*
Then, EDT *which encrypts at most* L*-block messages is* (q_E, q_D, t, ϵ)*-*CIML2*-secure in the unbounded leakage model with*

$$\epsilon_{\mathsf{sTPRP}} + \epsilon_{\mathsf{CR}} + q_D\epsilon_{\mathsf{roPR}} + (q_D+1)2^{-n} + \frac{q(q-1)}{2^{n+1}} - \frac{q_E(q_E-1)}{2^{n+1}}$$

with $q = q_E + q_D + 1$, $t + t_{\mathsf{ch}(1,\mathcal{KH})} + t_{\mathsf{chn}(2,\mathcal{B})} + qt_{\mathsf{H}} + (q-1)(2L-1)t_{\mathsf{F}}) \leq t_1$,
$t + t_{\mathsf{chn}(2,\mathcal{B})} + qt_{\mathsf{H}} + (q-1)(2L-1)t_{\mathsf{F}} + t_{\mathsf{f}^*(2q-1))} \leq t_2$
and $t + t_{\mathsf{chn}(2,\mathcal{B})} + qt_{\mathsf{H}} + (q-1)(2L-1)t_{\mathsf{F}} + t_{\mathsf{f}^*(2q-2)} \leq t_3$.

Proof. **Game 0.** Let Game 0 be the CIML2-game where the (q_E, q_D, t)-adversary A^{L} tries to produce a valid and fresh ciphertext when he plays against EDT. Let E_0 be the event that the adversary wins this game, i.e., that the output of the game is 1.

Game 1. Let Game 1 be Game 0, where we have replaced the sTPRP F^* with a tweakable random permutation, named f^*. Let E_1 be the event that the adversary wins this game.

Transition between Game 0 and Game 1. To bound the difference $\Pr[E_0] - \Pr[E_1]$ we build a $(2q-1, t_1)$-sTPRP adversary B against F^* based on A.

The $(2q, t_1)$**-sTPRP adversary** B**.** B has to distinguish if he is interacting against an oracle implemented with the sTPRP F^* or with the random tweakable permutation f^*. To do this, he uses the CIML2-adversary A.
At the start of the game, B picks two random values p_A, p_B uniformly at random in $\mathcal{B}$ with $p_A \neq p_B$ and a key s for the hash function H uniformly at random in $\mathcal{KH}$. Then, he relays them to A. Moreover, he picks a list $\mathcal{S}$, which is empty at the start. This takes time $t_{\mathsf{ch}(1,\mathcal{KH})} + t_{\mathsf{chn}(2,\mathcal{B})}$.
When A does an encryption query on input (r^i, m^i), B first, (1) he calls his oracle on input $(0, r^i, +1)$, obtaining k_1^i and he parses $m^i = (m_1^i, ..., m_{l^i}^i)$. From k_1^i, he (2) computes $c_1^i = \mathsf{F}_{k_0^i}(p_B) \oplus m_1^i$, after that, for every $i' \in [2, l^i]$, he computes $k_{i'}^i = \mathsf{F}_{k_{i'-1}^i}(p_A)$, $y_{i'} = \mathsf{F}_{k_{i'}^i}(p_B)$ and

$c^i_{i'} = y^i_{i'} \oplus m^i_{i'}$ [41]. Then, (3) he simply computes $h^i = \mathsf{H}_s(r^i \| C^i)$ with $C^i = (c^i_0, c^i_1, ..., c^i_{l^i})$, after that, (4) he calls his oracle on input $(1, h^i, +1)$, receiving τ^i, Finally (5), he answers A $c^i = (C^i, \tau^i)$ and the leakage k^i_1 and he adds $\{c^i\}$ to $\mathcal{S}$.
This takes time $t_{\mathsf{H}} + (2l^i - 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L-1)t_{\mathsf{F}}$. Moreover, 2 oracle queries are needed.
When A asks a decryption query on input (r^i, c^i) with $c^i = (C^i, \tau^i)$, B first, (1) he computes $h^i = \mathsf{H}_s(r^i \| C^i)$ and (2) he calls his oracle on input $(1, \tau^i, -1)$, obtaining $\tilde{h}^i$. Then (3), if $h^i \neq \tilde{h}^i$, B (3a) answers to A and the leakage $\tilde{h}^i$ and stops the execution; otherwise, (3b) he queries his oracle on input $(0, r^i, +1)$ obtaining k^i_1 and he parses $C^i = (c^i_0, c^i_1, ..., c^i_{l^i})$. From k^i_1, he (4b) computes $m^i_1 = \mathsf{F}_{k^i_1}(p_B) \oplus c^i_1$, after that, for every $i' \in [2, l^i]$, he computes $k^i_{i'} = \mathsf{F}_{k^i_{i'-1}}(p_A)$, $y^i_{i'} = \mathsf{F}_{k^i_{i'}}(p_B)$ and $m^i_{i'} = y^i_{i'} \oplus c^i_{i'}$ [42]. Finally (5b), B answers $m^i = (m^i_1, ..., m^i_{l^i})$ and the leakage k^i_1 to A
This takes time $t_{\mathsf{H}} + (2l^i - 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L-1)t_{\mathsf{F}}$. Moreover, 2 oracle queries are needed.
When A outputs its forgery (r^{q_D+1}, c^{q_D+1}) with $c^{q_D+1} = (C^{q_D+1}, \tau^{q_D+1})$, B first, (1) he computes $h^{q_D+1} = \mathsf{H}_s(r^{q_D+1} \| m^{q_D+1})$ and (2) he calls his oracle on input $(1, \tau^{q_D+1}, -1)$, obtaining $\tilde{h}^{q_D+1}$. Then (3), if $h^{q_D+1} = \tilde{h}^{q_D+1}$ and $c^{q_D+1} \notin \mathcal{S}$, B outputs 1; otherwise 0.
This takes time t_{H}. Moreover, one oracle query is needed.
Thus, B does at most $2q-1$ queries to his oracle. He needs time $t + t_{\mathsf{ch}(1,\mathcal{KH})} + t_{\mathsf{chn}(2,\mathcal{B})} + qt_{\mathsf{H}} + (q-1)(2L-1)t_{\mathsf{F}}) \leq t_1$.

Bounding $|\Pr[E_0] - \Pr[E_1]|$**.** Clearly if the oracle B faces is implemented with $\mathsf{F}^*_k(\cdot)$, he correctly simulates Game 0 for A otherwise Game 1. Thus,

$$|\Pr[E_0] - \Pr[E_1]| = |\Pr[\mathsf{B}^{\mathsf{F}^*_k(\cdot)} \Rightarrow 1] - \Pr[\mathsf{B}^{\mathsf{f}^*(\cdot)} \Rightarrow 1]| \leq \epsilon_{\mathsf{sTPRP}}$$

where the last inequality is due to the fact that B is a $(2q-1, t_1)$-adversary and F^* is a $(2q-1, t_1, \epsilon_{\mathsf{sTPRP}})$-$\mathsf{sTPRP}$.

Game 2. Let Game 2 be Game 1, where we suppose that there are no collisions for the hash function. Let E_2 be the event that A wins this game.

Transition between Game 1 and Game 2. Clearly, Game 1 and Game 2 are identical if the following event HC (*Hash collision*) does not

[41] For $i' = l^i$, $c^i_{i'} = \pi_{|m^i_{i'}|}(y_{i'})^i \oplus m^i_{i'}$
[42] For $i' = l^i$, $m^i_{i'} = \pi_{|m^i_{i'}|}(y_{i'})^i \oplus c^i_{i'}$

happen:

$$HC := \{\exists i, j \in \{1, ..., q_E\} \cup \{1, ..., q_D + 1\} \text{ with } i \overset{\%}{=} j \text{ s.t. } h^i = h^j\}.$$[43]

To compute this event, we build a t_2-CR-adversary C.

The t_2-CR-adversary C. C has to find a collision for the hash function H_s. At the start of the game, he is provided a key s for the hash function. Moreover, he picks two values $p_A, p_B \in \mathcal{B}$ with $p_A \neq p_B$. Then, he relays s, p_A and p_B to A^{L}. Moreover, he has a list $\mathcal{H}$, which is empty at the start. This takes time $t_{\mathsf{chn}(2,\mathcal{B})}$.
When A does an encryption query on input (r^i, m^i), C first, (1) he lazy samples $k_1^i = \mathsf{f}^*(0, \tau^i, +1)$, and he parses $m^i = (m_1^i, ..., m_{l^i}^i)$. From k_0^i, he (2) computes $c_1^i = \mathsf{F}_{k_0^i}(p_B) \oplus m_1^i$, after that, for every $i' \in [2, l^i]$, he computes $k_{i'}^i = \mathsf{F}_{k_{i'-1}^i}(p_A)$, $y_{i'} = \mathsf{F}_{k_{i'}^i}(p_B)$ and $c_{i'}^i = y_{i'}^i \oplus m_{i'}^i$ [44]. Then, (3) he simply computes $h^i = \mathsf{H}_s(r^i \| C^i)$ with $C^i = (c_1^i, ..., c_{l^i}^i)$, adds $\{(r^i \| C^i, h^i)\}$ to $\mathcal{H}$, after that, (4) he lazy samples $\tau^i = \mathsf{f}^*(1, h^i, +1)$, receiving τ^i. Finally (5), he answers A $c^i = (C^i, \tau^i)$ and the leakage k_1^i.
This takes time $t_{\mathsf{H}} + (2l^i - 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L - 1)t_{\mathsf{F}}$ and the time needed to lazy sample f^* twice.
When A asks a decryption query on input (r^i, c^i) with $c^i = (C^i, \tau^i)$, C first, (1) he computes $h^i = \mathsf{H}_s(r^i \| C^i)$, adds $\{(r^i \| C^i, h^i)\}$ to $\mathcal{H}$, and (2) he lazy samples $\tilde{h}^i = \mathsf{f}^*(1, \tau^i, -1)$. Then, (3), if $h^i \neq \tilde{h}^i$, (3a) C answers to A $\perp$ and the leakage $\tilde{h}^i$to A; otherwise, (3b) he lazy samples $k_1^i = \mathsf{f}^*(1, \tau^i, +1)$ and he parses $c^i = (c_1^i, ..., c_{l^i}^i)$. From k_0^i, he (4b) computes $m_1^i = \mathsf{F}_{k_1^i}(p_B) \oplus c_1^i$, after that, for every $i' \in [2, l^i]$, he computes $k_{i'}^i = \mathsf{F}_{k_{i'-1}^i}(p_A)$, $y_{i'}^i = \mathsf{F}_{k_{i'}^i}(p_B)$ and $m_{i'}^i = y_{i'}^i \oplus c_{i'}^i$ [45]. Finally (5b), C answers $m^i = (m_1^i, ..., m_{l^i}^i)$ to A.
This takes time $t_{\mathsf{H}} + (2l^i - 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L - 1)t_{\mathsf{F}}$ and the time needed to lazy sample f^* twice.
When A asks outputs its forgery (r^{q_D+1}, c^{q_D+1}) with
$c^{q_D+1} = (C^{q_D+1}, \tau^{q_D+1})$, C first, (1) he computes $h^{q_D+1} = \mathsf{H}_s(r^{q_D+1} \| C^{q_D+1})$ and he adds
$\{(r^{q_D+1} \| C^{q_D+1}, h^{q_D+1})\}$ to $\mathcal{H}$, and (2) he lazy samples
$\tilde{h}^{q_D+1} = \mathsf{f}^*(1, \tau^{q_D+1}, -1)$.
This takes time $t_{\mathsf{H}}+$ and the time needed to lazy sample f^* once.
At the end of the game, he looks up the list $\mathcal{H}$ to find a collision. If he

[43] $i \overset{\%}{=} j$ means that if i comes from a encryption query and j from a decryption query, or viceversa, then, they are considered differently.
[44] For $i' = l^i$, $c_{i'}^i = \pi_{|m_{i'}^i|}(y_{i'})^i \oplus m_{i'}^i$
[45] For $i' = l$, $m_{i'}^i = \pi_{|m_{i'}^i|}(y_{i'})^i \oplus c_{i'}^i$

finds it, he outputs it; otherwise, 0^n and 1^n.
Thus, in total, he needs to lazy sample f^* at most $2q-1$ times. Moreover, he needs time $t+t_{\mathsf{chn}(2,\mathcal{B})}+qt_{\mathsf{H}}+(q-1)(2L-1)t_{\mathsf{F}})+t_{\mathsf{f}^*(2q-1))} \leq t_2$.

Bounding $|\Pr[E_1]-\Pr[E_2]|$**.** Observe that if event HC happens, C wins. Since C is a t_2-adversaries and H is a $(t_2, \epsilon_{\mathsf{CR}})$-collision resistant hash function, we can bound:

$$|\Pr[E_1]-\Pr[E_2]| \leq \epsilon_{\mathsf{CR}}.$$

Game 3. Let Game 3 be Game 2, where we suppose that for every $\tilde{h^j}$ obtained in a not valid decryption query, the adversary is not able to ask a valid decryption query $c^{j'}$ such that $h^{j'} = \tilde{h}^j$. Let E_3 be the event that the adversary wins this game.

Transition between Game 2 and 3. We build a sequence of q_D+1 games Game 3^I $I = 0, ..., q_E$ [46]. In Game 3^I, for the first I decryption queries, if they are invalid, the adversary has never found a value x s.t. $\mathsf{H}_s(x) = \tilde{h}^j$ where, $\forall j = 1, ..., I$, where $\tilde{h}^j$ is the check hash computed during the jth decryption query, $\forall j = 1, ..., I$. Let E_2^I be the event that the adversary wins in Game 2^I.

Transition between Game $2^{i'-1}$ and Game 2^I. Clearly, Game 2^{I-1} and Game 2^I are equal if the following event

$$PR^I := \left\{ \begin{array}{l} \exists j \in \{1, ..., q_D+1\} \text{ s.t. } \mathsf{Dec}_k(c^j) \neq \perp \wedge h^j = \tilde{h}^I \\ \text{with } c^j \text{ the } j\text{th decryption query which is fresh.} \end{array} \right\}$$

does not happen. We build a t_3-roPR adversary D_I to bound the probability that event PR^I happens.

The D_I adversary. The adversary D_I receives a random value x, and he has to find a hash pre-image for it. He is based on A^{L}.
To obtain a hash pre-image for x, the idea is that D_I sets $\tilde{h}^I := x$ and uses a valid forgery (r^j, c^j) with $c^j = (C^j, \tau^j)$ made by A^{L} where $h^j = \tilde{h}^I$ with $h^j = \mathsf{H}_s(r^j \| C^j)$ where (r^j, m^j) is the couple randomness-message retrieved during the jth decryption query. Formally:
At the start of the game, D_I is provided a key s for the hash function and a target value x. Moreover, he picks two values $p_A, p_B \in \mathcal{B}$ with $p_A \neq p_B$. Then, he relays s, p_A and p_B to A^{L}. Moreover, he has two lists $\mathcal{H}$ and $\mathcal{S}$, which are empty at the start. This takes time $t_{\mathsf{chn}(2,\mathcal{B})}$.

[46] $\tilde{h}^{q_D+1}$ cannot be used as a target in subsequent decryption query

When A does an encryption query on input (r^i, m^i), D_I first, (1) he lazy samples $k_1^i = \mathsf{f}^*(0, r^i, +1)$, and he parses $m^i = (m_1^i, ..., m_{l^i}^i)$. From k_1^i, he (2) computes $c_1^i = \mathsf{F}_{k_1^i}(p_B) \oplus m_1^i$, after that, for every $i' \in [2, l^i]$, he computes $k_{i'}^i = \mathsf{F}_{k_{i'-1}^i}(p_A)$, $y_{i'} = \mathsf{F}_{k_{i'}^i}(p_B)$ and $c_{i'}^i = y_{i'}^i \oplus m_{i'}^i$ [47]. Then, (3) he simply computes $h^i = \mathsf{H}_s(r^i \| C^i)$ with $C^i = (c_1^i, ..., c_{l^i}^i)$, and he adds $\{(r^i \| C^i, h^i)\{$ to $\mathcal{H}$, after that, (4) he lazy samples $\tau^i = \mathsf{f}^*(1, h^i, +1)$. Finally (5), he answers A $c^i = (C^i, \tau^i)$ and the leakage k_1^i.
This takes time $t_{\mathsf{H}} + (2l^i - 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L - 1)t_{\mathsf{F}}$ and the time needed to lazy sample f^* twice. Moreover, D_I adds $\{(c^i, \perp)\}$ to $\mathcal{S}$
When A asks a decryption query on input (r^i, c^i) with $c^i = (C^i, \tau^i)$, D_I first (1), he computes $h^i = \mathsf{H}_s(r^i \| C^i)$, and he adds $\{(r^i \| C^i, h^i)\}$ to $\mathcal{H}$. After that (2), if (a) it is the Ith decryption query and $\mathsf{f}^*(1, \tau^I, -1)$ has never been samples, he sets $x := \tilde{h}^I = \mathsf{f}^*(0, \tau^I, -1)$ [48]; otherwise, (b) he lazy samples $\tilde{h}^i = \mathsf{f}^*(1, \tau^i, -1)$. Then, (3), if $h^i \neq \tilde{h}^i$, (3c) D_I answers to A $\perp$ and the leakage $\tilde{h}^i$; otherwise, (3d) he lazy samples $k_1^i = \mathsf{f}^*(0, \tau^i, +1)$ and he parses $C^i = (c_1^i, ..., c_{l^i}^i)$. From k_1^i, (4d) he computes $m_1^i = \mathsf{F}_{k_1^i}(p_B) \oplus c_1^i$, after that, for every $i' \in [2, l^i]$, he computes $k_{i'}^i = \mathsf{F}_{k_{i'-1}^i}(p_A)$, $y_{i'}^i = \mathsf{F}_{k_{i'}^i}(p_B)$ and $m_{i'}^i = y_{i'}^i \oplus c_{i'}^i$ [49]. Finally, he (5) answers to A $m^i = (m_1^i, ..., m_{l^i}^i)$ and the leakage k_1^i.
This takes time $t_{\mathsf{H}} + (2l^i - 1)t_{\mathsf{F}} \leq t_{\mathsf{H}} + (2L - 1)t_{\mathsf{F}}$ and the time needed to lazy sample f^* once, if $i = I$; otherwise, twice.
When A asks outputs its forgery (r^{q_D+1}, c^{q_D+1}) with
$c^{q_D+1} = (C^{q_D+1}, \tau^{q_D+1})$, D_I first, (1) he computes $h^{q_D+1} = \mathsf{H}_s(r^{q_D+1} \| C^{q_D+1})$, and he adds
$\{(r^{q_D+1} \| C^{q_D+1}, h^{q_D+1})\}$ to $\mathcal{H}$, then, (2) he lazy samples $\tilde{h}^{q_D+1} = \mathsf{f}^*(1, \tau^{q_D+1}, -1)$.
This takes time t_{H} and the time needed to lazy sample f^* once.

---

[47] For $i' = l^i$, $c_{i'}^i = \pi_{|m_{i'}^i|}(y_{i'})^i \oplus m_{i'}^i$

[48] Note that it may be the case that it is not possible to set $x = \mathsf{f}^*(1, \tau^I, -1)$ since f^* is a tweakable permutation. But, this may happen only if

a either $f^*(1, \cdot, +1)$ has been sampled on input x

b or if sampling $f^*(1, \cdot, -1)$ has given x as an output.

This is not a problem. In fact:

a it means that we have asked to sample $f^*(1, x, +1)$. This situation may happen only in an encryption query. Moreover, when it happens, the input is the hash of the randomness and the message, that is, $x = \mathsf{H}_s(r^j \| C^j)$ for the previous jth encryption query, thus, D_I proceeds normally, lazy sampling $f^*(1, \cdot, +1)$. Additionally, he has already found a pre-image for x, which is precisely $r^j \| C^j$.

b it means that we have sampled $x = \tilde{h}^j = f^*(1, \tau^j, -1)$. This situation may happen only in a decryption query. We call this decryption query the jth with $J < I$. But, since in both games Game 2^{I-} and Game 2^I, event PR^j does not happen, so it is impossible that D_I wins finding a pre-image for x.

[49] For $i' = l^i$, $m_{i'}^i = \pi_{|m_{i'}^i|}(y_{i'})^i \oplus c_{i'}^i$

At the end of the game, D_I looks into his list $\mathcal{H}$ if he finds a pre-image for x. If he finds it, he outputs the corresponding entry, that is, if there is an entry $(r^j \| m^j, h^j)$ in $\mathcal{H}$ s.t. $h^j = x$, he outputs $r^j \| m^j$; otherwise, 0^n.
Thus, in total, he needs to lazy sample f^* $2q-2$ times[50]. Moreover, he needs time $t + t_{\mathsf{chn}(2,\mathcal{B})} + qt_{\mathsf{H}} + (q-1)(2L-1)t_{\mathsf{F}}) + t_{\mathsf{f}^*(2q-2)} \leq t_3$.

Bounding $|\Pr[E_2^{I-1}] - \Pr[E_2^I]|$. Observe that if event PR^I happens, D_I wins. Since D_I is a t_3-adversaries and H is a $(t_3, \epsilon_{\mathsf{roPR}})$-range-oriented pre-image resistant hash function, we can bound:

$$|\Pr[E_2^{I-1}] - \Pr[E_2^I]| \leq \Pr[PR^I] \leq \epsilon_{\mathsf{roPR}}$$

Bounding $|\Pr[E_2] - \Pr[E_3]|$. Since Game 2 is Game 2^0 and Game 3 is Game 2^q we can bound

$$|\Pr[E_2] - \Pr[E_3]| \leq \sum_{I=1}^{q_D} |\Pr[E_2^{I-1}] - \Pr[E_2^I]| \leq q_D \epsilon_{\mathsf{roPR}}.$$

Game 4. In Game 4 we suppose that all decryption queries are deemed invalid (if the ciphertext c^i is fresh).

Transition between Game 3 and 4. We build a sequence of $q_D + 2$ games Game 3^i $i = 0, ..., q_D + 1$. In Game 3^i, for the first i decryption queries, if h^i is fresh, then, they are invalid. That is, $\tilde{h}^i \neq \mathsf{f}^*(0, \tau^i, -1)$. Thus, all the first i verification queries are invalid if fresh. Let E_3^i be the event that the adversary wins in Game 3^i.

Bounding $|\Pr[E_3^{i-1}] - \Pr[E_3^i]|$. Note that the difference between Game 3^{i-1} and Game 3^i are equal if the ith decryption query is not both fresh and valid. Thus:

$$|\Pr[E_3^{i-1}] - \Pr[E_3^i]| = \Pr[c^i \text{ fresh and valid}].$$

Since we are in Game 3 nor event CR nor events $PR^1, ..., PR^q$ have happened. Thus, the only possibility is that $\mathsf{f}^*(1, \tau^i, -1)$ has never been computed where the ith decryption query is (r^i, c^i) with $c^i = (C^i, \tau^i)$. Thus

$$|\Pr[E_3^{i-1}] - \Pr[E_3^i]| \leq \frac{1}{2^n - q_E - i + 1}$$

since $\mathsf{f}^*(1, \tau^i, -1)$ is picked uniformly at random with the constraint that $\mathsf{f}^*(1, \cdot)$ remains a permutation, thus, there at most $q_E + i - 1$ values that

[50] D_I does not have to lazy sample $\mathsf{f}^*(0, \tau^I, -1)$ because it is equal to x.

cannot be picked in $\mathcal{B}^*$. Thus, the probability that $\mathsf{f}^*(0, \tau^i, -1) = h^i$ is $\frac{1}{2^n - q_E - i + 1}$.

Bounding $|\Pr[E_3] - \Pr[E_4]|$**.** Since Game 3 is Game 3^0 and Game 4 is Game 3^{q_D+1} we can bound

$$|\Pr[E_3] - \Pr[E_4]| \leq \sum_{i=1}^{q_D+1} |\Pr[E_3^{i-1}] - \Pr[E_3^i]| \leq \sum_{i=1}^{q_D+1} \frac{1}{2^n - q_E - i + 1} \leq$$

$$(q_D + 1)2^{-n} + \frac{q(q-1)}{2^{n+1}} - \frac{q_E(q_E - 1)}{2^{n+1}}.$$ [51]

Thus, putting everything together, we can conclude:
Bounding $\Pr[E_0]$**.** Since $\Pr[E_4] = 0$ (since, in Game 4, all decryption queries are invalid (if fresh)), we can conclude:

$$\Pr[E_0] = \Pr[E_0] - \Pr[E_4] = \sum_{i=1}^{4} |\Pr[E_{i-1}] - \Pr[E_i]| \leq$$

$$\epsilon_{\mathsf{sTPRP}} + \epsilon_{\mathsf{CR}} + q_D \epsilon_{\mathsf{roPR}} + (q_D + 1)2^{-n} + \frac{q(q-1)}{2^{n+1}} - \frac{q_E(q_E - 1)}{2^{n+1}}.$$

□

B.7 Proof of the CIML2-security of CONCRETE

Theorem 9. *Let* $\mathsf{F}^* : \mathcal{K}^* \times \mathcal{B}^* \times \mathcal{TW}^* \to \mathcal{B}^*$ *be a leak free* $(q+1, \epsilon_{\mathsf{sTPRP}})$*-strong tweakable pseudorandom permutation (*sTPRP*), let* $\mathsf{F} : \mathcal{K} \times \mathcal{B} \to \mathcal{B}$ *be a* $(2, \epsilon_{\mathsf{PRF}})$*-pseudorandom function (*PRF*) and let* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \to \mathcal{B}'$ *be a* ϵ_{CR}*-collision resistant hash function. Let* $\mathcal{B}^* = \mathcal{K} = \mathcal{B}$ *and* $\mathcal{TW}^* = \mathcal{B}'$*. Let* $\mathcal{B} = \{0,1\}^n$*.*
Then, the mode CONCRETE*, which encrypts messages which are at most* L*-block long, is* (q_E, q_D, ϵ)*-*CIML2 *secure in the unbounded leakage model with*

$$\epsilon \leq \epsilon_{\mathsf{sTPRP}} + \frac{(q_E + q_D)(q_E + q_D - 1)}{2^{n+1}} + \epsilon_{\mathsf{CR}} +$$

$$\frac{(q_D + 1)(L + 1)(q_D + 2q_E)}{2^{n+1}} + \frac{q_D + 1}{2^n} + (q_D + 1)\epsilon_{\mathsf{PRF}}.$$

with $q = q_E + q_D$ *and*

$$t_1 = t_{\mathsf{ch}'} + (q+1)(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}})$$

$$t_2 = t_{\mathsf{ch}'} + (q+1)(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}) + t_{\mathsf{f}(Q+1)}.$$

Proof. We use a series of games. For simplicity, we call the decryption query induced by the output of A^{L} as the $q_D + 1$ decryption query.

Game 0. This is the real CIML2 game where A^{L} attacks scheme CONCRETE. Let E_0 be the event that A^{L} wins Game 0.

Game 1. In this game, we replace sTPRP $\mathsf{F}^*_k(\cdot,\cdot)$ with the random tweakable permutation $\mathsf{f}^*(\cdot,\cdot)$. Let E_1 be the event that A^{L} wins Game 1.

Transition from Game 0 to Game 1. It is easy to build a $(q{+}1, t{+}t_1)$-sTPRP adversary B whose sTPRP advantage is $|\Pr[E_0] - \Pr[E_1]|$.

The sTPRP adversary B. The adversary B has access to an oracle which is either implemented either via $\mathsf{F}^*_k(\cdot,\cdot)$ and $\mathsf{F}^{*,-1}_k(\cdot,\cdot)$ or via $\mathsf{f}^*(\cdot,\cdot)$ and $\mathsf{f}^{*,-1}(\cdot,\cdot)$[52]. B has to distinguish the situations. In detail:
At the start of the game, B picks 2 constants $p_A, p_B \in \{0,1\}^n$ with $p_A \neq p_B$ and a key $s \xleftarrow{\$} \mathcal{KH}$ and sends to A^{L} (p_A, p_B, s). Moreover, B sets $\mathcal{S}$ as an empty set.
When A does an encryption query on input (r^i, m^i), with $m^i = (m^i_1, ..., m^i_{l^i})$, B simply (1) sets $k^i_0 := r^i$, then, (2) from the ephemeral key k^i_0, he computes $(c^i_0, ..., c^i_{l^i})$ [53] [54], after that, (3) he computes $h^i = \mathsf{H}_s(c^i_0 \| ... \| c^i_{l^i})$ and (4) he queries his oracle on input $(h^i, k^i_0, +1)$ obtaining $c^i_{l^i+1}$, finally (5) B answers A c^i with $c^i = (c^i_0, ..., c^i_{l^i}, c^i_{l^i+1})$[55]. Then, he updates the set $\mathcal{S}$, adding the ciphertext c^i. For every encryption query B does 1 oracle query, moreover, he evaluates F $2l^i + 1 \leq 2L + 1$ times and once the hash function H; thus, answering to A takes at most $(2L+1)t_{\mathsf{F}} + t_{\mathsf{H}}$ time.
When A makes a decryption query on input c^j with $c^j = (c^j_0, ..., c^j_{l^j}, c^j_{l^j+1})$, B (1) computes the hash $h^j = \mathsf{H}_s(c^j_0, ..., c^j_{l^j})$, (2) queries his oracle on input $(h^j, c^j_{l^j+1}, -1)$ obtaining k^j_0, (3) computes $\tilde{c}^j_0 = \mathsf{F}_{k^j_0}(p_B)$, then (4) if $c^j_0 \neq \tilde{c}^j_0$ he sets $m^j = \perp$, that is, he answers "invalid", otherwise, (5) from k^j_0, B is able to compute $m^j = (m^j_1, ..., m^j_{l^j})$[56] [57], finally, (6) B sets $\mathsf{L}_D(c^j, k) := k^j_0$ and he answers (m^j, k^j_0). For every decryption query B does 1 oracle query, moreover, he evaluates once the hash function H,

[52] To make notation simpler, the adversary uses the third input, which is either $+1$ or -1 to distinguish if he is doing an evaluation query or an inverse query.
[53] Via $c^i = y^i \oplus m^i$ with $y_i = \mathsf{F}_{k_i}(p_B)$ and $k_i = \mathsf{F}_{k_{i-1}}(p_A)$
[54] If $i = l$, $c^i = \pi_{|m^i_{l_i}|}(y^i) \oplus m^i$
[55] We have already showed that we can assume that there is no leakage in encryption.
[56] Via $m^i = y^i \oplus c^i$ with $y_i = \mathsf{F}_{k_i}(p_B)$ and $k_i = \mathsf{F}_{k_{i-1}}(p_A)$
[57] If $i = l$, $m^i = \pi_{|c^i_{l_i}|}(y^i) \oplus m^i$

once F if c^j is deemed invalid, otherwise $2l^j + 1 \leq 2L + 1$ times; thus, answering to A takes at most $(2L+1)t_{\mathsf{F}} + t_{\mathsf{H}}$ time.
When A outputs the challenge ciphertext $c = c^{q_D+1}$, B proceeds as for the others decryption queries. Thus, he uses for this decryption query again 1 oracle query and at most time $(2L+1)t_{\mathsf{F}} + t_{\mathsf{H}}$. If the decryption query c is valid and $c \notin \mathcal{S}$, B outputs 1, 0 otherwise.
Thus B runs in time bounded by $t + (q+1)(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}})$ and does at most $q+1$ oracle queries.

Bounding $|\Pr[E_0] - \Pr[E_1]| \leq \epsilon_{\mathsf{sTPRP}}$. Clearly if the oracle is implemented with $(\mathsf{F}_k^*(\cdot,\cdot), \mathsf{F}_k^{*,-1}(\cdot,\cdot))$ B is correctly simulating Game 0; otherwise, Game 1.

$$\text{Thus } \Pr[\mathsf{B}^{\mathsf{F}_k^*(\cdot,\cdot),\mathsf{F}_k^{*,-1}(\cdot,\cdot)} \Rightarrow 1] = \Pr[\mathsf{A} \text{ wins Game 0}] = \Pr[E_0]$$

$$\text{and } \Pr[\mathsf{B}^{\mathsf{f}^*(\cdot,\cdot),\mathsf{f}^{*,-1}(\cdot,\cdot)} \Rightarrow 1] = \Pr[\mathsf{A} \text{ wins Game 1}] = \Pr[E_1].$$

Consequently

$$|\Pr[E_0] - \Pr[E_1]| = \left|\Pr[\mathsf{B}^{\mathsf{F}_k^*(\cdot,\cdot),\mathsf{F}_k^{*,-1}(\cdot,\cdot)} \Rightarrow 1] - \Pr[\mathsf{B}^{\mathsf{f}^*(\cdot,\cdot),\mathsf{f}^{*,-1}(\cdot,\cdot)} \Rightarrow 1]\right|$$

which is bounded by $\epsilon_{\mathsf{sTPRP}}$ since $\mathsf{F}^*(\cdot,\cdot)$ is a a $(q+1, t+(q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{F}}), \epsilon_{\mathsf{sTPRP}})$-strong tweakable pseudorandom permutation (sTPRP) and B is a $(q+1, t+(q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{F}})$-$\mathsf{sTPRP}$ adversary.

Game 2. Game 2 is Game 1, where we suppose that all hash values h^i are different provided that their inputs are different. Let E_2 be the event that A wins Game 2.

Transition between Game 1 and Game 2. We introduce a failure event HC, so defined:

$$HC := \left\{ \begin{array}{c} \exists i, i' \in \{1,...,q_E\} \cup \{1,...,q_{D+1}\}, i \overset{\%}{\neq} i' \\ \text{s.t. } h^i = h^{i'} \text{and } (c_0^i, ..., c_{l^i}^i) \neq (c_0^{i'}, ..., c_{l^{i'}}^{i'}) \end{array} \right\}$$

(With the symbol $\overset{\%}{\neq}$ we mean that the inequality $i \neq i'$ *always* holds if one index is picked from $\{1,...,q_E\}$ and the other from $\{1,...,q_{D+1}\}$)
To compute the probability of event HC, which clearly consists on a collision for the hash function H_s, we build a collision resistant adversary C.

The $(0, t+(q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{F}}) + t_{\mathsf{f}(q+1)})$ collision resistance adversary C. The collision resistant adversary C wants to output a

collision for the hash function $\mathsf{H}_s(\cdot)$ he has access to and he is based on the CIML2 adversary A. To emulate either Game 1 or Game 2 for A^{L}, C simply picks two values $p_A, p_B \leftarrow \{0,1\}^n$ and a tweakable random permutation f^*. To make the adversary more efficient we allow him to lazy sample [17] the tweakable random permutation f^*. Then he behaves as adversary B emulating Game 0 (or 1) for A with two differences: first, to obtain c^i_{l+1} in encryption queries (step 4) and k^j_0 in decryption queries (step 2), instead of querying his oracle and using its answers, C lazy samples $\mathsf{f}^*(\cdot,\cdot)$; second, he has a list $\mathcal{H}$ which he updates adding $(h^i, (c^i_0, ..., c^i_{l^i}))$ every time he has to compute the hash function $\mathsf{H}_s(\cdot)$ (that is, he keeps track of all inputs and outputs of the hash function). At the end of the game, C looks up into his list $\mathcal{H}$ if he finds a collision: if it is the case, he outputs it, otherwise he outputs $(0,1)$. C does no query and he runs in time bounded by $t + (q+1)(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}) + t_{\mathsf{f}(q+1)}$, where $t_{\mathsf{f}(q+1)}$ is the time needed to lazy sample f^* $q+1$ times.

Bounding $\Pr[HC]$**.** If event HC happens, clearly C wins because he has output a collision. Thus

$$\Pr[HC] \leq \Pr[\mathsf{C} \text{ produces a collision}] \leq \epsilon_{\mathsf{CR}}$$

since the hash function H is $(t + (q+1)(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}) + t_{\mathsf{f}(q+1)}, \epsilon_{\mathsf{CR}})$-collision resistant and C is a $t+(q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{F}})+t_{\mathsf{f}(q+1)})$-adversary.

Bounding $|\Pr[E_1] - \Pr[E_2]|$**.** Since Game 1 and Game 2 are identical if event HC does not happen, then,

$$\Pr[E_1] \leq \Pr_{[} E_2] + \Pr[HC] \leq \Pr[E_2] + \epsilon_{cr}.$$

Game 3. Game 3 is Game 2, where we suppose that all fresh decryption queries are invalid. Let E_3 be the probability that A wins Game 3. Clearly $\Pr[E_3] = 0$.

Transition between Game 2 and 3. To bound the difference $|\Pr[E_2] - \Pr[E_3]|$, we build a sequence of $q_D + 2$ games Game $2^0, ...,$ Game 2^{q_D+1}.

Game 2^i**.** Game 2^i is Game 2 where the first i decryption queries c^j, for $j = 1, ..., i$, if they are fresh, are answered with $(\perp, k^j_0)$ with $k^j_0 = \mathsf{L}_D(c^j)$. Let E^i_2 be the event that adversary wins Game 2^i.
Clearly, Game 2 is Game 2^0 and Game 3 is Game 2^{q_D+2}.

Transition between Game 2^{i-1} **and Game** 2^i**.** We observe that the only difference between Game 2^{i-1} and Game 2^i is how the ith decryption

query is treated. Consider the following event:

$$F_i := \{ \text{ the } i\text{th decryption query is valid and fresh}\}$$

If event F_i does not happen, Game 2^{i-1} and Game 2^i are indistinguishable since the answer to the ith decryption query is the same. Thus:

$$\left|\Pr[E_2^{i-1}] - \Pr[E_2^i]\right| \leq \Pr[F_i].$$

Bounding $\Pr[F_i]$**.** Let $c^i = (c_0^i, c_1^i, ..., c_{l^i}^i, c_{l^i+1}^i)$ be the ith decryption query. There are two possibilities:

F^1 The partial ciphertext $(c_0^i, ..., c_{l^i})$ is *fresh*,

F^2 The partial ciphertext $(c_0^i, ..., c_{l^i})$ is *not fresh*

Clearly, every fresh ciphertext falls in exactly one of the previous case. We call event F_i^j event $F^j \cap F_i$.

Event F_i^1**.** Since the partial ciphertext $(c_0^i, ..., c_{l^i}^i)$ is fresh, then, its hash h^i is fresh because event HC has not happened. That is, there are no collisions for the hash function. Since at least one of the input (the tweak in this case, which is equal to h^i) of the tweakable random permutation $\mathsf{f}^{*,-1}$ is fresh, then, $k_0^i := \mathsf{f}^{*,-1}(h^i, c_{l^i}^i)$ is picked uniformly at random via lazy sampling of the tweakable random permutation. Thus, event F_i^1 happens only if $\mathsf{F}_{k_0^i}(p_B) = c_0^i$ for a random key. This event is called CO^i (collision output). To compute $\Pr[CO^i]$, we introduce Game 3^i where, we replace $\mathsf{F}_{k_0^i}(\cdot)$ with the random function $\mathsf{f}^i(\cdot)$ in the computation of the ith decryption query.

Game 3^i**.** Game 3^i is Game 2^i where we replace in the ith decryption query $\mathsf{F}_{k_0^i}$ with the random function f_0^i. We call event CO_2^i the event that $c_0^i = \tilde{c}_0^i$ in Game 2^i, and event CO_3^i the event that $c_0^i = \tilde{c}_0^i$ in Game 3^i.

Transition between Game 2^i **and Game** 3^i**.** To do it, we build a $(2, t + (q+1)(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}) + t_{\mathsf{f}(q+1)})$-PRF adversary D^i based on A.

The $(2, t+(q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{F}})+t_{\mathsf{f}(q+1)})$-PRF **adversary** D^i**.** The PRF adversary D^i has access to an oracle which is implemented either with $\mathsf{F}_{k_0^i}(\cdot)$ where k_0^i is a key picked uniformly at random or with a random function $\mathsf{f}^i(\cdot)$. D^i has to distinguish the two situations. To emulate Game 2^i for A, D^i simply picks two values $p_A, p_B \leftarrow \{0,1\}^n$, a key for the hash function, $s \xleftarrow{\$} \mathcal{KH}$, and a tweakable random permutation f^*, which, for

efficiency, he lazy samples. Then, he behaves as adversary C emulating Game 1 for A^{L}, before the ith decryption query. When D^i receives the ith decryption query on input $c^i = (c_0^i, ..., c_{l^i}^i, c_{l^i+1}^i)$, by hypothesis (F^1) the partial ciphertext is fresh, thus, its hash h^i is fresh $(\overline{HC})$. Then, D^i calls his oracle on input p_B, receiving y as an answer and he sets $\tilde{c}_0^i = y$. After that, he calls his oracle on input p_A receiving y' and he sets $k_1^i = y'$. Moreover, he picks a random key $\tilde{k}_0^i$ and he sets the leakage $\mathsf{L}_D(c^i; k) = \tilde{k}_0^i$. From now on, he behaves as C in Game 1. At the end of the game, if $\tilde{c}_0^i = c_0^i$, D^i outputs 1, otherwise he outputs 0. D^i does only two queries to his oracle, and he runs in time bounded by

$$t+(q_E+i-1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{F}})+t_{\mathsf{f}(q+1)} \leq t+(q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{F}})+t_{\mathsf{f}(q+1)}.$$

(Even if adversary D^i had not answered correctly to the ith decryption query, or if he had not simulated the game correctly after that query, this would not have created any problem since D^i's output does not depend on what the adversary A^{L} does after his ith decryption query.)
Moreover, concerning the correctness of the simulation, we observe that, if the key k_0^i is a key which has already been used in the game as a key for F during a previous encryption or decryption query, it is not possible to replace $\mathsf{F}_{k_0^i}(\cdot)$ with a random function only in the ith decryption query. Apart from this case, the computation of $\tilde{c}_0^i$ is correctly simulated if the oracle is implemented with $\mathsf{f}^i(\cdot)$ for Game 3^i, while if the oracle is implemented with $\mathsf{F}_{k_0^i}(\cdot)$ for a random key k_0^i, Game 2^i is correctly simulated by D^i. Thus, we define the event KC^i (key collision):

$$KC^i := \left\{ \begin{array}{l} \exists \text{ a previous encryption query } {}^{(r^j, m^j)} \text{ s.t. } \exists \lambda \text{ s.t. } k_0^i = k_\lambda^j \\ \text{or } \exists \text{ a previous decryption query } {}^{c^j} \end{array} \right\}$$

where among encryption queries j can run only among the encryption queries A has done before the ith decryption query, (which are $\leq q_E$) and with the first $i-1$ decryption queries.

Bounding $\Pr[KC]^i$**.** Since k_0^i is randomly picked and since there are at most $i-1+q_E$ possible values that it cannot have, we can bound $\Pr[KC^i] \leq \frac{q_E(L+1)+(i-1)(L+1)}{2^n}$.

Bounding $\left|\Pr[CO_2^i] - \Pr[CO_3^i]\right|$**.** Since D^i is a $(2, t+(q+1)(t_{\mathsf{H}}+(2L+1)t_{\mathsf{F}}) + t_{\mathsf{f}(q+1)})$-$\mathsf{PRF}$ adversary and $\mathsf{F}(\cdot)$ is a $(2, t + (q+1)(t_{\mathsf{H}} + (2L+1)t_{\mathsf{F}}) + t_{\mathsf{f}(q+1)}, \epsilon_{\mathsf{PRF}})$-$\mathsf{PRF}$ secure, thus

$$\left|\Pr[CO_2^i] - \Pr[CO_3^i]\right| = \left|\Pr[\mathsf{D}^{\mathsf{F}_{k_0^i}(\cdot)} \Rightarrow 1] - \Pr[\mathsf{D}^{\mathsf{f}^i(\cdot)} \Rightarrow 1]\right| \leq \epsilon_{\mathsf{PRF}}.$$

Bounding $\Pr[CO_3^i]$**.** We can compute $\Pr[CO_3^i] = 2^{-n}$, since $\mathsf{f}_0^i(\cdot)$ is a random function, thus, the probability that $\tilde{c}_0^i$ is equal to c_0^i is equal to $|\mathcal{T}|^{-1}$ with $\mathcal{T}$ the target space of the function $\mathsf{f}_0^i(\cdot)$.

We are finally able to bound $\Pr[F_i^1]$:

Bounding $\Pr[F_i^1]$**.** If D^i simulates correctly event F_i^1 happens iff event CO_i^2 happens, thus

$$\Pr[F_i^1] \leq \Pr[KC^i] + \Pr[CO_2^i] \leq \Pr[KC^i] + \Pr[CO_3^i] + \left|\Pr[CO_2^i] - \Pr[CO_3^i]\right|$$

Using the previous results, we obtain:

$$\Pr[F_i^1] \leq \Pr[KC^i] + \Pr[CO_3^i] + \left|\Pr[CO_2^i] - \Pr[CO_3^i]\right| \leq$$
$$\frac{q_E(L+1)+(i-1)(L+1)}{2^n} + 2^{-n} + \epsilon_{\mathsf{PRF}} =$$
$$\frac{q_E(L+1)+(i-1)(L+1)+1}{2^n} + \epsilon_{\mathsf{PRF}}$$

Bounding $\Pr[F_i^2]$**.** Now the partial ciphertext $(c_0^i, ..., c_{l^i}^i)$ is not fresh and it has been obtained in encryption[58] and/or decryption queries. Now we can reuse everything we used to bound $\Pr[F_i^1]$ since c_{l^i+1} is fresh with respect to the partial ciphertext $(c_0^i, ..., c_{l^i}^i)$ and thus k_0^i is fresh. Consequently, it is uniformy picked at random. Still, $\Pr[KC^i] \leq \frac{q_E(L+1)+(i-1)(L+1)}{2^n}$. Consequently we can bound

$$\Pr[F_i^2] \leq \Pr[KC^i] + \Pr[CO_3^i] + \left|\Pr[CO_2^i] - \Pr[CO_3^i]\right| \leq$$
$$\frac{q_E(L+1)+(i-1)(L+1)}{2^n} + 2^{-n} + \epsilon_{\mathsf{PRF}} =$$
$$\frac{q_E(L+1)+(i-1)(L+1)+1}{2^n} + \epsilon_{\mathsf{PRF}}$$

Bounding $\Pr[F_i]$**.** Since $\Pr[F_i] \leq \max\{\Pr[F_i^1], \Pr[F_i^2]\}$ we have bounded

$$\Pr[F_i] \leq \frac{q_E(L+1)+(i-1)(L+1)+1}{2^n} + \epsilon_{\mathsf{PRF}}$$

Bounding $\left|\Pr[\mathsf{A}\textbf{ wins Game } 2^{i-1}] - \Pr[\mathsf{A}\textbf{ wins Game } 2^i]\right|$**.** We have already proved that $\left|\Pr[\mathsf{A} \text{ wins Game } 2^{i-1}] - \Pr[\mathsf{A} \text{ wins Game } 2^i]\right| \leq \Pr[F_i]$ thus we obtain that

$$\left|\Pr[\mathsf{A} \text{ wins Game } 2^{i-1}] - \Pr[\mathsf{A} \text{ wins Game } 2^i]\right| \leq \Pr[F_i] \leq$$

[58] This may be done finding at most q_E different keys $k_0^1, ..., k_0^{q_E}$ s.t. $\mathsf{F}_{k_0^1}(p_B) = ... = \mathsf{F}_{k_0^{q_E}}(p_B)$ [very unlikely, but possible] and choosing $m_j^i = m_j^1 \oplus \mathsf{F}_{k_j^1}(p_B) \oplus \mathsf{F}_{k_j^i}(p_B)$. This can be done since the adversary chooses the k_0^is so he can anticipates all the values $\mathsf{F}_{k_j^i}(p_B)$ for every i and j. This is related to what we explain for the tidiness

$$\frac{q_E(L+1)+(i-1)(L+1)}{2^n}+\epsilon_{\mathsf{PRF}}$$

Bounding $|\Pr[\mathsf{A}$ **wins Game** $3]-\Pr[\mathsf{A}$ **wins Game** $2]|$**.** Since Game 2 is Game 2^0 and Game 3 is Game 2^{q_D+1} we have:

$$|\Pr[\mathsf{A} \text{ wins Game } 3]-\Pr[\mathsf{A} \text{ wins Game } 2]| \leq$$

$$\sum_{i=1}^{q_D+1}\left|\Pr[\mathsf{A} \text{ wins Game } 2^{i-1}]-\Pr[\mathsf{A} \text{ wins Game } 2^{i}]\right| \leq$$

$$\sum_{i=1}^{q_D+1}\left(\frac{q_E(L+1)+(i-1)(L+1)+1}{2^n}+\epsilon_{\mathsf{PRF}}\right)=$$

$$\frac{(q_D+1)q_E(L+1)}{2^n}+\frac{\frac{q_D(q_D+1)(L+1)}{2}}{2^n}+\frac{q_D+1}{2^n}+(q_D+1)\epsilon_{\mathsf{PRF}}=$$

$$\frac{(q_D+1)(L+1)(q+q_E)}{2^{n+1}}+\frac{q_D+1}{2^n}+(q_D+1)\epsilon_{\mathsf{PRF}}$$

Bounding $\Pr[E_3]$**.** Since all fresh decryption queries are deemed invalid in Game 3 there is no possibility that the adversary wins such Game thus $\Pr[E_3]=0$.

We now are able to finally conclude the proof.
Bounding $\Pr[E_0]$**.** Using all the bounds about the event E_i $(i=0,...,3)$ we obtain that

$$\Pr[E_0]\leq\epsilon_{\mathsf{sTPRP}}+\epsilon_{\mathsf{CR}}+\frac{(q_D+1)(L+1)(q+q_E)}{2^{n+1}}+\frac{q_D+1}{2^n}+(q_D+1)\epsilon_{\mathsf{PRF}}$$

Observation on the bound Since we have supposed that all k_0 used in the decryption query are different (and different from all other ephemeral keys), the previous bound covers also the difference $\epsilon_{\mathsf{sTPRP}}\setminus\epsilon_{\mathsf{tprf}}$ if we see F^* as a tweakable PRF, (that is, if $\mathsf{F}^*_k(\cdot,\cdot)$ is indistinguishable from a random function with the same signature). □

Observation on the proof The constraint, which we use for F, is stronger than what is, in reality, necessary, but, since, anyway we need that F is a $(2,t,\epsilon_{\mathsf{PRF}})$ to prove the pAE security (see the eprint version [28]) we assume this hypothesis. On the other hand, we would have been able to prove the CIML2 security, even if we had replaced F with the identity. The real security property that we need for the commitment $c_0=c_0(k_0)$ is the following:

$$\forall c_0\in\{0,1\}^n\ \Pr[\tilde{c}_0(k_0)=c_0 \text{ if } k_0\xleftarrow{\$}\mathcal{K}]\leq\epsilon_{\mathsf{COM}}$$

where $\tilde{c}_0(k_0)$ is the correct commitment for the ephemeral key k_0 which must be picked uniformly at random (in the bound of Thm. 9 we should replace ϵ_{PRF} with ϵ_{COM}). That is, given a commit c_0, the probability that it is correct for a random ephemeral key k_0, is negligible. In particular, this is true if the commitment is given by a PRF or by any injective function.
This also allows us to replace the term $\frac{(q_D+1)(L+1)(q+q_E)}{2^{n+1}}$ with $\frac{(q_D+1)(q+q_E)}{2^{n+1}}$, that is, we suppose only that all k_0 used in decryption queries are different.

B.8 Proof for HBC2 based on sUL

Before proving the theorem, we need the following technical lemma:

Lemma 1. *If* $2q \leq n$,

$$\sum_{s=1}^{q}(s-1)\cdot \Pr[\mathsf{MultiColl}(n,q)\geq s] \leq \frac{1}{n}\binom{q}{2}\left(1+\frac{2q}{n}\right).$$

Proof. Looking at the generic term for $s \geq 3$ after applying the Thm. 2 leads to

$$\frac{s-1}{n^{s-1}}\binom{q}{s} \leq \frac{1}{n^{s-2}}\binom{q}{s-1}\cdot\frac{q}{n} \leq \frac{1}{n}\binom{q}{2}\cdot\left(\frac{q}{n}\right)^{s-2}.$$

Then, the whole sum is upper-bounded by

$$\frac{1}{n}\binom{q}{2}\cdot\sum_{s=2}^{q}\left(\frac{q}{n}\right)^{s-2} \leq \frac{1}{n}\binom{q}{2}\frac{n}{n-q} = \frac{1}{n}\binom{q}{2}\left(1+\frac{q}{n-q}\right).$$

Hence, the result since $q \leq n-q$. □ □

Theorem 10. *Let* $\mathsf{F}^* : \mathcal{K}^* \times \mathcal{B}^* \longmapsto \mathcal{B}^*$ *be a* $(q_M, q_V, q_L, t, \epsilon_{\mathsf{sUL}})$*-strongly unpredictable block cipher in the presence of leakage, and* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \longmapsto \mathcal{B}'$ *be a hash function modeled as a random oracle that is queried at most* q_H *times. Let* $\mathcal{B}^* = \mathcal{B}' = \{0,1\}^n$ *and* $\mathcal{HM} = \{0,1\}^*$.
Then, HBC2 *is a* $(q_M, q_V, q_{\mathsf{L}}, t, \epsilon)$*-strongly unforgeable MAC in the unbounded leakage setting, with* $\mathsf{L}^* = (\mathsf{L}_M, \mathsf{L}_V)$ *defined above, where*

$$\epsilon \leq (q_H+q_V+1)(q_V+1)\epsilon_{\mathsf{sUL}} + (q_H+q_M+q_V+1)^2/2^n,$$

and $t_H(q_H+q_M+q_V+1)+(q_M+q_L-q)t_{\mathsf{F}}+(q_V+q)t_{\mathsf{F}^{-1}} \leq t$ *for any* $q \leq q_L$*, and where we assume that all the* H*-query involved in the* q_L *queries are already among the* q_H *queries, and if* $q_V \leq q_H$ *(which can be artificially fulfilled at the end of the experiment).*

Proof. To prove the theorem, we use a sequence of games. Given an adversary A, we start with Game 0 which is the $\mathsf{FORGEL}^{\mathsf{suf-vcma-L}}_{\mathsf{A},\mathsf{HBC2},\mathsf{L}^*}$ experiment and we end with a game where all the leaking verification queries deem the given input pair (m_i, τ_i) invalid, including the last verification at the finalization which is the $(q_V + 1)$-th verification query by convention.

Game 0. Let E_0 be the event that the adversary $\mathsf{A}^{\mathsf{L}^*}$ wins this game. That is, the output of the experiment is 1.

Game 1. We introduce a failure event F_1 with respect to Game 0, where F_1 occurs if among the at most $(q_H + q_M + q_V + 1)$ hash computations there is at least one collision. In Game 1, if F_1 occurs, we abort the game and return 0. We let E_1 be the event that the adversary $\mathsf{A}^{\mathsf{L}^*}$ wins this game.

Bounding $|\Pr[E_0] - \Pr[E_1]|$. Since Game 0 and Game 1 are identical as long as F_1 does not occur, if $Q = q_H + q_M + q_V + 1$, we have

$$|\Pr[E_0] - \Pr[E_1]| \leq \Pr[F_1] \leq Q(Q+1)/2^{n+1}.$$

Note: from now on, A wins if τ never appears in a leaking tag query. Moreover, $\tilde{h} = \mathsf{F}_k^{-1}(\tau)$ is fresh when τ appears in a leaking verification query for the first time if m was never used as input of H at that time. (See above.)

Game 2. We modify the winning condition of the previous game. In the finalization, once A outputs (m, τ) we say that A does not win and return 0 if A fails as before or if m appears as an input of H before the first apparition of τ during a leaking verification query. If we call F_2 the event that makes the adversary winning in Game 1 but loosing in Game 2, we have $|\Pr[E_2] - \Pr[E_1]| \leq \Pr[F_2]$, where E_2 is the event that A wins in this game.

Bounding $\Pr[F_2]$. If we call V_i the event that (m, τ) appears for the first time in the i-th leaking verification query (m^i, τ^i), we just have to bound $\Pr[F_2 \cap V_i]$, for all $i = 1$ to $q_V + 1$. By considering all the input-output pairs defined by H before the i-th leaking verification query, except those defined during a leaking tag query, we can build straightforwards reduction to the $\mathsf{SUL2}$-security of F. We thus have, $\Pr[F_2 \cap V_i] \leq (q_H + q_V + 1)\epsilon_{\mathsf{sUL}}$ and finally

$$\Pr[F_2] = \sum_{i=1}^{q_V+1} \Pr[F_2 \cap V_i] \leq (q_H + q_V + 1)(q_V + 1)\epsilon_{\mathsf{sUL}}.$$

Note: in Game 2, the adversary wins only if τ appears before m and τ first appears in a leaking verification. The random value of $\mathsf{H}(m)$ is still undefined at that time.

Game 3. In this game we follow the specification of $\mathsf{FORGEL}^{\mathsf{suf-vcma-L}}_{\mathsf{A,HBC2,L}^*}$ except that we always output 0 at the end of the game.

Bounding $|\Pr[E_3] - \Pr[E_2]| = \Pr[E_2]$. From the last note, we know that $\tilde{h} = \mathsf{F}_k^{-1}(\tau)$ must be reached from a new computation of H. Since any fresh H evaluation results in a uniform output which is thus independent of the view of τ, $\Pr[H(m') = \tilde{h}] = 1/2^n$ for all hash evaluations on some m' appearing after τ in a H-query or in a LVrfy query. But the number of targets $\tilde{h}$ during the game is actually the number of different τ' in the LVrfy queries when considering the hash evaluations after each such τ'. Then $\Pr[E_2] \leq q_V(q_H + q_V)/2^n$.

To summarize, we have

$$\Pr[E_0] \leq (q_H + q_V + 1)(q_V + 1)\epsilon_{\mathsf{sUL}} + \frac{q_V(q_H + q_V)}{2^n} + \frac{Q(Q+1)}{2 \cdot 2^n}$$

from which the result follows as $2q_V(q_H + q_V) \leq Q(Q-1)$. □
□

B.9 Proof for HTBC based on sUL

Theorem 11. *Let* $\mathsf{H} : \mathcal{KH} \times \mathcal{HM} \longmapsto \mathcal{B}'$ *be a hash function modeled as a random oracle, and* $\mathsf{F}^* : \mathcal{K} \times \mathcal{TW} \times \mathcal{B} \longmapsto \mathcal{B}$ *be a* $(q_T, q_V, q_\mathsf{L}, t, \epsilon_{\mathsf{sUL}})$*-strongly unpredictable tweakable block cipher with leakage* $\mathsf{L} = (\mathsf{L}_{\mathsf{Eval}}, \mathsf{L}_{\mathsf{Inv}})$*. Let* $\mathcal{HM} = \{0,1\}^*$*,* $\mathcal{B}' = \mathcal{TW} \times \mathcal{B}$ *and* $\mathcal{B} = \{0,1\}^n$*.*
Then, HTBC *is a* $(q_T, q_V, q_\mathsf{H}, q_\mathsf{L}, t, \epsilon)$-$\mathsf{suf}$-$\mathsf{L2}$ *strongly unforgeable* MAC *with unbounded leakage function pair* $\mathsf{L}^* = (\mathsf{L}_M, \mathsf{L}_V)$ *as defined above, where*

$$\epsilon \leq \frac{(q_H + q_T + q_V)^2}{2^{2n}} + (q_V + 1) \cdot \epsilon_{\mathsf{sUL}} + \frac{q_H^2 q_V}{2^n} \cdot \epsilon_{\mathsf{sUL}} + \frac{q_V(q_H + q_V)}{2^{2n}},$$

and $t_H(q_H + q_T + q_V + 1) + (q_T + q_L - q)t_\mathsf{F} + (q_V + q)t_{\mathsf{F}^{-1}} \leq t$ *for any* $q \leq q_L$*, and where we assume that all the* H*-query involved in the* q_L *queries are already among the* q_H *queries, as long as* $4 \leq q_H + q_T + q_V$*,* $4q_V \leq q_H$ *and* $10q_H \leq 2^n$*.*

Proof. To prove the theorem, we use a sequence of games. Given an adversary A, we start with Game 0 which is the $\mathsf{FORGEL}^{\mathsf{suf-vcma-L}}_{\mathsf{A,HTBC,L}^*}$ experiment and we end with a game where all the leaking verification queries

(m^i, τ^i) are deemed invalid, including the last and (q_V+1)-th verification which tests the validity of the potential forgery (m, τ). In the sequel, we note $\mathsf{H}(m) = h_1 \| h_2$.

Game 0. Let E_0 be the event that the adversary $\mathsf{A}^{\mathsf{L}^*}$ wins this game. That is, the output of the experiment is 1.

Game 1. We introduce a failure event F_1 for Game 0, where F_1 occurs if among the at most $(q_H + q_T + q_V + 1)$ distinct hash computations there is at least one collision. In Game 1, if F_1 occurs, we abort the game and return 0. We let E_1 be the event that the adversary $\mathsf{A}^{\mathsf{L}^*}$ wins this game.

Bounding $|\Pr[E_0] - \Pr[E_1]|$. Since Game 0 and Game 1 are identical as long as F_1 does not occur, and $4 \leq q_H + q_T + q_V$, we have

$$|\Pr[E_0] - \Pr[E_1]| \leq \Pr[F_1] \leq (q_H + q_T + q_V)^2 / 2^{2n}.$$

Note: from now on, in the case of a winning adversary, no TBCtriple of the form $(\star, h_2, \tau)$ appears when answering to a LMac query.

Game 2. We modify the winning condition of the previous game. In the finalization, once A outputs (m, τ) we say that A does not win and returns 0 if A fails as before or if $\mathsf{H}(m) = h_1 \| h_2$ appears before the first apparition of (h_2, τ) as input to F_k^{-1} in a leaking verification query. If we call F_2 the event that makes the adversary winning in Game 1 but loosing in Game 2, we have $|\Pr[E_2] - \Pr[E_1]| \leq \Pr[F_2]$, where E_2 is the event that A wins in this game.

Bounding $\Pr[F_2]$. If we call V_i the event that the first time (h_2, τ) appears during the computation of the answer to a leaking verification query is in the i-th leaking verification query (m^i, τ^i), we just have to bound $\Pr[F_2 \cap V_i]$, for all $i = 1$ to $q_V + 1$. The event $F_2 | V_i$ means m appears before m_i ($m = m_i$ included) and, if $\mathsf{H}(m^i) = h_1^i \| h_2^i$, we have $(h_2^i, \tau^i) = (h_2, \tau)$ while this never happens before in a previous LVrfy query. If $m = m_i$, $F_2 | V_i$ reduces to the strong unpredictability of F with leakage. Indeed, it is easy to emulate LMac and LVrfy from LEval and LInv and to output the final triple (h_1^i, h_2^i, τ^i) against F at the time the i-th query is made in order to win with the same probability since (h_1^i, h_2^i) was never a LEval query (as m was never a LMac query in F_2 and there is no more collision since Game 1) and (h_2^i, τ^i) was never a LInv query before by definition of V_i. However, when $m \neq m^i$, we cannot follow such a simple strategy. Since $m \neq m^i$ we know that $h_1 \neq h_1^i$ as there is no collision by assumption and $(h_1^i, h_2^i = h_2, \tau^i = \tau)$ is not a winning triple against F. Of course, we could simply make an LInv query on (h_2^i, τ^i) to emulate the

i-th LVrfy query but in that case we will actually "consume" our chance to win against F with (h_1, h_2, τ) because the input (h_2, τ) for inversion will be no more fresh after answering the i-th LVrfy query, and the reduction will fail. Fortunately, at the time we should emulate the i-th LVrfy query, we now that the history of the hash evaluations already contains the forged message m, that h_2 collides with h_2^i. Moreover, that τ^i is its valid tag. We thus have to make several hybrids on the collisions with h_2^i to anticipate the right m' to win against F with the triple $(h'_1, h'_2 = h_2^i, \tau^i)$. This number of hybrids is the number of collisions with h_2^i which remains small with high probability as it implies multi-collision in $\{0,1\}^n$. We note that only the hash evaluations of a H-query or of an LVrfy query matter. In the following, we write $\mathsf{H}_2(m') = h'_2$ when $\mathsf{H}(m') = h'_1 \| h'_2$ for some h'_1.

Concretely, if q_i is the number of H evaluations made from all the H-queries and the LVrfy queries until the i-th LVrfy query, including $\mathsf{H}(m^i)$, and if S_i is the random variable counting the number of H_2-collisions with h_2^i, for $i = 1$ to q_V, we have

$$\begin{aligned}\Pr[F_2 \cap V_i] &= \sum_{s=1}^{q_i} \Pr[F_2 \mid V_i \cap S_i = s] \cdot \Pr[V_i \cap S_i = s] \\ &\leq \Pr[F_2 \mid V_i \cap S_i = s \cap \textstyle\bigcup_{s=1}^{q_i} H_{i,s}] \\ &\quad + \sum_{s=2}^{q_i} \sum_{j=1}^{s-1} \Pr[F_2 \mid V_i \cap S_i = s \cap H_{i,j}] \cdot \Pr[\mathsf{H}_2\text{-}\mathsf{Coll}(q_i) \geq s]\end{aligned}$$

where $H_{i,j}$ is the event that among the s distinct messages that H_2-collide on h_2^i the j-th one is the forged message m. By convention, we always see m^i as the s-th and last such message even if the computation of $\mathsf{H}(m^i)$ appears earlier than in the i-th LVrfy query[59]. The case $\bigcup_{s=1}^{q_i} H_{i,s}$ thus corresponds to $m^i = m$ and the related probability in the expression is upper-bounded by ϵ_{sUL} as explained above. Moreover, for each $j = 1$ to $s-1$, it is now easy to see that the event $F_2 \mid V_i \cap S_i = s \cap H_{i,j}$ reduces to sUL by using the j-th message and τ^i as our guess against the TBC.

[59] This assumption is without loss of generality as the enumeration only matters in the reduction at the time we get the adversary's i-th LVrfy query (m^i, τ^i). The choice of (which is) the j-th message colliding on h_2^i can be made once the s messages are known. At that time, if $(h_2^i, \tau^i) = (h_2, \tau)$ as implied by V_i, $\mathsf{F}_k^{-1}(h_2, \tau)$ was never computed anyway.

Therefore,

$$\Pr[F_2 \cap V_i] \leq \epsilon_{\mathsf{sUL}} + \epsilon_{\mathsf{sUL}} \sum_{s=2}^{q_i} (s-1) \cdot \Pr[\mathsf{H}_2\text{-}\mathsf{Coll}(q_i) \geq s]$$
$$\leq \epsilon_{\mathsf{sUL}} + \epsilon_{\mathsf{sUL}} \cdot \frac{1}{2^n} \binom{q_i}{2} \left(1 + \frac{2q_i}{2^n}\right)$$

by lemma 1, since $2q_i \leq 2(q_H + q_V) \leq 2^{n-2} \leq 2^n$ by assumption on the number of queries. In addition, $1 + 2q_i/2^n \leq 5/4$. Summing on all the i's, with $\Pr[F_2 \mid V_{q_V+1}] \leq \epsilon_{\mathsf{sUL}}$, gives

$$\Pr[F_2] \leq (q_V + 1) \cdot \epsilon_{\mathsf{sUL}} + \epsilon_{\mathsf{sUL}} \cdot \frac{1}{2^n} \cdot \frac{5}{4} \cdot \sum_{i=2}^{q_V} \binom{q_i}{2}.$$

Some basic computation shows that $\sum_{i=1}^{q_V} \binom{q_i}{2} \leq \sum_{i=1}^{q_V} \binom{q_H+i}{2} \leq \frac{1}{2} q_H^2 q_V (1 + \frac{2q_V}{q_H})$, if $q_V \leq q_H$. But then, as $q_V \leq q_H/4$ by assumption, we have

$$\Pr[F_2] \leq (q_V + 1) \cdot \epsilon_{\mathsf{sUL}} + \frac{q_H^2 q_V}{2^n} \cdot \epsilon_{\mathsf{sUL}}.$$

Game 3. In this game we follow the specification of $\mathsf{FORGEL}^{\mathsf{suf-vcma-L}}_{\mathsf{A},\mathsf{HBC2},\mathsf{L}^*}$ except that we always output 0 at the end of the game.

Bounding $|\Pr[E_3] - \Pr[E_2]| = \Pr[E_2]$. We end by showing that winning while the TBCinput (h_2, τ) for inversion appears when answering a leaking verification query before the computation of $\mathsf{H}(m)$ is negligible. For each (h_2^i, τ^i) that appears when answering a leaking verification query and *before* m' the only way for the adversary to win with the pair (m', τ') is to "hope" that $\mathsf{H}(m') = \tilde{h}_1^i \| h_2^i$. However, before the computation of $\mathsf{H}(m')$ its value remains independent of the adversary's view. Therefore, the probability that the random output of $\mathsf{H}(m')$ hits the full target $\tilde{h}_1^i \| h_2^i$ is $1/2^{2n}$. We now count the number of tries a winning adversary can make in it that case. Clearly, we do not have to count the tries related to any LMac. Considering the event V_i as in the previous analysis of Game 2, in $E_2 \cap V_i$ there at most $q_H + q_V + 1 - i$ hash evaluations left after the i-th LVrfy query. Then, $\Pr[E_2 \mid V_i] \leq (q_H + q_V)/2^{2n}$ for $i = 1$ to q_V. Note that $\Pr[E_2 \mid V_{q_V+1}] = 0$ by definition. Finally, we get

$$\Pr[E_2] \leq \frac{q_V(q_H + q_V)}{2^{2n}}.$$

Hence, the bound of the theorem.

□