Security & Indistinguishability in the Presence of Traffic Analysis

Cristina Onete[1] Daniele Venturi[2]

[1]Darmstadt University of Technology & CASED, Germany
www.minicrypt.de
[2]SAPIENZA University of Rome, Italy

Abstract. Traffic analysis (TA) is a powerful tool against the security and privacy of cryptographic primitives, permitting an adversary to monitor the frequency and timing characteristics of transmissions in order to distinguish the senders or the receivers of possibly encrypted communication. Briefly, adversaries may leak implementation-specific information even for schemes that are provably secure with respect to a classical model, resulting in a breach of security and/or privacy.

In this work we introduce the notion of *indistinguishability in the presence of traffic analysis*, enhancing *any* classical security model such that no adversary can distinguish between two protocol runs (possibly implemented on different machines) with respect to a TA oracle (leaking information about each protocol run). This new notion models an attack where the adversary taps a single node of in- and outgoing communication and tries to relate two sessions of the same protocol, either run by two senders or for two receivers.

Our contributions are threefold: (1) We first define a framework for indistinguishability in the presence of TA, then we (2) fully relate various notions of indistinguishability, depending on the adversary's goal and the type of TA information it has. Finally we (3) show how to use our framework for the SSH protocol and for a concrete application of RFID authentication.

Keywords. foundations, traffic analysis, provable security, SSH

1 Introduction

Cryptographic protocols aim at securing communication between two or more parties which interact via an insecure channel. In classical security models, adversaries may observe communication or interact with honest and dishonest parties in certain, fixed ways. Provable secure in this sense is essential, but not sufficient, particularly due to the increasing popularity and efficiency of side-channel attacks. Many recently proposed attacks, which effectively break "provably secure" schemes, exploit the physical, implementation-specific characteristics of a primitive, e.g. running-time [28, 10], power consumption [29], and electromagnetic radiation [37, 21, 30].

In order to provide for side-channel attacks, one must build primitives that retain provable security even when run on machines prone to (as wide a class as possible of) side-channels. Starting from the seminal works of Micali and Reyzin [32] and Ishai et al. [24], great progress has been made in the area of leakage resilience, where the main idea is to "ease" an adversary $\mathcal{A}$'s task of winning a game against a primitive Π by giving $\mathcal{A}$ access to a leakage oracle. The adversary can forward *arbitrary* functions $f(\cdot,\cdot)$ and input values x to a leakage oracle, which evaluates $f(sk, x)$ for some secret key sk. By using game information *and* evaluations of the functions f, the adversary must win the security game. The choice of f is restricted to exclude trivial cases, e.g. the adversary could ask for the key itself. In fact, several existing models allow for positive results (see e.g., [19, 2, 36, 27, 34, 14, 41, 17, 18, 20, 8, 9, 7]).

This paper shows a different approach from leakage resilience. Instead of querying functions f of sk, adversaries tap a single in- and outgoing communication node running a primitive. Receivers in communication are called *users*, associated with some global parameters, and senders are implementations of the primitive. A sender-receiver pair, their parameters, and the interchanged communication define *instantiations*. Receivers can be viewed as parties in a protocol, whereas senders are applications/machines of a single sending party, running the same primitive (e.g. encryption can be used from an SSH terminal sender and an eMail client sender). For each instantiation of the security game, the adversary queries a predefined leakage oracle, *not* necessarily depending on the secret key. This oracle models traffic analysis of the tapped node, i.e. fingerprinting in- and outgoing communication, timing processing delays, counting the messages received by the node, etc.

Consider the case of a military basis (the tapped communication node) in contact with various military camps (receivers). The basis uses multiple transceivers (senders), either domestic or belonging to incoming, new personnel. An adversary may now want to know which receivers interact with the basis, or whether new personnel (new senders) has arrived to the basis. To achieve this, the adversary may "fingerprint" the communication by detecting transmission-specific characteristics of senders and receivers. If this is achievable, an adversary can accurately distinguish between two senders or between two receivers of a message.

There are three fundamental security notions involved in our framework. We start from classical security games in a special template, called a canonical form. Here, adversaries $\mathcal{A}_{small}$ attempt to break the appropriate security notion (which could be indistinguishability, pseudo-randomness, privacy, etc.) in a classical way, having access to a number of primitive-specific oracles. This notion is called the *classical security* of the primitive.

As a first step, classical security can be extended to give $\mathcal{A}_{small}$ access to a traffic analysis (TA) oracle on top of game-specific oracles. This idea appears for a particular model (IND-CCA security) in [35]. Our general TA oracle models analysing transmission patterns, thus leaking data. Now $\mathcal{A}_{small}$ attempts to break classical security, but with an additional oracle. The enhanced model, called *security in the presence of traffic analysis*, covers, though it is not limited to, non-adaptive leakage resilience (see Appendix A).

Finally, the main achievement of this paper is to further enhance security by considering adversaries $\mathcal{A}_{big}$ that may create instantiations of classical security games (this adversary $\mathcal{A}_{big}$ taps a node's transmissions in several sender-receiver sessions) and may play these instantiations as in the classical game. However, $\mathcal{A}_{big}$ also has access to the TA oracle, and it aims to distinguish between two instantiations for different senders (was the transmission sent from the SSH terminal or from the eMail client?), or for different receivers (was the transmission sent to Alice or for Bob?). This new notion is called *indistinguishability in the presence of traffic analysis*. Note that by modelling indistinguishability of senders or of receivers, we cover classical traffic analysis issues, which focus on anonymous routing. We describe in more detail how indistinguishability compares with existing concepts in the literature in Appendix A.

Our notion is used as follows: for a pre-set, non-adaptively chosen traffic analysis oracle Θ and some primitive Π, we consider instantiations of the appropriate classical security notion for Π. An adversary with access to the instantiations and to Θ, must now distinguish between two instantiations. Note that such an adversary will trivially win if it breaks either classical security, or security in the presence of TA. However, the reverse is not true: indistinguishability is a strictly stronger notion, as we show in Section 3 and in Appendix C. In fact, indistinguishability combines two notions, security *and* sender/receiver privacy in a single model; this saves us from having to model privacy for each notion, and allows us to write a single proof of both security and indistinguishability. For applications which do not require such strong security, the proofs can be done in the weaker model of security in the presence of TA (if the TA oracle always returns $\perp$, this is equivalent to classical security); however, high-level security applications in say, the military, should consider attaining the strong indistinguishability notion rather than just classical security.

A Running Example. Although our framework is general, we give an intuition of it by means of a running example for public key encryption (PKE), which is formally a tuple of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ for which the classical security game is indistinguishability against chosen ciphertext attacks (IND-CCA) as described in

Appendix B. This game has a so-called *canonical* form (see Section 2). We consider two receivers for a PKE scheme, Alice and Bob, both communicating with a common node, Carol, in the presence of an adversary Eve, who taps Carol's transmissions over an insecure channel. Alice and Bob are associated with tuples (sk_A, pk_A) and (sk_B, pk_B); Carol uses various senders to encrypt plaintexts m under either pk_A or pk_B, obtaining ciphertexts c. Eve intercepts and possibly changes such values of c. In the *classical* IND-CCA security game, Eve must break the PKE scheme: given ciphertext c, which is the encryption of either message m_0 or m_1 (both chosen by Eve) under $pk \in \{pk_A, pk_B\}$, and given pk, Eve wants to distinguish which of the two plaintexts was encrypted. Decryption Dec requires the secret key corresponding to pk; Eve does not have the secret key (she just knows pk), but she may submit non-challenge ciphertexts c_i to a decryption oracle, which outputs the corresponding plaintexts m_i. Intuitively, Eve wins against IND-CCA if it learns at least one bit of plaintext-information.

For *indistinguishability in the presence of traffic analysis*, Eve does not need to break the encryption scheme, and she does not have access only to a public key and a decryption oracle. Eve can additionally query a TA oracle, leaking additional information from the tapped channel (e.g. by measuring processing time, Eve could learn a part of the randomness used to encrypt). In our model, Carol sends messages to Alice and Bob in various sessions (determined by the receiver – Alice or Bob – and the sender – Carol's machine/application). Eve's new goal is one of the following: either (a) to distinguish, given a ciphertext c generated for a particular public key pk, which sender Carol used, or (b) to distinguish, given ciphertext c generated for either pk_A or pk_B, who the receiver was (Alice or Bob).

In the first case, called *sender* Indistinguishability in the presence of Traffic Analysis, Eve must learn which application or machine has generated c. To see why this attack is motivated by practice, imagine that Carol encrypts eMails to Alice regarding an auction. Carol eMails Alice for every bid. At the same time, Carol also encrypts information to Alice over SSH for a different purpose. If Eve can distinguish whether Carol's ciphertexts were encrypted in the eMail client or in SSH, she will know if Carol keeps bidding on the auction (and may try to outbid her, or she may even delay the encrypted eMail ciphertext till the auction is finished). The second notion we consider, called *receiver* Indistinguishability in the presence of Traffic Analysis, models the case where Eve must learn who is the receiver of the ciphertext, i.e. either Alice or to Bob. This is important in our auction scenario if Carol now organises the auction and, at the end of the bidding process, Carol contacts the winner to confirm the purchase. Now Eve should not learn who has won the auction.

1.1 Our Contributions

The Models. In this paper we formalise indistinguishability (in the presence of traffic analysis) between runs of a security game for a cryptographic primitive. Here, traffic analysis is abstractly modelled by an oracle leaking some pre-set sender- and receiver-specific information. For cryptographic primitives as defined in Section 2, we consider classical, so-called canonical security games, played between adversaries $\mathcal{A}_{small}$ and challengers $\mathcal{G}$. Separate sessions of the classical security game, containing an adversary instance $\mathcal{A}_{small}$, a challenger instance $\mathcal{G}$, and additional parameters (UPar, IPar), are called *instantiations*. Recall that we call receivers *users*: the parameters UPar associated with them are *global* (they are run at game setup). Sender-specific *local* parameters IPar are created after setup, including values used for challenge ciphertexts, pseudorandom output, etc. Two instantiations are *same-user* if they share their global parameters, and *mixed-user* otherwise. In our framework, same-user instantiations model *implementations of a cryptographic primitive on different senders* e.g. an application (an SSH-terminal or an eMail client) or a machine (a laptop). A sender is thus simply a set of parameters corresponding to the application or machine.

In the indistinguishability model, adversaries $\mathcal{A}_{big}$ create many instantiations of the security game for a given primitive and must distinguish between two instantiations. In order to do so they either manipulate $\mathcal{A}_{small}$ in each instantiation, or they query a pre-set traffic analysis (TA) oracle which, on input an instantiation, leaks some information. The TA oracle may be restricted, i.e. yielding only information about local parameters, or unrestricted, if it leaks information about both local and global parameters. The strength of the TA oracle reflects the physical capabilities of an adversary, which may, for instance, only be able to learn a single bit of

local input, or on the contrary, it may fully fingerprint either the receiver or the sender of a transmission.

The aim of $\mathcal{A}_{big}$ can be either (a) to distinguish between two same-user instantiations (sender indistinguishability in the presence of traffic analysis – indSenderTA), or (b) to tell apart two mixed-user instantiations (receiver indistinguishability in the presence of traffic analysis, indReceiverTA). The former goal models attacks trying to distinguish the sender (i.e. the implementation running the game), whereas the latter models trying to distinguish the intended receiver. We show how our notion relates to existing work in Appendix A.

Relating the Models. We formally prove that indistinguishability for *unrestricted* traffic analysis oracles is strictly stronger than indistinguishability for *projections* of such oracles; these projections are *restricted* TA oracles. We also show that *indistinguishability* in the presence of TA is strictly stronger than *security* in the presence of the same oracle (see above). Finally, we show that *sender* and *receiver* indistinguishability in the presence of TA are independent. In particular, the ability to tell apart two same-user instantiations does *not* guarantee the ability to distinguish between two mixed-user instantiations (one's ability to distinguish the latter depends on the TA oracle output). A complete diagram relating these notions appears in Figure 2.

Using our Framework. Besides providing a formal definition for indistinguishability in the presence of traffic analysis, we apply our framework in two different scenarios.

Encode-then-Encrypt & MAC. Recently Paterson et al. [16, 35] have shown that the "provably secure" SSH-$NPC protocol is vulnerable to implementation-specific information not included in IND-CCA security (i.e. leakage of the length field in an encrypted packet, dependence of the decryption process on the length field, byte-by-byte processing, etc). They extend the security model by explicitly considering all this side-channel information, and prove that the related SSH-CTR protocol achieves this extended notion.

As discussed in more detail in Appendix A, length-leakage is exactly one of the means used in traffic analysis to achieve *classification* of SSH flows. This is consistent with our framework: in Appendix C.4 we show a simple attack against the indSenderTA security of *any* "Encode-then-Encrypt & MAC" scheme (such as SSH). This is actually a separation proving that indTA is a strictly stronger notion than security in the presence of traffic analysis, secTA (see the proof of Lemma C.8). We also show a theoretical solution to achieve indSenderTA, based on an ideal distribution for the padding length. This solution introduces a large overhead, however, and is hard to implement, as no such ideal distribution is known in practice (though some heuristics, e.g. [31, 23] exist).

RFID authentication. Our second scenario is tag-to-reader authentication for RFID systems, more concretely the destructive-private protocol due to Vaudenay [43], where keys are updated by means of a Pseudo-Random Function (PRF). The RFID tag keeps only (a copy of) the current authentication key, whereas the RFID reader keeps the original key and looks for the number i of times that this key has been updated. In this scenario, a (possibly response-time based) traffic analysis oracle may reveal the value i. An attack against the indSenderTA security of this notion (where security is defined in the Juels and Weiss model [26]) is to output as a challenge a session where both tags have updated keys, and one where both have the original key. By using traffic analysis $\mathcal{A}_{big}$ will distinguish between the updated or not-updated tag.

In Appendix D we show that in order to render traffic analysis useless, either the tag must store multiple keys, called pseudonyms, or the reader must store a (potentially very large number of) intermediate states, such that it matches the updated key against a table, rather than search for i by means of computation.

2 Preliminaries

2.1 Notation

If S is a distribution over a set $\mathcal{S}$, then $x \leftarrow \mathsf{S}$ signifies that x is drawn at random from $\mathcal{S}$ according to S (unless otherwise specified, sets $\mathcal{S}$ are associated with the uniform distribution). If S is an algorithm, then $y \leftarrow \mathsf{S}(x)$ denotes an execution of S on input x with output y; in particular when S is probabilistic, y is a random variable.

If $\mathcal{A}$ is an algorithm and $\mathcal{O}$ is an oracle queried on some input, we write $\mathcal{A}^{\mathcal{O}(\cdot)^Q}$ for an adversary who queries $\mathcal{O}$ a number Q of times (for $Q \in \mathbb{N}$). A function in λ is *negligible*, written $\mathrm{negl}(\lambda)$, if it vanishes faster than the inverse of any polynomial in λ. An algorithm $\mathcal{A}$ is *probabilistic polynomial time* (PPT) if $\mathcal{A}$ uses some randomness as part of its logic (i.e. $\mathcal{A}$ is probabilistic) and for any input $x \in \{0,1\}^*$ the computation of $\mathcal{A}(x)$ terminates in at most $\mathrm{poly}(|x|)$ steps.

2.2 Games in Canonical Form

Cryptographic primitives. A *cryptographic primitive* $\Pi(\mathcal{P}) = (\mathsf{Setup}, \mathcal{F})$ for a player set $\mathcal{P} = \{P_1, \ldots, P_n\}$ consists of an initialisation algorithm Setup and a set of PPT algorithms $\mathcal{F}$. We often shorten the notation to just Π. There may be just two players (e.g., the prover and the verifier for authentication, or the signer and the verifier for signature schemes), or possibly n players executing a multiparty protocol (e.g., in an electronic voting scheme). The Setup algorithm takes as input parameters (λ, mpar) with $\lambda \in \mathbb{N}$ and mpar containing some master parameters (like the description of an underlying prime field, the parameters of a pairing-friendly curve, etc.) and outputs $(\mathit{ppar}, \mathit{spar}) \leftarrow \mathsf{Setup}(1^\lambda, \mathit{mpar})$ where $\mathit{ppar} \in \{0,1\}^*$ are public parameters and $\mathit{spar} \in \{0,1\}^*$ are private (secret) parameters for the primitive Π. The interactive use of algorithms $\mathcal{F}$ (by parties $P_i \in \mathcal{P}$) yields a transcript $\tau \in \{0,1\}^*$, containing the public inputs and outputs of the algorithms. If the primitive aborts during its execution, τ ends with $\perp$.

Classical Security – Hard games. Classical security of cryptographic primitives $\Pi(\mathcal{P})$ is typically defined through a *game* between an adversary $\mathcal{A}_{\mathit{small}}$ and an honest entity $\mathcal{G}$ called the *challenger*. We assume that $\mathcal{A}_{\mathit{small}}$ is stateful, i.e. it carries state from one step to another; $\mathcal{A}_{\mathit{small}}$ and $\mathcal{G}$ exchange messages via a shared communication tape, and $\mathcal{G}$ runs (a subset of) algorithms in $\mathcal{F}$ in a black-box way on $\mathcal{A}_{\mathit{small}}$'s input. At some point, $\mathcal{A}_{\mathit{small}}$ prompts $\mathcal{G}$ to issue a challenge, for which $\mathcal{A}_{\mathit{small}}$ must output a guess; given the challenge and this guess, $\mathcal{G}$ outputs a decision bit d. If $d = 1$ we say $\mathcal{A}_{\mathit{small}}$ has *won* the game, else the game is *lost*.

Our definitions apply to games in so-called *canonical* form as defined below; this notion is not too restrictive: most definitions in cryptography are canonical – see Appendix B. A canonical game Game is a tuple of PPT algorithms sharing state, $\mathsf{Game} = (\mathsf{Setup}, \mathsf{Learn}, \mathsf{ChGen}, \mathsf{Match})$ such that:

Setup. Upon input (λ, mpar) with $\lambda \in \mathbb{N}$, this algorithm runs the algorithm Setup of $\Pi(\mathcal{P})$ and outputs public/secret parameters $(\mathit{ppar}, \mathit{spar}) \leftarrow \mathsf{Setup}(1^\lambda, \mathit{mpar})$. Here, spar are used by $\mathcal{G}$, and ppar are used by $\mathcal{A}_{\mathit{small}}$. We require that $\mathcal{A}_{\mathit{small}}$ is equally successful in breaking the game regardless of which parameters are output by Setup.

Learn. Upon input a parameter set $\mathcal{Q}_i$, the challenger runs Learn, outputting a (partial) transcript $\tau_i \in \{0,1\}^*$.

ChGen. Upon input a set of parameters $\mathcal{Q}^*$, $\mathcal{G}$ runs ChGen outputting a challenge Υ^*. After ChGen has been successfully run, any further attempts to run it will output $\perp$.

Match. On input a guess Γ^* and a challenge Υ^*, Match outputs a bit d. We say the guess is correct if $d = 1$.

Security with respect to $\mathsf{Game} = (\mathsf{Setup}, \mathsf{Learn}, \mathsf{ChGen}, \mathsf{Match})$ is defined as follows.

Experiment $\mathbf{Exp}^{\mathsf{Game}}_{\Pi(\mathcal{P})}(\mathcal{A}_{small}, \lambda)$:

1. $(ppar, spar) \leftarrow \mathsf{Setup}(1^\lambda, mpar)$. Adversary $\mathcal{A}_{small}$ is given $ppar$.
2. $\mathcal{Q}^* \leftarrow \mathcal{A}_{small}^{\mathsf{Learn}(\cdot)^{Q'}}(ppar)$.
3. $\Upsilon^* \leftarrow \mathsf{ChGen}(\mathcal{Q}^*)$.
4. $\Gamma^* \leftarrow \mathcal{A}_{small}^{\mathsf{Learn}(\cdot)^{Q''}}(ppar, \Upsilon^*)$.
5. The experiment outputs $d \leftarrow \mathsf{Match}(\Upsilon^*, \Gamma^*)$.

Let $Q = Q' + Q''$ be the total number of queries to Learn (both before and after Υ^* is issued). If ChGen has no output, as e.g., in the case of MACs, cf. Appendix B, we set Υ^* to be the empty string.

Definition 2.1 (Hard games in canonical form) *Let* $\mathsf{Game} = (\mathsf{Setup}, \mathsf{Learn}, \mathsf{ChGen}, \mathsf{Match})$ *be the canonical security game for a primitive* Π. *We say* Π *is* (t, Q, ϵ)-secure *if for all PPT adversaries* $\mathcal{A}_{small}$ *running in time at most* t *and making at most* Q *queries in the experiment above, we have*

$$\left|\mathrm{Prob}\left[\mathbf{Exp}^{\mathsf{Game}}_{\Pi(\mathcal{P})}(\mathcal{A}_{small}, \lambda) = 1\right] - \delta\right| \leq \epsilon,$$

where the probability is taken over the randomness of the challenger $\mathcal{G}$ *and resp. that of* $\mathcal{A}_{small}$ *in a single execution. The parameter* δ *(with* $0 \leq \delta < 1$*) is the* hardness *of game* $\mathcal{G}$. *For any* $\mathcal{A}_{small}$ *against* Game, *we call* $\left|\mathrm{Prob}\left[\mathbf{Exp}^{\mathsf{Game}}_{\Pi(\mathcal{P})}(\mathcal{A}_{small}, \lambda) = 1\right] - \delta\right|$ *the* advantage *of* $\mathcal{A}_{small}$ *and we denote it* $\mathbf{Adv}^{\mathsf{Game}}_{\Pi(\mathcal{P})}(\mathcal{A}_{small})$.

As no exact functionality is specified for it, Match may in fact allow for restrictions in $\mathcal{A}_{small}$'s use of the oracles before and after the challenge. In the examples in Appendix B, Match conditions the $\mathcal{A}_{small}$'s success on the challenge input (in our PKE example, Match outputs 0 if the challenge messages m_0 and m_1 have different bit lengths, as required by IND-CCA). Similarly, we include restrictions on $\mathcal{A}_{small}$'s oracle use in Match. For PKE, Match outputs 0 if the decryption oracle is used on the challenge ciphertext.

Security in the presence of Traffic Analysis. If we additionally give $\mathcal{A}_{small}$ access to a pre-set, stateful traffic analysis (TA) oracle Θ, we have security in the presence of TA. Each run of the classical game is now an instantiation, and an adversary $\mathcal{A}_{small}$ in an instantiation Inst may run Θ on Inst receiving some output (Θ is stateful, thus $\mathcal{A}_{small}$ gets different output depending on when it queries the oracle). The game for security in the presence of traffic analysis is otherwise the same as that for classical security.[1] The goal of $\mathcal{A}_{small}$ is still to output Γ^* such that Match returns 1. We refer the reader to Appendix C for a formalization of this notion. Figure 1 depicts the essential differences between classical security, security in the presence of traffic analysis, and the notion of indistinguishability in the presence of traffic analysis, which we define in what follows.

3 Security Definitions

Classical security models usually take into account neither (a) weaknesses in implementation, nor (b) leakage by traffic analysis. In fact, the set of oracles $\mathcal{O}$ which $\mathcal{A}_{small}$ may query in a (canonical) security game (as defined previously) is primitive- and not machine- or implementation-specific. The implementation of a primitive may, however, leak important information, e.g. parts of the randomness used, or a fingerprint of the machine (sender) running it. We proceed to enhance security models to account for traffic analysis leakage. We first give an intuition of our notion, then proceed to the formal definitions.

[1] Note that the term *security* is somewhat ambiguous, as classical security for a primitive might be a notion of privacy as in the case of RFID (cf. Appendix B).

Informal Intuition. We consider *instantiations* of the security game, i.e. each run of the game is assumed to be dependent on the implementation and the machine running the primitive. The parameters of an instantiation are *global*, generated by Setup or *local*, (sender-specific as in Section 1). In our running example, the global parameters (pk, sk) are output by $\mathsf{Setup} = \mathsf{Gen}$, and pk is given to $\mathcal{A}_{small}$. We say that a user is *created* when Setup is run to output new parameters.[2]

Each user may run parallel *instantiations* of the game, having their own *local* parameters, e.g. challenge bits, or preimages. For IND-CCA security, once a user is generated, the public key pk is given to $\mathcal{A}_{small}$, who may query ChGen on inputs m_0, m_1. Then the challenger $\mathcal{G}$ picks a bit b, which is part of its local input. In fact, once $\mathcal{A}_{small}$ starts playing Game, we speak of an instantiation (uniquely determined by the oracle queries/replies of $\mathcal{A}_{small}$ and $\mathcal{G}$ and by the local and global parameters), to which there corresponds a transcript τ. By contrast, users correspond to the global parameters only. Instantiations are subject to traffic analysis (which is modelled by an oracle Θ), yielding information that is either local (the oracle is then *restricted*) or both local and global (this oracle is *unrestricted*). The model is non-adaptive in the sense that indTA-security is proved with respect to a pre-defined Θ oracle (e.g. we may choose to prove that encryption is secure in the presence of an oracle yielding half of the randomness used for encrypting ciphertexts).

Two instantiations sharing global parameters are *same-user* instantiations; instantiations with different global input are *mixed-user*. Our framework defines the notion of indistinguishability in the presence of traffic analysis, indTA, where the adversary must distinguish between two instantiations (either same- or mixed-user) of the security game. As aforementioned, an adversary that breaches the security (or the security in the presence of traffic analysis) of a protocol will also trivially win against indistinguishability, but this is not *the only* way to relate sessions. Indeed, fingerprinting an implementation will not necessarily break security, but it *will* make an adversary distinguish sessions.

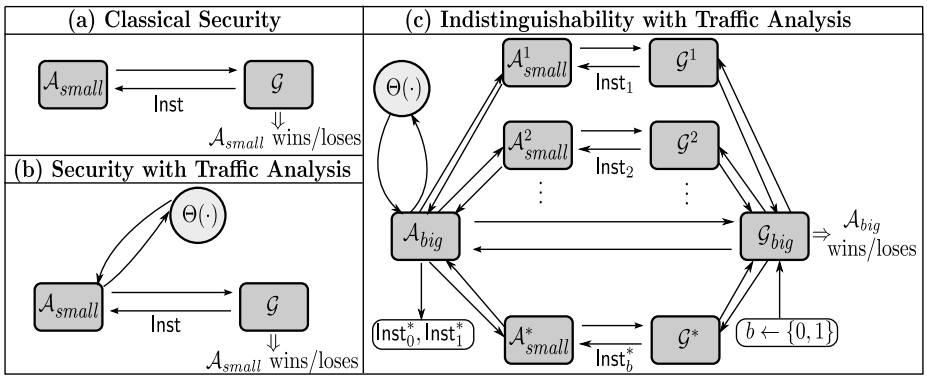

Figure 1: The three flavours of security considered in this paper: (a) Classical security, (b) Security in the presence of traffic analysis and (c) Indistinguishability in the presence of traffic analysis.

Formal Intuition. Concretely, let Π be a primitive with canonical security game $\mathsf{Game} = (\mathsf{Setup}, \mathsf{Learn}, \mathsf{ChGen}, \mathsf{Match})$. Adversaries $\mathcal{A}_{big}$ play against a challenger $\mathcal{G}_{big}$. Initially an algorithm Init is run on some λ to output the master parameters; then $\mathcal{A}_{big}$ can create new users (using Setup on input λ and *mpar*) or new instantiations Inst for a user (see also Figure 1).

A user User for Π and Game is a set of parameters UPar such that $(\mathsf{UPar}) \leftarrow \mathsf{Setup}(1^\lambda, mpar)$ for some λ and *mpar*. In our running example, $\mathcal{G}_{big}$ runs Setup to generate keys for users Alice and Bob. The adversary $\mathcal{A}_{big}$ receives pk for both users and may create instantiation Inst for, say Alice, creating a local $\mathcal{A}_{small}$ manipulated by $\mathcal{A}_{big}$ and a local $\mathcal{G}$ handled by $\mathcal{G}_{big}$. This corresponds to $\mathcal{A}_{big}$ tapping Carol's transmission to Alice generated for SSH or eMail.

[2]The term "user" is rather loose, as some models, e.g. RFID privacy (cf. Appendix B), create many tags in a single Setup.

Adversary $\mathcal{A}_{big}$ may either "play" a created Inst (by manipulating $\mathcal{A}_{small}$ and using Learn and ChGen with appropriate parameters) or may use Θ on Inst. Running ChGen in an instantiation invalidates it. Formally, an instantiation is a tuple $(\mathcal{A}_{small}, \mathcal{G}, \tau, \mathsf{UPar}, \mathsf{IPar})$, where: τ is a (potentially partial) transcript including a validity bit initialized to 1 and flipped to 0 upon successful running of ChGen; UPar are global (user-specific) parameters; and IPar are local parameters. We assume consecutive indexing of instantiations (in the order of their creation). Eventually $\mathcal{A}_{big}$ picks valid Inst_i^* and Inst_j^* and forwards them to $\mathcal{G}_{big}$, with challenge inputs $\mathcal{Q}_i^*$, resp. $\mathcal{Q}_j^*$.[3] Challenger $\mathcal{G}_{big}$ runs ChGen for both Inst_i^* and Inst_j^* and gets Υ_i^* and Υ_j^*, then picks a bit b, and forwards $\Upsilon_{big}^* = (\mathsf{Inst}^*, \Upsilon^*)$ to $\mathcal{A}_{big}$, where $(\mathsf{Inst}^*, \Upsilon^*) = (\mathsf{Inst}_i^*, \Upsilon_i^*)$ if $b = 0$ and $(\mathsf{Inst}^*, \Upsilon^*) = (\mathsf{Inst}_j^*, \Upsilon_j^*)$ for $b = 1$.[4] Finally, $\mathcal{A}_{big}$ outputs guess Γ_{big}^* for the bit b and wins if $\Gamma_{big}^* = b$. This notion is called indistinguishability in the presence of TA, denoted indTA.

We consider two flavours of indTA: sender indistinguishability – indSenderTA – where Inst_i^* and Inst_j^* are same-user instantiations, and receiver indistinguishability – indReceiverTA – where the chosen instantiations are mixed-user. There are two types of traffic analysis: restricted, denoted Θ_{rest}, which outputs only information about the local parameters IPar; and unrestricted, denoted Θ, yielding both IPar and UPar information.

INDTA AS A CANONICAL GAME. Our definition of indTA is a canonical indistinguishability-based security game $\mathsf{Game}_{big} = (\mathsf{Setup}_{big} = \mathsf{Init}, \mathsf{Learn}_{big}, \mathsf{ChGen}_{big} = \mathsf{Challenge}, \mathsf{Match}_{big})$. The algorithm Setup_{big} takes as input a parameter λ and outputs master parameters *mpar*. Adversary $\mathcal{A}_{big}$ plays game Game_{big} against challenger $\mathcal{G}_{big}$. The oracle Learn_{big} consists of the traffic analysis oracle Θ and a set $\mathcal{O}_{\mathcal{A}_{big}}$ of oracles (see the next section). The set $\mathcal{O}_{\mathcal{A}_{big}}$ includes the set $\mathcal{O}$ defined for Game, but also includes oracles to create new users and instantiations, and to play instantiations. The input $\mathcal{Q}_{big}^*$ to ChGen_{big} consists of: valid instantiations Inst_0^* and Inst_1^*, and respective challenge inputs $\mathcal{Q}_0^*$ and $\mathcal{Q}_1^*$. The algorithm ChGen_{big} picks a random bit b, runs ChGen for both instantiations, then outputs $\Upsilon_{big}^* = (\mathsf{Inst}_b^*, \Upsilon_b^*)$. The adversary $\mathcal{A}_{big}$ may play Inst_b^* or learn more information, and finally outputs a guess Γ_{big}^* for the bit b. The algorithm Match_{big} takes as input Υ_{big}^* and Γ_{big}^* and outputs 1 iff: (1) the bit $\Gamma_{big}^* = b$; (2) Inst_0^* and Inst_1^* are both valid; and either (3a) the two instantiations are same-user (for indSenderTA) or (3b) the two instantiations are mixed-user (for indReceiverTA).

3.1 Definitions of Indistinguishability for Traffic Analysis

There are two dimensions to our model: the adversary's goal and the strength of the TA oracle Θ. We have motivated the use in considering sender and receiver indistinguishability; we now briefly motivate our flavours of Θ before we formalise our definitions.

RESTRICTED AND UNRESTRICTED TA. In our framework, we prove protocols indistinguishable in the presence of certain, pre-set TA oracles Θ. On input an instantiation, Θ outputs information on this instantiation. In practice, Θ models an adversary's ability to learn information related to a primitive, e.g. a delay in a response could leak information regarding key-update status (see also Appendix D).

We consider two types of traffic analysis, depending on the their output. If Θ yields only local information, we say it is *restricted* and denote it Θ_{rest}; in practice, Θ_{rest} models sender-specific flaws for implementations which protect receiver data well (a sender might take longer to generate a challenge bit, or it may always send a tell-tale header). An oracle leaking both local and global information is called *unrestricted*. In our framework, indistinguishability for unrestricted oracles is stronger than for restricted oracles. The oracle is pre-set, and it may give more or less information: thus in the running example, a weak restricted oracle may yield a bit of the randomness used to generate the challenge ciphertext; a strong, unrestricted one may leak *sk*.

[3] We say that the adversary forwards the instances to the challenger if it forwards indices i and j.

[4] When we say the challenger forwards instantiation Inst^*, we mean $\mathcal{A}_{big}$ may use the handle Inst^* to play it or to use Θ on it, without knowing which instantiation it is.

We relate the two types of oracles by *projecting* unrestricted onto restricted oracles: Θ_{rest} is a projection $\mathsf{proj}(\cdot)$ of Θ if: (a) Θ_{rest} is restricted, and (b) the output of $\Theta_{rest} = \mathsf{proj}(\Theta)$ for any instantiation is included in the output of Θ for the same instantiation.

Algorithms and Oracles. As aforementioned, $\mathcal{G}_{big}$ begins the game by querying an oracle Init, which, on input a security parameter λ, outputs parameters *mpar*. We require that it is on average equally hard to break instances of the security game for different values of *mpar*. We now list the oracles that $\mathcal{A}_{big}$ may query in indTA against a primitive Π with canonical security game $\mathsf{Game} = (\mathsf{Setup}, \mathsf{Learn}, \mathsf{ChGen}, \mathsf{Match})$. Here $\mathcal{O}_{\mathcal{A}_{big}} = (\mathsf{NewUser}, \mathsf{NewInst}, \mathsf{Play})$.

$\mathsf{NewUser}$. On input λ and master parameters *mpar* (output by Init), this oracle runs Setup, outputting $\mathsf{UPar} = (ppar, spar)$ and a user identifier User. The values User, *ppar*, and *spar* are added on a new row of (an initially empty) table $\mathcal{D}$. By abusing notation we write $\mathsf{User} \in \mathcal{D}$ iff there exists a row in $\mathcal{D}$ containing User. An (empty) table $\mathcal{I}_{\mathsf{User}}$ is created for each user.

$\mathsf{NewInst}$. On input user User, if $\mathsf{User} \in \mathcal{D}$, then $\mathsf{NewInst}$ creates an instantiation Inst and adds it to $\mathcal{I}_{\mathsf{User}}$ and to a list $\mathcal{L}$. Else, $\mathsf{NewInst}$ outputs $\perp$.

Play. On input an instantiation Inst, an oracle $\mathsf{Mode} \in \{\mathsf{Learn}, \mathsf{ChGen}, \mathsf{Match}\}$, and some input state view, this algorithm forwards Mode to adversary $\mathcal{A}_{small}$ in Inst, who queries Mode and forwards the output value(s) out. At every query, the values $(\mathsf{Mode}, \mathsf{view}, \mathsf{out})$ are added to the transcript τ corresponding to Inst and to the table $\mathcal{I}_{\mathsf{User}}$. Furthermore, ChGen is only run if Inst is in the list of valid instantiations $\mathcal{L}$. Once the challenge is issued, Inst is removed from $\mathcal{L}$.

$\mathsf{Challenge}$. On input instantiations Inst_0^* and Inst_1^* and input $\mathcal{Q}_0^*$ resp. $\mathcal{Q}_1^*$, $\mathsf{Challenge}$ checks instantiation validity, then runs ChGen for both instantiations, removing them from $\mathcal{L}$. Then the algorithm picks a bit b at random and outputs instantiation Inst_b^* with relevant Υ_b^*.

Θ. On input an instantiation Inst, the oracle Θ outputs state information ϑ.

Considering the conditions on Init and Setup we may assume an uniformity of the instantiations, namely an adversary is about as likely to break the security of two (same-user or mixed-user) instantiations.

Sender Indistinguishability with Traffic Analysis. For sender indistinguishability we consider both *restricted* and *unrestricted* traffic analysis as defined above. We first define the restricted oracle Θ_{rest}:

Definition 3.1 (Restricted Θ oracle) *A Θ oracle is* restricted *iff for all mpar output by Init and for all PPT adversaries $\mathcal{A}_{small}$ having access only to the $\mathsf{NewUser}$, $\mathsf{NewInst}$, and Θ oracles, for all Inst_0 and Inst_1 output by $\mathcal{A}_{small}$ such that for all j with $\mathsf{User}_j \in \mathcal{D}$, $\mathsf{Inst}_0 \in \mathcal{I}_{\mathsf{User}_j}$ implies $\mathsf{Inst}_1 \notin \mathcal{I}_{\mathsf{User}_j}$, it holds that:*

$$\Theta(\mathsf{Inst}_0) = \Theta(\mathsf{Inst}_1).$$

In general, we write Θ_{rest} to denote that the Θ oracle is restricted.

We now define indSenderTA security. Let Π be a primitive with canonical security game $\mathsf{Game} = (\mathsf{Setup}, \mathsf{Learn}, \mathsf{ChGen}, \mathsf{Match})$. Consider the following experiment:

Experiment $\mathbf{Exp}^{\mathrm{indSenderTA}}_{\Pi(\mathcal{P}),\Theta_{rest}}(\mathcal{A}_{big}, \lambda)$

1. $mpar \leftarrow \mathsf{Init}(1^\lambda)$.
2. $\mathsf{User} \leftarrow \mathsf{NewUser}(1^\lambda, mpar)$. Let $\mathcal{O}_{\mathsf{User}} = (\mathsf{NewInst}(\mathsf{User}), \mathsf{Play}(\cdot))$.
3. $\mathcal{Q}^*_{big} = (\mathsf{Inst}_0^*, \mathsf{Inst}_1^*, \mathcal{Q}_0^*, \mathcal{Q}_1^*) \leftarrow \mathcal{A}_{big}^{\mathcal{O}_{\mathsf{User}}, \Theta_{rest}(\cdot)}$.

4. $\Upsilon^*_{big} = (\mathsf{Inst}^*_b, \Upsilon^*_b) \leftarrow \mathsf{Challenge}(\mathcal{Q}^*_{big})$ for a random bit b.
5. $\Gamma^*_{big} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{User}}, \Theta_{rest}(\cdot)}_{big}(\Upsilon^*_{big})$.
6. Output 1 iff $\Gamma^*_{big} = b$ and $(\mathsf{Inst}^*_0, \mathsf{Inst}^*_1) \in \mathcal{L}$.

Let Q, resp. Q_{TA} denote the number of queries to $\mathcal{O}_{\mathsf{User}}$, resp. Θ_{rest}. The adversary's advantage is:

$$\mathbf{Adv}^{\text{indSenderTA}}_{\Pi(\mathcal{P})}(\mathcal{A}_{big}, \Theta_{rest}) = \left|\text{Prob}\left[\mathbf{Exp}^{\text{indSenderTA}}_{\Pi(\mathcal{P}),\Theta_{rest}}(\mathcal{A}_{big}, \lambda) = 1\right] - \tfrac{1}{2}\right|.$$

Definition 3.2 (Restricted indSenderTA **security)** *The primitive* Π *is* $(t, Q, Q_{\mathsf{TA}}, \epsilon)$*-secure against restricted* indSenderTA *attacks for restricted oracle* Θ_{rest} *in a (canonical) game* Game *if for every PPT adversary* $\mathcal{A}_{big}$ *running in time at most* t *and making at most* Q*, resp.* Q_{TA} *queries it holds that:*

$$\mathbf{Adv}^{\text{indSenderTA}}_{\Pi(\mathcal{P})}(\mathcal{A}_{big}, \Theta_{rest}) \leq \epsilon.$$

We also define sender indistinguishability for unrestricted oracles Θ, leaking both local and global information. The adversary must still distinguish between same-user instantiations. We replace Θ_{rest} by Θ above to obtain $\mathbf{Exp}^{\text{indSenderTA}}_{\Pi(\mathcal{P}),\Theta}(\mathcal{A}_{big}, \lambda)$.

Definition 3.3 (Unrestricted indSenderTA **security)** *The primitive* Π *is* $(t, Q, Q_{\mathsf{TA}}, \epsilon)$*-secure against unrestricted* indSenderTA *attacks for unrestricted oracle* Θ *in a (canonical) game* Game *if for every PPT adversary* $\mathcal{A}_{big}$ *running in time at most* t*, and making at most* Q*, resp.* Q_{TA} *queries it holds that:*

$$\mathbf{Adv}^{\text{indSenderTA}}_{\Pi(\mathcal{P})}(\mathcal{A}_{big}, \Theta) \leq \epsilon.$$

Receiver Indistinguishability with Traffic Analysis. For indReceiverTA, Θ is unrestricted (in this sense, indReceiverTA is stronger than indSenderTA, see Section 3.2). We define indReceiverTA analogously to indSenderTA, for the following experiment:

Experiment $\mathbf{Exp}^{\text{indReceiverTA}}_{\Pi(\mathcal{P}),\Theta}(\mathcal{A}_{big}, \lambda)$
1. $mpar \leftarrow \mathsf{Init}(1^\lambda)$. Let $\mathcal{O}_{\mathsf{User}} = (\mathsf{NewUser}(1^\lambda, mpar), \mathsf{NewInst}(\cdot), \mathsf{Play}(\cdot))$.
2. $\mathcal{Q}^*_{big} = (\mathsf{Inst}^*_0, \mathsf{Inst}^*_1, \mathcal{Q}^*_0, \mathcal{Q}^*_1) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{User}}, \Theta_{rest}(\cdot)}_{big}$.
3. $\Upsilon^*_{big} = (\mathsf{Inst}^*_b, \Upsilon^*_b) \leftarrow \mathsf{Challenge}(\mathcal{Q}^*_{big})$ for a random bit b.
4. $\Gamma^*_{big} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{User}}, \Theta(\cdot)}_{big}(\Upsilon^*_{big})$.
5. Output 1 iff: (i) $\Gamma^*_{big} = b$; (ii) $\mathsf{Inst}^*_0, \mathsf{Inst}^*_1 \notin \mathcal{I}_{\mathsf{User}}$ for any user User output by $\mathsf{NewUser}$; and (iii) $\mathsf{Inst}^*_0, \mathsf{Inst}^*_1 \in \mathcal{L}$.

Again, Q, resp. Q_{TA} are the number of queries made to $\mathcal{O}_{\mathsf{User}}$, resp. Θ, and the advantage is:

$$\mathbf{Adv}^{\text{indReceiverTA}}_{\Pi(\mathcal{P})}(\mathcal{A}_{big}, \Theta) = \left|\text{Prob}\left[\mathbf{Exp}^{\text{indReceiverTA}}_{\Pi(\mathcal{P}),\Theta}(\mathcal{A}_{big}, \lambda) = 1\right] - \tfrac{1}{2}\right|.$$

Definition 3.4 (indReceiverTA **security)** *The primitive* Π *is* $(t, Q, Q_{\mathsf{TA}}, \epsilon)$*-secure against* indReceiverTA *attacks for unrestricted oracle* Θ *in a (canonical) game* Game *if for every PPT adversary* $\mathcal{A}_{big}$ *running in time at most* t*, and making at most* Q*, resp.* Q_{TA} *queries in the experiment above we have:*

$$\mathbf{Adv}^{\text{indReceiverTA}}_{\Pi(\mathcal{P})}(\mathcal{A}_{big}, \Theta) \leq \epsilon.$$

3.2 Relation between Security Notions

In the previous section we gave three security definitions (cf. Definitions 3.2 – 3.4), i.e. two flavours of indSenderTA, (restricted and unrestricted), and indReceiverTA. Here we state, and give an intuition for, the relations of the notions. Exact statements and the proofs are given in Appendix C.

Firstly, unrestricted indSenderTA is strictly stronger than restricted indSenderTA: the unrestricted Θ gives more data than $\Theta_{rest} = \mathsf{proj}(\Theta)$. A trivial separation is to consider Θ leaking Θ_{rest} data *and* a secret key. Relating unrestricted indSenderTA and indReceiverTA is more difficult (Θ is the same, but in one case the challenge instantiations are same-user and in the other, mixed-user). In fact, the two notions are independent, and the separation relies on the fact that one can define a TA oracle that is *only* useful for sender distinguishability, and not for receiver distinguishability, or viceversa.

We also show that the enhancement from *security in the presence of traffic analysis* – see Section 2 and Appendix C – to *indistinguishability in the presence of traffic analysis* is not trivial. In particular we prove that indTA is strictly stronger than secTA: if Π is indTA-secure with respect to Game, then it is also secTA-secure. The reverse is *not* true: our counterexample is the Encode-then-Encrypt & MAC paradigm with a length-leaking oracle. The adversary can manipulate the challenge input so that the TA oracle allows it to distinguish instantiations without necessarily breaking security.

These relationships are summarized in Figure 2. Formal statements are given in Theorems 3.1 – 3.3 and proved in Appendix C.2 – C.4.

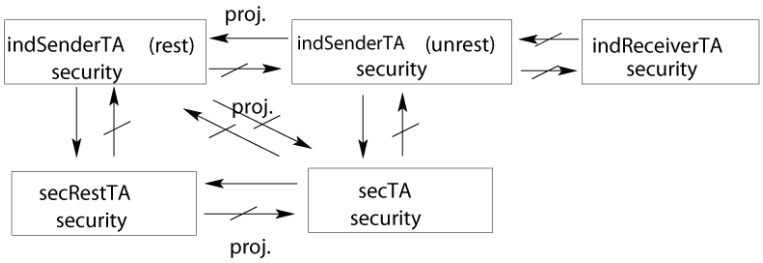

Figure 2: Relationships of the notions: arrows refer to implications, hatched arrows to separations; the notation "proj." refers to oracle projection, cf. Section 3.1.

Sender vs. Receiver Indistinguishability.

Theorem 3.1 (indSenderTA **and** indReceiverTA**)** *The following relationships hold:*

1. *If primitive* Π *is secure for a canonical game* Game *and unrestricted* indSenderTA*-secure for some oracle* Θ*, then it is also restricted* indSenderTA*-secure for any* Θ_{rest} *s.t.* $\Theta_{rest} = \mathsf{proj}(\Theta)$*.*
2. *There exists a primitive* Π *secure against a canonical game* Game*, such that one of the following holds:*
   (a) Π *is restricted* indSenderTA*-secure for* Θ_{rest}*, but unrestricted* indSenderTA-in*secure for some* Θ *satisfying* $\mathsf{proj}(\Theta) = \Theta_{rest}$*.*
   (b) Π *is* indReceiverTA*-secure for oracle* Θ*, but unrestricted* indSenderTA-in*secure for* Θ*.*
   (c) Π *is unrestricted* indSenderTA*-secure for oracle* Θ*, but* indReceiverTA-in*secure for* Θ*.*

Security vs. indistinguishability. We consider *security in the presence of traffic analysis* (see section 2 and Appendix C) for both restricted and unrestricted oracles, i.e. secRestTA-security and resp. secTA-security. We prove the following results:

Theorem 3.2 (Restricted and unrestricted security) *The following relationships hold:*

1. *If* Π *is secure for a canonical game* Game *and* secTA*-secure for unrestricted* Θ*, then it is also* secRestTA*-secure for any restricted* Θ_{rest} *s.t.* $\mathsf{proj}(\Theta) = \Theta_{rest}$*.*
2. *There exists a primitive* Π *that is: secure for a canonical game* Game*;* secRestTA*-secure for restricted* Θ_{rest}*; but* secTA-in*secure for unrestricted* Θ *s.t.* $\mathsf{proj}(\Theta) = \Theta_{rest}$*.*

Theorem 3.3 (Security and indistinguishability) *The following relationships hold:*

1. *Let* Π *be secure for a canonical game* Game*:*
   (a) *if* Π *is restricted* indSenderTA*-secure for restricted* Θ_{rest}*, then it is also* secRestTA*-secure for* Θ_{rest}*.*
   (b) *if* Π *is unrestricted* indSenderTA*-secure for oracle* Θ*, then it is also* secTA*-secure for* Θ*.*
2. *There exists a primitive* Π *secure for a canonical game* Game*, such that one of the following holds:*
   (a) Π *is* secRestTA*-secure for restricted* Θ_{rest}*, but restricted* indSenderTA-in*secure for* Θ_{rest}*.*
   (b) Π *is* secTA*-secure for oracle* Θ*, but unrestricted* indSenderTA-in*secure for* Θ*.*
   (c) Π *is restricted* indSenderTA*-secure for* Θ_{rest}*, but* secTA-in*secure for some* Θ *s.t.* $\mathsf{proj}(\Theta) = \Theta_{rest}$*.*
   (d) Π *is* secTA*-secure for oracle* Θ*, but restricted* indSenderTA-in*secure for some oracle* Θ_{rest} *s.t.* $\Theta_{rest} = \mathsf{proj}(\Theta)$*.*

4 Case Study: The "Encode-then-Encrypt & MAC" Paradigm

We apply our framework in the context of the Secure Shell (SSH) protocol [5]. SSH is a particular instantiation of the "Encode-then-Encrypt & MAC" paradigm, which we briefly recall below.

Let $\Pi^\star(\Psi, \Pi_E, \Pi_M) = (\mathsf{Gen}^\star, \mathsf{Enc}^\star, \mathsf{Dec}^\star)$ be a symmetric encryption scheme for encoding scheme Ψ, on block cipher $\Pi_E = (\mathsf{Gen}_E, \mathsf{Enc}, \mathsf{Dec})$ with input length B (in bytes) and on a message authentication code $\Pi_M = (\mathsf{Gen}_M, \mathsf{MAC}, \mathsf{VRFY})$. In what follows all lengths $|\cdot|$ are in bytes. The key-generation algorithm $\mathsf{Gen}^\star$ takes as input the security parameter λ and outputs $K = (K_E, K_M) \leftarrow \mathsf{Gen}^\star(1^\lambda)$, where $K_E \leftarrow \mathsf{Gen}_E(1^\lambda)$ and $K_M \leftarrow \mathsf{Gen}_M(1^\lambda)$. A message m of length $|m| = L$ bytes is encoded into $\Psi(m)$ for random padding u of length $|u|$, as follows:

$$\Psi(m) = (|\langle |u|, m, u\rangle|, |u|, m, u),$$

It is crucial that $|u|$ is standardized, chosen such that the length of the padded message is an integer multiple of the block length B of the underlying block cipher. Thus, $|u| \leq B$ bytes. In the following, denote by m_{pad} the padded message preceded by the padding length, i.e. $m_{\mathsf{pad}} = \langle |u|, m, u\rangle$. Note that decoding relies on the fact that the length $|m_{\mathsf{pad}}|$ is fixed by the standardization.

The encoded message $\Psi(m)$ is encrypted to $c \leftarrow \mathsf{Enc}_{K_E}(\Psi(m))$; since Π_E is a block-cipher, this requires the use of an operational mode, e.g. CBC. Finally, a tag $\phi \leftarrow \mathsf{MAC}_{K_M}(\Psi(m))$ of the encoded message is computed using key K_M. The ciphertext is then $c^\star = (c, \phi) \leftarrow \mathsf{Enc}^\star_K(m)$.

Decryption of $c^\star$ works as follows. Upon decrypting the first block of c, the length field $|m_{\mathsf{pad}}|$ is recovered; if something goes wrong, a special symbol $\perp_L$ is returned. Once the length field is known, $c^\star$ is parsed as (c, ϕ). If parsing fails, a special symbol $\perp_P$ is output. Then c is decrypted, yielding $\Psi(m) = \mathsf{Dec}_{K_E}(c)$; the tag ϕ is verified as in $\mathsf{VRFY}_{K_M}(\Psi(m), \phi)$. If verification fails a special symbol $\perp_M$ is output.[5]

[5]It is well known that an instantiation of the "Encode-then-Encrypt & MAC" using *any* CPA-secure encryption scheme Π_E and *any* secure MAC Π_M does *not* yield a CCA-secure scheme. There are, however, other combinations of symmetric encryption with MACs. For instance the "Encrypt-then-MAC" paradigm (where the MAC is applied to the encrypted message and not to the encoded plaintext) is known to be CCA-secure for *any* choice of CPA-secure Π_E and universally unforgeable Π_M.

The SSH protocol is a concrete example of the above paradigm for which $\Psi(m)$ has standard bounds $|u| \leq F_{\mathsf{pad}} = 1, |m_{\mathsf{pad}}| \leq F_{\mathsf{enc}} = 4$; messages m are encoded to reach a length compatible with the length B of the blockcipher, by using an amount $|u|$ of random padding as in:

$$|u| = B - (|m| + F_{\mathsf{enc}} + F_{\mathsf{pad}} \bmod B) = B - (|m| + 5 \bmod B).$$

Security in the Presence of Traffic Analysis. Bellare et al. [4] proved that two variations of the SSH protocol, namely SSH-$NPC (using CBC mode with explicit random IVs and random padding) and SSH-CTR (using counter mode), achieve IND-CCA security. However, very recently [16] shows a plaintext recovery attack against SSH-$NPC, despite the proof in [4]. This apparent contradiction is due to the fact that [16] exploited several characteristics of SSH implementations that are not modelled by classical IND-CCA, i.e. (1) the ability of adversaries to recover the (otherwise encrypted) packet length field and (2) the fact that decryption is stateful and byte-oriented. On a high level the attack goes as follows: the adversary sends a target ciphertext as the length field in the first block of a new SSH packet, then feeds the SSH connection with random blocks of ciphertext and measures the number of bytes needed before the MAC check fails. Due to CBC standardization, the adversary now knows the decryption of the length field block, i.e. the target.

To model such attacks, Paterson and Watson [35] naturally extend the standard IND-CCA model to allow for stateful decryption in a blockwise manner. Also the adversary may query a length-leaking oracle to get the (otherwise encrypted) packet length. Then [35] proves that SSH-CTR is secure in this model. The model in [35] can be modelled in our framework for secRestTA (cf. Definition C.1) as follows. We enhance IND-CCA security for $\Pi^\star$, with a TA oracle Θ_{length}. Whenever the adversary queries the decryption oracle, Θ_{length} updates a database with all the ciphertexts submitted by the adversary. When the challenge ciphertext is output in IND-CCA, it is added to the database. Then on input an instantiation Inst, Θ_{length} outputs the lengths $|m_{\mathsf{pad}}|$ of all encodings in ciphertexts in the instantiation's database. Thus Θ_{length} is restricted.

Privacy in the Presence of Traffic Analysis. The attack in [16] is only the tip of the iceberg. Several traffic analysis methods are known to breach SSH privacy. It is possible to e.g. very accurately classify traffic flows by exploiting length field leaks for SSH packets. This fact fits well in our framework: we prove in Appendix C.4 that "Encode-then-Encrypt & MAC" protocols like SSH are restricted indSenderTA-*in*secure (cf. Definition 3.2) for Θ_{length}, even if the scheme is secRestTA-secure (cf. Definition C.1) for Θ_{length}. This separates indSenderTA and secRestTA – cf. claim (2a) in Theorem 3.3 and Lemma C.8. Intuitively, in this attack an adversary $\mathcal{A}_{\mathit{big}}$ against indSenderTA simply chooses inputs $\mathcal{Q}_0^* := \{m_0^0, m_1^0\}, \mathcal{Q}_1^* = \{m_0^1, m_1^1\}$ in the two instantiations $\mathsf{Inst}_0^*, \mathsf{Inst}_1^*$ submitted to $\mathcal{G}_{\mathit{big}}$ in such a way that the messages in the input-set $\mathcal{Q}_0^*$ have different length from those in $\mathcal{Q}_1^*$. By using Θ_{length} on the challenge instantiation Inst_b^*, $\mathcal{A}_{\mathit{big}}$ distinguishes the two instantiations w.p. 1.

This attack can be prevented by ensuring that the padding *length* is not fixed and "hard to predict" (as aforesaid, the problem with SSH is that the padding length is predictable based on the message length and is always less than 1 block). A naïve solution is to require that all messages are padded until the maximum allowed length is reached: then the information leaked by Θ_{length} is useless. Whereas this guarantees indistinguishability, it also creates a huge overhead in practice. We show below how to achieve indistinguishability for less, though still a considerable, overhead; our method is a proof of concept rather than a practical solution.

Recall that the length of the padded message is $|\Psi(m)| = F_{\mathsf{enc}} + F_{\mathsf{pad}} + |m| + |u|$. The main idea is to let the padding length $|u|$ be drawn from a distribution $\mathcal{U}$ for which some trapdoor information allows a receiver to correctly parse the message and decrypt a ciphertext. More precisely, we modify the encoding scheme of $\Pi^\star$ as follows. Given message m with $|m| = L$, the length $|u|$ of the (random) padding u is drawn from $\mathcal{U}$. An encoding of m still has the form $\Psi(m) = (|m_{\mathsf{pad}}|, |u|, m, u)$, but now $|u|$ is not fixed anymore. Then the ciphertext $c^\star$ corresponding to m is computed in the same way as in the "Encode-then-Encrypt & MAC" paradigm. On the receiver side $c^\star$ is parsed as $c^\star = (c, \phi)$ and the encoding $\Psi(m)$ is recovered. Since $|u|$ is chosen at random by the sender, the receiver must recover it to get m. This is done by using a function f for some (private) trapdoor

key tk generated at setup. The function f takes as input the trapdoor and an encoding $\Psi(m)$ of message m and outputs $(|u|, |m_{\mathsf{pad}}|) = f_{tk}(\Psi(m))$, allowing to retrieve m at the receiver side.

Intuitively the distribution $\mathcal{U}$ should be such that the length of *padded* messages reveals negligible information about the length of *original* messages. This is not trivial, as shown by the following example. Assume that the adversary has access to two distributions for the plaintexts in the message space: one distribution always outputs very short messages and the other always outputs very long messages. Suppose that we choose $\mathcal{U}$ to be the uniform distribution over a space of fixed dimension. In this case an adversary can easily distinguish the encoded version of two arbitrary messages drawn from the two distributions, since the length of the padded messages will not change much: on average the encodings of the messages drawn from the first distribution will still be short, whereas the encodings of the messages drawn from the second distribution will still be long.

We formally consider the following experiment for the encoding scheme Ψ defined above (for distribution $\mathcal{U}$ and function f with trapdoor tk), an adversary $\mathcal{A}$, and a challenger $\mathcal{G}$. The challenger first chooses f, tk, and $\mathcal{U}$. The adversary $\mathcal{A}$ is given the security parameter and a description of f and may submit m to $\mathcal{G}$. The challenger chooses $|u| \leftarrow \mathcal{U}$, computes $m_{\mathsf{pad}} = \langle |u|, m, u \rangle$ where $u \leftarrow \{0,1\}^{|u|}$ and forwards $|m_{\mathsf{pad}}|$ to $\mathcal{A}$. At some point $\mathcal{A}$ chooses arbitrary m_0, m_1,[6] receives $\Upsilon^* = |m_{\mathsf{pad}}|_b$, and must guess the bit b. We say that the distribution $\mathcal{U}$ is *ideal* – and we denote it by $\mathcal{U}_{ideal}$ – if no adversary can guess b with probability larger than $1/2$.

We show that any distribution $\mathcal{U}$ which is indistinguishable from $\mathcal{U}_{ideal}$ yields restricted indSenderTA for Θ_{length}.

Proposition 4.1 (Towards security & privacy for SSH) *Let $\Pi^\star(\Psi, \Pi_E, \Pi_M)$ be $(t_{small}, Q, Q_{\mathsf{TA}}, \epsilon_{small})$-secure against* secRestTA *attacks for Θ_{length}. Moreover, let $\mathcal{U}$ be a distribution such that $\mathcal{U}$ and $\mathcal{U}_{ideal}$ are (t, ϵ)-computationally indistinguishable. If the encoding scheme Ψ in $\Pi^\star$ uses $\mathcal{U}$, then $\Pi^\star$ is $(t^\star, Q, Q_{\mathsf{TA}}, \epsilon^\star)$-secure against restricted* indSenderTA *attacks, where*

$$t^\star \approx t \qquad\qquad \epsilon^\star \leq \epsilon_{small} + \epsilon.$$

Proof. The proof uses game hopping. We denote by X_i the output of the experiment in the ith game.

Game 0. The first game is indSenderTA using Θ_{length}, where the encoding scheme Ψ uses the ideal distribution $\mathcal{U}_{ideal}$. Define E_{small} to be the event that there exists an adversary $\mathcal{A}_{small}$ able to break the secRestTA security of $\Pi^\star$ using Θ_{length}. Clearly,

$$\begin{aligned}
\mathrm{Prob}[X_0 = 1] &= \mathrm{Prob}[X_0 = 1 \mid E_{small}] + \mathrm{Prob}\left[X_0 = 1 \mid \overline{E_{small}}\right] \\
&\leq \mathrm{Prob}[E_{small}] + \mathrm{Prob}\left[X_0 = 1 \mid \overline{E_{small}}\right] \\
&\overset{(a)}{\leq} \epsilon_{small} + \mathrm{Prob}\left[X_0 = 1 \mid \overline{E_{small}}\right] \\
&\overset{(b)}{\leq} \epsilon_{small} + 1/2.
\end{aligned}$$

Here, (a) follows as $\Pi^\star$ is $(t_{small}, Q, Q_{\mathsf{TA}}, \epsilon_{small})$-secure against secRestTA and (b) holds because Ψ is based on the ideal $\mathcal{U}_{ideal}$.

Game 1. The second game is identical to Game 0, but we replace $\mathcal{U}_{ideal}$ by $\mathcal{U}$. We claim that:

$$|\mathrm{Prob}[X_1 = 1] - \mathrm{Prob}[X_0 = 1]| \leq \epsilon.$$

Assume this is not the case. We build an adversary $\mathcal{B}_{small}$ which can distinguish $\mathcal{U}$ from $\mathcal{U}_{ideal}$ with advantage $> \epsilon$, contradicting the indistinguishability of $\mathcal{U}$ and $\mathcal{U}_{ideal}$. Adversary $\mathcal{B}_{small}$ has access to an oracle returning

[6] Note that as the padding length is always random and hard to retrieve, we need not assume that $|\Psi(m_0)| = |\Psi(m_1)|$.

samples $|u|$ drawn from either $\mathcal{U}$ or $\mathcal{U}_{ideal}$. Intuitively $\mathcal{B}_{small}$ uses the fact that $\mathcal{A}_{big}$ can guess with only negligible probability against indSenderTA for $\mathcal{U}_{ideal}$: if $\mathcal{A}_{big}$ wins, $\mathcal{B}_{small}$ guesses that the encoding is done using $\mathcal{U}$ and *not* $\mathcal{U}_{ideal}$.

Adversary $\mathcal{B}_{small}$ simulates the environment for $\mathcal{A}_{big}$ as follows: (1) Whenever $\mathcal{A}_{big}$ asks to create a new instantiation of the IND-CCA security game, draw $K = (K_E, K_M, (\mathcal{U}, f, tk)) \leftarrow \mathsf{Gen}^\star(1^\lambda)$; (2) When $\mathcal{A}_{big}$ asks the encryption of a message m, query the oracle. Let $|u|$ be the sample returned by the oracle. Set $m_{\mathsf{pad}} = \langle |u|, m, u \rangle$ where $u \leftarrow \{0,1\}^{|u|}$. Answer $\mathcal{A}_{big}$'s query computing $c^\star$ using the key K as described in $\Pi^\star$; (3) When $\mathcal{A}_{big}$ asks the decryption of a ciphertext $c^\star$, answer with the message m obtained by decrypting $c^\star$ using the key K as described in $\Pi^\star$; (4) When $\mathcal{A}_{big}$ queries the Θ_{length} oracle with an instantiation Inst, return the values $\vartheta = |m_{\mathsf{pad}}|$ corresponding to the chosen instantiation. (5) When $\mathcal{A}_{big}$ outputs $\mathcal{Q}^*_{big} = (\mathsf{Inst}^*_0, \mathcal{Q}^*_0 = \{m^0_0, m^0_0\}, \mathsf{Inst}^*_1, \mathcal{Q}^*_1 = \{m^1_0, m^1_1\})$, query the oracle. For a random bit b, compute $c^\star_b$ encrypting a random message in $\mathcal{Q}^*_b$ as specified in $\Pi^\star$. Return the challenge $\Upsilon^*_{big} = (\mathsf{Inst}^*_b, \Upsilon^*_b)$ for $\mathcal{A}_{big}$; (6) Given the guess Γ^*_{big} from the adversary, conclude that the oracle is $\mathcal{U}$ iff $\Gamma^*_{big} = b$.

Clearly $\mathcal{B}_{small}$ runs in time $t \approx t^\star$. Moreover it is easy to see that when the samples $|u|$ are chosen from $\mathcal{U}$ the view of $\mathcal{A}_{big}$ is distributed like in Game 1, whereas when the samples $|u|$ are chosen from $\mathcal{U}_{ideal}$ the view of $\mathcal{A}_{big}$ is distributed like in Game 0. Thus $\mathcal{B}_{small}$ can tell apart $\mathcal{U}$ and $\mathcal{U}_{ideal}$ with advantage $> \epsilon$, against the assumption that $\mathcal{U}$ and $\mathcal{U}_{ideal}$ are (t, ϵ)-computationally indistinguishable. Putting all together we have shown that

$$\epsilon^\star := \mathrm{Prob}[X_1 = 1] - 1/2 \leq \epsilon_{small} + \epsilon,$$

as desired. □

It remains open the problem how to realize the distribution $\mathcal{U}_{ideal}$. To this extent, only heuristic solutions are known. For instance in [23] it is shown how to achieve indistinguishability of padded SSH flows using the optimal overhead, assuming one is given in advance the probability distributions of the packet lengths output by the application. However the overhead introduced is still very high (up to 35% of the transmitted data).

References

[1] *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*. IEEE Computer Society, 2010.

[2] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *TCC*, volume 5444 of *Lecture Notes in Computer Science*, pages 474–495. Springer, 2009.

[3] 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*. ACM, 2010.

[4] Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Breaking and provably repairing the SSH authenticated encryption scheme: A case study of the Encode-then-Encrypt-and-MAC paradigm. *ACM Trans. Inf. Syst. Secur.*, 7(2):206–241, 2004.

[5] Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. The secure shell (SSH) transport layer encryption modes. RFC 4344, January 2006. `http://www.ietf.org/rfc/rfc4344.txt`.

[6] Ron Berman, Amos Fiat, and Amnon Ta-Shma. Provable unlinkability against traffic analysis. In Ari Juels, editor, *Financial Cryptography*, volume 3110 of *Lecture Notes in Computer Science*, pages 266–280. Springer, 2004.

[7] Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 89–108. Springer, 2011.

[8] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In Tal Rabin, editor, *CRYPTO*, volume 6223 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2010.

[9] Zvika Brakerski, Yael Tauman Kalai, Jonathan Katz, and Vinod Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS* [1], pages 501–510.

[10] David Brumley and Dan Boneh. Remote timing attacks are practical. *Computer Networks*, 48(5):701–716, 2005.

[11] Christian Cachin. An information-theoretic model for steganography. In David Aucsmith, editor, *Information Hiding*, volume 1525 of *Lecture Notes in Computer Science*, pages 306–318. Springer, 1998.

[12] David Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–88, 1981.

[13] George Danezis and Richard Clayton. *Digital Privacy: Theory, Technologies, and Practices*, chapter 5, Introducing Traffic Analysis, pages 96–117. Eds. Auerbach Publications, 2007.

[14] Francesco Davì, Stefan Dziembowski, and Daniele Venturi. Leakage-resilient storage. In Juan A. Garay and Roberto De Prisco, editors, *SCN*, volume 6280 of *Lecture Notes in Computer Science*, pages 121–137. Springer, 2010.

[15] Nenad Dedic, Gene Itkis, Leonid Reyzin, and Scott Russell. Upper and lower bounds on black-box steganography. In Joe Kilian, editor, *TCC*, volume 3378 of *Lecture Notes in Computer Science*, pages 227–244. Springer, 2005.

[16] Jean Paul Degabriele and Kenneth G. Paterson. On the (in)security of IPsec in MAC-then-encrypt configurations. In Al-Shaer et al. [3], pages 493–504.

[17] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption schemes with auxiliary inputs. In Micciancio [33], pages 361–381.

[18] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS* [1], pages 511–520.

[19] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302. IEEE Computer Society, 2008.

[20] Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In Micciancio [33], pages 343–360.

[21] Karine Gandolfi, Christophe Mourtel, and Francis Olivier. Electromagnetic analysis: Concrete results. In Çetin Kaya Koç, David Naccache, and Christof Paar, editors, *CHES*, volume 2162 of *Lecture Notes in Computer Science*, pages 251–261. Springer, 2001.

[22] Vipul Goyal and Abhishek Jain. On the round complexity of covert computation. In Leonard J. Schulman, editor, *STOC*, pages 191–200. ACM, 2010.

[23] Alfonso Iacovazzi and Andrea Baiocchi. Optimum packet length masking. In *22nd International Teletraffic Congress (ITC)*, pages 1–8, 2010.

[24] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, 2003.

[25] 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*. Springer, 2009.

[26] Ari Juels and Stephen A. Weis. Defining strong privacy for RFID. In *International Conference on Pervasive Computing and Communications - Workshops (PerCom Workshops)*, pages 342–347. IEEE Computer Society Press, 2007.

[27] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In Mitsuru Matsui, editor, *ASIACRYPT*, volume 5912 of *Lecture Notes in Computer Science*, pages 703–720. Springer, 2009.

[28] Paul C. Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *CRYPTO*, volume 1109 of *Lecture Notes in Computer Science*, pages 104–113. Springer, 1996.

[29] Paul C. Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Michael J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 388–397. Springer, 1999.

[30] Markus G. Kuhn. *Compromising emanations: eavesdropping risks of computer displays*. PhD thesis, University of Cambridge, 2003. Technical Report UCAM-CL-TR-577.

[31] Marc Liberatore and Brian Neil Levine. Inferring the source of encrypted HTTP connections. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM Conference on Computer and Communications Security*, pages 255–263. ACM, 2006.

[32] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Moni Naor, editor, *TCC*, volume 2951 of *Lecture Notes in Computer Science*, pages 278–296. Springer, 2004.

[33] Daniele Micciancio, editor. *Theory of Cryptography, 7th Theory of Cryptography Conference, TCC 2010, Zurich, Switzerland, February 9-11, 2010. Proceedings*, volume 5978 of *Lecture Notes in Computer Science*. Springer, 2010.

[34] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In Shai Halevi, editor, *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 18–35. Springer, 2009.

[35] Kenneth G. Paterson and Gaven J. Watson. Plaintext-dependent decryption: A formal security treatment of SSH-CTR. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 345–361. Springer, 2010.

[36] Krzysztof Pietrzak. A leakage-resilient mode of operation. In Joux [25], pages 462–482.

[37] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (EMA): Measures and countermeasures for smart cards. In Isabelle Attali and Thomas P. Jensen, editors, *E-smart*, volume 2140 of *Lecture Notes in Computer Science*, pages 200–210. Springer, 2001.

[38] Charles Rackoff and Daniel R. Simon. Cryptographic defense against traffic analysis. In *STOC*, pages 672–681, 1993.

[39] Jean-François Raymond. Traffic analysis: protocols, attacks, design issues and open problems. *Designing Privacy Enhancing Technologies*, 2009:10–29, 2001.

[40] Ahmad-Reza Sadeghi, Ivan Visconti, and Christian Wachsmann. Anonymizer-enabled security and privacy for RFID. In *Advances in Cryptology — Asiacrypt'08*, volume 5888 of *Lecture Notes in Computer Science*, pages 292–299. Springer-Verlag, 2009.

[41] François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Joux [25], pages 443–461.

[42] François-Xavier Standaert, Oliver Pereira, Yu Yu, Jean-Jacques Quisquater, Moti Yung, and Elisabeth Oswald. *Leakage Resilient Cryptography in Practice*, chapter in Towards Hardware Intrinsic Security: Foundation and Practice, pages 105–139. Springer, 2010.

[43] Serge Vaudenay. On privacy models for RFID. In *Advances in Cryptology — Asiacrypt'07*, volume 4883 of *Lecture Notes in Computer Science*, pages 68–87. Springer-Verlag, 2007.

[44] Luis von Ahn, Nicholas J. Hopper, and John Langford. Covert two-party computation. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 513–522. ACM, 2005.

[45] Yu Yu, François-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In Al-Shaer et al. [3], pages 141–151.

A Existent and Future Work

Related Work. We have mentioned key-related leakage resilience as a related field. In fact, key-related leakage resilience and our notion of indistinguishability in the presence of traffic analysis are non-overlapping. Whereas our notion models *arbitrary* leakage (the TA oracle can yield information wholly unrelated to the secret key), leakage resilience allows the adversary to *adaptively* choose functions that it wishes to evaluate on the secret key (possibly with other adversary-chosen input parameters). This is not possible in our framework, which presets the traffic analysis oracle, and proves indistinguishability in the presence of that particular oracle. Note that a TA oracle can be composite, i.e. it can yield a possibly large (but not infinite) set of outputs, modelling for example a large set of pre-set functions of the secret key that need to be evaluated (but this set is still not adaptively updated). However, as argued by Standaert et al [42, 45], non-adaptivity is enough in many applications of side-channel attacks (and full-adaptivity is in general a too strong requirement).

Further groundwork on privacy for traffic analysis has also been laid (see [39] for a survey and further references). The explicit goal of traffic analysis is usually described as sender-recipient matching (if the sender of a message is known, identify the receiver, and if the receiver is known, identify the sender). This is exactly the notion we attempt to capture by our indistinguishability framework: indReceiverTA-security hides receivers, whereas indSenderTA hides senders. However, previous works on traffic analysis, building on Chaum's work on anonymous communication [12] and on Rackoff and Simon's work on anonymous traffic [38], mainly reflects an adversary's ability to monitor a communication channel and tell whether it is used or not. Usually one extensively elaborates on the topic of routing, arguing that anonymising routing would make it impossible for an adversary to tell the sender of a message even if it knows the recipient, and viceversa. The adversary described in this framework has the ability to manipulate nodes, time routes, and handle messages, but *not* to leak information from the implementation of a primitive. In fact, this aspect is explicitly outside the scope of [12, 38].

Our approach is more general. In particular, non-anonymous routing can be modelled by our traffic analysis oracle, but we do not limit ourselves to this. Even if routing is anonymised, thus making message delivery untraceable, it is still possible to match senders to recipients by other implementation characteristics, such as response duration or fingerprints in the communication. We note that though Rackoff and Simon's framework

[38] is thorough, it does not reflect an adversary's ability to use side-channel analysis and fingerprint communication. Here, the authors use techniques from Secure Multiparty Computation (SMC) and Non-Interactive Zero-Knowledge (NIZK) to achieve message "untraceability" in a network of synchronously communicating players.

Berman, Fiat, and Ta-Shma [6] address traffic analysis in the asynchronous case, introducing "obscurant networks" – i.e., networks able to obscure the destination of each particular player – further allowing the adversary to have some a-priori information about the communication patterns (e.g., some players exchange less messages than others). Whereas this is a particular case of our approach here, we consider far more general settings, allowing the adversary to learn further information than just the message load on a line of communication.

In fact, to the best of our knowledge our theoretical framework is the first to extend *general* provable-security models to account for traffic analysis and prove security *and* privacy in the presence thereof. We also comprehensively outline the relations between different flavours of traffic analysis.

Limitations of our Model. Classical traffic analysis can have two goals. One is distinguishing the *source of communication*. The second is to tell whether *communication is taking place at all*. In our model, we address only the first threat. The second notion – known as *steganography* [11] or *covert computation* [44] in its generalised form – is, however, much stronger and less practical to achieve [15, 22]. Communication can be hidden by simulating honest interactions on random communication lines, such that the set of honest protocol runs and the set of simulated ones are distributed in a computationally indistinguishable way with respect to time. For infrequently used communication lines, this is impossible to do in practice (here the set of simulated communication would be much larger than the set of honest communication, thus the overhead is overwhelming). A general, frequently used solution to hiding communication is frequency hopping, i.e. the properties of the communication channel are changed in a random fashion such that an adversary monitoring traffic cannot guess in advance which frequency the parties are using. This solution is not always useful, as an adversary may analyse the patterns of frequency hopping, thus distinguishing the randomiser that is being used.

Apart from its scope, our model is also limited by its generality. We try to encompass as many types of security notions and primitives as possible; however, in order to prove and generalise our results, we need to make some assumptions regarding the structure of the security game, something we call a canonical form. This canonical form is not too restrictive: indeed we show that many "classical" security definitions fit our framework. We have thus far considered only game-based security, leaving simulation-based notions for future work.

Finally, in this work we consider two concrete cases of traffic analysis. An interesting and natural extension would be to consider a larger, more abstract class of traffic analysis attacks (in a way similar to the context of leakage-resilient cryptography) for which to assess security. We leave this as an open problem for future research.

Traffic Analysis in Practice. Traffic analysis has its roots in the military context; indeed it has been widely used already during the World War I to learn the intentions and actions of the enemy. It is within the development of wireless communication, however, that TA has become a so powerful means, since radio communications allow to easily deduce information by monitoring patterns in communication. Traffic analysis is not only used for military purposes, but it is also a big concern in computer security, especially over the Internet. In this context several attacks are known to obtain traffic monitoring: traffic classification is possible based on statistical features of packet flows. Among the others features, packet length is one of the most used.

A concrete example is SSH [5], one of the most widely used secure network protocols. Several techniques are known (e.g. [23, 35] and cited references) to learn (otherwise encrypted) message lengths for SSH tunnels. This allows for very accurate traffic monitoring and classification, generating a huge privacy problem. See also [13] for applications of TA over the Internet, in contexts different from SSH.

The main attraction of TA with respect to side-channel leakage is that the former is much cheaper and easy to realize: traffic data can be automatically collected and processed to provide high level intelligence.

B Canonical Games in Cryptography

As stated before, the assumption that security games are canonical is not too strong. In this section we show how various known definitions fit our framework.

Public-key encryption. Let $\Pi(A, B) = (\mathsf{Setup} = \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme. The set of players $\mathcal{P} = (A, B)$ is made of only two parties, called Alice and Bob. The set $\mathcal{F}$ consists of $\mathsf{Enc}, \mathsf{Dec}$. The strongest notion of security is IND-CCA2 security, which is a canonical game with algorithms $(\mathsf{Setup}, \mathsf{Learn}, \mathsf{ChGen}, \mathsf{Match})$ as follows.

On input a security parameter λ, algorithm Setup runs the key-generation algorithm Gen on 1^λ, outputting $spar = sk$ and $ppar = pk$, i.e. a private-public key pair. The algorithm Learn on input any query $\mathcal{Q} = (\mathsf{Dec}, c)$ for ciphertexts c, outputs transcripts of the form $\tau = (\mathsf{Dec}, c, m)$, where $m \leftarrow \mathsf{Dec}_{sk}(c)$. For administrative purposes, this algorithm keeps a database recording the transcripts for all $i = 1, \ldots, Q = Q' + Q''$. Upon input set $\mathcal{Q}^* = (m_0, m_1)$, the challenge algorithm ChGen picks a bit $b \leftarrow \{0, 1\}$, then runs $\Upsilon^* \leftarrow \mathsf{Enc}_{pk}(m_b)$. Finally, on inputs a guess $\Gamma^* \in \{0, 1\}$ and a challenge Υ^*, the algorithm Match outputs $d = 1$ iff (2) $\Gamma^* = b$ (i.e. the bit that was chosen by the challenger), (2) the messages in $\mathcal{Q}^*$ have the same length $|m_0| = |m_1|$, and (3) there exists no $j \in \{Q' + 1, \ldots, Q\}$ such that $\mathcal{Q}_j = (\mathsf{Dec}, \Upsilon^*)$.

For the security of this game, note that the hardness of the game is $\delta = 1/2$.

Message authentication codes. Let $\Pi(A, B) = (\mathsf{Setup} = \mathsf{Gen}, \mathsf{MAC}, \mathsf{VRFY})$ be a Message Authentication Code (MAC). The set of players $\mathcal{P} = (A, B)$ consists again of only Alice and Bob. The set $\mathcal{F}$ consists of $\mathsf{MAC}, \mathsf{VRFY}$. The standard notion of security is universal unforgeability, which is a canonical game with algorithms $(\mathsf{Setup}, \mathsf{Learn}, \mathsf{ChGen}, \mathsf{Match})$ as follows.

On input a security parameter λ, algorithm Setup runs the key-generation algorithm Gen on 1^λ, outputting $spar = K$ and $ppar = \varepsilon$, i.e. the public parameters are the empty string ε. The algorithm Learn takes input of two forms. On input queries $\mathcal{Q} = (\mathsf{MAC}, m)$ for message m, the learning algorithm runs $\phi \leftarrow \mathsf{MAC}_K(m)$ and outputs transcripts of the form $\tau = (\mathsf{MAC}, m, \phi)$. On input queries $\mathcal{Q} = (\mathsf{VRFY}, m, \phi)$, the learning algorithm runs $b \leftarrow \mathsf{VRFY}_K(m, \phi)$ and outputs transcripts of the form $\tau = (\mathsf{VRFY}, m, \phi, b)$. For administrative purposes, this algorithm keeps a database recording all query-transcript pairs $(\mathcal{Q}_i, \tau_i)$ for all $i = 1, \ldots, Q = Q' + Q''$. The challenge phase is empty, i.e. on any input $\mathcal{Q}^*$, the algorithm ChGen outputs $\Upsilon^* = \varepsilon$. On input a guess $\Gamma^* = (m^*, \phi^*)$ for message m^* and tag ϕ^*, the algorithm Match outputs $d = 1$ iff (1) $\mathsf{VRFY}_K(m^*, \phi^*) = 1$ and (2) there exists no entry $j \in \{1, \ldots, Q\}$ in the database of the learning oracle such that $\mathcal{Q}_j = (\mathsf{MAC}, m^*)$. For the security of this game, note that the hardness of the game is $\delta = 0$.

Indistinguishability from random. Let $\Pi(\mathcal{P}) = (\mathsf{Setup} = \mathsf{Gen}, \mathsf{PRF})$ be a Pseudo-Random Function (PRF) with key space $\mathcal{K}$, domain $\mathcal{X}$ and range $\mathcal{Y}$. The set of players $\mathcal{P}$ consists of a single party (namely the user of the PRF). The security definition is based on indistinguishability between $\mathsf{PRF}_K(\cdot)$ (for a randomly chosen $K \leftarrow \mathcal{K}$) and a truly random function $\pi(\cdot)$ (chosen at random from all possible functions with domain $\mathcal{X}$ and range $\mathcal{Y}$). The security of the PRF, i.e. pseudorandomness, is a canonical game $(\mathsf{Setup}, \mathsf{Learn}, \mathsf{ChGen}, \mathsf{Match})$ defined as follows.

On input a security parameter λ, algorithm Setup runs the key-generation algorithm Gen on 1^λ, outputting a randomly chosen $spar = K \in \mathcal{K}$ and $ppar = \varepsilon$. A random bit b is drawn and kept secret by the challenger. The algorithm Learn, on input of the form $\mathcal{Q} = (\mathsf{PRF}, x)$, returns $\perp$ if Υ^* is undefined. Else, if Υ^* has already been output (see below), if the challenger's private bit is $b = 1$, algorithm Learn outputs $y = \mathsf{PRF}_K(x)$, or else if $b = 0$, it outputs a truly random value $y = \pi(x)$. The transcript output by Learn is $\tau = (\mathsf{PRF}, x, y)$. The challenge phase is empty, i.e. on any input $\mathcal{Q}^*$, the algorithm ChGen outputs $\Upsilon^* = \varepsilon$. On input a guess $\Gamma^* \in \{0, 1\}$ and the empty challenge ε, the algorithm Match outputs $d = 1$ iff $\Gamma^* = b$. For the security of

this game, note that the hardness of the game is $\delta = 1/2$.[7]

Authentication. We recall here the indistinguishability-based privacy experiment for RFID authentication, essentially the model in [26]. The primitive is defined as $\Pi(\mathcal{P}) = (\mathsf{Setup}, \mathcal{O}_{\mathsf{RFID}} = \{\mathsf{InitR}, \mathsf{Send}\mathcal{T}, \mathsf{Send}\mathcal{R}, \mathsf{Corrupt}\})$ for a set of players $\mathcal{P}$ consisting of tags $\mathcal{T}_1, \ldots, \mathcal{T}_n$ and a single reader $\mathcal{R}$. The integer number of tags n is a parameter of Setup. Informally security is defined as the adversary's inability to tell two tags $\mathcal{T}_i$ and $\mathcal{T}_j$ apart, i.e. the security game is canonical with algorithms $(\mathsf{Setup}, \mathsf{Learn}, \mathsf{ChGen}, \mathsf{Match})$ as follows.

On input a security parameter λ and an integer n, algorithm Setup runs the tag- and reader-initialisation algorithms Setup and InitR, such that to each tag there corresponds an (initial) key $K^0_{\mathcal{T}_i}$ and some initial state $\Sigma^0_{\mathcal{T}_i}$. The public output of this algorithm is a set of public parameters *ppar*, which contain information such as the public IDs of tags, the reader's public identifier, etc. The algorithm Learn runs on inputs of four types. On input $\mathcal{Q} = \mathsf{InitR}$, the learning algorithm initialises a new reader, and outputs the empty string in a partial transcript of the form $\tau = (\mathsf{InitR}, \varepsilon)$. On input $\mathcal{Q} = (\mathsf{Send}\mathcal{T}, j, m)$, if tag $\mathcal{T}_j$ is already associated with a partial transcript τ^j, the learning algorithm forwards the message m to this tag and collects its response r, appending the following entry to the partial transcript: $(\mathsf{Send}\mathcal{T}, j, m, r)$; if no partial transcript is associated with the tag, a new transcript is initiated with this entry. Then the learning algorithm outputs the updated transcript $\tau^j \leftarrow \tau^j || (\mathsf{Send}\mathcal{T}, j, m, r)$. Similarly, queries of the form $\mathcal{Q} = (\mathsf{Send}\mathcal{R}, j, m)$ will yield updated partial transcripts of the form $\tau^j \leftarrow \tau^j || (\mathsf{Send}\mathcal{R}, j, m, r)$ (the index j specifies that the message will be sent to the reader in the session between it and tag $\mathcal{T}_j$). Finally, on input $\mathcal{Q} = (\mathsf{Corrupt}, j)$, the learning algorithm updates the partial transcript of the current session associated with tag $\mathcal{T}_j$ with the entry $(\mathsf{Corrupt}, j, K^*_{\mathcal{T}_j}, \Sigma^*_{\mathcal{T}_j})$, i.e. the current key and state information of this tag. The updated partial transcript $\tau^j \leftarrow \tau^j || (\mathsf{Corrupt}, j, K^*_{\mathcal{T}_j}, \Sigma^*_{\mathcal{T}_j})$ is output by Learn. For administration purposes, each query to the learning oracle is indexed in a database $\mathcal{D}$ in the form $(\mathcal{Q}_i, \tau^i)$ for all queries $i = 1, \ldots, Q = Q' + Q''$. On input two tags $\mathcal{T}_i$ and $\mathcal{T}_j$, the algorithm ChGen picks a bit b and starts a new partial transcript (and corresponding session) τ^b for tag $\mathcal{T}_i$ if $b = 0$ and for tag $\mathcal{T}_j$ if $b = 1$. By abusing notation we will refer to $\mathcal{T}_b$ as taking the values $\mathcal{T}_i$ or $\mathcal{T}_j$, depending on the value of b. The initial partial transcript $\Upsilon^* = \tau^b$ is then output. On input a guess $\Gamma^* \in \{0, 1\}$, the algorithm Match outputs $d = 1$ iff (1) $\Gamma^* = b$, (2) there is no query of the form $(\mathsf{Corrupt}, b, *, *)$ in $\mathcal{D}$, and (3) there are no queries of the form $(\mathcal{O}_{\mathsf{RFID}}, i, *)$ or $(\mathcal{O}_{\mathsf{RFID}}, j, *)$ in $\mathcal{D}$. Else, the algorithm Match outputs $d = 0$.

C Proof of Theorems 3.1 – 3.3

C.1 Security in the presence of Traffic Analysis

In indTA an adversary $\mathcal{A}_{big}$ may query a traffic analysis oracle Θ which leaks information related to the implementation of a primitive. The aim of $\mathcal{A}_{big}$ is to distinguish between two instantiations of the security game, thus a single bit of information suffices to win against indTA. If we now allow local adversaries $\mathcal{A}_{small}$ in Game access to Θ, we have *security in the presence of traffic analysis*, where $\mathcal{A}_{small}$'s goal is to break the underlying canonical security game Game. If the traffic analysis oracle is restricted, we speak of security in the presence of restricted TA (secRestTA); else, for unrestricted Θ we have security for unrestricted TA (secTA). The formal treatment of secRestTA is as follows. For a primitive $\Pi(\mathcal{P})$ and player set $\mathcal{P}$, let Game be its canonical security game. We give adversary $\mathcal{A}_{small}$ against secRestTA access to the oracle Θ_{rest} (for secTA we consider an unrestricted oracle Θ). The security experiment is outlined below. Let $\Pi(\mathcal{P}) = (\mathsf{Setup}, \mathcal{F})$ be a primitive, with canonical-form security game $\mathsf{Game} = (\mathsf{Setup}, \mathsf{Learn}, \mathsf{ChGen}, \mathsf{Match})$. Consider the following experiment:

Experiment $\mathbf{Exp}^{\text{secRestTA}}_{\Pi(\mathcal{P}),\Theta_{rest}}(\mathcal{A}_{small}, \lambda)$:

[7] Let us stress that the standard definition for PRFs requires to compare the output of the distinguisher, rather than guessing the bit b. Nevertheless, it is immediate to see that the two definitions are equivalent.

1. $(ppar, spar) \leftarrow \mathsf{Setup}(1^\lambda)$. Adversary $\mathcal{A}_{small}$ is given $ppar$.
2. $\mathcal{Q}^* \leftarrow \mathcal{A}_{small}^{\mathsf{Learn}(\cdot)^{Q'}, \Theta_{rest}(\cdot)^{Q'_{\mathsf{TA}}}}(ppar)$.
3. $\Upsilon^* \leftarrow \mathsf{ChGen}(\mathcal{Q}^*)$.
4. $\Gamma^* \leftarrow \mathcal{A}_{small}^{\mathsf{Learn}(\cdot)^{Q''}, \Theta_{rest}(\cdot)^{Q''_{\mathsf{TA}}}}(ppar, \Upsilon^*)$.
5. The experiment outputs $d \leftarrow \mathsf{Match}(\Upsilon^*, \Gamma^*)$.

Let $Q = Q' + Q''$ denote the number of queries to Learn in both the learning and the guessing phases; similarly, $Q_{\mathsf{TA}} = Q'_{\mathsf{TA}} + Q''_{\mathsf{TA}}$ is the number of queries to the Θ_{rest} oracle. The advantage of the adversary in this game is defined as:

$$\mathbf{Adv}^{\text{secRestTA}}_{\Pi(\mathcal{P})}(\mathcal{A}_{small}, \Theta_{rest}) = \left| \text{Prob}\left[\mathbf{Exp}^{\text{secRestTA}}_{\Pi(\mathcal{P}), \Theta_{rest}}(\mathcal{A}_{small}, \lambda) = 1 \right] - \delta \right|.$$

Definition C.1 (secRestTA **security)** *The primitive* $\Pi(\mathcal{P})$ *is* $(t, Q, Q_{\mathsf{TA}}, \epsilon)$*-secure against* secRestTA *attacks for restricted* Θ_{rest} *and (canonical) game* Game *if for every PPT adversary* $\mathcal{A}_{small}$ *running in time at most* t*, and making at most* Q*, resp.* Q_{TA} *queries, we have:*

$$\mathbf{Adv}^{\text{secRestTA}}_{\Pi(\mathcal{P})}(\mathcal{A}_{small}, \Theta_{rest}) \leq \epsilon.$$

Analogously, we define secTA for unrestricted Θ by replacing Θ_{rest} in the definition above by Θ.

C.2 Receiver and Sender Indistinguishability

Lemma C.1 (Unrestricted indSenderTA $\Rightarrow$ **Restricted** indSenderTA**)** *If a primitive* Π *is* (t, Q, ϵ)*-secure with respect to a canonical security game* Game *and if there exists an adversary* $\mathcal{A}_{big}$ *which* $(t_{rest}, Q, Q_{\mathsf{TA}}, \epsilon_{rest})$*-breaks the restricted* indSenderTA *security of* Π *for* Θ_{rest}*, then there exists an adversary* $\mathcal{B}_{big}$ *which* $(t_{unr}, Q, Q_{\mathsf{TA}}, \epsilon_{unr})$*-breaks the unrestricted* indSenderTA *security of* Π *for any unrestricted* Θ *oracle such that* $\mathsf{proj}(\Theta) = \Theta_{rest}$*, where:*

$$t_{unr} \approx t_{rest} \qquad \text{and} \qquad \epsilon_{unr} \geq \epsilon_{rest}.$$

In other words, unrestricted sender indistinguishability in the presence of traffic analysis implies restricted sender indistinguishability.

Proof. This proof is trivial and follows from the definition of the two experiments. □

Lemma C.2 (Unrestricted indSenderTA $\not\Leftarrow$ **Restricted** indSenderTA**)** *There exists a primitive* Π*, a restricted* Θ_{rest} *oracle, and an unrestricted* Θ *oracle with* $\mathsf{proj}(\Theta) = \Theta_{rest}$ *such that the following holds: (1)* Π *is* (t, Q, ϵ)*-secure with respect to a canonical security game* Game *(for negligible* ϵ*); (2)* Π *is* $(t_{rest}, Q, Q_{\mathsf{TA}}, \epsilon_{rest})$*-secure against restricted* indSenderTA *attacks using* Θ_{rest} *(for negligible* ϵ_{rest}*); (3)* Π *is* in*secure against unrestricted* indSenderTA *attacks using* Θ*. In other words, restricted sender indistinguishability does* not *imply unrestricted sender indistinguishability.*

Proof. The intuition of the separation is to use the unrestricted Θ oracle to forward the adversary information that allows it to break the security of the underlying game, thus winning with probability 1.

We consider as a counterexample an encryption scheme $\Pi(A, B) = (\mathsf{Setup} = \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ with the canonical IND-CCA2 game outlined in Appendix B. Assume that Π is (t, Q, ϵ)-secure with respect to this game. Let Θ_{rest} be a restricted traffic analysis oracle such that Π is $(t_{rest}, Q, Q_{\mathsf{TA}}, \epsilon_{rest})$-secure against restricted indSenderTA attacks using Θ_{rest} (for negligible ϵ_{rest}).

Now consider the unrestricted oracle Θ such that $\Theta(\mathsf{Inst}) = (\Theta_{rest}(\mathsf{Inst}), sk_{\mathsf{User}})$ for $\mathsf{Inst} \in \mathcal{I}_{\mathsf{User}}$, where we write sk_{User} for the secret key of user User. We build an adversary $\mathcal{A}_{big}$ against unrestricted indSenderTA as follows: the game is first set up by the challenger and a user User is created. Then $\mathcal{A}_{big}$ starts (by using $\mathsf{NewInst}$)

two (same-user) instantiations Inst_0^* and Inst_1^* of User. Firstly, $\mathcal{A}_{big}$ queries Θ to learn sk_{User}. It then picks $\mathcal{Q}_0^* = (m_0, m_0)$ for Inst_0^* and $\mathcal{Q}_1^* = (m_1, m_1) \neq \mathcal{Q}_0^*$ for Inst_1^* and forwards input $\mathcal{Q}_{big}^* = (\mathsf{Inst}_0^*, \mathcal{Q}_0^*, \mathsf{Inst}_1^*, \mathcal{Q}_1^*)$. The challenger runs ChGen in both instantiations and forwards to $\mathcal{A}_{big}$ the challenge $\Upsilon_{big}^* = (\mathsf{Inst}_b^*, \Upsilon_b^*)$, where $\Upsilon_b^* \leftarrow \mathsf{Enc}_{pk}(m_b)$. By using sk_{User}, the adversary decrypts the challenge ciphertext and thus guesses bit b with probability 1. □

Lemma C.3 (Unrestricted indSenderTA $\not\Leftarrow$ indReceiverTA**)** *There exists a primitive* Π *and an unrestricted oracle* Θ *such that the following holds: (1)* Π *is* (t, Q, ϵ)*-secure with respect to a canonical security game* Game *(for negligible* ϵ*); (2)* Π *is* $(t_{Glob}, Q, Q_{\mathsf{TA}}, \epsilon_{Glob})$*-secure against* indReceiverTA *attacks using* Θ *(for negligible* ϵ_{Glob}*); (3)* Π *is* in*secure against unrestricted* indSenderTA *attacks using* Θ*. In other words, receiver indistinguishability does not imply unrestricted sender indistinguishability.*

Proof. The intuition for this counterexample is to consider a tailored TA oracle, which will only give the adversary valuable information if the challenge instantiations are same-user instantiations of Game. In this way, the adversary can only distinguish same-user instantiations. This is a rather artificial counterexample, but one which nonetheless reflects the gap between the two notions.

We again take the example of IND-CCA2 secure encryption. Let $\Pi(A, B) = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be an encryption scheme that is (t, Q, ϵ)-secure against IND-CCA2. We introduce an unrestricted traffic analysis oracle Θ as follows. Whenever this oracle is run before the challenge Υ_{big}^* is generated, Θ simply outputs $\perp$. The challenge is generated on instantiations Inst_0^*, Inst_1^*; if they are mixed-user instantiations, the Θ oracle continues to output the value $\perp$ for any future query. If Inst_0^* and Inst_1^* are same-user instantiations, the Θ oracle will output, when queried on input Inst_b^*, the secret key sk_b. In other words, if the two instantiations forwarded by $\mathcal{A}_{big}$ are same-instance, $\mathcal{A}_{big}$ learns the value of the secret key in the challenge instantiation.

Note that Π is (t, Q, ϵ)-secure against IND-CCA2 (since Θ cannot be used by adversaries $\mathcal{A}_{small}$ against Game). Furthermore, $\mathcal{A}_{big}$ against indReceiverTA learns no information from the traffic analysis oracle Θ (as its challenge instantiations are mixed-user). Finally, we describe an adversary $\mathcal{A}_{big}$ against unrestricted indSenderTA: this adversary generates same-user instantiations Inst_0^* and Inst_1^* and chooses m_0^0 and $m_1^0 \neq m_0^0$ as input for Inst_0^* and $m_0^1 \notin \{m_0^0, m_0^1\}$ and $m_1^1 \notin \{m_1^0, m_0^1, m_0^0\}$ (all four challenge messages are distinct) as input for Inst_1^*. The adversary $\mathcal{A}_{big}$ forwards this input to the challenger, who runs the challenge phases in both instantiations, and outputs $(\mathsf{Inst}_b^*, \Upsilon_b^*)$ to $\mathcal{A}_{big}$. The adversary $\mathcal{A}_{big}$ runs Θ on Inst_b^*, thus learning the value of sk_b, decrypting the challenge message, and distinguishing the challenge instantiation with probability 1.

□

Lemma C.4 (Unrestricted indSenderTA $\not\Rightarrow$ indReceiverTA**)** *There exists a primitive* Π *and an unrestricted* Θ *oracle such that the following holds: (1)* Π *is* (t, Q, ϵ)*-secure with respect to a canonical security game* Game *(for negligible* ϵ*); (2)* Π *is* $(t_{Loc}, Q, Q_{\mathsf{TA}}, \epsilon_{Loc})$*-secure against unrestricted* indSenderTA *attacks using* Θ *(for negligible* ϵ_{Loc}*); (3)* Π *is* in*secure against* indReceiverTA *attacks using* Θ*. In other words, unrestricted sender indistinguishability does not imply receiver indistinguishability.*

Proof. The intuition for this counterexample is to modify the primitive such that it reveals information specific to *users* (but which remains the same for all *instantiations* of the same user). In this way, indSenderTA security is preserved, but indReceiverTA can be broken.

Take again the example of an encryption scheme $\Pi'(A, B) = (\mathsf{Gen}', \mathsf{Enc}', \mathsf{Dec}')$ which is (t, Q, ϵ)-secure against IND-CCA2, and let Θ be *any* unrestricted TA oracle such that $\Pi'(A, B)$ is $(t_{Loc}, Q, Q_{\mathsf{TA}}, \epsilon_{Loc})$-secure against unrestricted indSenderTAattacks using Θ (for negligible ϵ_{Loc}).

We modify $\Pi'(A, B)$ to $\Pi(A, B) = (\mathsf{Setup} = \mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that at each instantiation of this scheme, Gen first picks a random session id sid, then runs Gen', and finally outputs $sk = sk'$ and $pk = (pk', sid)$. The output of Enc on message m is $(\mathsf{Enc}'_{pk'}(m), sid)$. The decryption algorithm Dec discards sid and simply runs Dec'. Note that Π is still (t, Q, ϵ)-secure against IND-CCA2 and $(t_{Loc}, Q, Q_{\mathsf{TA}}, \epsilon_{Loc})$-secure against unrestricted indSenderTA

attacks using Θ, as *sid* gives no information about the ciphertext and since all same-user instantiations Inst_0^* and Inst_1^* have the same *sid*. However, an adversary $\mathcal{A}_{big}$ against indReceiverTA security will create users User_0 and User_1, and instantiations Inst_0^* of User_0 and Inst_1^* of User_1, receiving the public keys of both users (thus in particular their session ids). Then $\mathcal{A}_{big}$ checks that $sid_0 \neq sid_1$ (this will happen w.h.p., but if the two id's are equal, $\mathcal{A}_{big}$ will continue creating users until it finds two users with distinct *sid* values). In the challenge phase, $\mathcal{A}_{big}$ picks a random message m, then forwards $\mathcal{Q}_{big}^* = (\mathsf{Inst}_0^*, \mathcal{Q}_0^* = (m, m), \mathsf{Inst}_1^*, \mathcal{Q}_1^* = (m, m))$ to the challenger, who runs ChGen_{big} on this input and outputs $\Upsilon_{big}^* = (\mathsf{Inst}_b^*, \Upsilon_b^*)$, where $\Upsilon_b^* = (\mathsf{Enc}'_{pk'_b}(m), sid_b)$. Now $\mathcal{A}_{big}$ guesses the bit b by comparing the value of sid_b with sid_0 and sid_1 and wins with probability 1. □

C.3 Restricted and Unrestricted Security with Traffic Analysis

Lemma C.5 (secTA ⇒ secRestTA) *If* Π *is* (t, Q, ϵ)*-secure with respect to a canonical security game* Game *and if there exists an adversary* $\mathcal{A}_{small}$ *which* $(t_{rest}, Q, Q_{\mathsf{TA}}, \epsilon_{rest})$*-breaks the* secRestTA *security of* Π *for* Θ_{rest}*, then there exists an adversary* $\mathcal{B}_{small}$ *which* $(t_{unr}, Q, Q_{\mathsf{TA}}, \epsilon_{unr})$*-breaks the* secTA *security of* Π *for any unrestricted* Θ *oracle such that* $\mathsf{proj}(\Theta) = \Theta_{rest}$*, where:*

$$t_{unr} \approx t_{rest} \qquad \text{and} \qquad \epsilon_{unr} \geq \epsilon_{rest}.$$

In other words, unrestricted security against traffic analysis implies restricted security against traffic analysis.

Proof. The proof for this statement is trivial by the definition of the security games, as in particular we can take $\Theta = \Theta_{rest}$. □

Lemma C.6 (secTA ⇏ secRestTA) *There exists a primitive* Π*, a restricted* Θ_{rest} *oracle, and an unrestricted* Θ *oracle with* $\mathsf{proj}(\Theta) = \Theta_{rest}$ *such that the following holds: (1)* Π *is* (t, Q, ϵ)*-secure with respect to a canonical security game* Game*(for negligible* ϵ*); (2)* Π *is* $(t_{rest}, Q, Q_{\mathsf{TA}}, \epsilon_{rest})$*-secure against* secRestTA *attacks for* Θ_{rest} *(for negligible* ϵ_{small}*); (3)* Π *is* in*secure against* secTA *attacks using* Θ*. In other words, restricted security against traffic analysis does not imply unrestricted security against traffic analysis.*

Proof. We can use the encryption counterexample for the proof of Lemma C.2 for this separation. Knowing the secret key by means of the unrestricted Θ oracle now allows the adversary to break the security of the IND-CCA2 game. □

C.4 Security and Indistinguishability with Traffic Analysis

Lemma C.7 (Restricted indSenderTA ⇒ secRestTA**)** *If a primitive* Π *is* (t, Q, ϵ)*-secure with respect to a canonical security game* $\mathsf{Game} = (\mathsf{Setup}, \mathsf{Learn}, \mathsf{ChGen}, \mathsf{Match})$ *with hardness* δ *and if there exists an adversary* $\mathcal{A}_{small}$ *which* $(t, Q, Q_{\mathsf{TA}}, \epsilon_{small})$*-breaks the* secRestTA *security of* Π *for a restricted* Θ_{rest}*, then there exists an adversary* $\mathcal{A}_{big}$ *which* $(t_{big}, Q + 2, Q_{\mathsf{TA}}, \epsilon_{big})$*-breaks the restricted* indSenderTA *security of* Π *for* Θ_{rest}*, where:*

$$t_{big} \approx t \qquad \text{and} \qquad \epsilon_{big} \geq \epsilon_{small}/2 - \epsilon/2.$$

In other words, restricted sender indistinguishability implies restricted security against traffic analysis.

Proof. We build adversary $\mathcal{A}_{big}$ as follows: (1) Firstly the challenger creates user User; (2) Then $\mathcal{A}_{big}$ creates instantiations Inst_0^* and Inst_1^* (this accounts for the 2 additional queries made by $\mathcal{A}_{big}$ to the learning oracles); (3) $\mathcal{A}_{big}$ runs $\mathcal{A}_{small}$ on Inst_0^*, answering all the TA queries that $\mathcal{A}_{small}$ makes by using its own Θ_{rest} oracle on input Inst_0^*; (4) When $\mathcal{A}_{small}$ returns an input $\mathcal{Q}_0^*$ for the challenge phase of Inst_0^*, adversary $\mathcal{A}_{big}$ chooses (same-user) instantiations Inst_0^* and Inst_1^* and forwards as input $\mathcal{Q}_0^*$ for the challenge in Inst_0^* and a random input $\mathcal{Q}_1^*$ for Inst_1^*; (6) The challenger enters the challenge phase, picks a bit b and outputs challenge $\Upsilon_{big}^* = (\mathsf{Inst}_b^*, \Upsilon_b^*)$, which $\mathcal{A}_{big}$ forwards to $\mathcal{A}_{small}$ in instantiation Inst_0^* and continues answering the TA queries by $\Theta_{rest}(\mathsf{Inst}_0^*)$; (7) Finally,

$\mathcal{A}_{small}$ outputs a guess Γ^*_{small}, which it forwards to $\mathcal{A}_{big}$, and $\mathcal{A}_{big}$ forwards it to the challenger in Inst^*_b, receiving a bit d (indicating if the game has been won or lost); (8) $\mathcal{A}_{big}$ outputs $1-d$ as its own guess.

Intuitively, the adversary $\mathcal{A}_{big}$ guesses in advance that the challenger will pick the bit $b=0$ and runs $\mathcal{A}_{small}$ on this instantiation; if the game is won, $\mathcal{A}_{big}$ outputs 0 as its guess, whereas if the game is lost, $\mathcal{A}_{big}$ assumes that it was the other instantiation and outputs a guess of 1. Note that the responses of $\mathcal{A}_{big}$ simulate $\mathcal{A}_{small}$'s TA queries perfectly if $b=0$. For the analysis of the game note that we can express $\mathcal{A}_{big}$'s advantage as:

$$\epsilon_{big} = \text{Prob}[\,\mathcal{A}_{big} \text{ wins}] - 1/2.$$

Recall that $\mathcal{A}_{big}$ wins if $1-d=b$. Hence we can write $\mathcal{A}_{big}$'s probability of winning as follows:

$$\text{Prob}[\,\mathcal{A}_{big} \text{ wins}] = \text{Prob}[\,b=0] \cdot \text{Prob}[\,\mathcal{A}_{small} \text{ wins } \mathsf{Inst}^*_0] + \text{Prob}[\,b=1] \cdot \text{Prob}[\,\mathcal{A}_{small} \text{ loses } \mathsf{Inst}^*_1]\,.$$

By assumption and by the fact that the underlying game Game has hardness δ, we have

$$\text{Prob}[\,\mathcal{A}_{big} \text{ wins}] \geq \tfrac{1}{2}\,(\epsilon_{small} + \delta) + \tfrac{1}{2}\,(1 - \text{Prob}[\,\mathcal{A}_{small} \text{ wins } \mathsf{Inst}^*_1])\,.$$

We now show that $\text{Prob}[\,\mathcal{A}_{small} \text{ wins } \mathsf{Inst}^*_1] \leq \delta + \epsilon$. Assume the contrary, i.e. $\mathcal{A}_{small}$ wins with probability greater than this value against random instantiation Inst^*_1, for challenge Υ^*_1 generated on random input $\mathcal{Q}^*_1$, with all queries to Θ_{rest} being answered as in Inst^*_0. In this case, we build an adversary $\mathcal{B}_{small}$ which breaks the underlying security of the game with advantage greater than ϵ (which would contradict the hypothesis that Π is ϵ-secure). Note that $\mathcal{B}_{small}$ essentially plays the same game as $\mathcal{A}_{small}$, but has no Θ_{rest} information. The adversary $\mathcal{B}_{small}$ runs $\mathcal{A}_{small}$ internally, but answers all its Θ_{rest} queries by random values of appropriate length, and outputs $\mathcal{A}_{small}$'s guess at the end. As a first game hop, we replace the parameters in $\mathcal{B}_{small}$'s game by the parameters in Inst^*_1. As instantiation Inst^*_1 was generated at random, this does not decrease $\mathcal{B}_{small}$'s winning probability. In the next game hop, we replace the challenge by Υ^*_1, which again does not decrease $\mathcal{B}_{small}$'s winning probability. Finally, we note that as no Θ_{rest} queries were made to Inst^*_1, $\mathcal{B}_{small}$'s responses to Θ_{rest} do not affect $\mathcal{A}_{small}$'s probability to win Inst^*_1. Thus $\mathcal{B}_{small}$ wins with advantage greater than ϵ (contradiction).

Thus we have shown that,

$$\text{Prob}[\,\mathcal{A}_{small} \text{ wins } \mathsf{Inst}^*_1] \leq \delta + \epsilon.$$

Plugging this value, we have:

$$\text{Prob}[\,\mathcal{A}_{big} \text{ wins}] \geq \tfrac{1}{2}(\epsilon_{small} + \delta) + \tfrac{1}{2}\,(1 - (\delta + \epsilon)) = \epsilon_{small}/2 - \epsilon/2 + 1/2,$$

and therefore we obtain the claimed bound.

□

Lemma C.8 (Restricted indSenderTA $\not\Leftarrow$ secRestTA**)** *There exists a primitive* Π *and a restricted* Θ_{rest} *oracle such that the following holds:(1)* Π *is* (t, Q, ϵ)*-secure with respect to a canonical security game* Game *(for negligible* ϵ*); (2)* Π *is* $(t, Q, Q_{\mathsf{TA}}, \epsilon_{small})$*-secure against* secRestTA *attacks using* Θ_{rest} *(for negligible* ϵ_{small}*); (3)* Π *is* in*secure against restricted* indSenderTA *attacks using* Θ_{rest}*. In other words, restricted security against traffic analysis does not imply restricted sender indistinguishability.*

Proof. We use as a counterexample the "Encode-then-Encrypt & MAC" paradigm exemplified by SSH-CTR (cf. Section 4). The underlying game is IND-CCA security. Concretely let $\Pi^\star(\Psi, \Pi_E, \Pi_M)$ be as in SSH-CTR. Claim (1) in the lemma comes from the fact that $\Pi^\star$ is IND-CCA secure, as proven in [4, Theorem 8.2].

We now formally define security in the presence of Traffic Analysis; let Θ_{length} be the restricted TA oracle of Section 4. Consider the following experiment:

Experiment $\mathbf{Exp}^{\text{CCARestTA}}_{\Pi^\star,\Theta_{length}}(\mathcal{A}_{small},\lambda)$

1. $K \leftarrow \mathsf{Gen}^\star(1^\lambda)$. Let $\mathcal{O}^\star = \{\mathsf{Enc}^\star_K(\cdot), \mathsf{Dec}^\star_K(\cdot)\}$.
2. $(m_0, m_1) \leftarrow \mathcal{A}^{\Theta_{length}(\cdot),\mathcal{O}^\star}_{small}(1^\lambda)$.
3. $\Upsilon^* = c^\star_b \leftarrow \mathsf{Enc}^\star_K(m_b)$ for a random $b \leftarrow \{0,1\}$.
4. $\Gamma^* \leftarrow \mathcal{A}^{\Theta_{length}(\cdot),\mathcal{O}^\star}_{small}(1^\lambda, c^\star_b)$.
5. Output 1 iff: (i) $\Gamma^* = b$; (ii) $\mathcal{O}^\star$ has not been queried on $c^\star_b$; and (iii) $|\Psi(m_0)| = |\Psi(m_1)|$.

Here, Q, resp. Q_{TA} denote the number of queries to $\mathcal{O}^\star$, resp. Θ_{length}.

In particular, $\Pi^\star$ is (t, Q, ϵ)-secure against secRestTA for restricted Θ_{length} if for every PPT adversary $\mathcal{A}_{small}$ running in time at most t and making at most Q, resp. Q_{TA} queries, we have

$$\text{Prob}\Big[\mathbf{Exp}^{\text{CCARestTA}}_{\Pi^\star,\Theta_{length}}(\mathcal{A}_{small},\lambda) = 1\Big] \leq \frac{1}{2} + \epsilon.$$

It follows from [35, Theorem 1], that $\Pi^\star$ is secRestTA-secure as claimed in (2).

To prove (3) we will build an adversary $\mathcal{A}_{big}$ breaking restricted indSenderTA security of $\Pi^\star$ (for Θ_{length}) in polynomial time and with probability 1. The intuition for $\mathcal{A}_{big}$ is simple: in order to distinguish between two instantiations it suffices to choose the challenge messages *in the two instantiations* with different length. Then the Θ_{length} leakage applied to the challenge ciphertext will allow the adversary to distinguish the two instantiations.

More formally $\mathcal{A}_{big}$ runs $\mathbf{Exp}^{\text{indSenderTA}}_{\Pi^\star,\Theta_{length}}(\mathcal{A}_{big},\lambda)$. Initially, user $\mathsf{User} \leftarrow \mathsf{NewUser}(1^\lambda, mpar)$ is created. The adversary then creates two instantiations of User: $\mathsf{Inst}^*_0 \leftarrow \mathsf{NewInst}(\mathsf{User})$ and $\mathsf{Inst}^*_1 \leftarrow \mathsf{NewInst}(\mathsf{User})$. Now $\mathcal{A}_{big}$ has to specify the inputs $\mathcal{Q}^*_0$, $\mathcal{Q}^*_1$ for the challenge phase of resp. Inst^*_0 and Inst^*_1. The attacker does this as follows:

$$\mathcal{Q}^*_0 = \{m^0_0, m^0_1\} \qquad\qquad \mathcal{Q}^*_1 = \{m^1_0, m^1_1\},$$

for any choice of m^0_0, m^0_1, m^1_0, m^1_1 in the message space, such that $\left|m^0_0\right| = \left|m^0_1\right| = L_0$ and $\left|m^1_0\right| = \left|m^1_1\right| = L_1 \neq L_0$. (Note that due to the encoding restrictions, this ensures also that $\left|\Psi(m^b_0)\right| = \left|\Psi(m^b_1)\right|$ for $b \in \{0,1\}$.)

The external challenger runs ChGen for both Inst^*_0 and Inst^*_1 updating their state; then a random bit $b \leftarrow \{0,1\}$ is drawn and $\Upsilon^*_{big} = (\mathsf{Inst}^*_b, \Upsilon^*_b)$ is passed to $\mathcal{A}_{big}$. (Recall that Υ^*_b in Υ^*_{big} consists of the challenge ciphertext for Inst^*_b, i.e. it is an encryption of a randomly chosen message in $\mathcal{Q}^*_b$.) Thus $\mathcal{A}_{big}$ can query the Θ_{length} oracle with input Inst^*_b and Υ^*_b, receiving $\vartheta_b = |m_{\mathsf{pad}}|_b$. Finally $\mathcal{A}_{big}$ outputs 1 iff $L_1 + |u_1| = \vartheta_b - 1$, where $|u_1|$ is computed as $|u_1| = B - (L_1 + 5 \bmod B)$. ($B$ is the input length of the underlying block-cipher Π_E.)

Clearly, $\mathcal{A}_{big}$ runs in polynomial time. Moreover by the definition of $\Psi(\cdot)$ and using the fact that $L_0 \neq L_1$, exactly one between L_0 and L_1 will satisfy the equation above. Hence $\mathcal{A}_{big}$ can guess the bit b with probability 1, finishing the proof. □

Lemma C.9 (Unrestricted indSenderTA $\Rightarrow$ secTA**)** *If a primitive* Π *is* (t, Q, ϵ)*-secure with respect to a canonical security game* Game *and if there exists an adversary* $\mathcal{A}_{small}$ *which* $(t, Q, Q_{\mathsf{TA}}, \epsilon_{small})$*-breaks the* secRestTA *security of* Π *for* Θ*, then there exists an adversary* $\mathcal{A}_{big}$ *which* $(t_{big}, Q+2, Q_{\mathsf{TA}}, \epsilon_{big})$*-breaks the unrestricted* indSenderTA *security of* Π *for the same* Θ *oracle, where:*

$$t_{big} \approx t \qquad \text{and} \qquad \epsilon_{big} \geq \epsilon_{small}/2 - \epsilon/2.$$

In other words, unrestricted sender indistinguishability implies unrestricted security against traffic analysis.

Proof. This proof is essentially the same as the proof of Lemma C.7, and we therefore do not repeat it here. □

Lemma C.10 (Unrestricted indSenderTA $\not\Leftarrow$ secTA**)** *There exists a primitive* Π *and an unrestricted* Θ *oracle such that the following holds: (1)* Π *is* (t, Q, ϵ)*-secure with respect to a canonical security game* Game *(for negligible* ϵ*); (2)* Π *is* $(t, Q, Q_{\mathsf{TA}}, \epsilon_{small})$*-secure against* secTA *attacks using* Θ *(for negligible* ϵ_{small}*); (3)* Π *is* in*secure against unrestricted* indSenderTA *attacks using* Θ*. In other words, unrestricted security against traffic analysis does not imply unrestricted sender indistinguishability.*

Proof. The same proof holds here as for Lemma C.8, as any restricted Θ_{rest} oracle is also a particular case of an unrestricted Θ oracle. □

Lemma C.11 (Restricted indSenderTA $\not\Rightarrow$ secTA**)** *There exists a primitive* Π*, a restricted* Θ_{rest} *oracle, and an unrestricted* Θ *oracle with* $\mathsf{proj}(\Theta) = \Theta_{rest}$ *such that the following holds: (1)* Π *is* (t, Q, ϵ)*-secure with respect to a canonical security game* Game *(for negligible* ϵ*); (2)* Π *is* $(t, Q, Q_{\mathsf{TA}}, \epsilon_{small})$*-secure against restricted* indSenderTA *attacks using* Θ_{rest} *(for negligible* ϵ_{small}*); (3)* Π *is* in*secure against* secTA *attacks using* Θ*. In other words, restricted sender indistinguishability does not imply unrestricted security against traffic analysis.*

Proof. The counterexample for this separation is the same as the one used for Lemma C.2, where the unrestricted Θ oracle also outputs the secret key *sk*. This breaks the security of the underlying game Game as well. □

Lemma C.12 (Restricted indSenderTA $\not\Leftarrow$ secTA**)** *There exists a primitive* Π*, a restricted* Θ_{rest} *oracle, and an unrestricted* Θ *oracle with* $\mathsf{proj}(\Theta) = \Theta_{rest}$ *such that the following holds: (1)* Π *is* (t, Q, ϵ)*-secure with respect to a canonical security game* Game *(for negligible* ϵ*); (2)* Π *is* $(t, Q, Q_{\mathsf{TA}}, \epsilon_{small})$*-secure against* secTA *attacks using* Θ *(for negligible* ϵ_{small}*); (3)* Π *is* in*secure against restricted* secTA *attacks using* Θ_{rest}*. In other words, unrestricted security against traffic analysis does not imply restricted sender indistinguishability.*

Proof. By taking $\Theta = \Theta_{rest}$, we reduce this proof to the proof of Lemma C.8. □

D Case Study II: RFID Authentication

A fast-developing technology used in manufacturing, logistics, transportation, and even personal identification (such as passports or IDs) is ***R***adio ***F***requency ***ID***entification (RFID). RFID systems consist of low-cost *tags*, which authenticate to readers by using RF signals (thus requiring no line of sight). The main security interest for RFID authentication is privacy, i.e. no adversary can trace an RFID tag back to its owner. Several privacy models exist for RFID authentication. Amongst these, the indistinguishability based notion due to Juels and Weiss [26] is presented (in canonical form) in Appendix B. In short, the adversary uses four oracles InitR, $\mathsf{Send}\mathcal{T}$,$\mathsf{Send}\mathcal{R}$, $\mathsf{Corrupt}$, and its goal is to distinguish between two tags which have not been corrupted.

An enhanced RFID privacy model due to Vaudenay [43] allows arbitrary corruptions; one of their security notions is so-called forward privacy, where past sessions of a tag are unlinkable after corruption. Vaudenay's model compares the adversary's success to that of a *blinded* adversary, i.e. an adversary which cannot corrupt tags. In this sense, this privacy model is simulation-based.

In practice, forward privacy requires updating keys (e.g. through a Pseudo-Random Function – PRF). We outline the following forward private protocol from [43].[8] The protocol works as shown in Figure 3, for two PRFs PRF_1 and PRF_2.

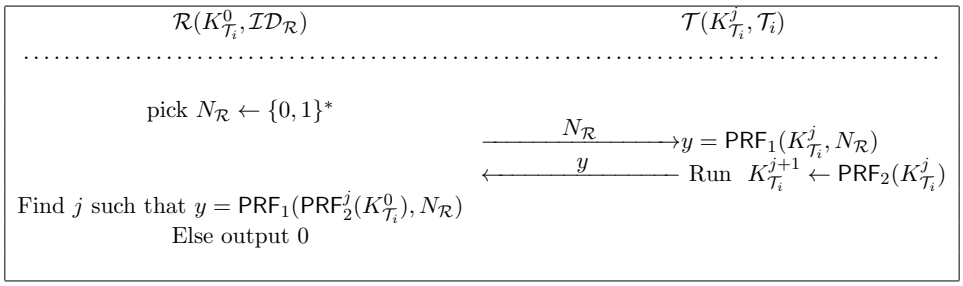

Figure 3: A forward-private protocol.

[8] This protocol actually achieves Vaudenay's destructive privacy notion, which is stronger than the model due to Juels and Weiss.

This protocol is not very efficient, as the reader must always search its entire database to find the correct value of the key and of the index j. The latter search is due to the fact that the reader retains only the original key for each tag, whereas the tag itself retains only the key's current value. This prevents desynchronization attacks, where an adversary tries to bring the reader and the tag out of sync with respect to the key. By doing so, the adversary also trivially distinguishes between tags.

We briefly introduce some terminology for the assessment of RFID privacy and security. We speak of reader impersonation when the adversary takes the part of a reader in its interaction with a tag, and tag impersonation when the adversary plays the part of a reader. The adversary can also play a relaying adversary, forwarding messages between the two parties: we say that the adversary then observes a legitimate protocol run.

Note that the run-time of the protocol in Figure 3 may leak information to an adversary regarding how many times the tag updates its key. Indeed, if the integer j for which y verifies is found quickly, then the adversary knows j is small. Let Θ be an unrestricted TA oracle, which, on input an instantiation of the privacy game of Appendix B, returns a partial transcript ϑ of all the values j which the reader has successfully identified up until the query is made.

We now show a simple attack to break the secTA of this primitive for oracle Θ. The adversary $\mathcal{A}_{small}$ creates only two tags $\mathcal{T}_0$ and $\mathcal{T}_1$ in the reader's database. It then runs a reader-impersonation on $\mathcal{T}_0$ (i.e. the adversary prompts the tag with a fresh random nonce making the tag respond *and* update its key). Then $\mathcal{A}_{small}$ chooses challenge tags $\mathcal{T}_0$ and $\mathcal{T}_1$ and forwards them to the challenger. Upon receiving the challenge tag $\mathcal{T}_b$, the reader runs an observation round on $\mathcal{T}_b$ and then runs Θ on this instantiation to receive the integer $j \in \{1, 2\}$, outputting guess $\Gamma^* = 0$ if $j = 2$ and $\Gamma^* = 1$ otherwise (during the observation run, the challenge tag has updated its key).

For the analysis note that $\mathcal{T}_0$ runs key $K^1_{\mathcal{T}_0}$ (because of the reader impersonation run) and $\mathcal{T}_1$ runs key $K^0_{\mathcal{T}_1}$, therefore the adversary guesses the correct tag and breaks the privacy of this primitive.

A naïve solution is to let the tag update a random number of times at every successful authentication. However, this enables the adversary to mount a statistical attack, considering the expected values of the number of updates. In order to render traffic analysis oracle useless, the reader should always search for an equal amount of time in its database. At the same time, tags must use different keys, thus ensuring forward privacy. Thus, a solution would be equipping each tag $\mathcal{T}_i$ with a number of distinct keys, or pseudonyms.[9] In its basic-most form, the solution has each tag equipped with a number k of keys, $K^1_{\mathcal{T}_i}, \ldots, K^k_{\mathcal{T}_i}$, also stored by the reader. After each authentication attempt, a key is discarded and the next key is used. We depict this in Figure 4.

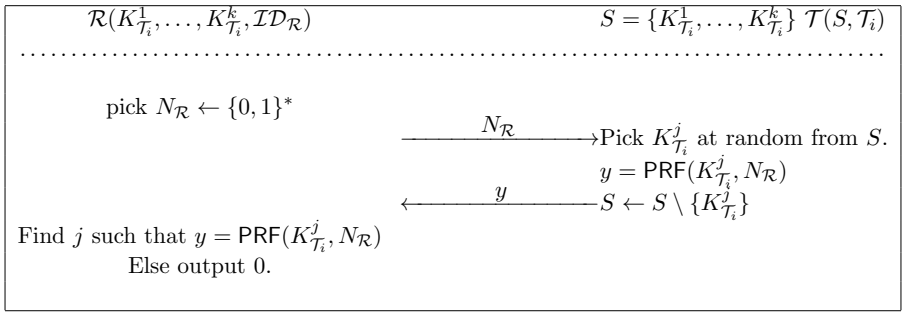

Figure 4: Pseudonym-based forward-private protocol.

Clearly, this protocol is both private in the sense of the definition in Appendix B and secTA-secure for oracle Θ. Indeed, we can prove the following:

Proposition D.1 (Privacy of the modified RFID protocol) *If there exists an adversary $\mathcal{A}_{small}$ that $(t_{small}, Q, \epsilon_{small})$-breaks the privacy of the scheme in Figure 4, then there exists a (t, ϵ)-distinguisher $\mathcal{B}_{small}$ against* PRF*,*

[9] This idea is extensively investigated in the literature, namely in [40]. Sadeghi et al. introduce the concept of an anonymizer – a third party in an RFID system, which creates new pseudonyms for tags such that they can authenticate with a fresh key every time.

with $t \approx t_{small}$ *and* $\epsilon = \epsilon_{small}$*. Furthermore, if there exists an adversary* $\mathcal{A}_{small}$ *that* $(t_{small}, Q, Q_{\mathsf{TA}}, \epsilon_{small})$*-breaks the* secTA *security of the scheme in Figure 4 for* Θ *as defined above, then there exists a* (t, ϵ)*-distinguisher* $\mathcal{B}_{small}$ *against* PRF*, with* $t \approx t_{small}$ *and* $\epsilon = \epsilon_{small}$*. Finally, if there exists an adversary* $\mathcal{A}_{big}$ *that* $(t_{big}, Q, Q_{\mathsf{TA}}, \epsilon_{big})$*-breaks the unrestricted* indSenderTA*, resp.* indReceiverTA *security of the scheme in Figure 4 for* Θ *as defined above, then there exists a* (t, ϵ)*-distinguisher* $\mathcal{B}_{small}$ *against* PRF*, with* $t \approx t_{big}$ *and* $\epsilon = \epsilon_{big}$*.*

Proof. We sketch the proof here. Note first that the proof for secTA follows from the proof for privacy, as the oracle Θ yields no useful information (the key is no longer updated and Θ always returns 0). Also, instantiations are as indistinguishable as the tags in a single instantiation, as the Θ oracle does not return any further information on the instantiations.

Therefore we only have to prove the security of the scheme against the underlying privacy game. The main steps of the proof go as follows: given an adversary $\mathcal{A}_{small}$ against the RFID authentication scheme in Figure 4, we build a distinguisher $\mathcal{B}_{small}$ against PRF. Adversary $\mathcal{B}_{small}$ receives outputs y_i from a challenger $\mathcal{G}$, such that y_i is either output by PRF or truly random. The distinguisher $\mathcal{B}_{small}$ returns random values to $\mathcal{A}_{small}$ in reader-adversary sessions (i.e. when prompted by $\mathcal{A}_{small}$ to start a new communication round), and it also returns random values for the corruption oracle. The response for $\mathsf{Send}\mathcal{T}$ queries are outputs y_i from $\mathcal{G}$. This is a perfect simulation for $\mathcal{A}_{small}$, which eventually returns two tags $\mathcal{T}_0$ and $\mathcal{T}_1$. The adversary picks a bit b and returns responses y_i for tag $\mathcal{T}_b$ and for tag $\mathcal{T}_{1-b}$. If the adversary correctly distinguishes between the tags, then $\mathcal{B}_{small}$ guesses that the output comes from PRF. Indeed, the keys used by the PRF are indistinguishable, hence this is a perfect simulation for $\mathcal{A}_{small}$. If $\mathcal{A}_{small}$ wins with advantage ϵ_{small}, then $\mathcal{A}_{small}$ wins with the same probability. □