8-13-2015

New Cryptographic Mechanisms for Enforcing Accountability

Qiang Tang
University of Connecticut - Storrs, qtang84@gmail.com

New Cryptographic Mechanisms for Enforcing Accountability

Qiang Tang, Ph.D.

University of Connecticut, 2015

Accountability ensures the proper use and distribution of digital data, including keys and programs. Without it, the digital world would become lack of order and in turn it will hinder the development and deployment of information technology itself. In this dissertation, we considered accountability for three different entities:

1. We systematically studied proactive deterring mechanisms for user accountability. Specifically, we study how to enforce the user to follow the key management policy in the single user setting of traditional public key infrastructure (PKI), and the multi-user setting of digital rights management. The crux of the deterring mechanisms is that a piece of user secret information is embedded into the public key/parameter. If the key owner makes a working device, e.g., a decryption device, and leaks it, then any recipient of the device can recover the secret information. This de-incentivizes illegal re-distribution and significantly advances the existing mindsets of detect-and-punish paradigm, which becomes ineffective when facing a private illegal redistribution.

2. We initiated the study of cliptography: preserving security of cryptographic primitives when *all* algorithms may be subverted by the adversary. This is regarding implementation provider accountability. Cliptography is possible since

the adversary also tries to avoid the detection of misbehavior. As a first step, we focused on dealing with key/public parameter generation algorithm, and provide the first secure one way functions, signature, and PRG in the aforementioned complete subversion model.

3. We revisited the current techniques for enforcing service provider accountability which aims at providing *undeniable* proofs when a malicious behavior is detected.

   Given that a fingerprinting scheme is usually used for copyrighting large files like a movie, we suggest to study and construct the first asymmetric fingerprinting scheme with an optimal communication rate.

   We for the first time show a generic transformation that converts any identity based encryption (IBE) to be accountable, and the ciphertext size only doubles that of the underlying IBE. Furthermore, our generic constructions can be extended without losing efficiency, to provide several properties like allowing identity re-use that are not known whether achievable before.

New Cryptographic Mechanisms for Enforcing Accountability

Qiang Tang

M.E., Graduate University, Chinese Academy of Sciences, 2009

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

at the

University of Connecticut

2015

APPROVAL PAGE

Doctor of Philosophy Dissertation

New Cryptographic Mechanisms for Enforcing Accountability

Presented by

Qiang Tang, M.E.

Co-Major Advisor _____
Aggelos Kiayias

Co-Major Advisor _____
Alexander Russell

Associate Advisor _____
Donald Sheehy

Associate Advisor _____
Marten Van Dijk

University of Connecticut

2015

ACKNOWLEDGEMENTS

My life as a Ph.D. student could never have been such a rewarding journey were it not for the support and guidance from my advisors, Aggelos Kiayias and Alexander Russell. I would like to first express my deep gratitudes to them. Aggelos and Alex showed me the best examples themselves that great scientists *can* at the same time be extremely decent and charming persons. Aggelos has a sharp intuition and is full of amazing ideas, every time I felt frustrated after got stuck on some problems, he can always magically bring me back to track and proceed forward. More importantly, he showed me a bigger world, and broadened my view on many aspects of life. There are so many good merits of him that I can always learn something fascinating. I remembered the enlightening discussions with him in the Athens metro, the Beijing subway, the Berlin underground, and the London tube. I've worked with Alex for not long, but his extreme talent and charisma gave me the impression that he is impeccable (even this vocabulary is taught by him) in many aspects. He can always pull out the hidden meaning from my vague expression, and explain it in an intuitive and a crystal clear way. The discussions with him are always illumining and our collaboration in my last semester is very efficient. And his manners give the best lesson about how to be a gentleman. Thank you Aggelos and Alex, you set up the best role models for me.

My sincere appreciation goes to Thomas Ristenpart, Donald Sheehy, and Marten Van Dijik for serving on my thesis committee and for many valuable comments. Specifically, I would like to thank Tom for taking me to the University of Wisconsin, Madison, where I spent a lovely summer and worked with him on the interesting topic of honey encryption.

Tom is vey smart, energetic and with great patience for tolerating me for all kinds of excuses delaying the polish of our paper during the graduation season.

Special thanks to Tatsuaki Okamoto for arranging my visit to NTT research, Tokyo, for a summer even if I requested very late. I had many inspiring discussions with him and also with several other wonderful researchers, in particular, Masayuki Abe, and Eiichiro Fujisaki. Also I thank all the NTT colleagues to make my stay very memorable.

I would like also to thank all my co-authors: Aggelos Kiayias, Nikos Leonardos, Helger Lipmaa, Ozgur Oksuz, Murat Osmanoglu, Kateryna Pavlyk, Thomas Ristenpart, Alexander Russell, Moti Yung and Hong-Sheng Zhou. Collaborating with them not only results in several interesting papers, but also teaches me a lot along the way.

Ph.D. time would be lonely and sometimes frustrating if there were no great friends. I am fortunate to have many of them: Sixia Chen, Justin Neumann, Ozgur Oksuz, Murat Osmanoglu, Lei Wan, Gang Wang, and Zuofei Wang. They helped me a lot during my stay in UConn. Particularly, I traveled many times with Murat to different places, and we had great memories about our conversations about movies, philosophy and almost everything sounds interesting. I wish the best luck of him when he goes back to Turkey, and I hope we can find chances for another road trips. Nikos Karvelas, Nikos Leonardos, Katerina Samari, Yannis Tselekounis, Thomas Zacharias, Bing-Sheng Zhang, thank you for the helps and fun times during my stay in Athens. I could only list a small number of names here, but I am grateful to all of the friends.

Last, but the foremost, my heartfelt gratitude to my dearest family, in particular, my grandma Yan-Sheng Guo, and my truly beloved wife, Lu Li, for their unconditional support and love, and the care and joy they bring to me. Were not because of them, it would have no meaning even if I may be lucky to achieve anything.

TABLE OF CONTENTS

Chapter 1

Introduction

1.1 Background

With the fast growing of information technology, digital data is produced and transferred in an unprecedented scale and speed. This revolutionizes many aspects of human creativity, but also, arouses serious concerns about data security, personal privacy, communication authenticity and many others. Accountability, among those properties, is one of the most important features that ensures the proper use and distribution of digital data, keys and programs.

Cryptography provides protection mechanisms for the information that people prefer to hide which may otherwise easily leaked in the Internet, e.g., private personal information, or even their identities. However it is a double-edged sword in this digital world. These powerful tools also provide convenient, even new ways for dishonest parties to misbehave and even to participate in illegal acts.

Accountability is indispensable, unfortunately, it does not come along automatically. Sometimes, people even have incentives to deviate from the commonly acknowledged principles that they are supposed to follow. For example, distributing pirate copies of digital goods will bring financial gains for the illegal distributor with essentially no cost. Even worse, in many cases, the dishonest parties are "insiders", they do not have to break the security of the cryptographic primitives Thus, we have to design new technical mechanisms to enforce them.

Since accountability aims at ensuring people to produce/use data, key or program according to certain policies, depending on the form of the enforcement, we may have three types of mechanisms: (i.) detect-then-punish, (ii.) proactively deter; and (iii.) even sometimes, prevent the malicious behavior at the beginning.

1.2 Contributions of This Dissertation

In this thesis, we will discuss the cryptographic tools developed by us for enforcing accountability in three different kinds of entities in different application scenarios, regarding all the three enforcing mechanisms mentioned above. We elaborate below.

User accountability. Encryption schemes are one of the most widely used cryptographic primitives. However, the security relies on the fact that the secret key is private. This requires a carefully designed key management policy. It follows that an equally important question is how to enforce the policy to be obeyed. A common requirement in a key management policy, (e.g., for a company public key infrastructure, or in a digital right management system), is that the key owner should not distribute the secret key to any third party. However, there is no built-in incentives for the key owners to follow; even worse, there may be motivation for them to leak (or "sell") a decryption device

(called a pirate box). In this case, we consider the key owners to be the adversary (while in classic security notions in cryptography, the secret key is not known to the adversary). It follows that it is very difficult to prevent the adversaries from misbehaving, (unless we introduce some extra hardware assumption which we don't use in this dissertation). Existing mindsets focus on designing techniques to detect the illegal re-distribution and then some actions maybe taken by the authority (e.g., the manager or the service provider) as punishment. One well-known cryptographic primitive aiming at enabling the detection of illegal content re-distribution is traitor tracing scheme (TTS) [36].

However, in order to make this detect-then-punish mechanism work, the authority has to have access to the pirate box. A foxy adversary can make a decryption device, (e.g., a program), and simply uploads it to, say, a private hacker forum, the access of which is granted to the members exclusively. The authority can not have access to this private form, even may not be aware of the existence of it. Since the members of the forum can download and use the program, they have no incentive to report to the authority about such a pirate box. This demonstrates the ineffectiveness of traditional tools such as a TTS, when dealing with private re-distribution.

We initiate the systematic study of proactive deterring mechanisms for enforcing key management policy. In particular, we introduced two new frameworks of *leakage-deterring* public key cryptography (LD-PKC), and *traitor deterring schemes* (TDS).

- LD-PKC: We first consider the accountability problem in the traditional public key infrastructure (PKI) of enforcing a key management policy that every key owner should keep the key to himself. To achieve this, we augment the PKI with a new property of leakage-deterrence. During the public key certification procedure, a piece of secret information s from the key owner will be securely embedded into her public key (or the certificate.) The system is also coupled with another

recovering procedure that given access to any partially working cryptographic functionality (e.g.,decryption) of the key owner, anyone can retrieve s. Having the new feature, when a key owner plans to leak her key or a device (say, decryption) in any form (even privately to a third party), she will bear with the potential cost/punishment that her secret s will be revealed. In this way, the key owner is deterred. Adding leakage deterrence turns out to be highly non-trivial. We did a systematic study for all the major asymmetric cryptographic primitives including public key encryption, signature schemes and identification schemes.

- TDS: Similarly, a TDS augments the classical TTS in the multi-recipient encryption system, with the new embedding and recovering mechanism we introduce. In a multi-recipient encryption system, there are a set of users, and each of them have a key that enables the decryption of ciphertext generated with the system encryption key. A TDS suggests that during the system initialization, a piece of secret information for each user is embedded into the public parameter. If a subset of the users collude to provide a decryption device, one may recover the secret of one of the colluders. While, in a TTS, only an identity of the colluders can be retrieved (even this requires a special property called public traceability). We gave two constructions for TDS with black-box recovery, and apply it to resolve a major open problem in digital signet with self-enforcement [51]. Along the way we introduce new cryptographic primitives of fuzzy locker and comparison predicate encryption with exponentially large domain, which are of their own interests.

- Bitcoin as a trusted collateral: Having the new mechanisms at hand, we identify an extremely suitable information which can be used as each user's collateral embedded in the public repository — Bitcoin. Bitcoin has a public ledger that

records every transaction ever happened. But only with a secret key corresponding to an account, any outgoing transaction (from that account) can be generated. Every user will be associated with one bitcoin account which is required to be frozen until the user cancels the service. The secret key of each account will be embedded into the public parameter. Our recovering mechanism enables the recipient of the pirate box to retrieve a secret key and then transfer the money out. The authority will also notice the transaction and further stop the service.

Implementation provider accountability. Consider conventional use of programs, including cryptographic primitives, a regular user simply runs the software or hardware obtained from some providers.

Kleptography, originally introduced by Young and Yung [130], studies how to steal information securely and subliminally from cryptosystems. The basic framework considers the (in)security of malicious implementations of a standard cryptographic primitive by embedding a "backdoor" into the system. Remarkably, crippling subliminal theft is possible even if the subverted cryptosystem produces output indistinguishable from a secure "reference implementation." After a long hiatus, interest in such issues was rekindled by the dramatic revelations of Edward Snowden, demonstrating that such deliberate attacks have been deployed and presumably used for massive surveillance. Notably, Bellare, Paterson, and Rogaway [11] initiated a formal study of attacks on symmetric key encryption algorithms.

Motivated by the original examples of subverting key generation algorithms in the kleptography papers from Young and Yung [130, 131], we initiate the study of cryptography in the setting where *all* algorithms are subject to kleptographic attacks—we call this **cliptography**. As a first step, we formally study the fundamental primitives of one-way function and trapdoor one-way function in this "complete subversion" model.

We describe a general, rigorous immunization strategy to clip the power of kleptographic subversions; concretely, we propose a general framework for sanitizing (trapdoor) one-way function index generation algorithms by hashing the function index, and prove that such a procedure indeed destroys the connection between a subverted function generation procedure and any possible backdoor. Along the way, we propose a split program model for practical deployment.

We then examine two standard applications of (trapdoor) one way functions in this complete subversion model. First, we consider construction of "higher level" primitives via black-box reductions. In particular, we show how to use our trapdoor one-way function to defend against key generation sabotage, and showcase a digital signature scheme that preserves existential unforgeability when *all* algorithms (including key generation, which was not considered to be under attack before) are subject to kleptographic attacks. Additionally, we demonstrate that the classic Blum–Micali pseudorandom generator (PRG), using our "unforgeable" one-way function, yields a backdoor-free PRG. Second, we generalize our immunizing technique to one way functions, and propose a new public immunization strategy to randomize the public parameters of a (backdoored) PRG. This notably contrasts with previous results of Dodis, Ganesh, Golovnev, Juels, and Ristenpart [47], which require an honestly generated random key.

Thus, we develop fundamental cryptographic primitives with meaningful security guarantees in a quite adversarial setting, where one cannot rely on private randomness and all associated algorithms, including key and index generation, are under attack.

Putting into the context of accountability, clipotography aims at enforcing honest implementation. In particular, in the case that the adversary ("big brother") wants also avoid the detection of misbehavior and thus her power is restricted to some extend. We

will essentially explore this opportunity to amplify the restriction so that the security can be preserved.

Service provider accountability. In most of the systems, there is service provider (SP) who sets up the system, thus the user secret key or the secret information is known to the SP. One nature question is do we have to trust the SP completely when considering the user accountability problem. In practice, one may think the service provider cares about its reputation and will well behave. However, there are potential system breaches and even insider attacks. More importantly, when a malicious user behavior is detected, if a third party (e.g., a judge) has to be involved, the user can simply emphasize the possible misbehavior of the SP. Thus excluding the capability of the SP to frame users can guarantee that one can form an *undeniable* proof when misbehaviors are detected. This will improve the applicability of the concept of detect-then-punish.

We considered two different scenarios for restricting SP. Although the concepts for SP accountability exists in both settings, there are different kinds of restrictions that hinder their wide use in practice. In this thesis, we significantly improve the state-of-the-art, and push them closer to be applicable in practice.

- Rate optimal asymmetric fingerprinting: The first setting is digital rights management (DRM) , e.g., a pay-TV service. Similar to the classical notion of TTS, a fingerprinting scheme enables the identification of a pirate box/data. However, the whole system is initialized by the SP, and all the user secret codewords are generated (thus known to) the SP. The concept of asymmetric fingerprinting suggests that via a secure two party protocol, the SP only learns half of the codeword of each user, and some references (e.g., commitments) of the other half of the codeword. The half code known to the SP can still be used to trace, and the SP can file a accusation. But now the SP has no knowledge of the other half code,

thus a judge can make a final decision to check whether the other half codeword also implicates the user.

However, security comes with a price of less efficiency. In particular, we examine the communication efficiency, which is one of the most critical considerations when transmitting big data, e.g., high resolution movies in the DRM setting. We suggests to systematically study the communication rate of an asymmetric fingerprinting protocol, i.e., the data needed over the actual communication should approach one very fast when the size of the data grows. We constructed the first such rate optimal asymmetric fingerprinting scheme, and along the way we formally analyze several overlooked security concerns.

- Accountable Identity Based Encryption: Identity based encryption (IBE) provides a compelling solution to replace PKI, and the flexibility of using any identity as a public key leads to many breakthroughs in cryptography. However an inherent weakness of the IBE system is the decryption key has to be generated by a trusted party, say a service provider. Accountable authority IBE (A-IBE) aims at easing this conflict between the privacy and the functionality, in a way that during key generation, the user randomly selects one key from an exponentially many keys generated by the authority. If the authority leaks a decryption device of the user, the key selected by the user and this extra key would be different, and they together forms an evidence that the authority misbehaves.

  Although A-IBE significantly improve the accountability of the authority, however, it is still not practical for a couple of reasons. (i.) existing constructions are either inefficient, or rely on special structure and not compatible with the standardized

designs, e.g., [26]. (ii.) when a user accidentally lost his secret key, the identity has to be abandoned and a new identity should be used for issuing a new key.

We proposed a *generic* way of compiling an IBE into an A-IBE, such that one can upgrade the efficient deployments like the Boneh-Franklin IBE [19]. More importantly, this generic transformation only doubles the ciphertext size (3 more group elements if using [19]). Furthermore, we improve the generic construction with public traceability and allow identity re-use, which drastically advance the usability of A-IBE schemes.

1.3 Organizations

We arrange the presentation as the order of proactive deterring mechanisms for user accountability; implementor accountability; and service provider accountability. In more detail, we will present leakage-resilient public key cryptography in chapter 2; and we then extend the notion to the multi-recipient encryption scenarios and present traitor deterring schemes in chapter 3. Then we shift gears to introduce and explain our initial works of cliptography in chapter 4. Furthermore, to improve the applicability of current solutions for SP accountability, we present the rate optimal asymmetric fingerprinting problem in chapter 5, and discuss our improvement of efficient generic transformation of A-IBE in chapter 6. And finally, we conclude and explain future works in chapter 7.

Chapter 2

Leakage Deterring Public Key Cryptography

2.1 Introduction

Consider any organization that maintains a PKI supporting various cryptographic functions including public-key encryption, signatures and identification. How is it possible to prevent individuals from sharing their cryptographic functions? Certified PKI members, out of convenience or even malice, can delegate their private keys to each other (or even to outsiders), thus violating accountability and organizational policy. Even worse, delegation can be partial: for instance, a public-key encryption user can share (or, in fact, even sell) an implementation that only decrypts messages of a certain form (e.g., only e-mails from a specific source). Seemingly, very little can be done to prevent this as the adversary in this case is the owner of the cryptographic key and hence she may freely choose to share it with others either directly or by incorporating its functionality within a larger system that is shared.

The above scenario puts forth the central problem of our work: how is it possible to prevent the sharing of cryptographic functions? The main challenge here is that the owner of the key is adversarial: she wishes to share a program or hardware device that (potentially only partly) implements her main cryptographic functionality. Given that she possesses the key (either in software or hardware), it is impossible for her to be prevented from delegating it. However, as we highlight, there can be ways for her to be *deterred* from doing so. A straightforward deterrence mechanism would be to identify and penalize the sharing behavior. However, the enforcement of a penalty mechanism is contingent to detecting the act of sharing — something that limits the effectiveness of penalties: a cautious adversary can keep itself "below the radar" and thus remain penalty-free. To address this we put forth and explore a more proactive approach.

A cryptographic scheme will be called *leakage-deterring* if the release of any implementation of the cryptographic function (e.g, decryption, signing), leads to the recovery of some private information (that the owner prefers to keep hidden) by anyone that possesses the implementation. Leakage deterrence is thus achieved in the sense that sharing the cryptographic function in any form incurs the penalty of revealing the private information (while non-sharing maintains its privacy).

Note that a leakage-deterring primitive should retain its original functionality (e.g., encryption, signing, identification) but it also offers two additional operations: first, it is possible to embed private data into the public-key of the primitive in a way that they are (at least) semantically secure – we call this property *privacy*. The embedding operation is facilitated through an interaction with an authority that vouches for the integrity of the private data and is akin to a PKI certification of the owner's public-key. In this fashion, the primitive's public-key becomes "enhanced" and is a carrier of private information itself (i.e., a ciphertext) — otherwise the intended functionality of the primitive should

remain unchanged. The second operation that is offered by a leakage-deterring primitive comes into play when the owner of the secret key produces an implementation of the main operation in the form of a "box" and shares it with other entities (in software or hardware). Given such a box, any entity that receives it can utilize a public recovering algorithm that will interact with the box and produce the private data that are embedded into the owner's enhanced public-key – we call this property *recoverability*.

In a nutshell, designing a leakage-deterring scheme requires the transformation of the public-key of the primitive into a (one-time) ciphertext that can be decrypted by *any* working implementation of the cryptographic functionality. The main challenge comes precisely from this latter requirement: any working implementation of the cryptographic functionality should be usable as a decryption key that unlocks the private data embedded into the public-key, even if the adversarial implementor takes into account the recoverability algorithm and the enhanced public-key that carries the private data when implementing the functionality.

To appreciate the complexity of the problem, consider a naive attempt to produce a leakage-deterring public-key encryption (PKE): the authority certifies as the enhanced public key the pair (pk, ψ) where $\psi = \mathbf{Enc}(pk, s)$ and s is the private data related to the owner. Recoverability can be attempted by feeding ψ into a decryption box. It is apparent that this construction can be defeated by an adversarial implementation of decryption that given input c, it decrypts it only in the case $c \neq \psi$ (or even $\mathbf{Dec}(c) \neq s$). The constructions we seek should facilitate recoverability even if the adversary releases implementations that work for *arbitrary* input distributions of her choice.

The applications of leakage-deterring cryptographic primitives are in any context where the intentional leakage of a cryptographic functionality should be deterred or restricted in some fashion or in a context where the leakage of an implementation should

enable the computation of a value that is otherwise hidden. In the most simple scenario, the enhanced public-key contains some piece of information that the owner prefers to keep secret (e.g., her credit-card number or similar piece of private information as suggested by Dwork, Lotspiech and Naor [51] that introduced the concept of *self-enforcement* – in a related but different context – see below). It follows that the system setup "self-enforces" the owner to keep the cryptographic functionality to herself. Depending on the deployment environment, different types of secret-information can be used. We describe more application scenarios of leakage deterring cryptographic primitives in section 2.7.

Our contributions. We introduce, formalize and implement leakage-deterring cryptographic primitives for public-key encryption, digital signatures, and identification schemes. The main technical contributions we provide are three different techniques for constructing leakage-deterring cryptographic primitives. Our techniques enable the secure embedding of private information into the public key of the primitive in a way that is recoverable given any (even partially) working implementation. Our first method, applies to encryption that is partially homomorphic; given a box that works only for some adversarially chosen distributions we show how to exploit the homomorphic property to appropriately manipulate a target ciphertext and make it decryptable by the adversarial decryption box. Our second method, which can rely on any encryption scheme, hides the key that unlocks the private information into an exponentially large key space that is encoded in the public-keys. By using appropriate redundancy in the public key space we enable the tracing of the vector of keys that identify the private information, out of any (even partially working) implementation. Achieving recoverability while maintaining small ciphertext size in this setting requires an involved recoverability algorithm which is one of the highlights of our contributions. Finally, our third method applies to signature and identification schemes. It uses the fact that working implementations of suitably

chosen such primitives can be used to build "knowledge extractors." These are algorithms that reveal information about the secret-key of the underlying primitives which we use to hide the private information. We note that the idea of using extractors for preventing the transfer of identification tokens has been used before [28, 30, 78, 98] in the sense that sharing a token implies sharing the key. Going beyond this we show here that secret key can be of sufficient entropy so that it simultaneously hides the embedded owner information while still maintaining the security of the underlying scheme. In fact we show that no additional intractability assumptions are necessary for achieving leakage-deterring signature and identification schemes.

Our first construction for PKE requires a standard homomorphic property and achieves constant size ciphertexts while offering recoverability for any (non-trivial) adversarial distribution. The second construction is generic and the size of ciphertexts is a parameter that increases as the min-entropy of the allowed adversarial distributions becomes smaller. We analyze our constructions in the IND-CPA setting and then present a generic transformation to obtain IND-CCA2 security[1]. It is evident that there is a trade-off between privacy and recoverability. For encryption schemes, we aim at maximizing the recoverability while privacy can only be achieved if no decryption query is allowed. For the case of signatures, we present a construction that maintains the privacy

[1]It may come as a surprise that recoverability and IND-CCA2 can actually coexist. Attaining IND-CCA2 intuitively means that a decryption oracle basically leaks no useful information about manipulated ciphertexts. Thus, the recovering algorithm can seemingly do nothing useful with access to a decryption implementation beyond decrypting valid ciphertexts, which if related to the enhanced public-key can be rejected. Still, the paradox can be resolved, if one observes that the decryption oracle should be useless only with respect to breaking the security of regular ciphertexts and not the security of the data that are somehow embedded into the enhanced public-key.

of the embedded information even if the adversary has access to the signing functionality (which is most desirable since digital signatures are typically publicly available). We still manage to enable recoverability by exploiting the random oracle model and non-black-box access to the implementation. Security properties of our identification schemes are shown in the standard model. To attain privacy in the standard model we utilize strong extractors for random variables with high conditional unpredictability.

Related work. The most relevant work to ours is [51] that introduced self-enforcement as a way of avoiding illegal content redistribution in a multi-user setting. Self-enforcement was argued there by ensuring (under nonfalsifiable assumptions) that an owner has only two options when implementing a decoder: either using her private key (that includes private personal information), or encoding a derived key that is of size proportional to the data to be decrypted. In our terminology, this means that the schemes of [51] exhibit a leakage-deterrence/program-length tradeoff and hence are not leakage-deterring *per se*. Furthermore, recoverability in [51] is only "white-box" as opposed to the black-box type that our constructions achieve. In another related line of work [28, 30, 78, 98] it was discussed how to deter a user from transferring her credentials (or the secret key directly) to others in the context of identification systems. The techniques from these works – by nature – were restricted to only identification schemes and digital signatures. In contrast, our work encompasses all major public key cryptographic primitives (including PKE). The primitive of circular encryption [28] might look promising at first sight to achieve leakage-deterring PKE, however, no recovery mechanism which works for all partial implementations is provided by this primitive. To resolve the main difficulty we explained in a previous paragraph, new techniques other than circular encryption are needed. Furthermore, for the case of identification and signature schemes, our method shows that leakage-deterrence can be achieved without any assumptions beyond the one employed

by the underlying primitive. Other forms of leakage deterring techniques were considered in various settings, e.g., limited delegation [62], data collection [64], e-payments [120] or designated verifier signatures in [95, 122] in the form of *non-delegatability* (which is a weaker notion than our leakage-deterring concept).

Another related notion, introduced in [101], dealt with the problem *copyrighting* a public-key decryption function: a single public-key decryption functionality should be implemented in many distinct ways so that if an implementation is derived from some of them, then it is possible to discover the index of at least one of the implementations that was used. This notion was further investigated in [87] and was related to a TTS [37]. In the context of PKE, the objective of copyrighting a function or of a TTS is orthogonal to ours. While in both cases we deal with adversarial implementations of cryptographic functionalities (hence the similarities in terminology), the adversarial goal is different: in the case of traitor tracing, the adversary has many different implementations of the same functionality and tries to produce a new one that is hard to trace back to the ones she is given. In an attack against a leakage-deterring scheme on the other hand, the adversary possesses an implementation of a cryptographic functionality and tries to modify it in a way that it cannot be used to extract some information that is hidden in the primitive's public-key. Combining the two functionalities in one is an interesting question and we will study it in next chapter (a step towards this general direction but in a much weaker model than ours was suggested in [89] but the leakage-deterring aspect (in our terminology) was found to be insecure in [88]).

Accountable authority identity based encryption (AIBE) [67, 68, 86, 93, 118] considers the problem of judging whether an implementation of decryption belongs to the owner or the PKG (in the context of IBE). In this setting, both the owner and the PKG may be the potential adversary who try to implicate the other. Hence, some property similar

to our recoverability is needed. In any case, the single bit decisional output required by AIBE is much weaker than our recoverability requirement in leakage-deterring PKE (even in the IBE setting) where by interacting with a decryption box, one should recover the whole private data embedded in the enhanced public-key.

Finally we should point out that the notion of leakage deterrence is different from the notion of leakage-resilience (see e.g., [52, 77]). Our notion aims at constructing schemes with the property that *intensional* leakage of the cryptographic functionality implies the revelation of some private owner information (hence they are "leakage-deterring"), while the leakage-resilience notion aims at ensuring that the unintentional leakage (as in the case of side channel attacks) provides no useful information to an adversary.

2.2 Preliminaries

First, we recall some known primitives and results which we utilize in our constructions or security analysis.

Proof of Knowledge: [6] A proof of knowledge (PoK) protocol is one that a prover convinces the verifier he knows a witness to a publicly known PPT predicate. This is a protocol between two parties P, V where P proves a statement $x \in L$ for a language L's instance x with its witness w from a witness set denoted by $W(x)$. Suppose $O_V[P(x,w) \leftrightarrow V(x)]$ denotes the output of the verifier V after interacting in the protocol with the prover P, a proof of knowledge protocol satisfies the following two properties:

- Completeness: Honest prover always convinces the verifier: if $w \in W(x)$, then $\Pr[O_V[(P(x,w) \leftrightarrow V(x)] = 1] = 1$.
- Soundness: There exist an expected PPT "knowledge extractor" $\mathbf{E}$ which interacts with a malicious prover P^*, and outputs a witness with overwhelming probability

as long as the success probability that P^* convinces V is non-negligible. Formally, for all x, w^*, whenever $\Pr[O_V[(P^*(x, w^*) \leftrightarrow V(x)] = 1]$ is non-negligible it holds that $\Pr[\mathbf{E}^{P^*}(x) \in W(x)]$ happens with overwhelming probability.

Σ-*Protocol:* [43] One frequently used type of PoK protocols is the class of Σ-protocols, which have a three move structure (a, e, z), starting with the prover sending a 'commit' message a, then the verifier sending a 'challenge' message e, and finally the prover answering with a 'response' message z. Using the Fiat-Shamir transformation [53], one can construct a signature scheme based on such protocol in the random oracle model [12]. Security of such signature schemes is comprehensively studied in [114], and it mainly relies on the existence of a knowledge extractor algorithm (which is implied by the soundness of the protocol).

General Forking Lemma: [9] The general forking lemma states that that if an adversary, on inputs drawn from some distribution, produces an output, then the adversary will produce another correlated output with different inputs from same distribution and same random tape. Rigorously, let $\mathcal{A}$ be a probabilistic algorithm, with inputs $(x, r_1, \ldots, r_q; \rho)$ that outputs a pair (J, σ), where ρ refers to the random tape of $\mathcal{A}$ (that is, the random coins $\mathcal{A}$ will make). Suppose further that x is sampled from some distribution X, and R is a super-polynomially large set and r_i is sampled uniformly from R. Let *acc* be the probability that $J \geq 1$. We can then define a "forking algorithm" as follows,

- on input x: pick a random tape ρ for $\mathcal{A}$.
- $r_1, \ldots, r_q \xleftarrow{\mathrm{r}} R$.
- $(J, \sigma) \leftarrow \mathcal{A}(x, r_1, \ldots, r_q; \rho)$
- If $J = 0$, return $(0, \epsilon, \epsilon)$.

- $r'_J, \ldots, r'_q \xleftarrow{r} R$
- $(J', \sigma') \leftarrow \mathcal{A}(x, r_1, \ldots, r_{J-1}, r'_J, \ldots, r'_q; \rho)$
- If $J' = J$ and $r_J \neq r'_J$ then return $(1, \sigma, \sigma')$, otherwise, return $(0, \epsilon, \epsilon)$.

Let *frk* be the probability that $\mathcal{A}$ outputs (b, σ, σ'), and $b = 1$, then $frk \geq acc(\frac{acc}{q} - \frac{1}{|R|})$.

Strong one-time signature [96] A signature scheme $\mathsf{Sig} = (\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$ is a strong one-time signature scheme if the success probability of any PPT adversary $\mathcal{A}$ in the following game is negligible:

- The challenger sends $\mathcal{A}$ a verification key vk.
- $\mathcal{A}$ asks one signing query on some message m chosen by her and the challenger returns a valid signature σ.
- $\mathcal{A}$ outputs a new signature (m^*, σ^*).

We say the adversary succeeds if: $\mathsf{Verify}(vk, (m^*, \sigma^*)) = 1 \wedge (m^*, \sigma^*) \neq (m, \sigma)$.

2.3 Definitions and Security Modeling

2.3.1 Definitions of leakage-deterring cryptographic primitives

A leakage-deterring cryptographic primitive includes two additional algorithms on top of the regular algorithms the primitive should normally possess: $\mathsf{EnKey}(\cdot)$, which embeds some (private) owner related information s into the public key, and $\mathsf{Rec}(\cdot)$ which recovers s from the public-key by interacting with any non-trivial implementation (or "box") and can be executed by anyone. While the concept of a leakage-deterring cryptographic primitive can be defined in abstract terms for a wide class of primitives we find it more instructive to present it for three main cryptographic primitives individually;

we focus on public-key encryption first; definitions of leakage-deterring signatures and identification are presented in the following part. With these three examples at hand, it is relatively straightforward to derive leakage-deterring definitions for other cryptographic primitives following the same general structure (see also remarks below).

Leakage-deterring Public Key Encryption:

- $\mathsf{KeyGen}(1^\lambda)$: On input security parameter λ, this algorithm returns a key pair (pk, sk).
- $\mathsf{EnKey}(O, A)$: This is a protocol between two parties O (owner) and A (authority), with inputs (pk, sk, s) and (pk, s) respectively that has the objective to embed the private owner's data s into his public-key; the protocol terminates with O obtaining an enhanced key pair (epk, esk) and A obtaining simply epk.
- $\mathsf{Enc}(epk, m)$: On input a message m, the user's enhanced public key epk, this algorithm returns a ciphertext c.
- $\mathsf{Dec}(esk, c)$: On input a ciphertext c and enhanced secret key esk, this algorithm returns m or fail $\perp$.
- $\mathsf{Rec}^{B,\mathcal{D}}(epk, \delta)$:[2] Using access to a decryption box B and a plaintext distribution $\mathcal{D}$ (which is supposedly the one that B is suited for and is correct with probability δ), as well as input the enhanced public key epk for a certain user, this algorithm outputs s or fail $\perp$.

[2]Having access to $\mathcal{D}$ is necessary; to see that, consider the following simple example: the box processes the input only if the message encrypted is of the form $sc||m$ for some secret string sc; otherwise it outputs $\perp$. It follows that without knowledge of $\mathcal{D}$, the box is useless. This counterexample applies to the other leakage-deterring primitives.

The definitions for other public-key primitives are similar and we present them below. They share the same basic structure in terms of the syntax of the recovering algorithm but there is some variability across primitives with respect to when this algorithm is supposed to operate. We tackle this question in the following section.

Leakage-deterring Signature Scheme: The definition of a leakage-deterring signature scheme is defined in a similar vein to the definition of leakage-deterring PKE. Specifically, algorithms $\mathsf{KeyGen}, \mathsf{EnKey}$ are identical. The rest are defined as follows.

- $\mathsf{Sign}(esk, m)$: On input a message m, the user's enhanced secret key esk this algorithm returns a signature σ.
- $\mathsf{Verify}(epk, m, \sigma)$: On input message-signature pair (m, σ), and enhanced public key epk, this algorithm returns 1 if valid, or 0 otherwise.
- $\mathsf{Rec}^{\mathcal{D}}(epk, B, \delta)$: Given a signing box B and a message distribution $\mathcal{D}$, and on input an enhanced public key epk, for a certain user, this algorithm outputs the private string s belongs to this user or fail $\perp$.

Leakage-deterring Identification Scheme: The case of leakage-deterring identification schemes is similar to digital signatures with the differentiation that the Sign and Verify algorithms are substituted by a protocol between two parties, the prover P and the verifier V. Normally, the owner is assumed to be the prover but the owner, having access to the secret-key, can issue implementations of the prover algorithm in the identification protocol. Specifically we have the following. First KeyGen and EnKey are the same as in the previous case of leakage-deterring signature schemes, for the other two,

- $\mathsf{Identify}(P, V)$: this is a protocol between the prover P with inputs epk, esk and the verifier V on input epk that terminates with the verifier outputting 1 (accepting) or 0 (rejecting the identification). We denote the transcripts as $P(epk, esk) \leftrightarrow V(epk)$.

- $\mathsf{Rec}(epk, B, \delta)$: Given an implementation B of the prover P algorithm in the identification protocol, and input the enhanced public key epk, this algorithm outputs s or fails $\perp$.

Remark 1. *One can think of* EnKey *as an extension of a public-key certification operation by an authority. The owner may still utilize* (pk, sk) *for the primitive's operation (as in a PKI one may still use an uncertified key) but epk is the key designated for public use.*

Furthermore, we note that in the Rec *algorithm, one may distinguish several ways that the algorithm may have access to the main functionality box (which is assumed to be resettable, i.e., it does not maintain state from one query to the next). Specifically, beyond black-box access we will also consider a certain type of non-black-box access.*

2.3.2 Correctness and security modelling

In this section we introduce the main security requirements for leakage-deterring cryptographic primitives. In general any leakage-deterring primitive should offer *privacy* for the owner (as long as no implementation of the primitive is leaked) and *recoverability*, i.e., that the recovering algorithm will be able to produce the private data of the owner when it has access to a *non-trivial* implementation of the cryptographic primitive. Finally, it is important that the introduction of the additional functionality does not disturb the standard cryptographic properties of the primitive. We examine these properties below.

Definition 1. Privacy *(of Owner's Data): For an honest owner who does not leak any non-trivial box, the privacy of its data bound in the enhanced public key should be protected. To define the property formally we introduce the following game between a challenger* $\mathcal{C}$ *and an adversary* $\mathcal{A}$*.*

- *The challenger runs* $\mathsf{KeyGen}(\cdot)$ *and sends to the adversary* $\mathcal{A}$ *the public key pk.*

- *The adversary $\mathcal{A}$ chooses two private strings s_0, s_1 and sends them to the challenger.*
- *$\mathcal{C}$ chooses b and simulates $\mathsf{EnKey}(\cdot)$ on s_b and pk, sk; it sends epk to the adversary.*
- *$\mathcal{A}$ returns his guess b' about b.*

If there is no PPT *adversary $\mathcal{A}$ can correctly guess b with non-negligible advantage, i.e., for all PPT $\mathcal{A}$, $|\Pr[b' = b] - \frac{1}{2}| \leq \epsilon$ where ϵ is a negligible function, we say the leakage-deterring cryptographic scheme achieves privacy (in the sense of indistinguishability).*

Furthermore, in the above game, we may allow $\mathcal{A}$ to observe the cryptographic functionality on a certain input distribution. If the above definition holds even if the adversary has access to an oracle $\mathcal{O}(esk, \cdot)$ (that is dependent on the enhanced secret-key of the owner, e.g, decryption oracle or signing oracle w.r.t. some plaintext distribution $\mathcal{D}$) we will say that the scheme achieves privacy with respect to the secret-key oracle $\mathcal{O}(esk, \cdot)$. Note that with respect to privacy we consider both owner and authority honest. It is possible to extend the model to the case of a dishonest authority but this goes beyond the scope of our current exposition (and intended use cases).

Definition 2. Recoverability *(of Owner's Data): If a dishonest owner releases a functional box B, anyone having access to B should be able to recover the owner's private data from the enhanced public key epk. Formally, consider the following game between a challenger and an adversary $\mathcal{A}$:*

- *The adversary $\mathcal{A}$ on input 1^λ generates a key pair (sk, pk) and submits it together with the owner private data s to the challenger.*
- *The challenger acting as the authority runs EnKey with the adversary (playing the role of the owner) to produce the enhanced key pair (epk, esk).*

- $\mathcal{A}$ *outputs an implementation* B *and a distribution* $\mathcal{D}$.
- *The challenger outputs the value* $s' = \mathsf{Rec}^{B,\mathcal{D}}(epk, \delta)$.

For a given δ*, we will say that the leakage-deterring cryptographic primitive satisfies black-box recoverability with respect to the class of input distributions* $\mathscr{D}$*, if for any* PPT *adversary* $\mathcal{A}$ *the following event in the game above happens with negligible probability.*

$$(B \text{ is } \delta\text{-correct w.r.t. } \mathcal{D}) \wedge (\mathcal{D} \in \mathscr{D}) \wedge (s' \neq s)$$

Definition 3. δ-correctness: *The predicate "*B *is* δ*-correct w.r.t.* $\mathcal{D}$*" takes a different form depending on the cryptographic primitive and is intended to capture the fact that the box produced by the adversary should have some minimum utility parameterized by* δ.

Consider the case of a public-key encryption scheme $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$*. The predicate for* δ*-correctness w.r.t.* $\mathcal{D}$ *in this case is as follows:*

$$\Pr[B(\mathsf{Enc}(epk, m)) = m] \geq \delta, \text{ where } m \leftarrow \mathcal{D}$$

where the random variables $epk, \mathcal{D}, B$ *are defined as in the game.*

For a digital signature scheme $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$*, the notion of* δ*-correctness of a box* B *for a message distribution* $\mathcal{D}$ *is defined as follows:*

$$\Pr[\mathsf{Verify}(epk, m, B(m)) = 1] \geq \delta, \text{ where } m \leftarrow \mathcal{D}$$

For an identification scheme $(\mathsf{KeyGen}, \mathsf{Identify})$*, we define* δ*-correctness of an implementation* B *as follows:*

$$\Pr[O_V[B(epk, esk) \leftrightarrow V(epk)] = 1] \geq \delta,$$

where $O_V[B(epk, esk) \leftrightarrow V(epk)]$ *denotes the output of verifier* V *after interacting with box* B *in an identification protocol.*

It is worth noting that the largest class of distributions $\mathscr{D}$ we can hope recoverability to work for a leakage-deterring PKE is one that includes those distributions whose *predicting probability*[3] is by a non-negligible amount smaller than δ; otherwise, one can implement a decryption box by always returning the most probable sample from $\mathcal{D}$.

Also, note that for a signature box to be "non-trivial", δ should be required to be non-negligible w.r.t. distributions $\mathcal{D}$ with super-logarithmic min-entropy[4]. In the case of identification schemes there is no general input passed in the identification box and hence the choice of distribution $\mathcal{D}$ is immaterial.

We will also consider a form of the above definition where a non-black-box technique is used for recovering the owner's private data. In this case we can think of the Rec algorithm as a family of algorithms parameterized by the box algorithm B (as opposed to being a single algorithm with black-box access to B).

We next compare privacy and recoverability and observe a natural trade-off between the two properties. *Privacy* w.r.t. a secret-key oracle $\mathcal{O}(esk, \cdot)$ for a distribution $\mathcal{D}$ (i.e., when adversarial access to the cryptographic primitive is allowed for input distribution $\mathcal{D}$) can not be achieved if the leakage-deterring cryptographic primitive satisfies black-box

[3] $p(\mathcal{D}) := 2^{-\mathbf{H}_\infty(\mathcal{D})}$, where $\mathbf{H}_\infty(\mathcal{D}) = -\log \max_x \Pr[x \in \mathcal{D}]$ is the min-entropy of $\mathcal{D}$.

[4] This entropy requirement for a non-trivial signing box B is necessary. Consider the following example. The key owner prepares a list containing a polynomial number of message-signature pairs, and implements B for a distribution $\mathcal{D}$ which has those messages as its support. B works by simply checking whether a queried message belongs to the list and if yes, it outputs the corresponding signature. Given that such implementation can be produced from publicly collected signatures, the recoverability of the owner secret from such implementation would imply that the privacy property collapses. To exclude this trivial implementation we require that the min-entropy of the message distribution is super-polynomial (and hence this trivial implementation has a super-polynomial description).

recoverability w.r.t. $\mathscr{D}$, in case $\mathcal{D} \in \mathscr{D}$. This easily follows from the fact that the privacy adversary can simulate the Rec algorithm with the help of the secret key oracle.

Security Properties. We next consider how the individual security properties for leakage-deterring primitives should be amended. In general, the original security property (e.g., IND-CPA or unforgeability) should be retained with respect to the enhanced public and secret-keys even in the presence of a corrupted authority running the EnKey protocol.

• *IND-CPA Security* (for leakage-deterring PKE with a dishonest authority): Consider the following game between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$:

- The challenger runs $\mathsf{KeyGen}(\cdot)$ to get (pk, sk) and returns pk to the adversary $\mathcal{A}$.
- The adversary $\mathcal{A}$ selects s and playing the role of the authority runs $\mathsf{EnKey}(\cdot)$ with the challenger on input pk, s.
- The adversary $\mathcal{A}$ chooses two messages m_0, m_1, and sends them to the challenger.
- $\mathcal{C}$ randomly picks a bit $b \in \{0, 1\}$, and gives $\mathcal{A}$ the encryption of m_b under epk.
- Finally, $\mathcal{A}$ returns a guess b' about b.

Suppose there is no PPT adversary $\mathcal{A}$ can guess b correctly with non-negligible advantage, i.e, $|\Pr[b' = b] - \frac{1}{2}| \leq \epsilon$, where ϵ is a negligible function. In this case, we say that the leakage-deterring encryption is IND-CPA-secure (with a dishonest authority).

If we allow the adversary to ask decryption queries at anytime before outputting the guess (it can be both before and after receiving the challenge ciphertext, with the only restriction being that the challenge ciphertext cannot be queried), then we refer to this property as IND-CCA2 security.

We can also consider the security definition with an honest authority, in which case both the algorithms $\mathsf{KeyGen}, \mathsf{EnKey}$ are executed by the challenger.

• *IND-CPA Security* (for leakage-deterring PKE with an honest authority) Consider the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

- The challenger runs $\mathsf{KeyGen}(\cdot)$ to get (pk, sk) and returns pk to $\mathcal{A}$.
- The adversary $\mathcal{A}$ selects s and playing the role of the owner runs $\mathsf{EnKey}(\cdot)$ with the challenger (as authority) to get epk.
- The adversary $\mathcal{A}$ chooses two messages m_0, m_1, and sends them to the challenger.
- $\mathcal{C}$ randomly picks a bit $b \in \{0, 1\}$, and gives $\mathcal{A}$ the encryption of m_b under epk.
- Finally, $\mathcal{A}$ returns a guess b' about b.

Suppose there is no PPT adversary $\mathcal{A}$ can guess b correctly with a non-negligible advantage, i.e, $|\Pr[b' = b] - \frac{1}{2}| \leq \epsilon$, where ϵ is a negligible function. In this case, we say that the leakage-deterring encryption is IND-CPA-secure (with honest authority). The only difference with standard IND-CPA security is that here we have an extra second step.

Below, we will only give unforgeability/impersonation resistance with a dishonest authority, it is straightforward to derive definitions in the setting of an honest authority by changing the roles of challenger and adversary play during EnKey protocol.

• *Unforgeability* (for leakage-deterring digital signatures): Consider the following game between the adversary and the challenger:

- The challenger runs $\mathsf{KeyGen}(\cdot)$ to get pk and returns pk to the adversary $\mathcal{A}$.
- The Adversary $\mathcal{A}$ selects s and playing the role of the authority runs $\mathsf{EnKey}(\cdot)$ with the challenger as a user on input pk, s to get epk;
- $\mathcal{A}$ is allowed to ask queries to a $\mathsf{Sign}(esk, \cdot)$ oracle.

- The adversary $\mathcal{A}$ outputs a message-signature pair (m^*, σ^*).

The adversary wins the game if m^* was never queried to the Sign oracle and $\mathsf{Verify}(epk, m^*, \sigma^*) = 1$. If for any PPT $\mathcal{A}$ the probability of winning the game is negligible, we say the leakage-deterring signature is unforgeable under adaptively chosen message attacks.

• *Impersonation Resistance* (for leakage-deterring identification schemes): Consider the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

- The challenger runs $\mathsf{KeyGen}(\cdot)$ to get pk and sends pk to the adversary $\mathcal{A}$.
- The Adversary $\mathcal{A}$ selects s and playing the role of the authority, and runs the $\mathsf{EnKey}(\cdot)$ protocol with the challenger on input pk, s to get epk;
- $\mathcal{A}$ is allowed to query for $\mathsf{Identify}(P, V)$ protocol transcripts.
- The adversary $\mathcal{A}$ engages in an $\mathsf{Identify}$ protocol execution with the challenger playing the role of the prover.

The adversary wins the game if at the end the challenger playing the role of the verifier accepts the interaction with the adversary. If it holds that for any efficient adversary $\mathcal{A}$, the probability of winning the above game is negligible, we say that the leakage-deterring identification protocol is impersonation resistant against passive attacks. Impersonation resistance against active attacks can be also expressed in a standard way (following e.g., [10]).

2.4 Leakage-deterring Public Key Encryption

In this section, we present constructions of leakage-deterring PKE schemes. We start with a construction from any *additive* homomorphic encryption to demonstrate

our first technique for implementing recoverability, then, we show a generic construction of IND-CPA secure leakage-deterring PKE from any IND-CPA secure encryption along with an improvement that achieves constant size ciphertext, this generic construction can be easily extended to the identity based setting as well. In section 2.5, we provide a general way to achieve IND-CCA2 security for all leakage-deterring encryption schemes.

2.4.1 CPA-secure leakage-deterring PKE from homomorphic encryption

Recall the trivial solution presented in the introduction (encrypting the owner's private data with its public-key). It does not work because an adversarial decryption box is able to test whether the queries fed by the recovering algorithm match the ciphertext stored in *epk*. A seeming fix is to query via rerandomizing the ciphertext contained in the enhanced public key. However, given that the private data are known to the attacker, the adversarial box can check for them and still rejects. So in some sense to go around the problem one has to re-randomize the plaintext as well! (so that after re-randomization, the plaintexts should be distributed according to $\mathcal{D}$ but still somehow be useful for decrypting the private data). We provide a solution along these lines.

Informally, an encryption algorithm $E(\cdot)$ has a homomorphic property if $E(m_1 + m_2) = E(m_1) \cdot E(m_2)$ for some operations $(+, \cdot)$ over plaintexts and ciphertexts respectively. For instance, we can submit a ciphertext $c^* \cdot E(r)$ to the decryption box B, and retrieve the message in c^* from the answer by subtracting r. This method would be effective for our purpose only if B satisfies correctness w.r.t. to random distributions over the whole message space. However we would like a solution that works *even* for adversarially chosen distributions that are unknown at the time of the generation of *epk*. The recovering technique we introduce below achieves this goal.

First assume that we have an underlying encryption $E : (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$ that is an IND-CPA secure PKE with a homomorphic property. Specifically, we assume that for any message m and any a, b from the message space, $\mathsf{Enc}(m)^a \cdot \mathsf{Enc}(b)$ is identically distributed to $\mathsf{Enc}(am + b)$. We call the following construction Scheme-I.

- $\mathsf{KeyGen}(1^\lambda)$: Run the KeyGen algorithm of E, return (pk, sk).

- $\mathsf{EnKey}(O, A)$: This is a protocol between O and A with input (pk, sk, s) and (pk, s) respectively. A randomly chooses $n = |s|$ messages $\omega_i, i = 1, \ldots, n$ according to the uniform distribution over $\{0, 1\}$. Then A calculates $s'_i = \omega_i \oplus s_i$, $\{c_i = E(pk, \omega_i)\}, s' = s'_1 \ldots s'_n$. The protocol terminates with O obtaining the enhanced key pair (epk, esk) where $epk = (pk, \{c_i\}, s')$, and $esk = sk$, while A gets only the enhanced public key epk.

- $\mathsf{Enc}(epk, m)$: This algorithm runs the encryption algorithm Enc, returning $c = \mathsf{Enc}(pk, m)$.

- $\mathsf{Dec}(esk, c)$: This algorithm runs the decryption algorithm Dec, returning $m = \mathsf{Dec}(sk, c)$.

- $\mathsf{Rec}^{B,\mathcal{D}}(epk, \delta)$: With access to a decryption box B and a distribution $\mathcal{D}$, that B supposedly works on with δ-correctness, the objective of this algorithm is to transform the ciphertexts $c_1, \ldots, c_n$ found in the epk to ciphertexts that look *inconspicuous* from the point of view of the box B. For each ciphertext c_i the algorithm will operate as follows. First it will calculate a sufficiently long sequence of pairs (x, y) (the exact length N of the sequence depends on the parameters of B and $\mathcal{D}$ and will be determined in our security analysis). For each pair, the algorithm first independently samples two plaintexts m_0, m_1 according to $\mathcal{D}$. Then

it calculates x, y by solving the following linear system:

$$\begin{cases} 0 \cdot x + y = m_0 \\ 1 \cdot x + y = m_1 \end{cases}$$

Let $(x_l, y_l)_{l=1,\ldots,N}$ be the pairs produced by running the above procedure N times and $m_{0,l}, m_{1,l}$ be the pair of plaintexts used as constant terms of the linear system for the l-th sample. Having calculated those, the algorithm computes $c'_{i,l} = c_i^{x_l} \cdot E(pk, y_l)$ for $l = 1, \ldots, N$, and feeds B with those ciphertexts (whose corresponding plaintexts follow $\mathcal{D}$). Let $a_1, \ldots, a_N$, be the answers of the box B where $a_l = \perp$ if B does not provide an answer for the l-th ciphertext. Now consider the modified answer sequence to be a string over $\{0, 1, \perp\}$ defined as follows:

$$a_l^* = \begin{cases} (a_l - y_l)/x_l & a_l \in \{m_{0,l}, m_{1,l})\} \wedge x_l \neq 0 \\ \perp & \text{otherwise} \end{cases}$$

Note that $a_l^* \in \{0, 1, \perp\}$. If the majority symbol among the non-$\perp$ symbols of $\langle a_1^*, \ldots, a_N^* \rangle$ is defined, the recovering algorithm calculates it as v_i and proposes it as the decryption of c_i (otherwise the algorithm fails). This procedure is repeated for all ciphertexts $c_1, \ldots, c_n$ thus forming $v = v_1 \ldots v_n$. Finally, the recovering algorithm proposes as the private data of the owner the string $s' \oplus v$ where s' is parsed from the *epk*.

Security Analysis: We will first sketch correctness and three security properties, i.e, *security, privacy, recoverability.* First observe that correctness is trivial, according to the correctness of the underlying encryption scheme E while IND-CPA security is also relatively obvious since the extra information exposed due to our extension are some independent values $(\omega_1 \ldots \omega_n, s)$. Now regarding the privacy property, we can see that

the EnKey algorithm in Scheme-I is a KEM/DEM mechanism [42], using a KEM which encrypts each bit of the key with a secure encryption. Given $\{c_i\}$, the adversary is not able to predict the bit ω_i with a sufficient bias, thus every ω_i is random conditioned on the adversary's view. This proves privacy (assuming no secret-key oracle $\mathcal{O}(esk, \cdot)$ is given). Regarding recoverability we can prove it w.r.t. essentially any distribution $\mathcal{D}$. The Rec algorithm produces a sequence of ciphertexts with plaintexts following $\mathcal{D}$ whose correct decryption reveals the bits ω_i by a majority argument. As long as the correctness of the box B is non-negligibly larger than the collusion probability of $\mathcal{D}$ (which is a minimal characteristic of "box usefulness") the recovering algorithm will produce the ω_i values with overwhelming certainty since it can do a perfect simulation of ciphertexts with $\mathcal{D}$ distributed plaintexts. The formal proofs are as follows:

Theorem 1. *Scheme-I achieves IND-CPA security (against dishonest authority) and privacy (without secret-key oracle) if the underlying PKE is IND-CPA secure. It also satisfies black-box recoverability w.r.t. any* $\delta > 0$ *and the class of distributions* $\mathscr{D} = \{\mathcal{D} \mid \exists \alpha : \alpha$ *is non-negligible and* $\mathbf{H}_\infty(\mathcal{D}) \geq \log \frac{1}{\delta - \alpha}\}$.

Proof. As explained above,the correctness and IND-CPA security is obvious. We will demonstrate the proof for *privacy* and *recoverability* as follows:

First we examine privacy. Let us define $E'(\omega)$ as $E(\omega_1) \dots E(\omega_n)$, a bitwise encryption using E. (w.l.o.g, we assume messages are bitstrings with length n, if not, challenger can do a proper padding on them). It is straightforward to see that E' is IND-CPA secure if E is, otherwise, one can easily distinguish $E(0), E(1)$ by distinguishing $E'(m_0), E'(m_1)$, where m_0, m_1 are identical except at one bit.

Suppose $\mathcal{C}$ is the challenger of E', $\mathcal{A}$ is the adversary who can break the privacy of scheme-I. We will build a simulator algorithm $\mathcal{S}$ which uses $\mathcal{A}$ as an oracle to break the IND-CPA security of E'.

$\mathcal{S}$ first forwards the public key pk from $\mathcal{C}$ to $\mathcal{A}$, he then chooses two random messages m_0, m_1, sends them to $\mathcal{C}$ and gets the challenge c, also he receives s_0, s_1 from $\mathcal{A}$. $\mathcal{S}$ randomly select b_0, b_1, computes $s' = s_{b_0} \oplus m_{b_1}$, and sends (pk, c, s') to $\mathcal{A}$ as epk. If $\mathcal{A}$ returns b_0 correctly, $\mathcal{S}$ outputs b_1 as his answer, otherwise he outputs $1 - b_1$.

It is easy to see that the simulator's advantage of breaking the semantic security of E' is at least half of the advantage that $\mathcal{A}$ has to break the privacy (denoted by Δ). If $c = E'(m_{1-b_1})$, then s' perfectly hides s_{b_1} since $s_{b_0} \oplus m_{b_1}$ now is independent with epk, $\mathcal{A}$ can only guess b_0 correctly with probability $1/2$. If $c = E'(m_{b_1})$, the epk is in a valid form and this time $\mathcal{A}$ will have advantage Δ. Thus, the simulator's advantage is $\frac{1}{2}(\frac{1}{2} + \frac{1}{2} + \Delta) - \frac{1}{2} = \frac{\Delta}{2}$.

Next, we examine recoverability. It is obvious that the KeyGen and EnKey procedures can be easily simulated. Note that the way we sample (x, y) in the Rec algorithm, every query is an encryption of a message independently sampled from $\mathcal{D}$ (the only exception is that $\mathcal{D}$ almost always outputs only one message but in this case, any box becomes "trivial"). Thus, the recovering query is identically distributed as normal decryption queries and B would have δ-correctness for every recovering query!

Now we analyze the number of repetitions needed (in terms of an asymptotic function of the security parameter λ) to guarantee we are almost certain that s will be returned in the Rec algorithm.

First, we call an experiment useful if $m_0 \neq m_1$, otherwise, we record a $\perp$ for this query. The probability of having one useful experiment after sampling N_0 pairs of (m_0, m_1) will be $1 - \mathbf{Col}(\mathcal{D})^{N_0}$, where $\mathbf{Col}(\mathcal{D})$ is the collusion probability of distribution $\mathcal{D}$ which denotes the probability of sampling a same element from two independent trials. Observe that $\mathbf{Col}(\mathcal{D}) < 1 - \gamma$, for some non-negligible γ, if $\mathcal{D}$ is not a trivial distribution

which has probability almost 1 over one single element. If we repeat $N_0 = O(\log^2 \lambda)$ times sampling, we will get a useful experiment with probability almost 1.

Further, for one useful query, only two answers x, $x + y$ (m_0, m_1 respectively) are considered correct, all others are ignored (denoted by $\perp$). The probability of getting at least one correct answer after repeating N_1 times useful experiments is $1 - (1 - \delta)^{N_1}$, if $N_1 = O(\log^2 \lambda)$, this probability is almost 1(with negligibly small difference).

Recall that $\delta = p(\mathcal{D}) + \alpha$, where $p(\mathcal{D}) = 2^{-\mathbf{H}_\infty(\mathcal{D})}$ denotes the predicting probability of $\mathcal{D}$. In one useful experiment, the probability of returning an incorrect but non-$\perp$ answer is at most $p(\mathcal{D})$, since this happens only when B returns m_b while the correct answer is m_{1-b}, for $b = 0, 1$ (recall that m_0, m_1 are independently sampled from $\mathcal{D}$).

Now we focus on non-$\perp$ answers only, the probability of an incorrect answer appearing among the non-$\perp$ answers is at most $q = \frac{p(\mathcal{D})}{2p(\mathcal{D})+\alpha}$ (This can be argued as follows: suppose $p(\mathcal{D}) = p + t$,where p is the probability of returning an incorrect but non-$\perp$ answer in one query and $t \geq 0$; then, the probability of obtaining an incorrect but non-$\perp$ answer among all non-$\perp$ answers is at most $\frac{p}{\delta+p} \leq \frac{p+t}{\delta+p+t} = q$). Suppose X is the random variable that denotes the number of appearances of incorrect but non-$\perp$ answers after collecting N_2 non-$\perp$ answers, μ denotes the expectation of X which is no bigger than $N_2 q$. The probability that correct answers do not constitute the majority is $\Pr[X \geq \frac{N_2}{2}]$. Using the upper tail of the Chernoff bound:$\Pr[X > (1 + \beta)\mu)] \leq e^{\frac{-\beta^2\mu}{3}}$, we can verify that this probability is bounded by $\exp(-N_2\alpha^2/(24p(\mathcal{D})^2 + 6p(\mathcal{D})\alpha))$.

So if we collect more than $N_2 = O(\alpha^{-2} \log^2 \lambda)$ non-$\perp$ symbols, the majority will be occupied by the correct answers with probability almost 1 (negligibly small difference), and hence we can recover the bit.

Combining above, if we repeat the recovering procedure $N_0 \times N_1 \times N_2 = O(\alpha^{-2} \log^6 \lambda)$ times for each bit of s we will successfully recover s with probability almost 1. $\square$

Remark 2. *Note that the restriction for the class of distribution is optimal in the sense that otherwise, the box would be "trivial".*

2.4.2 Generic CPA-secure leakage-deterring PKE with an honest authority

In this section, we relax further the requirements of leakage-deterring PKE to minimal by constructing a scheme based on *any* secure PKE. We will only consider IND-CPA security with honest authority in this section and we will show how to go beyond this and achieve security against dishonest authorities (and actually IND-CCA2) in the next section.

Linear-size construction. To make the exposition more accessible we present first a less efficient construction (with linear size ciphertexts in the length of hidden information); then we show our main generic construction with constant size ciphertext. Consider a semantically secure PKE E. The main idea of the construction is as follows. For each bit of private data there is a pair of public keys, and the owner has only one of the secret keys. The ambiguity of which secret key the owner has offers the opportunity for the recovering algorithm to work. We call this construction Scheme-II, details are as follows:

- $\mathsf{KeyGen}(1^\lambda)$: This algorithm generates $n = |s|$ key pairs $(pk_1, sk_1), \ldots, (pk_n, sk_n)$.
- $\mathsf{EnKey}(O, A)$: (O, A) have inputs $(pk_1, \ldots, pk_n, s, sk_1, \ldots, sk_n)$, and $(pk_1, \ldots, pk_n, s)$ respectively, where $s \in \{0,1\}^n$. A randomly generates $r \in \{0,1\}^n$ which we call indicating string, and n new random public keys $pk'_1, \ldots, pk'_n$. The enhanced public key epk is n pairs of public keys $(pk_1^0, pk_1^1), \ldots, (pk_n^0, pk_n^1)$, together with $s' = r \oplus s$, where for $i = 1, \ldots, n$, $pk_i^{r_i} = pk_i, pk_i^{1-r_i} = pk'_i$, and the enhanced secret key is $esk = (sk, r)$, where $sk = (sk_1, \ldots, sk_n)$.

- $\mathsf{Enc}(epk, m)$:This algorithm first randomly picks $m_1, \ldots, m_{n-1}$, and computes $m_n = m - \sum_{i=1}^{n-1} m_i$ (wlog we assume that additive secret-sharing works over the plaintext space). It outputs the ciphertext $c = [(c_1^0, c_1^1), \ldots, (c_n^0, c_n^1)]$, where $c_i^0 = \mathsf{Enc}(pk_i^0, m_i), c_i^1 = \mathsf{Enc}(pk_i^1, m_i)$.

- $\mathsf{Dec}(esk, c)$: To decrypt ciphertext c, this algorithm chooses from c the ciphertexts corresponding to the indicating string r, and returns $m = \sum_{i=1}^{n} \mathsf{Dec}(sk_i, c_i^{r_i})$.

- $\mathsf{Rec}^{B,\mathcal{D}}(epk, \delta)$: With access to a decryption box B and a plaintext distribution $\mathcal{D}$ for which the box supposedly works with δ-correctness, the algorithm recovers each bit s_i of s by repeating the following procedure N times (the exact value of N will be specified in the analysis):

  It first samples m, m' independently, according to $\mathcal{D}$, randomly chooses $m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_n$, and computes $m_i^0 = m - \sum_{j \neq i} m_j$, and $m_i^1 = m' - \sum_{j \neq i} m_j$.

  Then it feeds B with $[(c_1^0, c_1^1), \ldots, (c_n^0, c_n^1)]$, where for all $j \neq i$, c_j^0, c_j^1 encrypts the same message m_j while $c_i^0 = \mathsf{Enc}(pk_i^0, m_i^0), c_i^1 = \mathsf{Enc}(pk_i^1, m_i^1)$.

  The algorithm records a 0, if the response from the box is m, 1 if the response is m', and $\perp$ in any other case including the case $m = m'$. For each i, the algorithm will propose r_i to be the majority of the recorded non-$\perp$ values (the algorithm fails if majority is not well-defined).

  The above procedure is repeated for all $i \in \{1, \ldots, n\}$ to form a string r, and finally, the algorithm outputs $s = s' \oplus r$, where s' is parsed from epk.

Security analysis. We first sketch the security properties of Scheme-II. Let us call an encryption using a single pair of keys as a "unit building block". It is not hard to see that IND-CPA security of the unit building block implies the IND-CPA of scheme-II

(assuming authority is honest and taking into account the security of additive secret-sharing). Regarding privacy, observe that s is perfectly hidden within epk as a one-time pad ciphertext. Finally, regarding recoverability w.r.t any distribution $\mathcal{D}$, the recovering algorithm can attempt to query different encrypted messages in any single unit location. Due to the secret-sharing throughout all pairs, any box (even partly successful) has to include a key for each coordinate. Due to these facts, the recovering algorithm can detect which secret key does the owner possess at each location something that leads to the calculation of the indicating string and hence the recover of the private data. A detailed analysis is as follows:

Theorem 2. *Scheme-II achieves* privacy *(without secret-key oracle access). It also satisfies IND-CPA security (with honest authority) and black-box recoverability w.r.t. any* $\delta > 0$ *and the class of distributions* $\mathscr{D} = \{\mathcal{D} \mid \exists \alpha : \alpha$ *is non-negligible and* $\mathbf{H}_\infty(\mathcal{D}) \geq \log \frac{1}{\delta - \alpha}\}$*, if the underlying encryption is a regular IND-CPA secure PKE.*

Proof. The privacy property is trivially achieved because the owner private data is hidden with a one-time pad. Also it is easy to see correctness, owner has one key for each pair and can successfully decrypt one of the ciphertext in each pair. We prove the IND-CPA security with honest authority, and recoverability as follows.

We first prove the IND-CPA security of a simplified version of Scheme-II, (recall that we named it "unit building block") in which there is only one pair of public keys (pk_1^0, pk_1^1), the secret information s will be one bit only. During encryption, one encrypts message m under both public key (without doing secret sharing of the message).

Claim 3. *The unit building block* E^* *of Scheme-II is IND-CPA secure if the underlying encryption* E *is IND-CPA secure.*

Proof. of the claim: For any two message m_0, m_1, we define the pair $E(pk_1^0, m_i), E(pk_1^1, m_j)$ as E_{ij}. We would show that both E_{00}, E_{11} are indistinguishable from E_{01}, thus E_{00} is indistinguishable from E_{11}.

Suppose $\mathcal{A}$ can distinguish E_{00} from E_{01} with advantage Δ, one can use $\mathcal{A}$ to break the semantic security of E as follows:

Assume $\mathcal{C}$ is the challenger of E, when the simulator receives pk from $\mathcal{C}$ and a private bit s from $\mathcal{A}$, he randomly selects another public key pk', and sends $\mathcal{A}$ $(pk', pk, s \oplus 1)$ as epk. Then, the simulator forwards m_0, m_1 from $\mathcal{A}$ to $\mathcal{C}$. Whenever the simulator gets a challenge ciphertext c from $\mathcal{C}$, he computes $c' = E(pk', m_0)$, and sends (c', c) to $\mathcal{A}$ as his challenge. The simulator will outputs $\mathcal{A}$'s guess directly to $\mathcal{C}$ (suppose that $\mathcal{A}$ outputs 0 for E_{00}, and 1 for E_{01}).

It is easy to see that, (c', c) is exactly E_{00} if $c = E(m_0)$, or E_{01} if $c = E(m_1)$, so simulator breaks semantic security of E with the same advantage as $\mathcal{A}$ distinguishes E_{00}, E_{01}. Similarly, we can prove E_{01}, E_{11} are indistinguishable. Thus $E_{00} = E^*(m_0)$, and $E_{11} = E^*(m_1)$ are indistinguishable. □

With the above claim, we will reduce the security of the unit building block E^* to the security of Scheme-II. Suppose $\mathcal{C}$ is the challenger of E^*, $\mathcal{A}$ is the adversary who successfully breaks the semantic security of Scheme-II with advantage Δ, we will build a simulator to break the security of E^* using $\mathcal{A}$.

After receiving a private string s from $\mathcal{A}$, the simulator forwards the first bit s_1 to $\mathcal{C}$. After receiving (pk_1^0, pk_1^1, b_1) from $\mathcal{C}$, the simulator generates another $n-1$ pairs of public keys and secret keys $\{(pk_i^b, sk_i^b)\}_{i=2,\ldots,n-1;b=0,1}$, and sends $[(pk_1^0, pk_1^1), \ldots, (pk_n^0, pk_n^1), s']$ to $\mathcal{A}$ as epk, where $s'_i = b_i$, and $b_2 \ldots b_{|s|}$ are random bits.

After getting m_0, m_1 from $\mathcal{A}$, the simulator randomly selects $n-1$ messages $m_2, \ldots, m_n$, computes $m = m_0 - \sum_{i=2}^n m_i$, $m' = m_1 - \sum_{i=2}^n m_i$, and forwards m, m'

to $\mathcal{C}$. When the simulator receives a challenge $c = (c_1^0, c_1^1)$ from $\mathcal{C}$, then, for all $i \in \{2, \ldots, n\}$, $b \in \{0, 1\}$, he computes $\{c_i^b = \mathsf{Enc}(pk_i^b, m_b)\}$, and sends $\mathcal{A}$ his challenge $[(c_1^0, c_1^1), \ldots, (c_n^0, c_n^1)]$.

Simulator directly outputs $\mathcal{A}$'s guess as his own guess to $\mathcal{C}$ assumes that $\mathcal{A}$ outputs 0 for m_0 and 1 for m_1. It is easy to see that the simulator's advantage of breaking the IND-CPA security of Scheme-II is exactly Δ. □

Next we prove the recoverability holds. First we argue that adversarial box can not distinguish recovering queries (for some pair, two ciphertext contain different messages) from normal ciphertext. Essentially, we will argue that the box will have similar performance in both cases. Note that when a box gets a normal ciphertext and the message is sampled according to $\mathcal{D}$, it will return a correct answer with probability at least δ due to the δ-correctness, and in such cases, the box will return an incorrect but non-$\perp$ answer with probability at most $p(\mathcal{D})$ because of the fact that no information about the incorrect answer is contained in the decryption query.

Claim 4. *Suppose the underlying encryption scheme E satisfies ϵ-semantic security (i.e, no* PPT *adversary can distinguish $E(m_0)$ and $E(m_1)$ with advantage ϵ for any pair of different messages (m_0, m_1)), then any box with δ-correctness created by a* PPT *adversary will return a correct answer with probability at least $\delta - 2\epsilon$ and will return a incorrect but non-$\perp$ answer with probability at most $p(\mathcal{D}) + 2\epsilon$, when fed with a recovering query.*

Proof. of the claim: We prove for probability of returning a correct answer only, for the case of incorrect but non-$\perp$, it follows straightforwardly. First note that, in a recovering query, the messages are also sampled independently from $\mathcal{D}$. Suppose there is an adversary $\mathcal{A}$ producing a box $\mathcal{B}$ with δ-correctness when queried with normal ciphertext and with $(\delta - \Delta)$-correctness for a non-negligible Δ when queried with

recovering ciphertext, one can use $\mathcal{A}$ to distinguish two ciphertexts $E(m_0), E(m_1)$ in the IND-CPA game of E as follows:

First, the simulator receives public key pk from the challenger $\mathcal{C}$ of E, also it receives n public keys $(pk_1, \ldots, pk_n)$ and s from the adversary $\mathcal{A}$.

The simulator randomly chooses a position $i \in \{1, \ldots, n\}$, a random bitstring $r \in \{0,1\}^n$, and chooses $n-1$ random public keys $pk'_1, \ldots, pk'_{i-1}, pk'_{i+1}, \ldots, pk'_n$. The simulator sends $\mathcal{A}$ the enhanced public key $epk = [(pk_1^0, pk_1^1), \ldots, (pk_n^0, pk_n^1), r \oplus s]$, where for all $j \neq i$, $pk_j^{r_j} = pk_j, pk_j^{1-r_j} = pk'_j$, and $pk_i^{r_i} = pk_i, pk_i^{1-r_i} = pk$.

$\mathcal{A}$ then produces a decryption box B and a distribution $\mathcal{D}$. To create a challenge ciphertext for B, the simulator first randomly chooses two messages m, m' according to $\mathcal{D}$, then he randomly chooses $n-1$ messages $m_1, \ldots, m_{i-1}, m_{i+1}, \ldots, m_n$. Further, he computes $m_i^0 = m - \sum_{j \neq i} m_j$, and $m_i^1 = m' - \sum_{j \neq i} m_j$, then sends m_i^0, m_i^1 to $\mathcal{C}$, and receives a challenge c from $\mathcal{C}$. The simulator then randomly selects m_i^b from m_i^0, m_i^1, and feeds B with ciphertext $[(c_1^0, c_1^1), \ldots, (c_n^0, c_n^1)]$, where for all $j \neq i$, both c_j^0, c_j^1 contain the same message m_j, and $c_i^{r_i} = E(pk_i, m_i^b), c_i^{1-r_i} = c$.

If B returns the right message (if $b = 0$, it is m, if $b = 1$, it is m'), then the simulator returns b as his guess, otherwise, he returns $1 - b$.

It is obvious that if the simulator guesses correctly(m_i^b is the message $\mathcal{C}$ encrypts in c), the challenge for $\mathcal{A}$ is a normal ciphertext, otherwise, it is a recovering query. Thus, simulator will break the semantic security of E(return a correct b) with probability $\frac{1}{2}\delta + \frac{1}{2}(1 - \delta + \Delta) = \frac{1}{2} + \frac{\Delta}{2}$, thus $\Delta \leq 2\epsilon$.

Similarly, we can show the probability of returning an incorrect but non-$\perp$ answer will be 2ϵ close to $p(\mathcal{D})$, too. □

The rest parts are very similar to the analysis of theorem 1. For each bit of the owner data, in each query there are only two potentially correct answers (m, m' as we

used in the Rec algorithm in scheme-II). Since for each query, we have the probability of returning a correct answer is almost δ, and the probability of returning an incorrect but non-$\perp$ answer is almost $p(\mathcal{D})$, we only need to proceed to estimate the number of repetitions needed. The analysis is also composed of three main steps, we first estimate the number of samples needed for getting a useful experiment, and then estimate the number of repetition needed for collecting at least one correct answer, then we estimate the number of correct answers needs to be collected to ensure that the majority will be the correct answer. With the above claim, the performance of the adversarial box is negligibly close to that in scheme-I, thus the number of repetitions needed is also $O(\alpha^{-2}\log^6\lambda)$. For details of the calculation, we refer to the proof of theorem 1. □

Main generic construction. In Scheme-II, the sender splits the message into n pieces. This makes the ciphertext size (number of ciphertext units) linear in the length of the owner's private data. We now improve the generic construction to achieve a ciphertext size $O(\log\frac{1}{\delta})$ by using an error correcting code to create the indicating string, where δ is a specified minimum correct decryption probability that is assumed to be constant and is a parameter of the construction. We call this construction Scheme-III.

- $\mathsf{KeyGen}(1^\lambda)$: Same as in Scheme II.
- $\mathsf{EnKey}(O, A)$: (O, A) have inputs $(pk_1, \ldots, pk_m, s, sk)$, and $(pk_1, \ldots, pk_m, s)$ respectively where $s \in \{0,1\}^n$; the parameter m is selected based on n according to an ECC (e.g., from [70]) that corrects up to $\frac{m}{5}$ errors. A randomly generates $\tilde{r} \in \{0,1\}^n$, computes the indicating string $r = ECC(\tilde{r})$. Also, A selects m random public keys $pk'_1, \ldots, pk'_m$. The protocol terminates with O obtaining (epk, esk) and A obtaining epk, where epk is $(pk_1^0, pk_1^1), \ldots, (pk_m^0, pk_m^1)$, together with $s' = \tilde{r} \oplus s$, and for $i = 1, \ldots, m$, $pk_i^{r_i} = pk_i, pk_i^{1-r_i} = pk'_i$, and $esk = (sk, r)$.

- $\mathsf{Enc}(epk, m)$: To encrypt a message m, the algorithm first chooses a random subset $S \subseteq \{1, \ldots, m\}$ with size $t = 5 \ln \frac{4}{\delta}$. Then it randomly picks $m_1, \ldots, m_{t-1}$, and computes $m_t = m - \sum_{i=1}^{t-1} m_i$. The ciphertext $c = [S, (c_1^0, c_1^1), \ldots, (c_t^0, c_t^1)]$, where $c_i^b = \mathsf{Enc}(pk_{S_i}^b, m_i)$ for $b \in \{0, 1\}$.

- $\mathsf{Dec}(esk, c)$: To decrypt ciphertext c, this algorithm chooses from c the ciphertexts corresponding to the indicating string r projected on S, and returns $m = \sum_{i=1}^{t} \mathsf{Dec}(sk_{S_i}, c_i^{r_{S_i}})$.

- $\mathsf{Rec}^{B,\mathcal{D}}(epk, \delta)$: With access to a decryption box B and a plaintext distribution $\mathcal{D}$, the algorithm recovers each bit s_i of s by repeating the following procedure N times (the exact number will be specified in the analysis):

  It first randomly selects a subset $S \subseteq \{1, \ldots, m\}$ with size t.

  If $i \in S$, and i is the k-th element of S, the algorithm randomly chooses m, m' independently according to $\mathcal{D}$ as well as random values $m_1, \ldots, m_{k-1}, m_{k+1}, \ldots, m_t$. Then, it computes $m_k^0 = m - \sum_{j \neq k} m_j$, and $m_k^1 = m' - \sum_{j \neq i} m_j$. It feeds B with $[(c_1^0, c_1^1), \ldots, (c_t^0, c_t^1)]$ where, for all $j \neq k$, the pair c_j^0, c_j^1 encrypts the same message m_j using pk_i^0, pk_i^1 respectively, while $c_k^b = \mathsf{Enc}(pk_i^b, m_k^b)$ for $b \in \{0, 1\}$.

  If $i \notin S$, the algorithm proceeds by performing a regular encryption of a plaintext from $\mathcal{D}$.

  If $i \in S$ and the response of the decryption box is m, the algorithm records 0; if $i \in S$ and the response is m', this algorithm records a 1; Otherwise (in any other case including $i \notin S$ or $m = m'$), it records $\perp$. For each i the majority of the non-$\perp$ recorded values is proposed as the value of r_i. If no majority is defined, a random bit is produced as r_i.

The above procedure is repeated for all $i \in \{1, \ldots, m\}$, and a string r is formed. The decoding algorithm of ECC is now executed on r to obtain $\tilde{r}$. The algorithm terminates by returning $s = s' \oplus \tilde{r}$, where s' is parsed from epk.

Security analysis. The IND-CPA and privacy properties are essentially the same as in scheme II. We only discuss recoverability which is significantly more complex. The intuition is that because of the ECC the Rec algorithm would work as long as a linear fraction of bits of r can be recovered. As we will prove in the appendix, suppose that q is the number of positions among the m for which our recoverability procedure fails. We will show that the probability of correct decryption will become roughly smaller than $e^{-tq/m} = \delta^{5q/m}$. From this we derive that any decryption box operating with probability at least δ (as postulated) can make our algorithm fail in at most $m/5$ of the m secret keys which is sufficient for correct decoding. The full analysis is presented as follows:

Theorem 5. *Scheme-III parameterized by any $\delta > 0$, achieves* privacy *(without secret-key oracle access). Further, if the underlying public key encryption scheme is IND-CPA secure, it satisfies IND-CPA security (with honest authority) and black-box recoverability w.r.t. the class of distributions $\mathscr{D}_\delta = \{\mathcal{D} \mid \mathbf{H}_\infty(\mathcal{D}) \geq \log|s| + \log\frac{1}{\delta} - c\}$, where c is a constant (depending on the ECC) and $|s|$ the length of the embedded private information.*

Proof. All properties except recoverability are the same as Scheme-II. We only analyze recoverability below. The major difference in this analysis is that we need to examine the maximum number of positions for which the recoverability algorithm can fail while still maintaining that the decryption box B has more than δ-correctness.

First the number of experiments for each index i, denoted by N is calculated as in the proof of theorem 1 with the following modifications: First, we use a probability α_0 in place of α, which is defined as $(\delta - \kappa)/m^2$ where $\kappa = p(\mathcal{D}) = 2^{-\mathbf{H}_\infty(\mathcal{D})}$ denoting the

predicting probability. Second, we repeat more times due to the fact that randomly selecting S will not always contain i to be a "useful query" for recovering. But sampling S randomly (instead of always containing some i) aims at producing the recovering queries indistinguishable from normal ciphertext. Suppose now the target of the recovering algorithm is the i-th bit, in one selection, the probability $\Pr[i \in S]$ is $C_{m-1}^{t-1}/C_m^t = t/m$. From the lower tail of Chernoff bound, the probability of selecting N_3 times without hitting i once is smaller than $e^{-(\frac{N_3 t}{2m}-1)}$. After randomly sample $4m/t$ times, one will be sure that one of the subsets will contain i, and one useful query is created. In total, The recovering procedure repeats for $N_0 \times N_1 \times N_2 \times N_3$ times, where N_0, N_1, N_2 are as in the analysis of theorem 1 where α is substituted with α_0 (note that if we can reset the box across experiments the ciphertexts for which $i \notin S$ can be omitted).

The main challenge in the proof of the theorem is the fact that the box B might behave differently depending on i and thus force us to err in a number of locations i. We will prove that we can bound this number and hence our error-correction layer will be sufficient for recovering the hidden information in the *epk*. Let $\delta_i = \Pr[B \text{ decrypts correctly} \mid i \in S]$. We divide the indices $i \in \{1, \ldots, m\}$ in two sets, Bad and Good, according to the rule $i \in \mathsf{Good}$ if and only if $\delta_i \geq \kappa + \alpha_0$. Based on our choice of N, if $i \in \mathsf{Good}$ the recoverability will return the proper bit in the i-th coordinate with overwhelming probability. In order to upper bound the size of Bad consider the following. Let D be the event of correct decryption. We have that,

$$\Pr[\mathsf{D}] = \Pr[\mathsf{D} \mid S \cap \mathsf{Bad} = \emptyset] \cdot \Pr[S \cap \mathsf{Bad} = \emptyset] + \Pr[\mathsf{D} \mid S \cap \mathsf{Bad} \neq \emptyset] \cdot \Pr[S \cap \mathsf{Bad} \neq \emptyset]$$

Regarding $\Pr[S \cap \mathsf{Bad} = \emptyset]$ observe that if $k = |\mathsf{Bad}|$, the probability is bounded by $p(k,t) = C_{m-k}^t/C_m^t = \prod_{i=0}^{t-1}(1 - \frac{k}{m-i}) \leq (1 - \frac{k}{m})^t$. From inequality $e^x \geq 1 + x$, we can get $p(k,t) \leq e^{-kt/m}$. Regarding $\Pr[\mathsf{D} \mid S \cap \mathsf{Bad} \neq \emptyset]$ note that it is bounded by

$\sum_{i \in \mathsf{Bad}} \delta_i \leq m(\kappa + \alpha_0)$ (This bound follows directly from the fact that $\Pr[F | \cup_{i=1}^{n} A_i] \leq \sum_{i=1}^{n} \Pr[F|A_i]$, for any event F, A_i). We now derive the following,

$$\delta \leq \Pr[\mathsf{D}] \leq e^{-tk/m} + m(\kappa + \alpha_0)$$

From which we obtain the upper bound $k \leq \frac{m}{t} \cdot \ln(\delta - m(\kappa + \alpha_0))^{-1}$. Now observe that due to the condition for the min-entropy, we derive a bound on $\kappa \leq 2^c \delta / |s| = c' \delta / |s|$, for some constant c'. From the choice of α_0 we can prove that $\delta - m(\kappa + \alpha_0) \geq \delta/4$ as long as c is selected appropriately (taking into account the error-correcting rate which is constant). We plug this condition and the fact that $t = 5 \ln(4\delta^{-1})$ we conclude that $k \leq m \ln(4\delta^{-1})/5 \ln(4\delta^{-1}) = m/5$.

The rest is similar to the proof of theorem 2, whenever the i-th secret key is contained in the decryption box, as argued in the analysis of theorem 2, adversary will have similar performance when fed with a recovering query, and from the estimation of number of repetitions, if one repeat $O(\alpha_0^{-2} m \log^6 \lambda) = O(\alpha^{-2} \lambda^5 \log^6 \lambda)$ times(given that the length of the codeword $m = O(\lambda)$, and $\alpha = \delta - p(\mathcal{D})$), one can recover r_i correctly as long as $i \in \mathsf{Good}$ with overwhelming probability. So the number of errors in recovering r is at most k, while the ECC is able to correct up to $\frac{m}{5}$ errors, and $\frac{m}{5} \geq k$, thus r_0 will be recovered correctly with overwhelming probability and hence also s. □

Remark 3. *In this construction, the ciphertext size is parameterized by the correctness δ which is influenced by the min-entropy of the distribution the box works on. Essentially, if the desired min-entropy that the scheme should be leakage deterring against gets smaller, then the ciphertext size should increase.*

Remark 4. *The generic construction can also be easily adapted to the identity based setting. To accomplish this, observe that we can replace pk_i^b with $(ID|i, b)$, instantiate the IBE system with a hierarchical IBE scheme and apply the above generic construction to*

derive an identity based leakage-deterring encryption. We omit the detailed construction of identity based leakage-deterring encryption, but for the sake of completeness, we present the definitions needed in the appendix.

2.5 Generic CCA2 Secure Construction with Dishonest Authority

In this section, we introduce a general method to construct an IND-CCA2 secure leakage-deterring encryption with dishonest authority from any leakage-deterring PKE which satisfies IND-CPA security with honest authority, and any IND-CCA2 secure standard PKE. The main idea is to compose these two encryptions to form a nested encryption with the outer layer encryption to be the IND-CCA2 secure one. *Recoverability* could be maintained because the Rec algorithm can run the Rec algorithm of the inner leakage-deterring encryption to collect queries, and encrypt them using the outer layer public key to form its own recovering queries.

Construction Suppose E_1 is any IND-CPA secure leakage-deterring PKE (with an honest authority), and E_2 is any IND-CCA2 secure PKE. We call the following construction Scheme-IV.

- $\mathsf{KeyGen}(1^\lambda)$: This algorithm first executes the KeyGen algorithm of both E_1, E_2, and return $(pk_1, sk_1), (pk_2, sk_2)$.

- $\mathsf{EnKey}(O, A)$: This is a protocol between O, A with inputs (pk_1, sk_1, s) and (pk_1, s) respectively; it proceeds by executing the EnKey protocol of E_1 to get (epk_1, esk_1) first, and this protocol terminates with O obtaining enhanced key pair (epk, esk), and A obtaining epk only, where $(epk, esk) = ((epk_1, pk_2), (esk_1, sk_2))$.

- $\mathsf{Enc}(epk, m)$: To encrypt a message m, this algorithm runs the encryption algorithms of both of E_1, E_2, and returns the ciphertext as $c = \mathsf{Enc}(pk_2, \mathsf{Enc}(epk_1, m))$

- $\mathsf{Dec}(esk, c)$: To decrypt a ciphertext c, this algorithm runs the decryption algorithms of both E_1, E_2 and returns $m = \mathsf{Dec}(esk_1, \mathsf{Dec}(sk_2, c))$.

- $\mathsf{Rec}^{B,\mathcal{D}}(epk, \delta)$: With access to a decryption box B and a plaintext distribution $\mathcal{D}$, this algorithm calls the Rec algorithm R_1 of E_1. For each query c of R_1, this algorithm feeds B with $\mathsf{Enc}(pk_2, c)$. It then passes the responses of the box to R_1 and returns whatever R_1 returns.

Security Analysis *Correctness* is obvious. Privacy, recoverability and IND-CCA2 security follow easily from the security of the outer layer encryption and the properties of the inner-layer encryption. The details are presented below:

Theorem 6. *Scheme-IV is IND-CCA2 secure with dishonest authority if E_2 is a IND-CCA2 secure PKE, and achieves same privacy and recoverability as the underlying leakage-deterring PKE E_1.*

Proof. We will reduce the IND-CCA2 security of the outer layer encryption E_2 to the security of scheme-IV. Suppose the challenger of E_2 is $\mathcal{C}$, and the adversary of scheme-II is $\mathcal{A}$. We can build a simulator algorithm to attack the security of E_2 by using $\mathcal{A}$, the adversary on scheme-IV.

After receiving the public key pk from $\mathcal{C}$, the simulator randomly chooses a key pair (pk_1, sk_1) for E_1, the inner layer encryption. Also, he randomly chooses a secret string s. The simulator then sends $\mathcal{A}$ (pk_1, pk, s) and receives the enhanced key pairs $(epk, esk) = ((epk_1, pk_2), (esk_1, sk_2))$.

When $\mathcal{A}$ asks a decryption query c_i, the simulator forwards it to $\mathcal{C}$ and gets an answer of $c_1 = \mathsf{Dec}(sk, c_i)$, and then he uses esk_1 to decrypt the inner layer ciphertext, and returns $\mathcal{A}$ the answer. It is simple to see that the c_i is answered correctly.

After receiving m_0, m_1 from $\mathcal{A}$, the simulator encrypts them using the enhanced public key of the inner layer encryption E_1, sends $(m'_0, m'_1) = (E_1(epk_1, m_0), E_1(epk_1, m_1))$ to $\mathcal{C}$ and forwards the challenge c directly to $\mathcal{A}$.

The simulator continues to answer $\mathcal{A}$'s decryption queries as before. And sends $\mathcal{A}$'s guess b directly to $\mathcal{C}$ as his guess.

Since the challenge is exactly the encryption of m_0 or m_1, so the simulator would have the same advantage for $\mathcal{A}$ to distinguish encryption of m'_0, m'_1 using scheme-IV.

As the EnKey protocol is the same as in E_1, thus privacy of scheme-IV is the same as that of E_1; regarding recoverability, from the Rec algorithm we see that the ciphertext queries in scheme-IV and that in E_1 have a one-to-one correspondence, the algorithm will return a correct value as long as the Rec algorithm in E_1 can return one. □

Remark 5. *An alternative way to achieve IND-CCA2 security is to instantiate each PKE scheme in the generic construction of section 2.4.2 with a lossy trapdoor function, and apply the Peikert-Waters paradigm [109] to convert this IND-CPA secure scheme into an IND-CCA2 secure scheme by utilizing a strong one time signature. We will use only a unit building block as an example to demonstrate the idea, and the full construction can be derived straightforwardly. Using the notation from [109], $F_{ltdf}(s, x)$ denotes a lossy trapdoor function F evaluated at input x using public key s; $G_{abo}(s', vk, x)$ denotes an all-but-one trapdoor function evaluated at input x using public key s' and branch vk. For details of these two primitives, we refer to [109].*

The public keys will be $(s_0, s_1, s'_0, s'_1, h_0, h_1)$, and the secret key is t_b for a bit b, where t_b is the trapdoor corresponding to s_b, and h_0, h_1 are random universal hash functions. For Enc, the algorithm first generates a random key pair (sk, vk) for a strong one-time signature scheme, randomly selects x_0, x_1, and output the ciphertext as $c =$

$$(vk, [F_{ltdf}(s_0, x_0), G_{abo}(s'_0, vk, x_0), h_0(x_0) \oplus m], [F_{ltdf}(s_1, x_1), G_{abo}(s'_1, vk, x_1), h_1(x_1) \oplus m], \sigma),$$

where σ is a signature signed using sk on all the other components in the ciphertext. The Dec *algorithm just selects the corresponding ciphertext, and inverts $F_{ltdf}(s_b, x_b)$ using t_b to get x_b, and checks whether $G_{abo}(s'_b, vk, x_b)$ is well-formed and retrieves m by $h_b(x_b) \oplus m \oplus h_b(x_b)$. We can argue the IND-CCA2 security in a similar way to the analysis in [109], with the minor difference in the indistinguishability between lossy keys and injective keys where we use a pair of lossy trapdoor functions instead of one; similarly, for the hidden lossy branch property we use a pair of all-but-one trapdoor functions instead of one. We omit the detailed proof here. This alternative construction will yield a more efficient IND-CCA2 scheme when applied over the generic construction compared to the one we presented above. Furthermore, we can use the one-time signature paradigm [31] in the setting of identity based leakage-deterring encryption as well.*

To deal with the transformation from IND-CCA2 security with a honest authority to IND-CCA2 security with a dishonest authority without using the nested encryption, we can have the public keys in one of the pairs to be the same as the one generated by the user, for which the authority does not know the secret key. The encryption algorithm will always use this public key. It is easy to see that this "special" public key guarantees the security in a model with a dishonest authority. Note that the EnKey *protocol and the* Rec *algorithm in the above alternative construction are the same as those in the generic constructions and privacy and recoverability will not be affected.*

2.6 Leakage-deterring Signature & Identification

In this section, we design leakage-deterring signatures and identification schemes. The main idea is that we treat any functional box as an unforgeability or impersonation adversary, and take advantage of "witness extractability" in the security arguments of the underlying primitive to extract the secret-key, then unlock the private data. To

achieve this type of extractability we apply rewinding, and hence this means that a certain level of non-black-box access to the adversarial implementation is needed that was unnecessary before. Specifically, in the case of digital signatures we assume the recoverability algorithm can "hook" the hash function calls of the adversarial implementation while in the case of identification schemes the assumption is that the adversarial identification can be rewound to a previous state (something that is true for software adversarial implementations but not necessarily true in the case of a hardware adversarial implementation). We stress that our leakage-deterring constructions do not employ any additional intractability assumptions beyond the ones used in the underlying primitives.

2.6.1 Leakage-deterring signature In the random oracle model

We construct a leakage-deterring signature scheme based on a class of Σ-protocol-based signature schemes as in [114]. The security proofs of these signatures rely on the fact that if the adversary can forge one signature, then he could also forge another correlated signature for the same message with the same random tape but a different random oracle. Using these two forgeries that are correlated, one can extract the secret key of the owner.

Our construction of leakage-deterring signature is based on two independent digital signatures instances Sig_0 and Sig_1 that are unforgeable under adaptively chosen message attacks. Further, Sig_1 is required to be unforgeable in the random oracle(RO) model following [114]; specifically, the signature has the form of $(m, \sigma_1, h, \sigma_2)$ as in [114], and satisfies $h = H(m, \sigma_1)$, and σ_2 only depends on m, σ_1, h, where H is a RO. We call the following construction Scheme-V.

- $\mathsf{KeyGen}(1^\lambda)$: This algorithm executes the KeyGen algorithm of Sig_0, and returns the key pair (pk_0, sk_0).

- $\mathsf{EnKey}(O, A)$: This protocol is executed between O, A with inputs (pk_0, sk_0, s), and (pk_0, s) respectively. A runs KeyGen algorithm of Sig_1 to generate a key pair (pk_1, sk_1). The protocol terminates with O obtaining (epk, esk), and A obtaining epk, where $epk = (pk_0, pk_1, H(sk_1) \oplus s)$, and $esk = (sk_0, sk_1)$.

- $\mathsf{Sign}(esk, m)$: On input a message m, this algorithm returns the signature $\sigma = (\sigma_0, \sigma_1)$, where $\sigma_0 = \mathsf{Sign}_0(sk_0, m)$, and $\sigma_1 = \mathsf{Sign}_1(sk_1, m) = (\sigma_1^1, h_1, \sigma_1^2)$.

- $\mathsf{Verify}(epk, m, \sigma)$: On input a message-signature pair (m, σ) and enhanced public key $epk = (pk, pk')$, this algorithm returns 1 if both of the two signatures are valid, 0 otherwise.

- $\mathsf{Rec}^{\mathcal{D}}(epk, B, \delta)$: The recovering algorithm follows the security proof argument of [114]: Whenever the box B asks a random oracle query (suppose total number of such queries is bounded by q), the algorithm selects a uniform response from the range of the random oracle and feeds it to the box; it also maintains a table of all these queries. The recovering algorithm samples a message m according to $\mathcal{D}$ and simulates the box B on m. When the box outputs a valid signature σ_0, σ_1, where $\sigma_1 = (\sigma_1^1, h_1, \sigma_1^2)$, algorithm checks the table and identifies the index i of the first query from B on (m, σ_1^1). Then, it rewinds B to the state prior to the i-th query, and continues the simulation picking new random query responses.

  The above procedure is repeated until the box outputs another valid signature (σ_0', σ_1') on the same message m, where $\sigma_1' = (\sigma_1^1, h_1', \sigma_2')$, and also the index i that (m, σ_1^1) was queried is the same for both σ_1 and σ_1'.

  Now the algorithm can extract the second secret key sk_1 from $(m, \sigma_1^1, h_1, \sigma_1^2), (m, \sigma_1^1, h, \sigma_2')$ using the Σ protocol properties of the scheme that define Sig_1. The recovery of s follows immediately.

Security Analysis: We first give some brief intuition about the three properties. It is easy to see that unforgeability against adaptively chosen message attacks can be derived from the property of Sig_0 as any forgery will imply also a forgery of Sig_0. Note that signing queries are easy to simulate because the simulator has the secret key for Sig_1, and can ask signing queries to the challenger for Sig_0. *Privacy* w.r.t. a secret-key oracle for any distribution can be achieved because any successful privacy attacker will have to eventually query sk_1 to the random oracle hence violating the unforgeability of Sig_1. Note that *recoverability* cannot violate *privacy* w.r.t. an arbitrary secret key oracle, since it is achieved now via a non-black-box technique. It uses the fact that rewinding the signing box and controlling the random coins in an execution, one can always find a pair of signatures that reveal the secret key, something that yields the private data. Note that we consider only the "non-trivial" signing box works for super polynomially many messages, otherwise, it can be produced without containing the secret key. Details of the analysis are as follows:

Theorem 7. *Scheme-V is unforgeable under adaptively chosen message attacks if the underlying signature* Sig_0 *is unforgeable under adaptively chosen message attacks. It achieves privacy w.r.t. any signing oracle if* Sign_1 *is unforgeable under adaptively chosen message attacks. Also, Scheme-V achieves non-black-box recoverability w.r.t any non-negligible* δ *and any message distribution* $\mathcal{D}$ *with super logarithmic min-entropy.*

Proof. Correctness is relatively obvious, the validity of a secret key for a signature scheme can be easily verified, owner would check the validity when receiving the additional key pairs from the authority. We will only demonstrate the security properties as follows:

We first examine the unforgeability. Suppose $\mathcal{A}$ is an adversary who breaks the unforgeability of Scheme-V, and $\mathcal{C}$ is the challenger in the security game of Sig_0.

After receiving s, pk_0 from $\mathcal{A}, \mathcal{C}$ respectively, the simulator generates a random key pair (sk_1, pk_1), and runs the EnKey protocol with $\mathcal{A}$ and terminates with $\mathcal{A}$ obtaining epk, and the challenger obtaining epk, sk_1, where $epk = (pk_0, pk_1, s')$ and $s' = H(sk_1) \oplus s$.

Whenever $\mathcal{A}$ asks a signing query on m, the simulator asks such signing query to $\mathcal{C}$ to get σ_0, signs with sk_1 to get σ_1, and returns $\mathcal{A}$ with (σ_0, σ_1).

If $\mathcal{A}$ outputs a forgery (σ_0^*, σ_1^*) on m^* which is never asked for signing query, then the simulator outputs m^*, σ_0^* as his forgery to $\mathcal{C}$.

Next, we will show privacy (with secret key oracle access). We will reduce the unforgeability of $\mathsf{Sign}_1$to the privacy property.

First, the simulator $\mathcal{S}$ gets pk_1 from the unforgeability challenger $\mathcal{C}$ and creates another key pair (pk, sk), and sends pk to the adversary $\mathcal{A}$. After receiving s_0, s_1 from $\mathcal{A}$, $\mathcal{S}$ randomly chooses r and a bit b, and sends $(pk, pk', r \oplus s_b)$ as epk.

$\mathcal{S}$ can always answer the signing queries since he can sign with sk to get the first half of a signature, and asks $\mathcal{C}$ for the second half.

Note that only when one queries an input a to the random oracle, it is possible for him to predict even one bit of the output $H(a)$. If the adversary can predict at least one bit of s, that means he can predict at least one bit of r, thus he has to ask a random oracle query about sk (which is the input of the random oracle for r) at some point. The simulator simply collects all the random oracle queries of $\mathcal{A}$, and checks whether it is the secret key corresponding to pk_1, whenever sk_1 is found, the simulator stops and outputs a signature using sk_1 on a message which $\mathcal{C}$ is never asked for a signing query.

Further, we examine the recoverability. According to the notations in the general forking lemma [9], we can see that $acc = \Pr[J \geq 1] = \Pr[F_1 \wedge F_2] = \Pr[F_2] \Pr[F_1|F_2]$, where F_1 denotes the event that B makes a call to the hash function for a query m, and F_2 denotes the event that B outputs a valid signature for m. First, all messages

submitted to be signed by Rec are sampled from $\mathcal{D}$, thus B will output a valid signature for each query with probability at least δ; hence, we have $\Pr[F_2] \geq \delta$. Second, an efficient B can only store polynomially many hash values or message-signature pairs. This is seen as follows: let L be the list of messages m for which it holds that $B(m)$ outputs a signature with non-negligible probability without querying the hash function on that message. We claim that the size of this list L is is bounded by Q=poly(λ). Indeed, if the list is super-polynomial one can execute B repeatedly and obtain all valid signatures that form list L. The signatures determine a list of values of size Q from the table of the hash function that B never queries to its hash oracle. It follows that this corresponds to a superpolynomial amount of information that is encoded in the description of B. By a standard information-theoretic argument one can show that this is impossible as it suggests that the adversary that constructs the box can be used to encode a super-polynomial amount of information in a polynomial size description (from which the encoded information can still be recovered).

Now recall that $H_\infty(\mathcal{D})$ is $\omega(\log \lambda)$, thus for a random sample m from $\mathcal{D}$ the probability that m is included in the list L is at most $Q/2^{\omega(\log \lambda)}$. It follows that $\Pr[F_1|F_2] \geq 1 - Q/2^{\omega(\log \lambda)} \geq \alpha$ for some non-negligible α.

Combining the above fact with $\Pr[F_2] \geq \delta$, we obtain that $acc \geq \alpha\delta$. Following the general forking lemma, we can see that the probability of finding a successful pair of signatures on a same message in one rewinding is no smaller than $(\alpha\delta)^2/q - \epsilon$, where ϵ is a negligible function so that $1/\epsilon$ is the size of the range of random oracle, and q is the number of random oracle queries. After repeating the rewinding procedure for some polynomial in λ number of times, where λ is the security parameter, one can find a successful pair with probability $1 - \epsilon'$ where ϵ' is a negligible function. With such a

successful pair, one can extract the secret key sk_1 as the knowledge extractor does in the Σ-protocol (or as in the reduction in [114]), and outputs $s = s' \oplus H(sk_1)$. □

2.6.2 Leakage-deterring identification

We will construct a leakage-deterring identification scheme by using a similar approach as in the signature case. However here we will show our construction secure in the *standard model*, and thus we need a novel method to embed the owner private data into the enhanced public key. In fact we will need no additional assumption beyond the one employed for the underlying scheme.

Our construction of a leakage-deterring identification scheme is based on the class of identification schemes which are derived from zero-knowledge proofs of knowledge protocols that can be parallelly composed. We utilize the fact that given access to the code of any box that implements the identification functionality, one can rewind the box and implement the knowledge extractor assured to exist due to the soundness property of the zero-knowledge proof (this idea was used before to obtain the related notion of non-transferability of credentials in [30,98]). We call the following construction Scheme-VI and is based on a parameter t that we specify below.

- $\mathsf{KeyGen}(1^\lambda)$: This algorithm executes the KeyGen algorithm of the underlying identification scheme, and returns the key pair (pk, sk).

- $\mathsf{EnKey}(O, A)$: This is a protocol executed between O, A with inputs (pk, sk, s), and (pk, s) respectively. A runs the KeyGen algorithm to generate t new key pairs $(pk_1, sk_1), \ldots, (pk_t, sk_t)$, and further, A calculates $s' = r \oplus s$, where $r = Ext(sk_1 || \ldots || sk_t, \rho)$ and Ext is a strong randomness extractor (see below for implementation remarks) while ρ is the random seed. The protocol terminates with

O obtaining (epk, esk), and A obtaining epk, where $epk = (pk, pk_1, \ldots, pk_t, s', \rho)$, and $esk = (sk, sk_1, \ldots, sk_t)$.

- $\mathsf{Identify}(P, V)$: This protocol is executed between P, V with inputs (epk, esk), and epk respectively. The protocol is the parallel composition of the $t + 1$ underlying identification schemes. The protocol terminates with V outputting 1 if he accepts the proof of knowledge of all secret keys, and 0 otherwise.

- $\mathsf{Rec}(epk, B)$: Given B, Rec runs the knowledge extractor algorithm for the parallel composition of the t schemes until all the secret keys $\{sk_1, \ldots, sk_t\}$ are recovered. Then it applies the extractor on ρ and returns $s = s' \oplus Ext(sk_1 || \ldots || sk_t, \rho)$.

Security Analysis: We first sketch the security properties. *Recoverability* is essentially the same as the *recoverability* of Scheme-V. *Impersonation resistance* is also similar to the *unforgeability* property of Scheme-V; this property mainly relies on the fact that nothing related to the original secret key of the owner sk is added to the epk, therefore the security of identification using the original (pk, sk) can be reduced to the impersonation resistance of Scheme-VI. Regarding *privacy*, according to impersonation resistance, after seeing a polynomial number of transcripts of interaction between P, V, there is still unpredicatability on the secret key, (otherwise, one can impersonate by eavesdropping) then applying the strong extractor one can get pure randomness out of the secret-keys, and thus the owner data is hidden computationally. Details are given as follows:

Theorem 8. *Scheme-VI is impersonation resistant if the underlying identification schemes are impersonation resistant under parallel composition. It achieves privacy w.r.t. the secret key oracle that plays the role of the prover and performs with the adversary the identification protocol. Also, Scheme-VI achieves non-black-box recoverability w.r.t any non-negligible δ.*

Proof. Correctness follows straightforwardly from the correctness of the underlying identification scheme. Also, as explained above, *impersonation resistance* and *recoverability* are very similar to the unforgeability and recoverability of Scheme-V, thus we only focus on the proof for *privacy* here.

We first show that every secret key sk_i still has sufficient conditional unpredicatability given that the adversary adaptively makes queries for transcripts during identification.

Claim 9. *The conditional unpredicatability of each sk_i is at least $\omega(\log \lambda)$, where λ is the security parameter, if the underlying identification scheme is impersonation resistant against a passive adversary.*

Proof. of the claim: After seeing adaptively queried transcripts, if there exists an adversary $\mathcal{A}$ who can predict one of the secret keys sk_i with non-negligible probability (conditional unpredicatability is asymptotically smaller than $\omega(\log \lambda)$), then one can build a simulator which breaks the impersonation resistance of the underlying identification protocol. In more details:

Suppose $\mathcal{C}$ is the challenger in the impersonation resistance game, when the simulator receivers pk from $\mathcal{C}$, he generates $t-1$ key pairs $(pk_1, sk_1), \ldots, (pk_{i-1}, sk_{i-1}), (pk_{i+1}, sk_{i+1}, \ldots, (pk_t, sk_t)$, and sends $\mathcal{A}$ $(pk_1, \ldots, pk_{i-1}, pk, pk_{i+1}, \ldots, pk_t)$.

The simulator asks the identification transcripts for pk from $\mathcal{C}$, and for other public keys, the transcripts can be perfectly simulated since he knows the secret keys.

When $\mathcal{A}$ outputs a guess for sk_i, the simulator uses this sk_i to execute $\mathsf{Identify}$ as a prover with $\mathcal{C}$ as a verifier.

It is obvious that $\mathcal{C}$ will accept the identification attempt of the simulator on behalf of the prover with the same probability that $\mathcal{A}$ correctly outputs sk_i. □

Under the above claim, the unpredictability of all the sk_i's concatenated together is sufficiently large, therefore after applying the randomness extractor, the output r is uniform, thus the owner's private data is hidden due to the fact that the embedding works as a one-time pad. □

Remark 6. *The strong randomness extractor Ext should work on any source with sufficient conditional unpredictability along the lines of [74]. For instance, we can use the extractor derived from the Goldreich-Levin hard-core predicate [60]. Intuitively, one can think of the view (protocol transcripts adaptively queried) of the adversary as the output of a one way function on input* $\{sk_i\}$*. Using this, [60] implies an extractor of* $\log \lambda$ *bits per instance and thus* t *should be as long as* $|s|/\log \lambda$*.*

Remark 7. *If one is willing to allow additional intractability assumptions, a more compact construction for leakage-deterring signature (in the RO model) and leakage-deterring identification is also possible*[5]*. The construction would utilize two key pairs* $(pk_0, sk_0), (pk_1, sk_1)$ *and the secret information will be embedded as* $E(pk_1, s)$*, thus only* sk_1 *will be used by the recoverability algorithm. Observe now that privacy will rely on the security of the encryption scheme (and thus may require assumptions going beyond the underlying identification scheme). Furthermore reusing the same key for signing and decrypting may not always be secure and some specialized systems would need to be employed, for instance cf. [72].*

[5] We thank an anonymous reviewer for pointing this out.

2.7 Leakage-deterring Cryptography Applications

In this section we explore in more detail practical scenarios where leakage-deterring cryptosystems can be used to provide novel solutions to security problems related to sharing and transferring cryptographic functions.

Let us start with a more detailed motivating scenario: consider a user that maintains all her e-mail encrypted on a mailserver. The user is approached by someone wishing to buy all e-mails sent by the e-mail address x@y in the past, present and future. Using a regular encryption, the user may release to the attacker an implementation of her decryption function that works only if the plaintext is an e-mail sent by x@y (and rejects all other input). If the user does not care about the secrecy of the e-mails from x@y, she has no strong reason to be deterred from releasing the implementation (all her other messages can still be relatively safe assuming the implementation is sufficiently obfuscated or delivered in hardware). Using our encryption however, she is deterred: if she releases the above implementation (even in the form of a hardware token) an adverse action is guaranteed to take place (via the recoverability algorithm): her private information will be revealed. Obviously, a determined secret-key owner can always decrypt and release the plaintexts corresponding to those e-mails individually. But this has to be done one by one, at a potentially high cost. In this scenario, leakage-deterring public-key encryption ensures there is no way to optimize this operation: if one wants to provide access to his decryption he has to do it on a "per-ciphertext" basis. Within a PKI this enforces secret-key owners to practice more responsible secret-key management.

Recall privacy w.r.t. to secret key oracles (that would be the CCA flavor of our privacy property) and recoverability can not be achieved simultaneously in the general case: the two properties are mutually exclusive. Thus, one needs to choose a proper

trade-off if he wants to implement leakage-deterring public key schemes. Regarding PKE, our objective in this work is to maximize the scope of recoverability: it should work for all (even partially functional) implementations; this makes our primitive most useful from a self-enforcement perspective and necessitates the restrictions we have made in terms of the privacy property. If the user wishes the private information to remain hidden, she should provide no access to her secret-key. In the case of signature/identification schemes the situation is more tricky since by nature of the functionality, the user is expected to release signatures/identification transcripts publicly (which in some cases they may even be adaptively selected). Thus, we must compromise and weaken our recoverability property in some way. We resolved this by adopting a non-black-box recoverability algorithm. As expected, if the implementation becomes "obfuscated" then recoverability would be infeasible. We believe the trade-offs we utilized are natural for the primitives studied, but of course different tradeoffs can be possible between privacy and recoverability, and we leave them as future work.

Depending on different application scenarios, we can embed various types of private owner information to deter the leakage of a cryptographic functionality. We list three relevant scenarios below.

Self-enforcement. In the context of self-enforcement the owner of the cryptographic functionality has embedded into her enhanced public-key some private information that she normally prefers to keep secret. This can be e.g., her credit-card number or similar piece of private information as suggested by Dwork, Lotspiech and Naor [51] that introduced self-enforcement (in a different context - see related work in the introduction). In this way, when using our leakage-deterring primitives, if the owner releases any implementation of the cryptographic functionality, any recipient of the implementation will become privy to the hidden information. This property "self-enforces" the owner to

keep the functionality to herself and solves the problem of how to deter the sharing of software or hardware devices that implement cryptographic functionalities.

All-or-nothing sharing of cryptographic functionalities. In this scenario, the owner is obliged to embed the secret key of the cryptographic primitive itself into the enhanced public-key (in practice this can be done e.g., by a trusted key generator algorithm which will be running the embedding algorithm that is executed by the authority in our model). Using our techniques this means that any working implementation of the cryptographic functionality would leak the whole secret-key. In this sense, the cryptographic functionality becomes "unobfuscatable", any program that partially implements it, say for some types of inputs, can be transformed to a program that implements it perfectly. Leakage-deterring primitives used in this way suggest a type of *all-or-nothing* property for cryptographic keys: owners of a cryptographic functionality cannot partially share it, they either have to keep it to themselves or share it fully. In practice, one can expect that this is also a type of self-enforcing mechanism: either all information about the cryptographic key will be leaked or none.

Anonymity revocation from implementations. In this setting, the owner of the cryptographic functionality operates it under a pseudonym (i.e., the enhanced public-key is certified but without openly identifying the owner). However, the embedded information is ensured by the authority to be either the owner's real identity or an identity credential that the owner prefers to hide. In this setting, using our methodology, if any working implementation of the functionality is confiscated, it will be possible to use the recovering algorithm to reveal the hidden identity credential. This in turn, ensures some level of accountability: the owner remains pseudonymous as long as he does not share the cryptographic functionality but can be identified in case any (even partially working) implementation is leaked.

2.8 Leakage-deterring Identity Based Encryption

Definition of IBE: [19, 123] IBE is a public key encryption mechanism that any string can be used as a public key, while the corresponding private key of a string can be extracted by an authority. It is composed of four algorithms:

- **Setup**(1^λ): On input a security parameter, this algorithm generates a master key pair (mpk, msk).
- **Extract**(ID, msk): On input an identity ID, and the master secret key msk, this algorithm outputs a key pair (pk_{ID}, sk_{ID}) for this identity ;
- $\mathsf{Enc}(ID, m, mpk)$: On input a message m and an identity ID, this algorithm returns a ciphertext c;
- $\mathsf{Dec}(sk_{ID}, c)$: On input a ciphertext c and a secret key sk_{ID}, this algorithm returns the message m.

Regarding to 2-layer hierarchical IBE (HIBE) [58], there is an extra **Derive** algorithm, that given the secret key of any identity $ID \in \{0,1\}^t$, it derives the secret key for any identity $ID||v \in \{0,1\}^{2t}$ which has ID as a prefix. For more details about the definition of HIBE schemes, we refer to [58].

ID-based Leakage-deterring Encryption: In identity based setting, KeyGen and EnKey can be merged into the EnExtract algorithm. Details are as follows:

- **Setup**(1^λ) This algorithm generates a master key pair (mpk, msk).
- $\mathsf{EnExtract}(O, A)$ This is a protocol between a user O and a key generator A with inputs (ID, s), and $[ID, (mpk, msk), s]$ respectively. The protocol terminates with both parties obtaining the enhanced key pair $(epk, esk) = ((pk_{ID}, s'), sk_{ID})$ for the input identity.

- $\mathsf{Enc}(ID, m, mpk)$ On inputs message m and identity ID's enhanced public key, this algorithm returns a ciphertext c;

- $\mathsf{Dec}(sk_{ID}, c)$ On input ciphertext c and secret key sk_{ID}, this algorithm returns the message m.

- $\mathsf{Rec}^{B,\mathcal{D}}(epk, \delta)$ Given oracle access to a decryption box B, and a message distribution $\mathcal{D}$, which B has δ-correctness on, and with input the enhanced public key for a certain identity, this algorithm returns s or $\perp$.

ID-IND-CPA Security (for ID-based leakage-deterring encryption): The IND-CPA security for identity based leakage-deterring encryption is slightly different with that of leakage-deterring PKE which is defined in 2.3.2, as the authority is always assumed to be honest in this setting. Details are described in the following game between the adversary and the challenger:

- The challenger runs the Setup algorithm and returns mpk to the adversary.

- The adversary adaptively chooses identities $ID_1, \ldots, ID_q$, and $s_1, \ldots, s_q$ as the private data for each identity, and then interacts with the challenger in the $\mathsf{EnExtract}$ protocol to get enhanced public and secret keys for these identities.

- The adversary chooses two messages m_0, m_1, a target identity ID which was not queried for secret key before, and a string s^*, and sends them to the challenger.

- The challenger randomly choose a bit b, and sends $c = \mathsf{Enc}(ID, m_b, mpk)$ and epk to the adversary, where epk is the enhanced public-key of the user that corresponds to ID with private information s^*.

- The adversary outputs his guess b'.

We say an ID-based leakage-deterring encryption is ID-IND-CPA secure if $\Pr[b' = b] \leq 1/2 + \epsilon$ in the above game, where ϵ is a negligible function. If the adversary is required to claim the target identity and the string in the beginning (before the secret key queries), we call it selective-ID-IND-CPA secure. Furthermore, if decryption queries are allowed for the target identity (with the exclusion of the challenge ciphertext) before the adversary outputs his guess, we call it ID-IND-CCA2 secure.

We note that the enhanced public-key *epk* will not be used for encryption, to be compliant with the id-based nature of the primitive. However, the *epk* still carries the private information of the user and is assumed to be a public-value that is available to anyone. The only time that this value is relevant is in the operation of the recoverability algorithm that may happen only after a leakage incident takes place.

As in remark 4 at the end of section 2.4.2, following the generic construction of leakage-deterring PKE, we can similarly construct an identity based leakage-deterring encryption scheme by generating exponentially many secret keys for one identity, and using the "indicating string" to select corresponding keys as in the generic construction. We omit the detailed description of the construction here.

2.9 Conclusions and Open Problems

We introduced the notion of leakage-deterring cryptosystems. Our schemes have the property that whenever an owner releases an (even partially) "functional" box for others to use instead of herself, anyone who has access to the box can recover some private information that is embedded into the public-key of the owner. We defined the security properties of these primitives and we provided several constructions for public key encryption, signatures, identification.

Since this is the first step in the formal investigation of such primitives, several interesting open questions remain. A natural question is how to combine the notion with traitor tracing and other multi-user oriented cryptosystems. Another direction is with respect to CCA2 security: our construction can potentially be optimized for efficiency and avoid the nesting of two encryptions. A third direction is to see to what extent it is feasible to construct leakage-deterring signatures and identification with black-box recoverability in the standard model or more generally explore the tradeoff between recoverability and privacy in a comprehensive fashion. Last but not least, it would be desirable to see how the trust to the authority can be reduced (and e.g., obviate the need for the authority to know the secret information).

Chapter 3

Traitor Deterring Schemes

3.1 Introduction

We put forth and construct a new primitive we call a *traitor deterring scheme* (TDS): a multi-recipient encryption scheme where each recipient (henceforth also called a user) has some secret information that is provided as a collateral and hidden in a public directory. If the user is honest and keeps her key to herself her secret information remains hidden. On other hand, if some users collude to produce an unauthorized decryption device, any recipient of the device will be able to recover one of the colluders' collateral secret information.

TDS strictly strengthens the notion of a traitor tracing scheme (TTS) that was introduced by Chor, Fiat, Naor [36] and further studied in numerous works cf. [20–22, 25, 33, 56, 87, 90]. TTS's are multi-user encryption schemes that when some users (aka traitors) collude to produce an unauthorized decryption device (aka a pirate decryption

box), it is possible to recover at least one of their identities. TTS's de-incentivize piracy, in the sense that colluders may be identified once an unauthorized device is detected.

A TDS provides a stronger mechanism for de-incentivizing piracy. Observe that in a TTS, recovering the identity of a traitor can only happen when the authority becomes aware of the unauthorized decryption device. This means that if the traitors operate stealthily there is nothing the authority can do to deter them. Furthermore, the penalty that the authority may inflict to the traitors (e.g., canceling their subscription or increasing their subscription fee) can only be applied "after-the-fact", i.e., only after the unauthorized decoder has been recovered and analyzed by the authority. In contrast, in a TDS a penalty can be inflicted immediately when a user becomes a traitor and shares her decryption key with the adversary. Indeed, given a user key in a TDS, the adversary can recover the users' hidden collateral that is -presumably- of some value and the user prefers it to be hidden (e.g., it can be a user's credit card number). Of course, additional penalties may be applied after the authority notices the pirate decryption device, exactly as in a TTS.

Compared to TTS's, the main difficulty of constructing a TDS is that one needs to enable a public recovering procedure which returns the user's secret information which is an element of an exponentially sized domain — in other words linear number of bits in the security parameter λ need to be extracted from the pirate box. Contrary to that, in a TTS, the authority only needs to recover the identity of a traitor, which is an element of merely a polynomially sized domain — in other words just logarithmic number of bits in the security parameter λ need to be extracted from the pirate box. As in TTS, the recovering procedure should work given only black-box access to the pirate decryption box which may be only partially working and furthermore, it should operate without utilizing any private-key information, as in a TTS with public traceability [33].

A TDS (or a TTS) can also be considered in a stronger adversarial model that we call "the known ciphertext model." In this model the adversary aims at communicating a pirated copy consisting of a sequence of plaintexts that corresponds to a given set of (polynomially many) ciphertexts; without loss of generality we can assume the pirate copy is in the form of a pirate box that acts on the known sequence of ciphertexts. The adversary aims at producing a pirate box of smaller size than the sequence of plaintexts; we capture this in the model by requiring the "space rate" of the attacker to be $o(1)$. This problem was first considered by Dwork, Lotspiech and Naor [51] that proposed an $o(1)$ space rate solution assuming non-black-box recoverability and under an unfalsifiable assumption [103] (specifically, the existence of incompressible functions of a specific type). Constructing a TDS or a TTS in the known ciphertext model under a falsifiable assumption is an open question.

Our contributions. We formalize traitor deterring schemes and we give two different construction methods that we instantiate in various ways; further, we formalize the known-ciphertext model in the spirit of [51] and we provide both feasibility and infeasibility results for TDS in this model. In more detail:

1. We put forth the formal model for TDS. Such schemes enable the embedding of hidden user-specific information in a public parameter, and have three security properties: (i) security of plaintexts which is formalized in a standard fashion as in public-key encryption; (ii) privacy of user information that is hidden in the public parameters. This property should be upheld even if all other users conspire against a user as long as the secret key of the user is not compromised; finally, (iii) traitor deterring suggests that there is a recoverability algorithm that given black-box access to some working implementation of a decryption device, it is

capable of recovering the private information of at least one traitor, using only public information.

2. We give two construction methods for TDS's. The first one is based on fingerprinting codes and a new primitive we call a fuzzy locker. In a fuzzy locker, the message is encrypted using a random codeword C_i of a fingerprinting code [36, 83] as the key; the decryption operation returns the message given any codeword C^* that would result in the i-user being accused in the underlying fingerprinting code. In the TDS construction, the recovering procedure will first retrieve a pirate codeword C^* from the decryption device, and the traceability of the fingerprinting code will guarantee that one of the collusion's codewords will be identified. We then give a concrete construction of a fuzzy locker for CFN codes [36] using fuzzy extractors [50] paired with efficient list decoding for Reed-Solomon codes [71, 126]. Our second construction method for TDS's generalizes the constructions of [22, 25] that are based on comparison predicate encryption (CPE). Contrary to these works however, for our method to work, we require that the user identity space is exponentially large and sparse, so that a randomly chosen user identity can be used as a secret key to hide the user secret information directly. To recover the hidden information given a pirate decryption decoder we utilize a binary search type of recovering mechanism to navigate through the exponentially sized domain and discover one of the traitor identities. Given this identity we proceed to unlock the user hidden data via trial and error. A CPE scheme can be directly obtained via functional encryption (FE) using indistinguishability Obfuscation (iO) [55]. In order to obtain a construction based on standard assumptions we resort to bounded collusion FE [63, 65]. We provide a more efficient construction for this

primitive via a combinatorial argument and we then use it to instantiate a CPE with exponential size domain. Our TDS constructions are summarized in Fig. ??.

3. We revisit the problem of digital signets and self-enforcement [51] and we formulate the "known ciphertext model" for TDS where the adversary knows the target set of (polynomially many) ciphertexts before implementing the pirate box. In an attack in this model, the adversary tries to achieve a favorable "space rate", i.e., produce a decryption box that has size smaller than the total plaintext material that is encoded in the known ciphertexts without leaking any of the traitors' collaterals. Constructing a TDS in the known ciphertext model is equivalent to the problem of constructing digital-signets with self-enforcement as defined in [51] which is open under falsifiable assumptions; the construction of [51] assumes an *incompressible function* of a specific type (this is an unfalsifiable assumption) and works for any space rate $o(1)$. With our TDS constructions we resolve the open question showing feasibility under falsifiable assumptions for space rates $O(\log \lambda/\lambda)$ while we show infeasibility for space rates $\Omega(\log^2 \lambda/\lambda)$ in the black-box recoverability setting. It remains still open whether the weaker non-black-box recoverability setting of [51] can allow for higher space rates; however we warn that such "white-box" recoverability is too weak of a property for TDS's since it provides no solid guarantee that the collateral hidden traitor data will be made public (hence the open question that remains seems to be of less relevance to practice). In our result, we exploit bounds on the false positive rate of the membership testing problem to show how our TDS schemes can be used while our negative result applies Bloom filters [15] to provide an efficient attacker strategy (in this way we resolve the open question circumventing incompressible functions altogether).

	Assumption	Ciphertext size	Upper bound on t	Recoverability
Construction I	PKE	$O(t^2 \log^2(n/\epsilon))$	$O(\log(n/\epsilon)/\lambda)$	Black-box
Construction I	PKE	$O(t^4\lambda)$	n	Black-box
Construction II	LWE	$O(t^{3+e}\text{poly}(\lambda))$	n	Black-box
Construction II	iO	$O(1)$	n	Black-box

Figure 1: Comparison of our TDS's; t is the collusion size, n is total number of users, $e = 1/\text{poly}(\lambda)$, ϵ is the error term in the fingerprinting code which is $\mathsf{negl}(\lambda)$ and λ is the security parameter. PKE denotes public-key encryption, LWE denotes the learning with errors problem, and iO denotes general indistinguishability obfuscation.

4. We finally describe how one can use Bitcoin as a collateral in a TDS deployment. Given the power of TDS's, the idea is simple: since the collateral can be an arbitrary string we can have the SP store as collateral the secret-key that corresponds to a bitcoin address of the user. For the purpose of the subscription that corresponds to the service the account should remain frozen (i.e., no outgoing transaction should be made from this account otherwise the subscription will be cancelled). As long as the user respects the service agreement the collateral remains safe and the user may reclaim it when the service contract terminates.

Related primitives. As discussed above, a TTS aims at providing "a posteriori" deterrence of malicious users while TDS provides, in addition, a proactive way of deterrence. Furthermore, traitor tracing is possible only when the authority gains access to the pirate box, while in a TDS, the mere act of sharing a decryption key (or any implementation of a decryption algorithm containing such key) will lead to the leakage

of one of the traitors' secrets. We show that a traitor deterring scheme implies a publicly traceable traitor tracing scheme [33] (cf. Section 4.3.1.1).

Another very related notion to TDS is digital signets for self-enforcement proposed by Dwork, Lotspiech and Naor [51]. In this multi-user encryption system, the adversary that controls a set of traitor user keys and wants to retransmit a certain plaintext that was transmitted, will either send a message as long as the plaintext itself, or will have to leak some of the traitors' private data. The formalization of the problem in [51] assumes that recoverability of the collateral information requires direct access to the traitor keys (also called white-box access). In our terminology, they provide a symmetric-key TDS that is only secure in the non-black-box sense. The construction provided by [51] relies on the unfalsifiable assumption that $f(x) = g_1^x||g_2^x||\ldots||g_\ell^x$ is incompressible (incompressible means given x, f, no adversary can come up with an intermediate value y, such that: (1). $|y|/|f(x)| = o(1)$; (2). one can recover $f(x)$; (3). x remains hidden).

Kiayias and Tang studied the problem of leakage deterring (LD) public key cryptography [85] in which if a key owner leaks any partial implementation of her decryption box, a piece of secret information that is embedded in her public key will be revealed to the recipient. Our notion of TDS is a generalization of LD from the single user setting to the multi-user setting. We note that because of collusion attacks in the multi-user setting the techniques for recoverability from [85] are not directly applicable for building a TDS (even a scheme with ciphertext size linear in the number of users is not obvious).

3.2 Preliminaries

3.2.1 Definitions

Definition 4. *A comparison predicate encryption is a predicate encryption for comparison predicate P, where $P(x, y) = 1$ if $x \leq y$. It is composed of four algorithms:*

- *CPE.Setup* *This algorithm inputs security parameter and outputs a master key pair* (mph, msk).

- *CPE.KeyGen*(msk, v) *This algorithm takes master secret key, and an attribute* v *as inputs and outputs a secret key* sk_v.

- *CPE.Enc*(mph, x, m) *This algorithm takes master public key, message* m *and an attribute* x *as inputs and it outputs a ciphertext* c_x.

- *CPE.Dec*(sk_v, c_x) *This algorithm takes a secret key* sk_v *and a ciphertext* c_x *as inputs, it outputs* m *or* $\perp$.

A comparison predicate encryption satisfies the following properties:

Correctness. If $x \leq v$, then $\mathsf{CPE.Dec}(sk_v, c_x) = m$;

Payload hiding. This property is essentially IND-CPA security (when the adversary does not hold any key which can decrypt the challenge ciphertext) for the message of CPE schemes. Consider the following game between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- The challenger $\mathcal{C}$ runs the $\mathsf{CPE.Setup}$ algorithm and sends $\mathcal{A}$ the master public key mpk.

- $\mathcal{A}$ selects arbitrary number of attributes $v_1, \ldots, v_q$ and asks for the secret keys; $\mathcal{C}$ sends $sk_{v_1}, \ldots, sk_{v_q}$ to $\mathcal{A}$, where $sk_{v_i} = \mathsf{CPE.KeyGen}(msk, v_i)$;

- The adversary $\mathcal{A}$ sends two messages m_0, m_1 and an attribute x to $\mathcal{C}$, and the challenger randomly selects a bit b, computes $c_x^b = \mathsf{CPE.Enc}(mph, x, m_b)$ and returns c_x^b as the challenge ciphertext.

- $\mathcal{A}$ chooses other set of attributes and requests keys. At the end, $\mathcal{A}$ outputs her guess b'.

We require in the above game that for all $\{v_i\}$ attributes she used to request keys, $x > v_i$. $\mathcal{A}$ wins if she guesses correctly with probability close to 1/2, i.e., the advantage of $\mathcal{A}$ in the payload hiding game $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{ph}} := |\Pr[b' = b] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$.

Attribute hiding. We also require the ciphertext hides the attribute in a IND-CPA fashion, even if the adversary has keys allowing her to decrypt the challenge ciphertext. Consider the following game between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- The challenger $\mathcal{C}$ runs the $\mathsf{CPE.Setup}$ algorithm and sends $\mathcal{A}$ the master public key mpk.
- $\mathcal{A}$ selects arbitrary number of attributes $v_1, \ldots, v_q$ and asks for the secret keys; $\mathcal{C}$ sends $sk_{v_1}, \ldots, sk_{v_q}$ to $\mathcal{A}$, where $sk_{v_i} = \mathsf{CPE.KeyGen}(msk, v_i)$;
- The adversary $\mathcal{A}$ sends two messages m_0, m_1 and two attributes x_0, x_1 to $\mathcal{C}$, and the challenger randomly selects a bit b, computes $c^b_x = \mathsf{CPE.Enc}(mph, x_b, m_b)$ and returns c^b_x as the challenge ciphertext.
- $\mathcal{A}$ chooses other set of attributes and requests keys. At the end, $\mathcal{A}$ outputs her guess b'.

We require $m_0 = m_1$ in the above game that if there exists v_i that the adversary requested a key for and $x \leq v_i$; and also we require that for each of the attributes $v_1, \ldots, v_q$ it is either larger than $\max(x_0, x_1)$ or smaller than $\min(x_0, x_1)$. These two requirement are to rule out the trivial attacks. $\mathcal{A}$ wins if she guesses correctly with probability close to 1/2, i.e., the advantage of $\mathcal{A}$ in the payload hiding game $\mathsf{Adv}^{\mathcal{A}}_{\mathsf{ph}} := |\Pr[b' = b] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$.

Remark: the two definitions can be merged into one as attribute hiding defined as above implies payload hiding, we describe them separately for the ease of later presentation.

Approximate membership testing problem. We also provide the definition of membership testing problem used in the known ciphertext model TDS.

Definition 5. [15, 32] *For a subset V randomly chosen from a universe U, the approximate membership testing problem with a false positive probability η is to produce a data structure T such that, for a random element $x \in U$, if $x \in V$, $T(x)$ always outputs 1, while if $x \notin V$, $T(x)$ outputs 0 with probability at least $1 - \eta$ (i.e., it may output 1 with probability at most η, a false positive).*

3.2.2 Building blocks.

First, we put the simple parallel repetition construction of q-query secure FE from a 1-query secure FE. Given a 1-query secure FE, $\langle \mathsf{OneFE.Setup}, \mathsf{OneFE.KeyGen}, \mathsf{OneFE.Enc}, \mathsf{OneFE.Dec} \rangle$, a q-bounded FE is as follows:

- $\mathsf{Setup}(\lambda)$: This algorithm inputs the security parameter λ and runs $\mathsf{OneFE.Setup}$ N times to generate $(MPK, MSK) = \{(mpk_1, msk_1), \ldots, (mpk_q, msk_q)\}$, it returns (MPK, MSK).

- $\mathsf{KeyGen}(MSK, C, i)$: This algorithm takes the master secret key MSK, a circuit C and a counter i as inputs. It computes $sk_{C,i} = \mathsf{OneFE.KeyGen}(msk_i, C)$ and returns $sk_C = (sk_{C,i}, i)$ and updates $i = i + 1$.

- $\mathsf{Enc}(MPK, m)$: This algorithm takes the master public key, and message m as inputs. It simply encrypts m under each master public key, i.e., $CT = CT_1, \ldots, CT_q$, where $CT_j = \mathsf{OneFE.Enc}(mpk_j, m)$.

- $\mathsf{Dec}(sk_C, CT)$: This algorithm takes a secret key sk_C and ciphertext CT as inputs. It picks the corresponding ciphertext and runs the decryption algorithm of the 1-query secure FE decryption algorithm, i.e., it returns $C(m) = \mathsf{OneFE.Dec}(sk_{C,i}, CT_i)$.

We see that if the underlying 1-secure FE is succinct [63], then the resulting q-secure FE is with ciphertext size $O(q \cdot \text{poly}(\lambda))$.

In order to apply this construction to our traitor deterring scheme, we have to choose $q = n$, where n is the total number of users. To see this, if any two keys sk_{C_1}, sk_{C_2} are generated using a same master secret key, say msk_1, then no security can be guaranteed for the first 1-query secure FE instance, and thus the whole resulting CPE. Whenever $q \leq n$, there must exist a pair of users, their secret keys are generated using the same master secret key. If such two users collude, we can not ensure the security of the TDS.

We also recall the bounded collusion FE from [65]. This construction is stateless. They can guarantee security as long as there are no more than q keys corrupted together, even if there are more than q secret keys issued. Given a 1-query secure FE, $\langle \mathsf{OneFE.Setup}, \mathsf{OneFE.KeyGen}, \mathsf{OneFE.Enc}, \mathsf{OneFE.Dec} \rangle$, a stateless q-bound FE is as follows:

- $\mathsf{Setup}(\lambda)$: This algorithm inputs the security parameter λ and runs $\mathsf{OneFE.Setup}$ N times to generate $(MPK, MSK) = \{(mpk_1, msk_1), \ldots, (mpk_N, msk_N)\}$, it returns (MPK, MSK) and parameters $para = (d, D, N, S, v)$.

- $\mathsf{KeyGen}(MSK, C_i)$: This algorithm takes the master secret key MSK and a circuit C_i as inputs. It first randomly selects a subset $\Gamma_i \subset [N]$ with size $dD + 1$, and a subset $\Delta_i \subset [S]$ with size v. It then generates the secret key $\{sk_{G_i,j}\}_{j \in \Gamma_i}$, where $sk_{G_{i,j}} = \mathsf{OneFE.KeyGen}(msk_j, G_i)$, and the circuit G_{i,Δ_i} is

defined as $G_{i,\Delta_i}(x_1,\ldots,x_\ell,r_1,\ldots,r_S) = C_i(x_1,\ldots,x_\ell) + \sum_{i\in\Delta_i} r_i$. It returns $sk_{C_i} = (\Gamma_i, \Delta_i, \{sk_{G_i,j}\}_{j\in\Gamma_i})$

- $\mathsf{Enc}(MPK, para, m_1,\ldots,m_\ell)$: This algorithm takes the master public key, public parameters and a message vector $\langle m_1,\ldots,m_\ell\rangle$ as inputs. It first generate ℓ random degree-d polynomials $\mu_1,\ldots,\mu_\ell$ satisfying $\mu_i(0) = m_i$, and then it does a $d+1$-out-of-N secret sharing on each of the messages. Also, the algorithm selects S random polynomials $\zeta_1,\ldots,\zeta_S$ with degree dD satisfying $\zeta_i(0) = 0$. The encryption algorithm returns the ciphertext $CT = \{CT_i\}$, where $CT_i = \mathsf{OneFE.Enc}[mpk_i, (\mu_1(i)),\ldots,\mu_\ell(i)), (\zeta_1(i),\ldots,\zeta_S(i))]$ for $i = 1,\ldots,N$.

- $\mathsf{Dec}(sk_{C_i}, CT)$: This algorithm takes a secret key sk_{C_i} and a ciphertext CT as inputs. The algorithm first runs the decryption algorithm of the 1-query secure FE on corresponding ciphertext that it has a secret key, i.e., for each $j \in \Gamma_i$, it runs $\mathsf{OneFE.Dec}(sk_{G_i,j}, CT_i) = g_j$, then use those values $\{g_j\}_{j\in\Gamma_i}$ to interpolate and returns $g_0 = C_i(m_1,\ldots,m_\ell)$.

Observe that for each $i \in \Gamma$, $g_i = G_{C,\Delta}(x_1,\ldots,x_\ell,r_1,\ldots,r_S) = C(\mu_1(i),\ldots,\mu_\ell(i)) + \sum_{j\in\Delta}\zeta_j(i)$ and this corresponds to polynomial $\eta(\cdot) = C(\mu_1(\cdot),\ldots,\mu_\ell(\cdot)) + \sum_{i\in\Delta}\zeta_i(\cdot)$ which is with degree dD, because the inner layer $\mu_i(\cdot)$ is with degree d and the outer layer $C(\cdot)$ is with maximum degree D and $\zeta_i(\cdot)$ also with degree D. With $dD+1$ values g_i, one can do a standard interpolation to obtain $g_0 = C(\mu_1(0),\ldots,\mu_\ell(0)) + \sum_{i\in\Delta}\zeta_i(0) = C(m_1,\ldots,m_\ell)$.

For security, as 1-query secure FE is used as the underlying scheme, if colluded users obtains secret keys from one same master secret key, say msk_i, then the security of such FE instance is broken; we have to be sure that t colluded users can not break $d+1$ such

instances so that the secret sharing with still guarantees security. This condition will put a first restriction that $|\cup_{j\neq i}(\Gamma_i \cap \Gamma_j)| \leq d$.

The purpose of using the random polynomials $\{\zeta_i(\cdot)\}$ is to randomize the outcome of $\mathsf{OneFE.Dec}(sk_C, (CT_i))$ so that each ciphertext containing the shares can be simulated properly. This requires that there exists at least one ζ_i appears only for one secret key. More specifically, we require that for every $i \in [q]$, it satisfies that $\Delta_i \backslash \cup_{j\neq i} \Delta_j \neq \emptyset$.

In [65], they chose parameters $d = \Theta(q^2\lambda)$, $N = \Theta(D^2q^2d)$, $v = \Theta(\lambda)$ and $S = \Theta(vq^2)$, thus the resulting ciphertext size for the q-bound FE is $\Theta(N \cdot S \cdot \mathrm{poly}(\lambda)) = \Theta(q^6D^2\mathrm{poly}(\lambda))$ even if the underlying 1-secure FE is succinct.

3.3 Definition and Security Modeling for TDS

We provide the formal definition and security model of traitor deterring schemes and demonstrate their relationship to traitor tracing schemes.

Syntax of traitor deterring schemes. Informally, a TDS is a multi-user encryption scheme with a deterring mechanism such that if some of the receivers collude to leak an implementation of a (potentially only partially working) decryption device, any recipient of the device will be able to recover one of the colluding user's secret information which is hidden in the public parameter of the system. Formally we have the following:

- $\mathsf{Setup}(1^\lambda, s_1, \ldots, s_n)$: This algorithm takes the security parameter, users' secrets $s_1, \ldots, s_n \in \{0,1\}^\lambda$ as inputs, and outputs system parameter $para$, an encryption key pk, and a set of decryption keys $sk_1, \ldots, sk_n$.

- $\mathsf{Enc}(pk, m)$: This algorithm inputs $para, pk$ and a message m, and outputs ciphertext c.

- $\mathsf{Dec}(sk_i, c)$: This algorithm takes $para, pk$, one of the secret keys sk_i and a ciphertext c as inputs, and outputs m.

- $\mathsf{Rec}^{B,\mathcal{D}}(pk, para)$: This algorithm takes $para, pk$ as inputs and has oracle access to a device B and a distribution $\mathcal{D}$. The algorithm returns a string in $\{0,1\}^\lambda$ or the symbol $\perp$.

The correctness of the scheme is standard and entails that $\mathsf{Enc}(pk, \cdot)$ can be inverted by $\mathsf{Dec}(sk_i, \cdot)$ for any $i = 1, \ldots, n$. The intended functionality of the algorithm Rec is that if B is created by a collusion of receivers with secret keys $sk_{i_1}, \ldots, sk_{i_t}$ and operates correctly for ciphertexts whose corresponding plaintext follows a distribution $\mathcal{D}$, the algorithm outputs at least one of the strings $s_{i_1}, \ldots, s_{i_t}$. We clarify the conditions under which this is supposed to happen (as well as the other necessary security properties) in the next paragraph.

Security model. There are three security requirements for a traitor deterring scheme, *security* of plaintexts, *privacy* of user's secrets, and *traitor deterring*.

IND-CPA security. Regarding security of plaintexts we consider a security property of IND-CPA defined in a standard fashion: the challenger $\mathcal{C}$ runs setup for any $s_1, \ldots, s_n$ and provides the adversary $\mathcal{A}$ with $para, pk$. In response, $\mathcal{A}$ provides two plaintexts m_0, m_1 to $\mathcal{C}$. Subsequently $\mathcal{C}$ computes $\psi = \mathsf{Enc}(pk, m_b)$ for a random $b \in \{0,1\}$ and provides ψ to $\mathcal{A}$. $\mathcal{A}$ returns a bit b' and $\mathcal{C}$ terminates with 1 if $b = b'$ and 0 otherwise. The probability that $\mathcal{C}$ returns 1 means that $\mathcal{A}$ is successful and we denote it by $\mathsf{Succ}_{\mathcal{A}}^{\mathtt{indcpa}}(1^\lambda)$. For security to hold it must be that $\Pr[\mathsf{Succ}_{\mathcal{A}}^{\mathtt{indcpa}}(1^\lambda)] \leq 1/2 + \mathsf{negl}(\lambda)$. The notion of security can be extended in a straightforward manner to IND-CCA2.

Privacy. Regarding the privacy of user secret information it should be the case that each s_i value remains hidden within the public parameter even all other users are corrupted. Formally, consider the following game:

- The adversary $\mathcal{A}$ sends an index i as well as private information $s_1, \ldots, s_{i-1}$, $s_{i+1}, \ldots, s_n$ and the pair $s_{i,0}, s_{i,1}$ to the challenger.
- The challenger randomly selects a bit b, and simulates the Setup algorithm on input $s_1, \ldots, s_{i-1}, s_{i,b}, s_{i+1}, \ldots, s_n$. It sends to $\mathcal{A}$ the values $para, pk, sk_1, \ldots, sk_{i-1}, sk_{i+1}, \ldots, sk_n$.
- $\mathcal{A}$ returns a single bit b' and $\mathcal{C}$ returns 1 if and only if $b = b'$.

The event that $\mathcal{C}$ returns 1 means $\mathcal{A}$ is successful and we denote it by $\mathsf{Succ}_{\mathcal{A}}^{\mathtt{priv}}(1^\lambda)$. For privacy of secret information to hold it must be that $\Pr[\mathsf{Succ}_{\mathcal{A}}^{\mathtt{priv}}(1^\lambda)] \leq 1/2 + \mathsf{negl}(\lambda)$ for any PPT adversary $\mathcal{A}$.

It is also possible to define a weaker variant of the above definition where the adversary $\mathcal{A}$ is restricted to a number t of secret-keys.

Traitor-deterring. Finally we define the traitor deterring property. In order to specify the definition we need first to define the notion of δ-correctness with respect to a public-key pk and a plaintext distribution $\mathcal{D}$. A device B is $\delta-$correct with respect to $\mathcal{D}$ and pk if it satisfies that $\Pr[B(\mathsf{Enc}(pk, m)) = m : m \leftarrow \mathcal{D}] \geq \delta$. With the public parameter, and a non-trivial pirate decryption box B which is created by the collusion of all users, the recovering algorithm should determine of the colluder's secret information s_i. Formally, consider the following game:

- The challenger $\mathcal{C}$ simulates the Setup algorithm and the adversary $\mathcal{A}$ receives pk. $\mathcal{A}$ then provides a vector of secret information $s_1, \ldots, s_n$ as well as an arbitrary

subset $T \subseteq \{1, \ldots, n\}$ to the challenger $\mathcal{C}$ and $\mathcal{A}$ receives the secret keys of all users in T, $\{sk_i \mid i \in T\}$ as well as the public parameter $para$.

- $\mathcal{A}$ outputs an implementation B and a distribution $\mathcal{D}$.
- $\mathcal{C}$ returns 1 if and only if $\mathsf{Rec}^{B,\mathcal{D}}(pk, para) \notin \{s_i \mid i \in T\}$.

We define by $\mathsf{Succ}^{\mathtt{deter}}_{\mathcal{A}}(1^\lambda)$ the event that $\mathcal{C}$ returns 1. We say a scheme achieves fully collusion resilient, black-box traitor deterring w.r.t. a class of distributions $\mathfrak{D}$ (that may depend on δ) if for any PPT adversary $\mathcal{A}$ it holds that

$$\Pr[\text{B is } \delta\text{-correct w.r.t. } \mathcal{D} \wedge \mathcal{D} \in \mathfrak{D} \wedge \mathsf{Succ}^{\mathtt{deter}}_{\mathcal{A}}(1^\lambda)] = \mathsf{negl}(\lambda).$$

In the above experiment we assume that Rec has resettable black-box access to B. Even though we do not consider them here, weaker variants of the above formulation may be relevant in some settings and can be "t-collusion resilient (as opposed to fully-collusion resilient) or they may extend Rec algorithm's access to B (e.g., in a non-black-box setting Rec may have access to the traitor keys).

Definition 6. *$\langle \mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rec} \rangle$ is a (fully-collusion resistant, black-box) traitor deterring scheme if it satisfies, (i) correctness, (ii) IND-CPA security, (iii) privacy and (iv) fully-collision resistant, black-box traitor deterring.*

TDS and TTS. We conclude the section by a brief argument that a traitor deterring scheme is a strict generalization of a traitor tracing scheme (in fact of a traitor tracing scheme with "public-traceability" [33]). Given a traitor deterring scheme $\langle \mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Rec} \rangle$, the reduction is easy with the following simple observation. First we set $s_i = i$ for all $i = 1, \ldots, n$. It follows that the Setup algorithm requires no other input other than the security parameter λ. Observe now that the Rec algorithm will output

one of the indices of the colluding users who jointly produce the decryption box B with only access to pk, hence it is a traitor tracing scheme with public-traceability.

3.4 Traitor Deterring from Fingerprinting Codes

In this section, we will present our first technique of constructing a TDS from fingerprinting codes. We first formalize a new encryption scheme we call fuzzy locker (w.r.t a fingerprint scheme), from which together with a public key encryption, we will construct a TDS. We then give a concrete construction of fuzzy locker for the CFN codes [36].

First, let us recall the definition of fingerprinting codes [83]. A *q-ary fingerprinting code* is a pair of algorithms $(\mathsf{Gen}, \mathsf{Accuse})$, where Gen is a probabilistic algorithm with input a security parameter ϵ and two numbers n, t denoting the number of users and the maximum collusion size respectively, and $t \in [n] = \{1, \ldots, n\}$. It outputs n q-ary strings $\mathcal{C} = \{C_1, \ldots, C_n\}$ (called codewords), where $C_i = \mathsf{c}_1^i \ldots \mathsf{c}_\ell^i$ for $i \in [n], j \in [\ell], \mathsf{c}_j^i \in Q$–the alphabet set with size q and a tracing key tk. Accuse is a deterministic algorithm with input a "pirate" codeword C^*, and a user codeword C_i (sometimes a tracing key tk) and output in $\{0, 1\}$.

Suppose adversary $\mathcal{A}$ corrupts up to t users (whose indices form a set $\mathcal{U}_{cor} \subset [n]$), and outputs a pirate codeword $C^* = \mathsf{c}_1^* \ldots \mathsf{c}_\ell^*$. We define the accused user set as $\mathcal{U}_{acc} = \{i \in [n] : \mathsf{Accuse}(tk, C^*, C_i) = 1]$. A fingerprinting code is called $t-$*collusion resistant* (*fully collusion resistant* if $t = n$) if it satisfies: (i) *traceability*, if the strategy of producing C^* satisfies the "marking assumption", (for each $i \in [n], \mathsf{c}_i^* = \mathsf{c}_i^j$ for some $j \in \mathcal{U}_{cor}$), then one of the colluders must be accused, i.e., $\Pr[\mathcal{U}_{acc} \cap \mathcal{U}_{cor} = \emptyset] \leq \epsilon$; and

(ii) *soundness*, the probability that an innocent user is accused is bounded by ϵ, i.e., $\Pr[([n] - \mathcal{U}_{cor}) \cap \mathcal{U}_{acc} \neq \emptyset] \leq \epsilon$. [1]

3.4.1 TDS from fuzzy lockers.

Fingerprinting codes are combinatorial designs that enable testing whether a codeword is used in generating a pirate codeword. They were demonstrated to be very useful in building TTS in a line of previous works, e.g., [21, 36, 87]. The basic idea is that each user will be assigned an "identity" which is represented by a codeword, and the secret keys for the user will be selected from a group of keys according to his codeword. The encryption algorithm will cover all the user keys. And the tracing algorithm will first recover a "pirate codeword" by feeding the pirate decryption device with malformed (but seemingly valid in the view of $\mathcal{A}$) ciphertext, and then it will run the tracing algorithm of the fingerprinting code to identify at least one of the colluded users who participated in producing the pirate codeword.

The main challenge of upgrading the above paradigm to a TDS is the way of embedding and recovering of the secret information of the users. To address this, we formalize a new primitive we call fuzzy locker w.r.t a (publicly traceable) fingerprinting code. In a fuzzy locker, a message is encrypted using a random codeword C_i. The message can be decrypted ("unlocked") only if one provides a pirate codeword C^* such that C_i will be accused by the accusation algorithm, otherwise, the message will remain IND-CPA secure. Given such a primitive, one can construct a TDS as follows: the embedding of the user private information can be simply done via encryption using the user's codeword (which is normally randomly selected according to the Gen algorithm). The *privacy* requirement can be easily achieved via the security of the fuzzy locker. The

[1] In many fingerprinting schemes, the soundness holds unconditionally, (e.g.,[36, 127]).

recover algorithm will first retrieve a “pirate codeword" from the pirate box and then it will try decrypting all locked data using this pirate codeword. The *traitor deterring* property can be guaranteed by the traitor tracing property of the fingerprinting code, since at least one of the codewords used in producing the pirate codeword will be accused and thus the private user data can be retrieved.

We first give the formal definition and security model of a fuzzy locker. W.l.o.g, we can think of the Gen algorithm of the fingerprinting code $\mathcal{C}$ to operate in two phases, first, using n, t and the security parameter to produce a secret state st and then uses a $\mathcal{C}.\mathsf{Sample}$ subroutine that produces the codewords one-by-one while updating st.

Definition 7. *A fuzzy locker w.r.t a (publicly traceable) fingerprinting code $\mathcal{C}$ consists of the following two algorithms:*

- $\mathbf{FL.Enc}(C_i, m)$*: Given a codeword $C_i \leftarrow \mathcal{C}.\mathsf{Sample}$ and a message m, the encryption algorithm outputs a ciphertext c.*
- $\mathbf{FL.Dec}(C^*, c)$*: Given a ciphertext c and a string C^*, the algorithm outputs a message m or $\perp$.*

Correctness: If $\mathcal{C}.\mathsf{Accuse}(tk, C_i, C^*) = 1$:

$$\Pr[\mathbf{FL.Dec}(C^*, c) = m] \geq 1 - \mathsf{negl}(\lambda).$$

Security of a fuzzy locker. We define t-resilient security (fully resilient if $t = n$) of a fuzzy locker scheme in the sense of IND-CPA security, by considering the following game between a challenger $\mathcal{C}$ and an adversary $\mathcal{A}$:

- The challenger produces st using Gen on input ϵ, t, n and sends $C_1, \ldots, C_t \leftarrow \mathcal{C}.\mathsf{Sample}(st)$ to $\mathcal{A}$.
- $\mathcal{A}$ selects two messages m_0, m_1 and sends them to $\mathcal{C}$.

- The challenger randomly samples a codeword $C_0 \leftarrow \mathcal{C}.\mathsf{Sample}(st)$, randomly flips a coins b, and sends $c = \mathbf{FL.Enc}(C_0, m_b)$ to the adversary $\mathcal{A}$.
- $\mathcal{A}$ outputs her guess b'.

A fuzzy locker is t-resilient IND-CPA secure if:

$$\mathsf{Adv}_{\mathsf{FL}}^{\mathcal{A}} = |\Pr[b' = b] - \frac{1}{2}| \leq \mathsf{negl}(\lambda).$$

Construction-I of TDS. Given a fuzzy locker and a public key encryption (PKE) with the algorithms (KeyGen, Enc, Dec), we can construct a TDS as follows:

- $\mathsf{Setup}(1^\lambda, s_1, \ldots, s_n, t)$: The algorithm first runs the codeword generation algorithm $\mathcal{C}.\mathsf{Gen}$ which inputs the security parameter, and t, n, it returns $tk, \{C_i\}_{i\in[n]}$, where $C_i = \mathsf{c}_1^i, \ldots, \mathsf{c}_\ell^i$. The algorithm then runs the KeyGen of the PKE and returns $q\ell$ key pairs:

$$(pk_{1,1}, sk_{1,1}), \ldots, (pk_{1,\ell}, sk_{1,\ell});$$
$$(pk_{2,1}, sk_{2,1}), \ldots, (pk_{2,\ell}, sk_{2,\ell});$$
$$\ldots$$
$$(pk_{q,1}, sk_{q,1}), \ldots, (pk_{q,\ell}, sk_{q,\ell})$$

  Finally, the Setup algorithm takes users' secrets $s_1, \ldots, s_n$, $tk, C_1, \ldots, C_n$ and all those key pairs as inputs, and it outputs system parameter $para$, an encryption key pk, and a set of decryption keys $sk_1, \ldots, sk_n$. Specifically, pk contains all the public keys above; $sk_i = \{sk_{1,\mathsf{c}_1^i}, \ldots, sk_{\ell,\mathsf{c}_\ell^i}\}$ for $i \in [n]$; and $para$ contains tk and $\langle \omega_1, \ldots, \omega_n \rangle$, where $\omega_i = \mathbf{FL.Enc}(C_i, s_i)$.

- $\mathsf{Enc}(pk, m)$: This algorithm is given pk and a message m. It first randomly samples $m_1, \ldots, m_{\ell-1}$, then computes $m_\ell = m - \sum_{i=1}^{\ell-1} m_i$ and $ct_{i,j} = \mathrm{Enc}(pk_{i,j}, m_j)$ for $i \in [q], j \in [\ell]$; it outputs the ciphertext $ct = \{ct_{i,j}\}$.

- $\mathsf{Dec}(sk_i, ct)$: This algorithm takes inputs $para, pk$, a secret key sk_i and a ciphertext ct. It parses the secret key and the ciphertext, and computes $m_j = \mathrm{Dec}(sk_{j,\mathsf{c}_j^i}, ct_{i,j})$ and further $m = \sum_{i=1}^{\ell} m_i$. The algorithm outputs m.
- $\mathsf{Rec}^{B,\mathcal{D}}(pk, para)$: This algorithm inputs $para, pk$ and has oracle access to a device B and a distribution $\mathcal{D}$. It first runs the following procedure for each index $k \in [\ell]$ to extract a pirate codeword from B:
  1. It initialize the pointer value $i_0 = 1$.
  2. It samples $m \leftarrow \mathcal{D}$, samples messages $\{m_i\}_{i \neq k}$ and computes $m_k = m - \sum_{i \neq k} m_i$.
  3. It feeds ciphertext $\{ct_{i,j}\}$ to B where $c_{i,j} = \mathrm{Enc}(pk_{i,j}, m_j)$ if $j \neq k$ or $i > i_0$; and $c_{i,k} = \mathrm{Enc}(pk_{i,k}, r_i)$ for $i \leq i_0$ and r_i is a random message.
  4. If in N runs (value will be determined in the analysis), the number of times n_1 that B returns m is sufficiently smaller than that in the $(i_0 - 1)$ experiment (the difference is denoted by n_0 when $i_0 = 1$, the experiment is using all valid ciphertexts), or it returns $r_{i_0} + \sum_{j \neq k} m_j$ the algorithm returns $\mathsf{c}_k^* = i_0$; otherwise, it stores n_1, sets $i_0 = i_0 + 1$, and repeats from step 2.

  The pirate codeword retrieved is C^*. The algorithm parses $para$ to identify the data $\langle \omega_1, \ldots, \omega_n \rangle$. It then runs the decryption algorithm of the fuzzy locker on all of them, i.e, for $i \in [n]$, it runs $\mathbf{FL.Dec}(C^*, \omega_i) = s_i$. The algorithm stops if $\exists s_i \neq \perp$ and it returns s_i.

Security analysis. Due to lack of space, we present here only a brief sketch about the security properties, and refer to the full version for the detailed proofs.

Regarding *IND-CPA security*, it follows in a straightforward manner from the security of the underlying PKE scheme.

Regarding *privacy* of the honest user secrets, follows easily from the security of the fuzzy locker.

Regarding the *black-box traitor deterring* property, note that the Rec algorithm proceeds in two steps, it first recovers a pirate codeword C^* from the box B. If there exists a colluder codeword C_i, such that $\mathsf{Accuse}(tk, C^*, C_i) = 1$, then in the second step, according to the correctness of the fuzzy locker, the decryption of $\mathbf{FL.Dec}(C^*, \omega_i)$ will return s_i. The security of the fingerprinting code guarantees that if any pirate codeword is produced following the "marking assumption", it can be used to accuse at least one of the colluders. The IND-CPA security of the underlying PKE scheme essentially enforces the "marking assumption." To see this, suppose the collusion user secret keys are $\{sk_i\}$ for $i \in \mathcal{U}_{cor}$, for each index j, the alphabet c_i^* for the pirate codeword at index i can not be any c_i^k for $k \notin \mathcal{U}_{cor}$ with probability significantly larger than the guessing probability $\delta - \alpha$. Otherwise, these keys may be used to decrypt a ciphertext encrypted under a public key $pk_{k,i}$.

The choice of N, n_0 can be easily determined as follows. As there must exist an index i_0 such that the probability of returning a correct plaintext (denoted by p_1) is at least $[\delta - (\delta - \alpha)]/q = \alpha/q$ smaller than that for $i_0 - 1$ (denoted by p_2). From the Chernoff bound $\Pr[X < (1-\omega)\mu] \leq e^{-\omega^2\mu/2}$, let us use $X_i^1 = 1$ denote the event that decryption query for $i_0 - 1$ is answered correctly while $X_i^2 = 0$ denote that for i_0 it is not answered correctly. Also we use $X_i = 1$ denote the above joint event, i.e., $\Pr[X_i = 1] = \Pr[X_i^1 = 1 \wedge X_i^2 = 0] = p_1(1 - p_2) \geq p_1 - p_2 \geq \alpha/q$. When we set $N = \frac{q}{\alpha}\log^2\lambda, n_0 = \frac{\alpha}{2q}N$, where λ is the security parameter, a pirate codeword must be identified.

Theorem 10. *Given a public key encryption scheme, and a fully secure fuzzy locker (for a q-ary fingerprinting code), there exists a (public key) TDS satisfying: fully collusion*

resilient, black-box traitor deterring property w.r.t to any message distribution $\mathcal{D}$ that has min-entropy $\mathbf{H}_\infty(\mathcal{D}) \geq -\log(\delta - \alpha)$, where δ is the correctness required by the adversarial device, α is a non-negligible amount that is significantly smaller than δ, and the parameters $N = \frac{q}{\alpha}\log^2\lambda, n_0 = \frac{\alpha}{2q}N$.

3.4.2 A Fuzzy locker for CFN codes.

We now propose a construction for a fuzzy locker w.r.t. CFN codes [36] using fuzzy extractors [50]. Consider the CFN fingerprinting scheme where the collusion size is set to be t; in order generate a codeword for a user j, the authority randomly samples $\mathsf{c}_1^j, \ldots, \mathsf{c}_\ell^j \leftarrow [q]^\ell$. The tracing algorithm accuses the user whose codeword has the largest number of locations that share the same symbol with the pirate codeword. Observe that this accusation procedure is identical to finding the "closest" among all user codewords to the pirate codeword. To put it in another way, the user codewords are random strings, but the tracing property of the CFN code guarantees that under the "marking assumption", any pirate codeword produced by a collusion of no more that t random codewords will have a small L_1-distance to one of the colluder codewords. [2] To facilitate the construction of the fuzzy locker we employ a fuzzy extractor [50] which enables one to retrieve the same random string from two different but correlated strings that have high entropy (cf. the fuzzy vault scheme [81]).

In more detail, most of the fuzzy extractors follow a correct-then-extract strategy. When the two strings are close enough, an error correcting code (ECC) can be used to eliminate their discrepancies and then a randomness extractor [32] is applied to extract a uniform output (which will later be used as a key) from the high entropy codeword

[2] Actually, from the analysis of CFN one infers that if the pirate codeword and user codeword agree on more than ℓ/t symbols, the user can be accused.

(which will be the source from the point of view of the extractor). However for the fuzzy locker for CFN codes, the portion of disagreement (errors) between the codeword used for encryption and the pirate codeword extracted for decryption is quite large and beyond the decoding capability of a unique decoding ECC. We thus give up perfect correctness for the fuzzy locker, and turn to the use of list decoding [71, 126]. In a list decodable code, the error correction returns multiple candidates, but it can decode efficiently a number of errors up to portion almost 1 (as opposed to unique decoding). One last thing we need to be careful is that the rate of the ECC should be selected in a way that the entropy loss will not prohibit randomness extraction.

Combining the above tools, we will use the uniform string extracted from the fuzzy extractor as a secret key to encrypt the user secret data. We further will assume the valid messages are easily identifiable, and that decryption using a wrong key will almost never yield a valid message. These two assumptions are easy to achieve by including in the plaintext a message authentication code or a signature on the message, for details about this technique, we refer to [50, 99].

Fuzzy locker for CFN codes.: We present below the fuzzy locker for CFN codes; the choices of the parameters will be specified later. Given a randomness extractor Ext and a secure symmetric key encryption $(\mathsf{SE.Enc}, \mathsf{SE.Dec})$:

- **FL.Enc**(C, m): The algorithm takes input $C = \mathsf{c}_1 \dots \mathsf{c}_\ell \xleftarrow{U} \mathcal{F}_q^\ell$, and message m. It first samples a random $(\ell, \kappa)_q$ Reed-Solomon code $X = x_1, \dots, x_\ell$ which can correct up to $\ell - \ell/t$ errors, and computes $Y = \{y_1, \dots, y_\ell\}$, where $y_i = \mathsf{c}_i + x_i \mod q$; It then selects a random bitstring s and computes $k = \mathsf{Ext}(s, C)$, [3] and $c = \mathsf{SE.Enc}(k, m)$. The algorithm outputs $ct = (Y, s, c)$.

[3] we assume here extractors can be applied to large alphabet, if not, we can simply use the bit string representing C to be the input to the extractor.

- **FL.Dec**(C^*, ct): The algorithm inputs a pirate codeword $C^* = \mathsf{c}_1^* \dots \mathsf{c}_\ell^*$ and ciphertext ct. It first computes $C' = \mathsf{c}_1' \dots \mathsf{c}_\ell'$ where $\mathsf{c}_i' = y_i - \mathsf{c}_i^* \mod q$, and it runs the list decoding algorithm on C' to get a list of RS codewords $\{X_1', \dots, X_L'\}$. It then computes a list of possible user codewords $\{C_1, \dots, C_L\}$ where $C_i = Y_i - X_i'$, where "-" stands for component-wise modular subtraction. The algorithm tries the following for every user codeword: it computes $r_i = \mathsf{Ext}(s, C_i)$ and $m_i = \mathsf{SE.Dec}(r_i, c)$. If there exists an $m \neq \perp$, the algorithm outputs m, otherwise, $\perp$.

Security analysis. Regarding *correctness.* First we recall the basics of the CFN code. It randomly samples $C \leftarrow \mathcal{F}_q^\ell$. Suppose t users (w.l.o.g, we assume they have codewords $C_1, \dots, C_t$) collude to produce a pirate codeword C^*. Due to the marking assumption, each symbol c_i^* it equals to one of the corresponding symbols in $C_1, \dots, C_t$. It follows easily that there exists a C_i, such that C^* and C_i agree on at least ℓ/t locations. We now check the decryption algorithm on $ct_i = \mathbf{FL.Enc}(C_i, m_i)$. $C' = Y - C^* = X + (C - C^*)$ mod q, thus $\{\mathsf{c}_1', \dots, \mathsf{c}_\ell'\}$ agree with $x_1, \dots, x_\ell$ on at least ℓ/t locations. For a Reed-Solomon code RS: $\Sigma^\kappa \to \Sigma^\ell$, it can decode at most $\ell - \sqrt{\ell\kappa}$ errors. If we have $\ell/t \geq \sqrt{\ell\kappa}$, then RS would return a list of possible candidates which contains the actual X. Then $Y - X$ would yield the user codeword C_i; correctness then follows easily.

Regarding *security*: for any honest user whose codeword C that is uniformly selected, we can think it is selected after the pirate code C^* is produced. Following the probabilistic analysis from [36], if $\ell \geq 4t \log \frac{n}{\epsilon}/3$, and $q \geq 4t$, it holds that $\Pr[C^*, C \text{ agree on } \ell/t \text{ locations}] \leq \epsilon$. It follows that the decoding algorithm will not return any user codeword. A bit more formally, we can think of the ciphertext (Y, s, c) as being generated following the KEM/DEM [42] framework, where Y, s encrypt the session key k which is used to encrypt the data in c. Conditioned on Y, s, the min-entropy of C can be calculated as $\ell \log q - (\ell - \kappa) \log |\Sigma|$ as s is independent of C, Y is

of length ℓ, and the original random codeword has entropy $\kappa \log |\Sigma|$. Now if we have $\ell \log q - (\ell - \kappa) \log |\Sigma| \geq \Theta(\lambda)$, the strong extractor can output a sufficiently long uniform key k, thus Y, s form a secure KEM. Now the IND-CPA security of the message follows from the security of the symmetric key encryption.

Setting up the parameters. There are multiple restraints about selecting the right parameters for the first construction of traitor deterring scheme from the CFN code. More specifically, for parameters $\ell, \kappa, n, t, \epsilon, \lambda$ being dimension and degree of the RS code, number of users, bound of colluders, error term in the fingerprinting code and the security parameter respectively, they have to satisfy: (1). $\ell \geq \max(\kappa t^2, 4t \log \frac{n}{\epsilon})$; (2). $\ell \log q - (\ell - k) \log \ell \geq \Theta(\lambda)$.

When $\kappa t^2 \leq 4t \log \frac{n}{\epsilon}$, we can choose $\ell = q = 4t \log n\epsilon$, and $\kappa = \Theta(\lambda)$. The resulting traitor deterring scheme will have ciphertext size $O(t^2 \log^2 \frac{n}{\epsilon})$, and the upper bound of the collusion size is $t = O(\log \frac{n}{\epsilon}/\lambda)$;

When $\kappa t^2 \geq 4t \log \frac{n}{\epsilon}$, we can choose $\ell = q = \lambda t^2$, and $\kappa = \Theta(\lambda)$. The resulting traitor deterring scheme will have ciphertext size $O(\lambda t^4)$ for any collusion size t.

To summarize, if we select the parameters in a way that all the conditions above are satisfied, then the correctness and security of the fuzzy locker for CFN code follows. Then from the general construction, we can conclude that:

Corollary 11. *Given PKE, there exists a (public key) TDS satisfying: fully-collusion resilient, black-box traitor deterring property w.r.t. to any message distribution $\mathcal{D}$ that has min-entropy $\mathbf{H}_\infty(\mathcal{D}) \geq -\log(\delta - \alpha)$, where δ is the correctness probability required by the adversarial device and α is a non-negligible amount significantly smaller than δ. And it is with ciphertext size $O(\log \frac{n}{\epsilon}/\lambda)$ when $t \leq 4 \log \frac{n}{\epsilon}/\lambda$; and $O(t^4\lambda)$, if $t \geq 4 \log \frac{n}{\epsilon}/\lambda$.*

Proof. We first analyze the properties of the fuzzy locker for CFN codes.

Regarding correctness, suppose the pirate codeword $C^* = \mathsf{c}_1^*, \ldots, \mathsf{c}_\ell^*$ makes a user codeword $C = \mathsf{c}_1, \ldots, \mathsf{c}_\ell$ accused, i.e., $\mathsf{Accuse}(tk, C^*, C) = 1$. That means for $C - C^* := \mathsf{c}_1 - \mathsf{c}_1^*, \ldots, \mathsf{c}_\ell - \mathsf{c}_\ell^*$, there are at least $\frac{\ell}{t}$ zeros for parameter of collusion size t. The decryption algorithm first computes $C' = Y - C^* = X + (C - C^*) \mod q$, thus $\{\mathsf{c}'_1, \ldots, \mathsf{c}'_\ell\}$ agree with $x_1, \ldots, x_\ell$ on at least ℓ/t locations.

While the RS code can correct up to $\ell - \ell/t$ errors, running the decoding algorithm on C' will return the correct codeword X. It follows that $Y - X = C$. Then the decryption algorithm simply runs the extractor on s, C and retrieves a session key $k = \mathsf{Ext}(s, C)$ and runs the decryption algorithm on c to get the message $m = \mathsf{SE.Dec}(k, c)$.

Regarding security, we first show a simple argument. Given two symmetric key encryption schemes $\mathsf{SE}_1 = (\mathsf{SE.Enc}_1, \mathsf{SE.Dec}_1)$ and $\mathsf{SE}_2 = (\mathsf{SE.Enc}_2, \mathsf{SE.Dec}_2)$, the KEM/DEM mechanism using $\mathsf{SE}_1, \mathsf{SE}_2$ is IND-CPA secure if both of them are IND-CPA secure, i.e., we have a new symmetric key encryption $\mathsf{SE} : (\mathsf{SE.Enc}, \mathsf{SE.Dec})$ defined as: $\mathsf{SE.Enc}(k, m) : c_1 = \mathsf{SE.Enc}_1(k, r), c_2 = \mathsf{SE.Enc}_2(r, m)$ for a randomly chosen r.

Let us check the following game sequence. Game G_0, it is the original IND-CPA game for SE, in which the challenge ciphertext is generated as $\mathsf{SE.Enc}_1(k, r_0)$, $\mathsf{SE.Enc}_2(r_0, m_0)$;

In game G_1, the challenge ciphertext will be generated as $\mathsf{SE.Enc}_1(k, r_1)$, $\mathsf{SE.Enc}_2(r_0, m_0)$, using two different r_0, r_1;

In game G_2, the challenge ciphertext will be generated as $\mathsf{SE.Enc}_1(k, r_1)$, $\mathsf{SE.Enc}_2(r_0, m_1)$;

In game G_3, the challenge ciphertext will be generated as $\mathsf{SE.Enc}_1(k, r_0)$, $\mathsf{SE.Enc}_2(r_0, m_1)$;

G_0, G_1 are indistinguishable because of the IND-CPA security of SE_1. To see this, a simulator the challenge $\mathsf{SE.Enc}_1(k, r_b)$ and simulates $\mathsf{SE.Enc}_2(r_0, m_0)$ and forwards them to the adversary $\mathcal{A}$ distinguishing G_0, G_1. It is easy to see that the answer from $\mathcal{A}$ can be used directly to predict b;

G_1, G_2 are indistinguishable because of the IND-CPA security of SE_2, as the SE_1 part now is essentially independent with the SE_2 part carrying the messages;

G_2, G_3 are indistinguishable because of the IND-CPA security of SE_1, and can be similarly argued as the indistinguishability between G_0, G_1.

Now we instantiate SE_1 as follows: given a random CFN codeword C, and a uniform string s, an extractor is applied to define the message $k = \mathsf{Ext}(s, C)$; Now the output of $\mathsf{SE}.\mathsf{Enc}_1(C, k)$ is defined as (Y, s), i.e., C is used the secret key to $\mathsf{SE}.\mathsf{Enc}_1$, while k plays the role of message. Note that Y, s information theoretically hides k, if we ignore$\mathsf{SE}.\mathsf{Dec}_1(\cdot)$. To see this, $\mathbf{H}_\infty(C|Y, s) = \ell \log q - \ell \log |\Sigma| + \kappa \log |\Sigma|$, and the parameter selection guarantees that $\mathbf{H}_\infty(C|Y, s) \geq \Theta(\lambda)$ thus the distribution of k is statistically close to uniform and $\mathbf{H}_\infty(k|Y, s) \approx \lambda$ (All other informations are independent thus we omit them here). Thus, SE_1 defined in this way is IND-CPA secure, and further the fuzzy locker for CFN code is fully-resilient IND-CPA secure.

Then following Theorem 10, and the parameter selection, we can conclude that we can construct a traitor deterring scheme from any PKE schemes. □

3.5 Construction from Comparison Predicate Encryption

In this section, we will present our second technique of constructing traitor deterring schemes based on comparison predicate encryption with an exponentially large attribute space. We first give the general construction of TDS, then instantiate the comparison predicate encryption from (optimized) bounded collusion functional encryption. The resulting TDS exhibits better efficiency than our CFN construction for larger collusions.

3.5.1 TDS from CPE.

In a CPE, decryption succeeds only when $v \leq x$, where x, v are the attributes for the the ciphertext and the secret key respectively. Moreover, besides standard security, it also requires an attribute hiding property that no adversary $\mathcal{A}$ can distinguish c_0, c_1 which have attributes x_0, x_1 (assuming $x_0 < x_1$) respectively, as long as $\mathcal{A}$ does not have a secret key sk_v such that $x_0 \leq v < x_1$ (even if $\mathcal{A}$ has secret key sk_v that can decrypt both c_0, c_1). (This corresponds to the fully attribute hiding of predicate encryption [**?**]).

It was shown in [22,25] that a weaker version of CPE (called private linear broadcast encryption in [22], which has only a polynomially large attribute/identity space) implies a TTS. In the construction, each user is assigned an integer index as identity, and the encryption scheme has the property that $Enc(pk, i, m)$ is indistinguishable from $Enc(pk, i+1, m)$ provided $\mathcal{A}$ does not hold sk_i. Thus the tracer can do a linear scan in the identity space and feed ciphertexts generated using attributes from 0 to $n+1$ for each test. If he notices a gap between the responses for some i, and $i+1$, then the user i will be accused. And the gap is guaranteed to exist as all users can decrypt $Enc(pk, n+1, m)$ and no user can decrypt $Enc(pk, 0, m)$.

To construct a TDS, we observe that if the indices are chosen randomly from an exponentially large space, they could be used as secret keys to hide the user private information. Unfortunately, it is not clear how to generalize [22,25] to an exponentially large identity space. We tackle this problem by constructing CPE's for an exponential large attribute space from functional encryption; furthermore, we apply a binary search type of tracing. In particular, in each step of search (feeding a sequence of tracing ciphertexts using a corresponding pivot identity), the recovering algorithm only consider two states for the pirate box. It is functioning, if the decryption probability is close to the claimed correctness of the pirate box, δ; or not functioning, otherwise. Then

the recovering algorithm decides to move to a smaller pivot or a larger one.Given a CPE (CPE.Setup,CPE.KeyGen,CPE.Enc,CPE.Dec) and an authenticated encryption (AE.Enc, AE.Dec), our second construction of TDS (construction-II) is as follows:

- $\mathsf{Setup}(\lambda, n, s_1, \ldots, s_n)$: The setup algorithm first runs the CPE.Setup algorithm to output a master key pair (mpk, msk), then it randomly selects n bitstrings $id_1, \ldots, id_n$ with length ℓ, (that is as an integer, each $id_i \in [2^\ell - 1]$), and runs the CPE.KeyGen algorithm to generate secret keys for the users. For user i it assigns the identity id_i and then generates the secret key sk_i =CPE.KeyGen(msk, id_i). It embeds the secret information of the user s_i as the ciphertext $\omega_i = \mathsf{AE.Enc}(id_i, s_i)$. The setup algorithm outputs public key $mpk, para$, and secret keys $sk_1, \ldots, sk_n$, where $para = \langle \omega_1, \ldots, \omega_n \rangle$.

- $\mathsf{Enc}(mpk, m)$: This algorithm runs $\mathsf{CPE.Enc}(mpk, 2^\ell, m)$ and returns the corresponding output c as ciphertext.

- $\mathsf{Dec}(sk_i, c)$: This algorithm runs CPE.Dec with input sk_i and ciphertext c and returns m or $\perp$.

- $\mathsf{Rec}^{B,\mathcal{D}}(mpk, para)$: The algorithm maintains a counter j with initial value $\ell - 1$ and repeats the following procedure until $j = 0$: It first samples a sequence of messages $m_1, \ldots, m_N$ from $\mathcal{D}$; then it generates the query ciphertexts $c_1, \ldots, c_q$, where c_i =CPE.Enc(mpk, p, m_i) for the position $p = 2^j$ and records how many correct answers does the box B produce; If the number of correct decryptions is more than n_0, the algorithm will set the pivot for the next test position to be $p := p - 2^{j-1}$, otherwise $p := p + 2^{j-1}$; The algorithm then decreases the counter $j := j - 1$ and repeats the procedure. The values for the parameters N, n_0 will be determined in the analysis. Suppose the above algorithm stops at position p. The

Rec algorithm then runs $\mathsf{AE.Dec}(p, \omega_i)$ on all ω_i and returns the first non-$\perp$ value s_i.

Remark: We may implement the authenticated encryption as $\mathsf{SE.Enc}(k, s||\sigma)$ where $\sigma = Sig(s)$, where Sig is a signature scheme and the verification key is included in $para$ where $\mathsf{SE.Enc}$ is any secure symmetric key encryption scheme.

Analysis. *Correctness* and *privacy* follow straightforwardly, so we focus on the intuition of the *black-box traitor-deterring* property. Let us first present the following observations. (1). If all the colluder identities are smaller than the index p used in the tracing ciphertext, the box will decrypt correctly with probability close to δ. This holds because of the the attribute hiding property that $\mathsf{CPE.Enc}(mpk, p, m)$ is indistinguishable from the regular ciphertext $\mathsf{CPE.Enc}(mpk, 2^\ell, m)$. From this it can be deduced that failing to decrypt with probability close to δ suggests that at least one colluder identity is larger than p, thus the algorithm will not err by moving to a larger pivot. (2). Similarly, if all colluder identities are larger than the pivot index p used in the tracing ciphertext, the box will work with just negligible probability because of the payload hiding property. It follows that decrypting with a probability close to δ (non-negligible) implies at least one colluder identities is smaller than the attribute in the tracing ciphertext, and hence the tracing algorithm will not err by moving to a smaller position attribute. (see lemmas in the appendix.) The above observations imply that every move is towards a subspace containing some pirate identities.

To be a bit more formal, consider a complete binary tree which represents the whole identity space. We can think of the path walked by the Rec algorithm as moving along such tree. It starts from the root (represented by index $2^{\ell-1}$) and moves to the root of a complete subtree in each step. We will show via strong induction that in each move, the subtree will contain at least one colluding identity.

Theorem 12. *Construction-II satisfies fully collusion resilient, black-box traitor deterring property w.r.t to any message distribution $\mathcal{D}$ that has min-entropy $\mathbf{H}_\infty(\mathcal{D}) \geq -\log(\delta-\alpha)$ for some non-negligible α, s.t., $\delta \geq 1.5\alpha$, where δ is the correctness probability provided by the adversarial device, and the parameters $N = \alpha^{-2}\log^2 \lambda$, $n_0 = (\delta - \frac{\alpha}{2})N$.*

Proof. Correctness follows directly from the correctness of the underlying CPE scheme. *Privacy* is also straightforward, as the user identity is uniformly sampled, the IND-CPA security of the underlying encryption scheme guarantees no information about the plaintext is leaked.

Regarding the *traitor-deterring* property: Suppose a pirate box B is created using secret keys of the users $id_1, \ldots, id_t$, and it is with $\delta-$correctness w.r.t a message distribution $\mathcal{D}$, s.t., $\mathbf{H}_\infty(\mathcal{D}) \geq -\log \delta_0$, where $\delta_0 = \delta - \alpha$, for some α. We first present three lemmas that follow easily from the payload hiding and attribute hiding properties of CPE.

Lemma 13. *If the underlying CFE is payload hiding, and the tracing ciphertext C is created using a pivot p and message m randomly sampled from $\mathcal{D}$, and $id_i > p$ for all $i = 1, \ldots, t$, then: $|\Pr[B(C) = m] - \delta_0| = \mathsf{negl}(\lambda)$.*

Lemma 14. *If the underlying CFE is attribute hiding, and two tracing ciphertexts C_1, C_2 are created using message m, and pivots p_1, p_2 respectively, and for all $i = 1, \ldots, t$, $id_i \notin [p_1, p_2)$, then: $|\Pr[B(C_1) = m] - \Pr[B(C_2) = m]| = \mathsf{negl}(\lambda)$.*

Lemma 15. *If the underlying CFE is attribute hiding, the tracing ciphertext C is created using a message m randomly sampled from $\mathcal{D}$ and a pivot p, and $id_i \leq p$ for $i = 1, \ldots, t$, then $|\Pr[B(C) = m] - \delta| = \mathsf{negl}(\lambda)$.*

We then estimate the parameters n_0, N for determining whether B works. Following the Chernoff bounds, $\Pr[X < (1-\omega)\mu] \leq e^{-\omega^2\mu/2}$, and $\Pr[X > (1+\omega)\mu] \leq e^{-\omega^2\mu/3}$,

when $X = \sum X_i$, $\{X_i\}$ are independent random variables over $\{0, 1\}$, $0 < \omega < 1$, and $\mu = E(X)$. In this setting, X_i is the event denoting when Rec feeds the $i-$th ciphertext which encrypts a random message m sampled from $\mathcal{D}$, the box B returns the plaintext correctly. It follows that, if the traitor indices are all smaller than the pivot, B works with δ-correctness, $\Pr[X_i = 1] \geq \delta$. After repeating N times, the probability that at most $n_0 = (\delta - \frac{\alpha}{2})N$ correct answers are returned by B is bounded by $e^{-\alpha^2 N/8}$. On the other hand, if the traitor indices are all larger than the pivot, B works with only probability $\delta - \alpha$. The probability that B returns more than n_0 correct answers is bounded by $e^{-\alpha^2 N/12}$.

Setting parameters $N = \alpha^{-2}\log^2 \lambda, n_0 = (\delta - \frac{\alpha}{2})N$, less than n_0 correct answers means that there must be a traitor index larger than the pivot; more than n_0 correct answers means there must be a traitor index smaller than the pivot.

Now we are ready to proceed to prove the theorem. We can represent all users as leaves in a complete binary tree indexed by $\{1, \ldots, 2^\ell\}$; given this Rec moves a pivot performing a binary search in this tree by selecting a sequence of subtrees $S_0, S_1, \ldots$ in the following fashion: at move $j \geq 1$, the pivot p_j defines the subtree S_{j-1} as the subtree of the complete binary tree that is rooted at a node v that has p_j as the index of the rightmost leaf of the left subtree of S_{j-1}. Observe that S_0 is the whole tree. We will prove by strong induction that for all $j \geq 0$, S_j contains a traitor. The base, $j = 0$, is straightforward. Suppose that the statement is true for $S_0, S_1, \ldots, S_{j-1}$. We will prove for S_j.

Case 1. Suppose that S_j is a left subtree of S_{j-1}. This means that there is a traitor with index at most p_j (otherwise, if all traitors had a bigger index, then by lemma 13 the pirate box would be unsuccessful and the recovering algorithm would move to the right subtree of S_{j-1}). Now suppose that none of the traitors belong to S_j and let u be

the largest index of a traitor that has index at most p_j. By the fact that u does not belong to S_j we know that at least one of the subtrees $S_1, \ldots, S_{j-1}$ is a right subtree of its containing parent subtree. Let S_k be such a subtree with the largest $k \leq j-1$. Now note that when the recovering algorithm used pivot p_k (which lies in the center of subtree S_{k-1}) it holds that: $u \leq p_k$. Observe that there is no traitor with index in the set $\{p_k + 1, \ldots, p_j\}$. Based on lemma 14 the decision of Rec when testing with pivot p_j and pivot p_k should be the same (with overwhelming probability). This leads to a contradiction as Rec moved to the right (resp. left) when testing with index p_k (resp. p_j).

Case 2. Suppose that S_j is a right subtree of S_{j-1}. This means that there is a traitor with index bigger than p_j (otherwise, if all traitors had an index that at most p_j, then by lemma 15, the pirate box would have a close-to-δ-correctness and the recovering algorithm would move to the left subtree of S_{j-1}). Now suppose that none of these traitors belongs to S_j and let u be the smallest index of traitor that has index larger than p_j. By the fact that u does not belong to S_j we know that at least one of the subtrees $S_1, \ldots, S_{j-1}$ is a left subtree of its containing parent subtree. Let S_k be such a subtree with the largest $k \leq j-1$. Now note that when the recovering algorithm used pivot p_k (which lies in the center of subtree S_{k-1}), it holds that: $u > p_k$. Observe that there is no traitor with index in the set $\{p_j + 1, \ldots, p_k\}$. Based on lemma 14, the decision of Rec when testing with pivot p_j and pivot p_k should be the same (with overwhelming probability). This leads to a contradiction as Rec moved to the left (resp. right) when testing with index p_k (resp. p_j).

We can conclude that S_ℓ is a single leaf node and it also denotes a traitor. □ □

3.5.2 Instantiation of comparison predicate encryption.

Next we will give a concrete construction of CPE which supports an exponentially large attribute space. We first note that, a straightforward instantiation can be obtained from general functional encryption (FE) which can be constructed using indistinguishability obfuscation (iO) [55]. The resulting TDS will have only a constant size ciphertext however it will rely on assumptions related to multilinear maps [55].

We now present an instantiation from standard assumptions. We note that there exists a construction of bounded collusion FE from standard assumptions. In a traitor deterring scheme there is only a potentially small (and in any case polynomially bounded) subset of users that is colluding to produce a pirate box. We show how to construct a CPE from bounded collusion FE.

Instantiation-I. General functional encryption secure for a single key query with succinct ciphertext was constructed in [63]. A trivial way to amplify this scheme to a q-query secure FE is to run q independent 1-query secure FE schemes in parallel. Each user secret key is generated using a different master secret key (this step will require that the authority maintains and updates a private state to keep track of which master secret keys have been used), while each master public key will be used to encrypt the message resulting in a vector of q ciphertexts encrypting the same message (see Appendix 6.2 for details). Unfortunately using this scheme to instantiate the CPE for a TDS would force $q = n$. To see this, even if we choose $q = n - 1$, there exist a pair of users i, j such that their secret keys are generated using a same master secret key (say the k-th master secret key). When user i, j are corrupted together, then no security can be guaranteed for the k-th 1-query secure FE instance, and the CPE scheme cannot be shown secure. Thus the resulting TDS will have ciphertext size $O(n \cdot \mathrm{poly}(\lambda))$ which is not preferable

especially given that the collusion t might be much smaller than n. We then show how to improve the ciphertext complexity.

Instantiation-II. A stateless q bounded FE was constructed in [65] from a 1-query secure FE using techniques from secure computation, and their scheme can guarantee security under arbitrary collusion with size q, even if more keys are issued (say n). We can use such a t-bounded FE to instantiate a CPE facing t corrupted users. Unfortunately, the parameters in [65] were chosen in a way that the ciphertext size is as big as $O(D^2t^6\lambda\tau)$, where D is the maximum degree of the polynomial representing the circuits describing the functions that FE supports, and τ is the ciphertext size of the underlying 1-query secure FE. For some parameters d, N, in the construction, there are N 1-query secure FE instances. The encryption algorithm will do a $(d+1, N)$ secret sharing on the message and will encrypt each share independently under the N 1-query FE instances. Each user will be assigned a random subset (denoted by Γ_i, and $|\Gamma_i| = dD+1$) of keys each of which is generated using the corresponding master secret key. Note that prior to encrypting each share is padded with additional randomness to ensure the simulation can succeed. In total there are N ciphertexts each encrypting $O(t^2\lambda)$ number of plaintext elements. (See Appendix 6.2 for details).

[65] requires that the collusion of size t can not break enough 1-query secure FE instances to get $d+1$ shares and obtain extra information about the message. More specifically, it requires $|\cup_{i\neq j}(\Gamma_i \cap \Gamma_j)| \leq d$. We observe that if we can replace this condition to be $|\cup_{i_1,\ldots,i_a}(\cap_{i_j=i_1}^{i_a}\Gamma_{i_j})| \leq d$, for any integer $a \geq 2$, through a probabilistic analysis, we can bring down N to $O((Dt)^{1+e}\lambda)$ (for $e = 1/(a-1)$). Doing this optimization requires us to use an a-query secure FE as the underlying building block. We can obtain a succinct a-query secure FE for some polynomially bounded a by applying the technique of [65] to the succinct 1-query secure FE of [63]. In this way we obtain a a-query FE

that has ciphertext size $O(\text{poly}(\lambda))$ which is independent from the number of users. Then we can apply the extended probabilistic analysis explained above and to obtain a t-query FE with ciphertext $O(t^{3+e}\text{poly}(\lambda))$. Note that we are using circuits for the comparison predicate only and thus the degree D of the polynomial representing the circuits is at most λ.

Our CPE instantiation. Our final CPE instantiation will be a hybrid of the two instantiations above. When $t \leq n^{\frac{1}{3+e}}$, we use *instantiation-II*, the optimized t-FE; when $t > n^{\frac{1}{3+e}}$, we simply use *instantiation-I* of n-query secure FE. The resulting TDS will be with ciphertext size $\min[O(t^{3+e} \cdot \text{poly}(\lambda)), O(n \cdot \text{poly}(\lambda))]$. As the succinct 1-query FE can be built on fully homomorphic encryption [27] and attribute based encryption [66], both of which can be based on the LWE assumption [115] efficiently. We summarize the above, and refer detailed analysis to section 3.2.

Corollary 16. *Under the subexponential LWE assumption, there exists a TDS satisfying: fully collusion resilient, black-box traitor deterring w.r.t to any message distribution $\mathcal{D}$ with $\mathbf{H}_\infty(\mathcal{D}) \geq -\log(\delta - \alpha)$, where δ is the correctness probability provided by the pirate box, α is a non-negligible amount and $\delta \geq 1.5\alpha$. It has ciphertext length $\min[O(t^{3+e} \cdot \text{poly}(\lambda)), O(n \cdot \text{poly}(\lambda))]$, where n, t are total number of users and corrupted users, $e = 1/\text{poly}(\lambda)$, and λ is the security parameter.*

Proof. We first analyze the parameters of the instantiation of comparison predicate encryption in section 3.5.2.

For the parallel repetition construction, as every secret key is generated from a different master secret key, and the ciphertext $CT = \{CT_i\}$, and $CT_i = \mathsf{OneFE.Enc}(mpk_i, m)$ for the same message m. It is obvious that due to the security of the underlying 1-query secure FE, every ciphertext is simulatable given $C(m)$ by running the corresponding 1-query secure FE simulator.

Next, we will analyze the optimized q-query secure FE. Following the analysis of [65], the only difference of our scheme is that we use a c-query secure FE as the building block. We can replace the first restriction to be $|\cup_{i_1,\ldots,i_c}(\cap_{i_j=i_1}^{i_c}\Gamma_{i_j})| \leq d$, as each instance now can assure security given that the for less than d instances, the collusion of t users have c keys. Now we can analyze that this condition will improve the size of N.

To be more specific, suppose X_{ij} denotes the expected size of the intersection of two random subsets X_i, X_j. It satisfies that

$$E(X_{ij}) = \sum_X E(X_{ij}|X_i = X)\Pr[X_i = X] = E(X_{ij}|X_i = X).$$

To see this, for any X with size $dD+1$, the expected value $E(X_{ij}|X_i = X)$ is the same and the conditional distribution follows the hypergeometric distribution with $dD+1$ good balls and $tD+1$ draws, thus $E(X_{ij}|X_i = X) = (tD+1)^2/N$.

For X_{ijk} denoting the expected size of intersection of three random subsets X_i, X_j, X_k,

$$\begin{aligned} E(X_{ijk}) &= \sum_i \Pr[X_{ij} = i]E(X_{ijk}|X_{ij} = i) = \sum_i \Pr[X_{ij} = i]i \cdot (dD+1)/N \\ &= \frac{(dD+1)}{N}E(X_{ij}) = \frac{(tD+1)^3}{N^2}. \end{aligned}$$

Similarly, we can generalize the second formula to the expected size of intersection of c random subsets which is $(tD+1)^c/N^{c-1}$.

The expected size E_c of the disjoint of all possible intersection of c subsets: $|\cup_{i_1,\ldots,i_c}(\cap_{i_j=i_1}^{i_c}\Gamma_{i_j})|$ (we denote as γ) is the summation of all combinations of $X_{i_1,\ldots,i_c}$, i.e.,

$$E_c = q(q-1)\ldots(q-c+1)\cdot(dD+1)^c/N^{c+1} \leq (2qdD)^c/N^{c-1}.$$

If we let $E_c \leq \frac{d}{2}$, i.e. let $N = 4d(Dq)^{1+e}$, where $e = 1+1/c$, then following the Chernoff bound: for any $\delta > 0$, $\Pr[X > (1+\delta)E[X]] \leq e^{\frac{-\delta^2}{2+\delta}E[X]}$, thus:

$$\Pr[\gamma \geq t] = \Pr[\gamma \geq (1+1)E_c] \leq e^{-\frac{E_c}{3}} = e^{-d/6}.$$

While we are focusing on comparison predicate, which can be easily implemented e.g., for two numbers x, v represented using bits $x_1 \dots x_\ell, v_1 \dots v_\ell$, the comparison predicate $[x \leq v] \iff [x = v] \vee [x_1 = 0 \wedge v_1 = 1] \vee [(x_1 = v_1) \wedge (x_2 = 0 \wedge v_2 = 1)] \vee \dots \vee [(x_1 \dots x_{\ell-1} = v_1 \dots v_{\ell-1}) \wedge x = v]$, is with degree at most $\ell = O(\lambda)$.

Summarizing the above analysis, if we set $d = \lambda$, $N = O(q^{1+e}\text{poly}(\lambda))$, we have $\Pr[|\cup_{i_1,\dots,i_c}(\cap_{i_j=i_1}^{i_c}\Gamma_{i_j})| \geq d] \leq e^{-\lambda/6} = \mathsf{negl}(\lambda)$. Then following the analysis of [65], all the ciphertexts can be simulated.

The ciphertext efficiency of the optimized scheme is that $O(N \cdot S \cdot \tau) = O(q^{3+e} \cdot \text{poly}(\lambda))$, where N is the number of ciphertexts, S is the number of plaintext elements in each ciphertext, and τ is the size of ciphertext for each plaintext element of the underlying succinct $c-$bound FE, where we can choose integer $c = \text{poly}(\lambda)$.

On the other hand, the parallel repetition based construction is simply with ciphertext efficiency $O(n \cdot \text{poly}(\lambda))$.

As the succinct 1-query secure FE can be constructed assuming succinct FHE and ABE for circuit, and the following two can be based on LWE assumption. Then following theorem 12, we can conclude as in the corollary. □

3.6 Traitor Deterring in the Known Ciphertext Model

In the known ciphertext model for TDS, the adversary has a weaker goal: it aims to produce a pirate box that works w.r.t. a *given* sequence of ciphertexts.

Because the sequence of ciphertexts is fixed there is a trivial way to implement the pirate decoder: simply store a database of all the plaintexts. Thus, in the known ciphertext model, the adversary should only win when the size of the decoder is smaller than the trivial attack; formally, we will associate an attack in this model with a "space rate" that is equal to the size of the pirate box divided by the length of the total plaintext

contained in the known ciphertext. An ideally secure scheme should work with respect to any space rate $o(1)$.

The known ciphertext model is applicable to the setting of distributing content via CDs, or granting access to an encrypted database, since in these cases, the attack occurs after the target ciphertexts become known. (In contrast, the original traitor deterring model is applicable to all other settings, e.g., online streaming, and movie distribution in Pay-TV etc.). Traitor deterring in the known ciphertext model reformulates in the black-box public-key setting the problem of constructing digital signets as posed in [51]. In [51] a construction for any space rate $o(1)$ is presented however it requires the unfalsifiable assumption that the function $f(x) = g_1^x||g_2^x||\ldots||g_\ell^x$ is incompressible (as well as they assume non-black-box recoverability). They leave as open question whether a construction exists that is secure under a falsifiable assumption; using our TDS's we resolve this open question in this section.

3.6.1 Definition: the known ciphertext model.

We provide the formal definition of the known ciphertext model that strengthens our traitor deterring definition from Section 3.3.

$(1-\epsilon)$-*correctness.* Since the pirate box B may work only for a fixed set of ciphertext $SC = \{c_1, \ldots, c_n\}$, we require for SC, it almost always works, i.e., $\Pr[B(i, c_i) = m_i] \geq 1 - \epsilon$, where $\epsilon = \mathsf{negl}(\lambda)$ and m_i is the $\Theta(\lambda)$ bit plaintext that is encrypted in c_i (note we also allow B to receive the index of the ciphertext).

Privacy: This is the same as in section 3.3.

Traitor Deterring for Known Ciphertexts. The main difference with the traitor deterring property is that the adversary is aware of the ciphertexts before making the device B,

and hence can embed some information into B so that B is able to check the decryption queries and only work for the given ciphertexts. Formally,

- The challenger $\mathcal{C}$ simulates the Setup algorithm and the adversary $\mathcal{A}$ receives pk. $\mathcal{A}$ then sends to $\mathcal{C}$ a vector of secret information $s_1, \ldots, s_n$, an arbitrary subset $T \subseteq \{1, \ldots, n\}$ as well as a distribution $\mathcal{P}_k$ with support set of a vector of k plaintexts for some $k = O(\text{poly}(\lambda))$. [4] $\mathcal{A}$ receives the secret keys of all users in T, $\{sk_i \mid i \in T\}$ as well as the public parameter $para$.

- $\mathcal{C}$ samples $(m_1, \ldots, m_k)$ from $\mathcal{P}_k$ and sends $\mathcal{A}$ to a sequence of ciphertexts $SC = \langle c_1, \ldots, c_k \rangle$; finally, $\mathcal{A}$ outputs an implementation B.

- $\mathcal{C}$ outputs 1 if and only if $\mathsf{Rec}^B(pk, para) \notin \{s_{i_1}, \ldots, s_{i_t}\}$.

We denote the event that $\mathcal{C}$ outputs 1 in the above game by $\mathsf{Succ}_{\mathcal{A}}^{\mathtt{KCdeter}}(1^\lambda)$. We say a scheme achieves black-box traitor deterring for known ciphertexts with space rate $\mathsf{s}(k, \lambda)$ if for any PPT adversary $\mathcal{A}$,

$$\Pr[\text{B is } (1-\epsilon)\text{-correct w.r.t } SC \wedge \frac{|B|}{k\lambda} = \mathsf{s}(k,\lambda) \wedge \mathsf{Succ}_{\mathcal{A}}^{\mathtt{KCdeter}}(1^\lambda)] = \mathsf{negl}(\lambda)$$

where $|B|$ denotes the size of the program B. Note that for $\mathsf{s}(k, \lambda) = \Theta(1)$ it is trivial to construct a device B that allows the adversary to win the above game — simply store all plaintexts $m_1, \ldots, m_k$ in B. Thus, the question that is raised is whether it is possible to deter with space rate that is $o(1)$.

3.6.2 Feasibility and infeasibility for the known ciphertext model.

At first sight, it may seem impossible to have a black-box recovering algorithm in the known ciphertext setting, since the Rec algorithm is restricted by the fact that

[4] This includes the case of encrypting one single long message (e.g., a movie file): it is first divided into k blocks and each block is encrypted individually.

the adversarial box is only guaranteed to work for a fixed set of ciphertexts. Indeed, although the size of B can be smaller than the size of the ciphertexts it is supposed to work for, there are ways for the adversary to embed some information and check whether a submitted ciphertext belongs to the targeted sequence, while reject all other ciphertexts submitted to it. We formalize this intuition and we show a simple attack following this principle and rules out the possibility of black-box traitor deterring for known ciphertexts for a range of space rates. However, we also observe that in order for the box B to perform a membership test in the targeted ciphertext sequence, the false positive probability of the testing algorithm increases as the storage used by B gets smaller. When the false positive probability becomes sufficiently high, a random sample of ciphertext will be answered by the box B with non-negligible probability δ, and thus B becomes a $\delta-$correct box in the regular model (as defined section 3.3); in this way, we can still apply our constructions of traitor deterring schemes against known ciphertext type of attacks. For ease of presentation, we consider only the 1-correct case, while all results will also follow for the case of $(1-\epsilon)$-correctness.

The intuition behind the proof of the following theorem is that when the suitable space bound is imposed on the pirate device, the device will have to resort to using the secret-key in a sufficiently large plaintext distribution that can be sampled with a non-negligible probability from the plaintext space. As a result, the decryption box, is a general purpose decryption box that is δ-correct for some non-negligible δ and thus our recoverability algorithms developed for traitor deterring can be applied in the known ciphertext model as well. We first present a lemma about the set membership problem.

Lemma 17. *For a universe U with size u, and $V \subset U$ with size v, and $v \ll u$, using space τ, the false positive η in the approximate set membership test problem satisfies $2^{\tau} \leq (2\eta)^{v}$.*

Proof. Let us denote the size of the universe as u, the size of V as v, suppose the size of the storage used for the approximate membership testing problem is no bigger than τ, and the false positive probability is η. Since we consider here only the case with false negative probability 0, we can think of the correspondence between the stored information and the subset of elements which the algorithm actually accepts. There are in total $C(u, v)$ many subsets from U with size v. By definition, with a false positive probability η, the algorithm will accept (outputs 1) $u' = v + \eta(u - v)$ many elements when they are randomly sampled from the universe. It follows that we can think of any specific memory string T as an encoding of $C(u', v)$ subsets that can be recognized by T. Since all subsets of size v are recognized it must be the case that $2^{\tau} \cdot C(u', v) \geq C(u, v)$; by expanding the inequality we have:

$$\begin{aligned} 2^{-\tau} \leq \frac{C(u', v)}{C(u, v)} &= \frac{u'(u'-1)\dots(u'-v+1)}{u(u-1)\dots(u-v+1)} \\ &= \frac{(\eta(u-v)+v)\dots(\eta(u-v)+1)}{u(u-1)\dots(u-v+1)} \\ &\leq (\frac{\eta(u-v)+v}{u-v})^v = (\eta + \frac{v}{u-v})^v \leq (2\eta)^v \end{aligned}$$

the last inequality holds when $u \gg v$. □

Theorem 18. *There exists a TDS with superpolynomial in λ plaintext space that satisfies black-box traitor deterring for known ciphertexts with space rate $\mathsf{s}(k, \lambda) = O(\log(\lambda)/\lambda) = o(1)$ for any $k = \Omega(\lambda)$.*

Proof. We will show that a TDS satisfying black-box traitor deterring with any pirate box with λ^{-c}-correctness is also a TDS satisfying black-box traitor deterring in the known ciphertext model for any $c \in \mathbb{N}$ for the stated space rate.

First, we recall a lower bound of the false positive probability in the approximate membership testing problem (see section 3.2.1 for definition) when the space of the

tester is upper bounded. For a universe U with size u, and $V \subset U$ with size v, and $v \ll u$, using space τ, the false positive η of any membership tester satisfies $2^{\tau} \leq (2\eta)^{v}$. (see Lemma 17 above). Applying logarithm to both sides, we can get $\eta \geq 2^{-\frac{\tau}{v}-1}$, thus if $\tau \leq c \cdot v \cdot \log \lambda$, we have $\eta \geq \lambda^{-c}$.

Next, we will use the above result to show that a useful decryption box B with size $O(k \cdot \log \lambda)$ will have non-negligible correctness w.r.t. uniform distribution over the message space. Specifically, we will build an approximate membership tester T (using B) for $V = \{(m_1, c_1), \ldots, (m_k, c_k)\}$, a subset of the universe U of all plaintext/ciphertext pairs, with a similar storage as follows. Whenever queried a uniformly random pair (m, c), T queries B with c, if B outputs m, T outputs 1, otherwise T outputs 0. It is easy to see that if $(m, c) \in V$, T always accepts; if $(m, c) \notin V$, T accepts with probability δ, where $\delta = \Pr[B(c) = m \wedge (m, c) \notin V]$. Furthermore, T only needs an extra storage of $O(\lambda)$ bits to store the query and compare whether the answer of B is valid. In the setting that $k = \Omega(\lambda)$, the storage of T is still $O(k \cdot \log(\lambda))$. Observe that if δ is negligible, T is a membership tester which violates the bound in Lemma 17.

With the above claim, we can see that for a randomly sampled ciphertext, the box B will answer with some probability δ and thus we can run the Rec algorithm and retrieve the corresponding secret information of one of the colluders assuming that the TDS works for δ w.r.t any distribution $\mathcal{D}$ for which it holds that $\delta \geq 2^{-H_{\infty}(\mathcal{D})} + \alpha$ where α is an arbitrary non-negligible function. □

Impossibility results. Next we will show that the above bound of the size of B is essentially tight, by describing a generic attack against any traitor deterring scheme for known ciphertexts. The attacking strategy is simple: using Bloom filters [15] the adversary produces a box that contains a membership tester built in so that it will answer only when the decryption query belongs to the ciphertext set. This makes two

boxes implemented using different keys indistinguishable via only oracle access, thus black-box recoverability will contradict privacy in this setting.

Proposition 19. *There does not exist any, even 1-resilient, black-box TDS in the known ciphertext model for space rate* $\mathsf{s}(k, \lambda) = \Omega(\log^2 \lambda/\lambda)$ *for any* k.

Proof. We will show an attack that uses up to $\Omega(k \cdot \log^2 \lambda)$ space can defeat the black-box traitor deterring if the privacy of the user data is also required.

The adversary $\mathcal{A}$ selects a set k of distinct plaintexts $S = \{m_1, \ldots, m_k\}$ and sets them as a distribution $\mathcal{P}_k$. $\mathcal{A}$ further corrupts user i and receives the corresponding set of ciphertexts $S = \{c_1, \ldots, c_k\}$. $\mathcal{A}$ creates a membership tester T with a false positive probability ϵ for S, such that T can be constructed using space $O(k \log \frac{1}{\epsilon})$. Using Bloom filters, [15, 108], if $\mathcal{A}$ uses space $\Theta(k \log^2 \lambda)$, then ϵ will be $\mathsf{negl}(\lambda)$. $\mathcal{A}$ produces a pirate box B with T built in, and when is given input c, B first checks whether $c \in S$ (besides the storage, this checking program has only a constant description). Assuming the test passes, the algorithm applies the key sk_i to decrypt the ciphertext.

Assume there is a black-box recovering algorithm recovers user i's secret information, given oracle access to B. There is another adversary corrupts a different user j and build a box B' s.t., it is the same as B when inputs passes the tester, B' uses sk_j to decrypt the query and respond. It is obvious that the input/output distribution of B, B' is statistically close (with the only difference because of the negligibly small false positive), these two boxes cannot be distinguished by any algorithm via only oracle access to them. Thus the Rec algorithm will also return the same output, i.e., secret information of user i when having oracle access to B', hence contradicting the privacy property. □

3.7 Using Bitcoin as a Collateral in a TDS

Bitcoin is a decentralized cryptocurrency [102] that uses a publicly maintained ledger to store transactions and record transfers between bitcoin accounts. Each bitcoin account is essentially a hash of a public-key and the owner of the secret-key has the ability to transfer funds from the account by posting a transaction in the bitcoin network that contains a signature generated by account's secret-key. The characteristic of bitcoin accounts is that the secret-keys represent complete ownership of the account.

We consider a TDS deployment for a broadcast service where a service provider (SP) wants to use a certain amount of bitcoin as collateral. Upon initiation of the service the SP generates bitcoin accounts corresponding to each of the n users setting $s_i = (a_i, k_i)$ where a_i is the bitcoin address and k_i is the associated secret-key. When a user joins the system it requests from the user to transfer some amount of x bitcoin to the a_i bitcoin account. The SP shares the account information (a_i, k_i) with the user so that it is ensured that the x bitcoin is a collateral and the user has the option to obtain the collateral back whenever she wishes (and cancel her subscription). At the same time the SP gives to the user the secret-key sk_i that corresponds to the account, and the user from this point on can use the service and decrypt ciphertexts associated with the service. In regular intervals the SP checks the public ledger to see whether any active account has an outgoing transaction (no matter it is transferred to the recipient or to the colluder herself). If there is such a case the subscription of the user should be cancelled (this would require the revocation of the key sk_i an issue that we do not explicitly deal here but can be handled generically via e.g., a re-key operation where the SP at regular intervals refreshes the keys of the system keeping the same collaterals for all the remaining subscribers). Observe that due to the properties of TDS for as long as

the user respects the service agreement and does not share her secret-key her collateral bitcoin remain safe. The user can collect her collateral bitcoin whenever she wants to terminate the service.

3.8 Conclusion and Open Problems

We formalize and construct a new cryptographic primitive of TDS to achieve proactive deterrence of unauthorized device redistribution. We also revisit the open problem of digital signets and reformulate it in a known ciphertext model, and show how we can utilize TDS to solve it under parameter choices that allow a possibility result. Finally we show how bitcoin can be used as a collateral for deployment of a TDS.

There are many interesting open problems remain. The first one is how to construct a TDS with constant size ciphertext under standard assumptions. This may require a fuzzy locker for, e.g., Tardos code [127] which currently uses a secret tracing algorithm. Also, a construction of unbounded collusion secure CPE is another alternative which has its own interests. Furthermore, combining a TDS with a revocation system as in [105] to obtain a "Trace Deterring and Revoke scheme" would be complementing this line of works. And finally, a game theoretic analysis to find the right amount of bitcoin required to deposit so that it might be better-off for the adversary not to corrupt more than t users will address important efficiency and security trade-offs for TDS's.

Chapter 4

Cliptography: Clipping the Power of Kleptographic Attacks

4.1 Introduction

Consider the conventional use of a cryptographic primitive, such as an encryption scheme: To encrypt a desired plaintext, the encryptor simply runs an implementation of the encryption algorithm obtained from a hardware or software provider with the plaintext as input. Although the underlying algorithms may be well-studied and proven secure, malicious implementations could still leak secret information exclusively to the provider/manufacturer without being noticed (through a covert channel that relies on an embedded backdoor). It is notable that *such leakage is possible even if the implementation produces "functionally and statistically clean" output that is indistinguishable from that of a faithful implementation.* While the underlying concept of kleptography was proposed by Young and Yung two decades ago [130, 131], the recent Snowden revelations [92, 110]

provided striking real-world examples that awakened the security community to the seriousness of these issues. As a result, the topic has recently received renewed attention; see, e.g., [7, 11, 47, 100]. In particular, Bellare, Paterson, and Rogaway [11][1] studied algorithm substitution attacks, with focus on symmetric key encryption. Soon after, Dodis, Ganesh, Golovnev, Juels, and Ristenpart [47] studied pseudorandom generators (PRG) in this backdoored setting.

4.1.1 Our contribution

We continue this line of pursuit. Specifically, we are motivated to develop cryptographic schemes in a *complete subversion model*, in which *all* algorithms of a scheme are potentially subverted by the adversary. This model thus significantly generalizes previously studied settings, which rely on trusted key generation or clean randomness that is assumed private from the adversary. We study two fundamental cryptographic primitives in the complete subversion model—one-way functions and one-way trapdoor funtions—and apply these primitives to construct other cryptographic schemes such as digital signatures and PRGs. Along the way, we identify novel generic defending strategies. We intend to stimulate a systematic study of **cliptography**, to provide a broader class of cryptographic building blocks and a larger set of defending strategies, eventually clipping out potential kleptographic attacks that arise from maliciously implemented components. As mentioned above, prior to our work kleptographic attacks on various primitives have been addressed in weaker models; see *Related work* in Section 4.1.2. In detail, we show the following:

- We study (trapdoor) one-way functions in the presence of kleptographic attacks. We first introduce a notion of *strong forgeability* that captures natural kleptographic

[1] This paper won the 2015 PET award.

attacks, namely: there is a specification that is proven secure for a (trapdoor) one-way function; the sabotaged function generation algorithm delivers a similar output distribution as the specification distribution; however, with a pre-chosen backdoor, the adversary can invert the entire family of functions that are generated by this subverted algorithm. We then show that such (adversarial) objects can indeed be constructed from (trapdoor) one-way functions by showing that random padding, in particular, renders the cryptosystem vulnerable. We also provide a weaker notion, *forgeability*, for one-way functions, that captures the case where the adversary only sets up the public parameters for which she keeps a backdoor.

- Our main goal is to provide defending mechanisms against kleptographic attacks. We construct *unforgeable* (trapdoor) one-way functions via a general transformation that "sanitizes" arbitrary OWFs by *randomizing the function index*. This transformation clips out all potential correlation between the function and the possible backdoor that the adversary may possess. Additionally, we introduce a *split-program* strategy to make the general method above applicable using only standard hash functions. In the split-program model, the function generation algorithm is composed of two parts: a (randomized) randomness generation algorithm RG that outputs an (ostensibly) uniform bit string, and a (deterministic) function generation algorithm dKG that converts such random string into the function index. We remark that our results even allow the sanitizing algorithm to be implemented by the adversary.

- In Section 4.4, we investigate how to construct backdoor-free PRGs. Previously, Dodis et al. [47] investigated "backdoored PRG" in which the adversary sets up a PRG instance (i.e., the public parameter), and is able to distinguish the output

from uniform with a backdoor. They then proposed immunizing strategies obtained by applying a keyed hash function to the output, *but assuming the key is unknown to the adversary* in the public parameter generation phase.

We construct backdoor-free PRGs in the complete subversion model. Our first construction is based on the classic Blum-Micali using our strongly unforgeable OWF and the Goldreich-Levin hardcore predicate [61]. In addition, in [47], Dodis et al. show that it is impossible to have a public immunizing strategy for all PRGs by applying a public function to the PRG output. This is because there always exists a PRG (having the immunizing function built in) that reveals the seed to the adversary bit by bit in each iteration. We circumvent their impossibility result via an alternative public immunizing strategy. Instead of randomizing the output of the PRG, we randomize the public parameter of PRG, which gives us a general construction for PRG in the complete subversion model.

- Having constructed unforgeable (trapdoor) one-way functions, we next explore the power of such primitives. In Section 4.5, we observe that unforgeable trapdoor one-way functions immediately give us a way to construct key generation algorithms (for digital signature schemes and stateless) against kleptographic attacks. We then showcase a concrete example of digital signature scheme in the complete subversion model. More concretely, we achieve this result by (1) using the unforgeable trapdoor one way permutation directly as a key generation algorithm, and then (2) instantiating the unique signature mechanism with the full domain hash. In previous works, [4, 11] demonstrated that a unique signature scheme is secure against kleptographic attacks, *assuming that the key generation algorithm is honest*

and all the message-signature pairs can be checked by the lab/user. Our result is the first digital signature scheme allowing the adversary to sabotage all algorithms.

Discussion. Careful readers might have noticed our general defending technique is different with previous known methods of destroying the steganographic channel. They either gave up the randomized algorithm to use deterministic ones; or using a trusted random source to re-randomize the output. While we can simply randomize the index and public parameter with even a potentially subverted hash function. A common feature of those randomized algorithms we consider in this paper is that they are only used once, i.e., generating the key/public parameter and other algorithms just use it. This property allows our simple strategy to destroy the connection to backdoors. To put it another way, we improve the state-of-the-art of defending kleptographic attacks from against essentially deterministic algorithms only to against one-time use randomized algorithms. Eventually, we would like to defend against any multi-use randomized algorithms.

4.1.2 Related work

The concept of *kleptography*—subverting cryptographic algorithms by modifying their implementations to leak secrets covertly, was proposed by Young and Yung [130,131] in 1996. They gave concrete examples showing that backdoors can be embeded into the public keys of commonly used cryptographic schemes; while the resulting public keys appear normal to the users, the adversary is nevertheless capable of learning the secret keys. It may not be surprising that defending against such deliberate attacks is challenging and only limited feasibility results exist. We next briefly describe these existing results.

In [80], Juels and Guajardo suggested the following idea: the user and a trusted certificate authority (CA) jointly generate the public key; as a part of this process, the user proves to the CA that the public key is generated honestly. This contrasts markedly with our setting, where the the user does not have any secret, and every component is provided by the big brother.

Bellare et al. considered a powerful family of kleptographic attacks that they call *algorithm substitution attacks*, and explore these in both symmetric key [11] and public key [7] settings. They first proposed a generic attack, highlighting the relevance of steganographic techniques in this framework: specifically, a sabotaged randomized algorithm can leak a secret bit-by-bit by invoking steganographic rejection-sampling; then an adversary possessing the backdoor can identify the leaked bits from the biased output, which appears unmolested to other observers. The analysis relies on the effectiveness of covert subliminal channels [73, 124, 125]. They then introduced a framework for defending against such attacks by focusing on algorithms that behave deterministically, determining a unique output for each input: relevant examples of such algorithms include unique ciphertext encryption algorithms (of encryption schemes).Their defending mechanism does not, however, address the (necessarily randomized) process of key generation—it implicitly assumes key generation to be honest. This state of affairs is the direct motivation of the current article: we adopt a significantly amplified *complete subversion model* where *all* cryptographic algorithms—including key generation—are subject to kleptographic (i.e., substitution) attacks. This forces us to manage certain randomized algorithms (such as key generation) in a kleptographic setting. The details of the model, with associated commentary about its relevance to practice, appear below.

Dodis et al. [47] studied an alternative family of kleptographic attacks on pseudorandom generators in order to formalize and study the notorious Dual_EC PRG

subversion [35, 107]. In their model, the adversary subverts the security of the PRG by opportunistically setting the public parameter while privately keeping some backdoor information (instead of providing an implementation). They prove the equivalence of such a "backdoored PRG" and public key encryption with pseudorandom ciphertexts. Then they proposed immunizing strategies obtained by applying a keyed hash function to the output (of the PRG). Note that the (hash) key plays a special role in their model: it is selected uniformly and is unknown to the adversary during the public parameter generation phase. These results likewise inspire our adoption of the amplified *complete subversion model*, which excludes such reliance on public randomness beyond the reach of the adversary. We remark that our results on strongly unforgeable OWFs can be applied to construct a specific "backdoor-free" PRG following the classic Blum-Micali framework. Moreover, our general immunizing strategy, randomizing the public parameter of a backdoored PRG instead of randomizing the PRG output, permits us to bypass an impossibility result established by Dodis et al. for general public immunization based on the PRG output.

Other works suggest different angles of defense against mass surveillance. For example, in [49, 100], the authors proposed a general framework of safeguarding protocols by randomizing the incoming and outgoing messages via a trusted (reverse) firewall. Their results demonstrate that with a clean trusted random source, many tasks become achievable. As they rely on a "subversion-free" firewall, these results require a more generous setting than provided by our *complete subversion model*.

Ateniese et al [4] continue the study of algorithm substitution attack on signatures and propose two defending mechanisms, one is using a unique signature scheme assuming the key generation and verify algorithms to be honest; the other is in the reverse firewall

model that assumes trusted randomness. Our work remove those assumptions and work in a complete subversion model.

4.2 Kleptographic Attacks and the Complete Subversion Model

In this section, we explain the kleptographic attacks in more detail and introduce the complete subversion model.

Classical cryptography assumes that the relevant cryptographic algorithms are faithfully implemented and, moreover, that participants have access to truly private randomness. In reality, cryptographic algorithms may be implemented by an adversary (e.g., the "big brother"); this potentially allows the adversary to exert malicious influence on the behavior of the algorithms and, for example, learn secret information that is not supposed to be leaked thru the algorithms. Kleptography studies how to apply these attacks on real-world cryptosystems with the extra condition that the attacks are undetectable. Specifically, while big brother wishes to monitor the system, he does not wish to be exposed.

The detection conditions. We work under the assumption that for every cryptographic scheme of interest, there exists a specification that is rigorously analyzed, proven secure, and designed by honest experts; furthermore, one can actively actively check whether implementations (supplied by the adversary) faithfully meet the specification. This "checking" procedure, however, is only assumed to have oracle access to the potentially sabotaged implementations. Following the formalization of Bellare et al. [11] of the detection condition, the adversary's algorithms must fool *all* PPT checkers.[2] Specifically, we say the adversarial subversion on cryptographic primitive Π is *undetectable*, if there

[2] Actually, we can even work in a model that relaxes the detection condition for our (trapdoor) one way functions and PRG.

exists a PPT adversary $\mathcal{A}$, for any PPT checker/detecter $\mathcal{A}$ that plays the following detection game (Figure 2), it holds that $|\Pr[b' = b] - 1/2| \leq \epsilon$, where $\epsilon = \mathsf{negl}(\lambda)$, $\mathbb{R}^1, \ldots, \mathbb{R}^k$ are the algorithms implemented for cryptographic primitive Π, and $\mathbb{R}^1_{\text{SPEC}}, \ldots, \mathbb{R}_{\text{SPEC}}$ are the corresponding specifications for Π and the probability is over the coins of the adversary $\mathcal{A}$ (for generating the backdoors or the implementations) and the detecter $\mathcal{D}$.

$\overline{\mathbb{R}}_0 \leftarrow \langle \mathbb{R}^1_{\text{SPEC}}, \ldots, \mathbb{R}^k_{\text{SPEC}} \rangle$
$\overline{\mathbb{R}}_1 = \langle \mathbb{R}^1, \ldots, \mathbb{R}^k \rangle \leftarrow \mathcal{A}$
$b \leftarrow \{0, 1\}$
$b' \leftarrow \mathcal{A}^{\overline{\mathbb{R}}_b}(\lambda)$

Figure 2: The detection game.

We emphasize that the algorithms $\langle \mathbb{R}^1, \ldots, \mathbb{R}^k \rangle$ may be randomized; indeed, the striking historical examples require the adversary to embed a (randomized) backdoor into the algorithms $\mathbb{R}^i$ in order to undetectably alter their behavior in a way that can, e.g., leak private data to the adversary. In any case, it follows that the adversary must certainly ensure that the output of any algorithm (randomized or deterministic) is computationally indistinguishable from the output of the specified algorithm, taken over any distribution of inputs of the checker's choice (and the random coins of the adversary).

We first formally present a simple observation that when a *deterministic* algorithm with a *public* input distribution (whether specified by the adversary or not) is subverted in a way that the implementation is inconsistent with the specification at a noticeable fraction of inputs, then it is easily detected.

Lemma 20. *If an adversarial subversion on deterministic algorithms* $\langle \mathbb{R}^1, \ldots, \mathbb{R}^k \rangle$ *(with the corresponding specifications* $\langle \mathbb{R}^1_{\text{SPEC}}, \ldots, \mathbb{R}^k_{\text{SPEC}} \rangle$*) is undetectable, then for every public input distribution of* $X_1, \ldots, X_k$*, where* X_i *corresponds to* $\mathbb{R}^i$*, it holds that:*

$\Pr[\mathbb{R}^i(x) \neq \mathbb{R}^i_{\text{SPEC}}(x) : x \leftarrow X_i] \leq \epsilon$, *where* $\epsilon = \mathsf{negl}(\lambda)$, *and the probability is over the coins for sampling inputs.*

Proof. Suppose there exists an implementation $\mathbb{R}^i$ for a deterministic algorithm, and a public input distribution X_i, such that, $\Pr[\mathbb{R}^i(x) \neq \mathbb{R}^i_{\text{SPEC}}(x) : x \leftarrow X_i] \geq \delta$, for a non-negligible amount δ; then there is a simple detecter algorithm $\mathcal{D}$ that samples $t = O(1/\delta^2)$ inputs $x^i_1, \ldots, x^i_t$ from X_i, and checks whether $\mathbb{R}^i(x^i_j) = \mathbb{R}^i_{\text{SPEC}}(x^i_j)$ for all $j = 1, \ldots, t$. Following the Chernoff bound, there exists a $j \in [1, t]$, such that $\mathbb{R}^i(x^i_j) \neq \mathbb{R}^i_{\text{SPEC}}(x^i_j)$ is with an overwhelming probability. □

Remark 8. *(i.) The above lemma states that for a deterministic algorithm, if the adversary wants to make the subversion undetectable, then for any public input distribution, the implementation will be inconsistent with the specification for at most a negligible fraction of the inputs; In the rest of the presentation, we sometimes use the specification directly for simplicity for deterministic algorithms with public input distribution. (ii.) The above lemma does not require the input distributions to be honestly generated.*

Corollary 21. *Assume a hash function* h_{SPEC} *is modeled as a random oracle, and* h *is a (potentially subverted) implementation. If the subversion of* h *is undetectable, then for any public input distribution* X, $\Pr[h(x) = h_{\text{SPEC}}(x) : x \leftarrow X] \leq \epsilon$, *where* $\epsilon = \mathsf{negl}(\lambda)$. *To put it in another way,* h *can be still modeled as a random oracle w.r.t to input distribution* X.

With the above detection condition only, it is impossible to achieve meaningful security for several cryptographic primitives. For example, as shown in [4,79], symmetric key encryption and signature schemes are impossible that the adversary may learn the secret key completely via the so-called "input-triggered subversion" [79]. In order to show feasibility for those primitives, we may require some extra assumption. Bellare et

al. introduced the decryptability assumption that all messages encrypted by a subverted encryption algorithm should be decrypted to the original message by the specification decryption algorithm [11]; and similarly Ateniese et al. used a verifiability assumption [4] that every signature produced by the subverted signing algorithm should pass the verification of the specification verification algorithm.

Degabriele et al. relaxed the perfect decryptability condition, and formalized a stronger detection model for symmetric key encryption in [79]. In particular, the detecter $\mathcal{D}$ may have access to the transcripts that the adversary uses to gain advantage. Subversion resistance in this case means that if the subverted algorithms leak information to the adversary, then from the recorded transcript, there exist an efficient detector that can notice the subversion. We choose this formalization for our signature and public key encryption scheme.

More formally, in the following game (Fig 3), the subversion advantage $\delta_s := |\Pr[b' = b] - 1/2|$, and the (strong) detection advantage δ_d is defined by $|\Pr[b'' = b] - 1/2|$. We say the subversion on the algorithms $\langle \mathbb{R}^1, \ldots, \mathbb{R}^k \rangle$ are strongly undetectable if δ_d is negligible; the algorithms $\langle \mathbb{R}^1, \ldots, \mathbb{R}^k \rangle$ are subversion resistant if $\delta_s \leq \delta_d$.

$$
\begin{array}{l}
\overline{\mathbb{R}}_0 \leftarrow \langle \mathbb{R}^1_{\text{SPEC}}, \ldots, \mathbb{R}^k_{\text{SPEC}} \rangle \\
\langle (\mathbb{R}^1, z^1), \ldots, (\mathbb{R}^k, z^k) \rangle \leftarrow \mathcal{A} \\
\overline{\mathbb{R}}_1 = \langle \mathbb{R}^1, \ldots, \mathbb{R}^k \rangle \\
\overline{z} = \langle z^1, \ldots, z^k \rangle \\
b \leftarrow \{0, 1\} \\
b' \leftarrow \mathcal{A}^{\overline{\mathbb{R}}_b}(\lambda, \overline{z}) \\
b'' \leftarrow \mathcal{A}^{\overline{\mathbb{R}}_b}(\lambda, \tau)
\end{array}
$$

Figure 3: The strong detection and subversion game, τ is the transcript of $\mathcal{A}$ for producing b'.

Remark 9. *(i.) The strong detection model essentially describes the situation that the "big brother" is worried about any possible detection. (ii.) When we use the stronger detection condition for signature and public key encryption scheme, the detection is public (verifying the signatures or encrypting the messages using a public key), this is in contrast with [11, 79] that the detecter needs the user secret key.*

Complete subversion model. Significant effort has been invested to defend against such kleptographic attacks. However, the state-of-the-art of defending mechanisms require the participants to have access to various trusted or private sources of randomness. For example, as described above, key generation is assumed to be honest in [11]. Unfortunately, key generation, as shown in the original papers of Young and Yung, [130, 131], can be directly subjected to kleptographic attacks. This motivates our focus on the *complete subversion model*, in which attacks may be launched against *any* of the relevant cryptographic algorithms. This includes, e.g., the *key generation* algorithm, and even the *defending algorithm*. We remark that the immunizing strategy of [47] requires an honestly generated uniform seed unknown to the adversary when she implements the algorithms. We will define security for each primitive we consider in this complete subversion model. Likewise, we analyze our defending mechanisms in this stringent model. We call this general paradigm **cliptography**.

4.3 One-Way Functions in the Complete Subversion Model

4.3.1 Formalizing kleptographic attacks on one way functions

Before studying the security for one-way functions (OWF) and trapdoor one-way functions (TDOWF) in the presence of kelptographic attacks, we first recall the conventional definitions of OWF and TDOWF.

One-way function (OWF). A function family $\mathcal{F} = \{f_i : X_i \to Y_i\}_{i \in I}$ is *one-way* if there are PPT algorithms $(\mathsf{KG}, \mathsf{Eval})$ so that (i) KG, given a security parameter λ, outputs a function index i from $I_\lambda = I \cap \{0,1\}^\lambda$; (ii) for $x \in X_i$, $\mathsf{Eval}(i, x) = f_i(x)$; (iii) $\mathcal{F}$ is one-way; that is, for any PPT algorithm $\mathcal{A}$, it holds that $\Pr[\mathcal{A}(i, y) \in f_i^{-1}(y) \mid i \leftarrow \mathsf{KG}(\lambda); x \leftarrow X_i; y := f_i(x)] \leq \mathsf{negl}(\lambda)$.

Trapdoor one way function (TDOWF). A function family $\mathcal{F} = \{f_i : X_i \to Y_i\}_{i \in I}$ is *trapdoor one-way* if there are PPT algorithms $(\mathsf{KG}, \mathsf{Eval}, \mathsf{Inv})$ such that (i) KG, given a security parameter λ, outputs a function index and the corresponding trapdoor pair (i, t_i) from $I_\lambda \times T$, where $I_\lambda = I \cap \{0,1\}^\lambda$, and T is the domain of t_i; (ii) $\mathsf{Eval}(i, x) = f_i(x)$ for $x \in X_i$; (iii) $\mathcal{F}$ is one-way; and (iv) it holds that $\Pr[\mathsf{Inv}(t_i, i, y) = x \mid i \leftarrow \mathsf{KG}(\lambda); x \leftarrow X_i; y := f_i(x)] \geq 1 - \mathsf{negl}(\lambda)$.

We note that, (trapdoor) one-way permutations can be defined similarly by setting $Y_i := X_i$. For simplicity, we often ignore Eval; for evaluating function with index i on input x, i.e., $\mathsf{Eval}(i, x)$, we write it as $f_i(x)$.

4.3.1.1 Strong-forgeability and unforgeability for OWF/TDOWF

In this subsection, we define *strong-forgeability* to capture kleptographic attacks on one-way functions. Note that the complemented security notion, *unforgeability*, provides the guarantee that OWF is immune to kleptographic attacks. Similarly, we can define strong-forgeability and the complemented notion, unforgeability, for trapdoor OWF. Next, let's start with expressing the intuition of capturing kleptographic attacks on one-way functions.

We begin with a "laboratory specification" version of the OWF, $(\mathsf{KG}_{\text{SPEC}}, \mathsf{Eval}_{\text{SPEC}})$, which has been rigorously analyzed and certified (e.g., by the experts in the cryptography community).

The adversary then provides an alternate implementation. Note that Eval is deterministic on publicly known input distribution ($i \times U_i$, where i is the output of KG, and U_i is the uniform distribution over X_i). Following lemma 20, the event that an input is sampled from KG and the X_i s.t., the implementation Eval is inconsistent with $\mathsf{Eval}_{\text{SPEC}}$ will happen with only a negligible probability. It follows that using $\mathsf{Eval}_{\text{SPEC}}$ directly for Eval will reduce the adversary advantage at most a negligible amount, thus we can consider it as honestly implemented.We therefore focus on KG (the algorithm which generates a function name).

The goal of the adversary is to privately maintain some "backdoor information" z so that the subverted implementation of KG will output functions that can be inverted using z. In addition, the adversary must be sure that the output distributions of $\mathsf{KG}(z)$ and that of specification function generation algorithm $\mathsf{KG}_{\text{SPEC}}$ are computationally indistinguishable, to avoid detection. Formally, we define *strongly-forgeable OWFs*; we note that this immediately allows us to define the complemented notion, *unforgeable OWFs*.

Definition 8. *A one-way function family $\mathcal{F} = \{f_i : X_i \to Y_i\}_{i \in I}$ (with the specification function generation algorithm $\mathsf{KG}_{\text{SPEC}}$) is* δ**-strongly forgeable** *if there exist PPT algorithms $(\mathsf{BG}, \mathsf{KG}, \mathsf{Inv})$ so that given a backdoor z produced by backdoor generation algorithm BG, i.e., $z \leftarrow \mathsf{BG}(\lambda)$, the function generation algorithm KG generates a function index i that is (1) invertible given z, and (2) computationally indistinguishable from that generated by the specification function generation algorithm $\mathsf{KG}_{\text{SPEC}}$. That is, for every $z \leftarrow \mathsf{BG}(\lambda)$, it holds that:*

(1) $\Pr[x' = x \mid i \leftarrow \mathsf{KG}(\lambda, z); x \leftarrow X_i; y := f_i(x); x' \leftarrow \mathsf{Inv}(z, i, y)] \geq \delta$

(2) $\{i \mid i \leftarrow \mathsf{KG}(\lambda, z)\} \overset{c}{\approx} \{i \mid i \leftarrow \mathsf{KG}_{\text{SPEC}}(\lambda)\}$

Correspondingly, we say a one-way function family is ***unforgeable*** *if it is* not *δ-strongly forgeable for any non-negligible function δ.*

The notion of strongly-forgeable OWF is closely related to the conventional notion, TDOWF. See the following lemmas. These results state that if we want to use public key cryptography, we have to accept the possibility of kleptographic attacks on OWFs.

Lemma 22. *A $(1-\epsilon)$-strongly forgeable OWF family is also a TDOWF family, where ϵ is a negligible function of the security parameter.*

Proof. Suppose $\mathcal{F} := (\mathsf{BG}_{\mathcal{F}}, \mathsf{KG}_{\mathcal{F}}, \mathsf{Inv}_{\mathcal{F}})$ is a $(1-\epsilon)$ strongly-forgeable OWF family. We can define $(\mathsf{KG}, \mathsf{Eval}, \mathsf{Inv})$ for the TDOWF family as follows. The generation algorithm KG is as follows: first run $\mathsf{BG}_{\mathcal{F}}(\lambda)$ and receive a string z; then run $\mathsf{KG}_{\mathcal{F}}$ with input (λ, z) and receive a function index i; finally output (i, z). The inversion algorithm Inv for the TDOWF is simply $\mathsf{Inv}_{\mathcal{F}}$, and the evaluation algorithm is defined as $\mathsf{Eval}(i, x) = f_i(x)$.

We can easily see that $(\mathsf{KG}, \mathsf{Eval}, \mathsf{Inv})$ is invertible once trapdoor is given: since $\mathcal{F}$ is a $(1-\epsilon)$ strongly-forgeable OWF family, it holds that by definition, $\Pr[\mathsf{Inv}_{\mathcal{F}}(z, i, y) = x \mid x \leftarrow X_i; y := f_i(x)] \geq 1-\epsilon$; therefore, $\Pr[\mathsf{Inv}(i, y, z) = x \mid x \leftarrow X_i; y := f_i(x)] \geq 1-\epsilon$.

We can also show the one-wayness: without z, no PPT algorithm can invert y for a random x; otherwise assume $(\mathsf{KG}, \mathsf{Eval}, \mathsf{Inv})$ is not one-way, then there exists an adversary $\mathcal{A}$ who for $i \leftarrow \mathsf{KG}_{\mathcal{F}}$, can invert $y := f_i(x)$ with non-negligible probability. We note that the specification function generation algorithm is one-way, i.e., for $i \leftarrow \mathsf{KG}_{\text{SPEC}}$, no one can invert $y := f_i(x)$ except negligible probability. Now, one can distinguish the output distribution of $\mathsf{KG}_{\mathcal{F}}(z)$ from the output distribution of $\mathsf{KG}_{\text{SPEC}}$, simply by trying inversion using $\mathcal{A}$. □

Next, we show how to construct a strongly-forgeable OWF from any TDOWF. This substantiates the folklore knowledge that sufficient random padding can render

cryptosystems vulnerable to backdoor attacks, e.g., [130, 131]. Specifically, the random padding in the malicious implementation can be generated so that it encrypts the corresponding trapdoor using the backdoor as a key.

Lemma 23. *One can construct a $(1-\epsilon)$-strongly forgeable OWF from a TDOWF, where ϵ is a negligible function on the security parameter.*

Proof. Consider a TDOWF $\mathcal{F} = \{f_i\}$ with the associated algorithms $(\mathsf{KG}_{\mathcal{F}}, \mathsf{Inv}_{\mathcal{F}})$. Assuming the trapdoors can be represented using $\ell(\lambda)$ bits, we construct a strongly forgeable OWF family $\mathcal{G} = \{g_{i,r}\}$, where $g_{i,r}(x) = f_i(x)||r$ and $r \in \{0,1\}^{\ell(\lambda)}$.

The specification version of the sfunction generation algorithm $\mathsf{KG}_{\text{SPEC}}$ is defined as follows: run the $\mathsf{KG}_{\mathcal{F}}$ algorithm and receive a function index/trapdoor pair (i, t_i); then discard t_i and sample randomly $r \leftarrow \{0,1\}^{\ell(\lambda)}$; finally output (i, r). It is easy to see that $g_{i,r}$ is one way because f_i is one way (without t_i).

While for the backdoored implementations, BG first outputs a random key k for a symmetric key encryption scheme $\mathsf{SE} = (\mathsf{SE.Enc}, \mathsf{SE.Dec})$ which is assumed to be a pseudorandom permutation (PRP). $\mathsf{KG}(k)$ is defined as follows: it first runs $\mathsf{KG}_{\mathcal{F}}$, and receives an index i together with the corresponding trapdoor t_i; the second part $\tilde{r}$ is generated by encrypting t_i using k, i.e., $\tilde{r} = \mathsf{SE.Enc}(k, t_i)$. $\mathsf{KG}(k)$ outputs $(i, \tilde{r})$. $(g_{i,\tilde{r}}(x) = f_i(x)||\mathsf{SE.Enc}(k, t_i))$. It is easy to see that with the backdoor k, one can define the Inv as follows: it first decrypts $\tilde{r}$ using k to retrieve t_i, and then inverts $f_i(x)$ by running $\mathsf{Inv}_{\mathcal{F}}$ with t_i as an input.

Furthermore, since $\mathsf{SE.Enc}$ is modeled as a PRP, the distributions of (i, r) returned by $\mathsf{KG}_{\text{SPEC}}$ and $(i, \tilde{r})$ returned by $\mathsf{KG}(k)$ for any k are computationally indistinguishable. □

The notions of strongly-forgeable OWF and TDOWF are similar in the sense that they both posit a secret that enables inversion of the OWF. Observe, however,

that a strongly-forgeable OWF has a further (critical) property: the distribution of function names, is indistinguishable from a particular specification distribution. The same principle yields a notion of strongly-forgeable TDOWF. Moreover, the adversary can invert using the backdoor, without referring to the regular trapdoor. The formal definition is presented below.

Definition 9. *A trapdoor one-way function family* $\mathcal{F} = \{f_i : X_i \to Y_i\}_{i \in I}$ *(with the specification algorithms* $\mathsf{KG}_{\text{SPEC}}, \mathsf{Inv}_{\text{SPEC}}$*) is* δ***-strongly forgeable*** *if there exist PPT adversarial algorithms* $\mathcal{A} = (\mathsf{BG}, \mathsf{KG}, \mathsf{Inv})$ *so that given a backdoor* z *produced by backdoor generation algorithm* BG*, the function generation algorithm* KG *generates a function index* i *and the corresponding trapdoor* t_i*, with the following properties: (1)* f_i *is invertible given* z *without providing* t_i*, and (2) the output of* KG *is computationally indistinguishable from that generated by the specification function generation algorithm* $\mathsf{KG}_{\text{SPEC}}$*. That is, for every* $z \leftarrow \mathsf{BG}(\lambda)$*, it holds that*

(1) $\Pr[x' = x \mid (i, t_i) \leftarrow \mathsf{KG}(\lambda, z); x \leftarrow X_i; y := f_i(x); x' \leftarrow \mathsf{Inv}_{\text{SPEC}}(z, i, y)] \geq \delta$ [3]

(2) $\{(i, t_i) \mid (i, t_i) \leftarrow \mathsf{KG}(\lambda, z)\} \overset{c}{\approx} \{(i, t_i) \mid (i, t_i) \leftarrow \mathsf{KG}_{\text{SPEC}}(\lambda)\}$

Correspondingly, we say a trapdoor one-way function family is ***unforgeable*** *if it is* not *δ-strongly forgeable for any non-negligible function δ.*

4.3.1.2 Forgeability and strong-unforgeability for OWF

The notion of strong forgeability models an attack where the adversary may provide a subverted implementation of the defining algorithms. In many cases, it may also be

[3] The inversion algorithm of the TDOWF never appears in the security definition, thus we omit the indistinguishability condition for this algorithm, considering it is honestly implemented as $\mathsf{Inv}_{\text{SPEC}}$, and focus only on KG.

interesting to consider a weaker form of attack that the adversary simply provides the function index, this is similar to the notion of backdoored PRG that the adversary sets up the public parameters (as in the Dual_EC PRG example [35]). This is a weaker definition in that the adversary only have to generate a backdoor that can be helpful for inverting one single function instead of a family of functions (as in the case of strongly-forgeable OWF). It is easy to see that this notion is equivalent to the standard notion of TDOWF if the $\mathsf{KG}_{\text{SPEC}}$ outputs the same distribution of the function index as the regular generation algorithm of the TDOWF, thus we only consider this notion for OWF. However, putting into the context of kleptography, it is suggested that when the experts recommend standard for OWF, TDOWF should not be a good candidate.

More importantly, we will later consider unforgeability for OWF, thus the reverse of the weaker notion will yield a stronger definition for unforgeability, and we will give a generic construction that converts any OWF into a strongly unforgeable OWF, which means we can destroy the trapdoor structure generically.

Definition 10. *A one-way function family* $\mathcal{F} = \{f_i : X_i \to Y_i\}_{i \in I}$ *(with the specification of function generation algorithm* $\mathsf{KG}_{\text{SPEC}}$*) is* δ**-forgeable** *if there exist PPT adversarial algorithms* $\mathcal{A} = (\mathsf{KG}, \mathsf{Inv})$ *so that*

1. $\Pr[x' = x \mid (i, z) \leftarrow \mathsf{KG}(\lambda); x \leftarrow X_i; y \leftarrow f_i(x); x' \leftarrow \mathsf{Inv}(z, i, y)] \geq \delta$
2. *The distribution of* i *is indistinguishable from the output distribution of* $\mathsf{KG}_{\text{SPEC}}$.

Correspondingly, we say a one-way function family is ***strongly-unforgeable*** *if it is not* δ*-forgeable for any non-negligible function* δ.

4.3.2 Eliminating backdoors

In this section, we discuss methods for safeguarding OWF generation against kleptographic attacks. We first present a general approach that immunizes any OWF generation procedure. We prove that hashing the function index is sufficient to eliminate potential backdoor information. However, in many cases of interest, function indices have specific algebraic structure. It is not clear in general how one can guarantee that a public hash function has such "structure preserving" properties. In order to apply our approach in more realistic settings, we propose a "split-program" model in which the function generation algorithm is necessarily composed of two parts: a random string generation algorithm RG that outputs random bits r, and a deterministic function index generation algorithm dKG which uses r to generate the index.

Before going to the details of our feasibility results, we remark that the definitions of (strongly) unforgeable OWFs can be found in Definitions 8 and 10. The definitions essentially insist the sabotaged KG looks the same as the specification $\mathsf{KG}_{\text{SPEC}}$ which is rigorously analyzed. To put it another way, invertibility in the backdoored mode will be restricted by the indistinguishability condition for the function index.

4.3.2.1 General feasibility results

We will show below that randomizing the function index (that is, the relationship between names and functions) can provide satisfactory immunization against possibly sabotaged function generation. The intuition behind this idea is that according to the definition of forgeable OWF, each backdoor that frequently appears can only be useful for inverting a sparse subset of one way functions (i.e., the range of $\mathsf{KG}(z)$ is exponentially sparse for every z, otherwise, one can use such backdoor to break the one-wayness of the functions generated by $\mathsf{KG}_{\text{SPEC}}$). Thus, randomizing the function index will map

the function index to a "safe" domain, and destroys the possible correlation with any selected backdoor. That said, it is difficult for the adversary to arrange a backdoor that works for a disturbed subset of function index (after hashing), even if she knows the immunizing strategy.

Constructing strongly-unforgeable OWFs. Given any OWF family $\mathcal{F} := (\mathsf{KG}_{\mathcal{F}}, \mathsf{Eval}_{\mathcal{F}}) := \{f_i\}_{i\in I}$ (which might be forgeable, and with specification $\mathsf{KG}_{\text{SPEC}}, \mathsf{Eval}_{\text{SPEC}}$) that is secure if $\mathsf{KG}_{\text{SPEC}}(\lambda)$ outputs uniform i from I_λ, we will construct a strongly unforgeable OWF family $\mathcal{G} := (\mathsf{KG}_{\mathcal{G}}, \mathsf{Eval}_{\mathcal{G}}) = \{g_i\}$. We assume a (public) hash function family $H_\lambda = \{h_\lambda : I_\lambda \to I_\lambda\}$ modeled as a random oracle (for each λ), and that h is randomly sampled from H_λ. The key generation algorithm $\mathsf{KG}_{\mathcal{G}}$ is the same as $\mathsf{KG}_{\mathcal{F}}$, and $\mathsf{Eval}_{\mathcal{G}}$ is given as $\mathsf{Eval}_{\mathcal{F}} * h$, and it is defined as $\mathsf{Eval}_{\mathcal{G}}(i, x) := \mathsf{Eval}_{\mathcal{F}}(h(i), x)$. (thus for each i, $g_i(\cdot) = f_{h(i)}(\cdot)$). [4] See also the pictorial illustration in Fig 4.

$$\boxed{\mathsf{KG}_{\mathcal{F}}} \to i \qquad i \to \boxed{h} \to \tilde{i} \qquad x \to \boxed{\mathsf{Eval}_{\mathcal{F}}(\tilde{i}, \cdot)} \to y$$

Figure 4: Immunization strategy for OWF.

Remark 10. *In the case that we can build hash functions that hash directly onto the index space I_λ, then we can use the above immunization strategy. However, in practice, it is not clear if we can always easily build hash functions on index space. To address this issue, in Section 4.3.2.2, we introduce a new framework called "split-program model". There, we can use standard hash function, e.g., SHA-256 to work on the random bits directly.*

[4] Note that the hash function can be implemented by the adversary, since it is deterministic and with a public input distribution (the output distribution of $\mathsf{KG}_{\mathcal{F}}$). We can also interpret the specifications for $\mathcal{G}$ as $(\mathsf{KG}_{\text{SPEC}}, \mathsf{Eval}_{\text{SPEC}} * h_{\text{SPEC}})$.

Theorem 24. *The OWF family $\mathcal{G}$ defined above is strongly unforgeable in the random oracle model.*

Proof. Suppose that $\mathcal{G}$ is δ-forgeable for a non-negligible function δ, and let $(\mathcal{A}_{\mathsf{KG}}, \mathcal{A}_{\mathsf{Inv}})$ be the adversarial algorithms of Definition 10. We will construct a simulator $\mathcal{S}$ which will break the one-way security of $\mathcal{F}$.

Suppose $(f_{\tilde{i}}, y)$ are the challenges $\mathcal{S}$ receives from the one way security challenger $\mathcal{C}$, where $y = f_{\tilde{i}}(x)$ for a randomly selected x. Then (i.) $\mathcal{S}$ first randomly samples a bit b to decide whether to embed j into the answers to the random oracle queries from $\mathcal{A}_{\mathsf{KG}}$ or $\mathcal{A}_{\mathsf{Inv}}$. (W.l.o.g., we assume all the random oracle queries are different.) (ii.) $\mathcal{S}$ runs $\mathcal{A}_{\mathsf{KG}}$. Suppose $\mathcal{A}_{\mathsf{KG}}$ makes q_1 random oracle queries $Q_1 = \{i_1, \ldots, i_{q_1}\}$. If $b = 0$, $\mathcal{S}$ randomly selects an index $t_1 \in \{1, \ldots, q_1\}$. When answering the random oracle queries $i_1, \ldots, i_{q_1}$ from $\mathcal{A}_{\mathsf{KG}}$, $\mathcal{S}$ answers $\tilde{i}$ for $h(i_{t_1})$; for all other queries, $\mathcal{S}$ answers with random elements from the index set I_λ. If $b \neq 0$, $\mathcal{S}$ answers all these queries using random elements from I_λ. $\mathcal{S}$ maintains a list for the query-answer pairs. $\mathcal{A}_{\mathsf{KG}}$ outputs a pair (i, z).

If $[b = 0 \wedge i \neq i_{t_1}]$, $\mathcal{S}$ aborts; otherwise, $\mathcal{S}$ runs $\mathcal{A}_{\mathsf{Inv}}$ with inputs (i, y, z). Assuming $\mathcal{A}_{\mathsf{Inv}}$ asks q_2 random oracle queries, $\mathcal{S}$ sets $\tilde{i}$ as $h(i)$ (even if i is not asked) and for all others queries, $\mathcal{S}$ answers with random elements from I_λ. $\mathcal{A}_{\mathsf{Inv}}$ outputs x'.

If $[b = 1 \wedge i \in Q_1]$, $\mathcal{S}$ aborts; otherwise, it returns x' as his answer to $\mathcal{C}$.

Probabilistic analysis. Now we bound the success probability of $\mathcal{S}$. Let us use W to denote the event that $\mathcal{S}$ aborts, W_1 to denote the event that $b = 0 \wedge i \neq i_{t_1}$, and W_2 to denote the event that $b = 1 \wedge i \in Q_1$. We have

$$\Pr[x' = x] = \Pr[x' = x|W] \Pr[W] + \Pr[x' = x|\overline{W}] \Pr[\overline{W}] \geq \Pr[x' = x|\overline{W}] \Pr[\overline{W}]$$

We first bound $\Pr[\overline{W}]$ as $1-\Pr[W]$, we have $\Pr[W]=\Pr[W_1\vee W_2]\leq \Pr[W_1]+\Pr[W_2]$. Assuming $\Pr[i\in Q_1]=\eta$, we bound $\Pr[W_1]$ as follows:

$$\begin{aligned}\Pr[W_1] &= \Pr[b=0\wedge i\neq i_t]=\Pr[b=0]\Pr[i\neq i_{t_1}]\\ &= \frac{1}{2}(\Pr[i\neq i_{t_1}|i\in Q_1]\Pr[i\in Q_1]+\Pr[i\neq i_{t_1}|i\notin Q_1]\Pr[i\notin Q_1])\\ &= \frac{1}{2}\left[\left(1-\frac{1}{q_1}\right)\eta+(1-\eta)\right]\end{aligned}$$

While $\Pr[W_2]=\Pr[b=1]\Pr[i\in Q_1]=\eta/2$, we have: $\Pr[W]\leq \frac{1}{2}\left(1-\frac{1}{q_1}\right)\eta+\frac{1}{2}\leq 1-\frac{1}{2q_1}$. Thus we can derive that $\Pr[\overline{W}]\geq 1/(2q_1)$.

Furthermore, conditioned on $\mathcal{S}$ not aborting, the input distributions of the adversaries $(\mathcal{A}_{\mathsf{KG}},\mathcal{A}_{\mathsf{Inv}})$ are identical to that of Definition 10. By definition of δ-forgeability, $\Pr[\mathcal{A}_{\mathsf{Inv}}(\tilde{i},f_{\tilde{i}}(x),z)=x]\geq\delta$, thus $\Pr[x'=x|\overline{W}]\geq\delta$.

Combing these facts, we conclude that if $\mathcal{G}$ is δ-forgeable for some non-negligible δ, then there exists an algorithm $\mathcal{S}$ that breaks the one-way security of $\mathcal{F}$ with probability at least $\delta/(2q_1)$ (which is non-negligible). This completes the proof. □

Remark 11. *(i.) Actually, it is not very hard to see from our analysis that we can even relax the detection condition for* KG *that it only has to pass one particular tester. (ii.) We may also prove a similar result in the standard model, that the random oracle is instantiated with a pseudorandom function* PRF*. However, in this case, the adversary* KG *can only have oracle access to the* PRF*, but the* Inv *can have full access to the* PRF *key. Still, this result will not be in the complete subversion model that the* PRF *requires a trusted key. Thus we leave as an open problem that how to establish a general immunizing result in the standard model. Furthermore, the above proof works even if the distribution of i is different with the output distribution of* $\mathsf{KG}_{\mathrm{SPEC}}$*, as long as it still belongs to the index set.*

4.3.2.2 Practical results in the split-program model

Indices (names) of a one-way function family may have structure. For example, for OWF based on discrete logarithm, $f_{g,p}(x) = g^x \bmod p$, the function index consists of an algebraically meaningful pair (p, g), where p is a prime and g a random generator. This would require that the hash function in the general immunization method above maps (g, p) to (g', p'); note that (g', p') is another algebraically meaningful pair with the same structure. Furthermore, for a TDOWF, the hash function needs to map the function/trapdoor pair to another function/trapdoor pair. It is not clear in general how one can guarantee that a public hash function has such "structure preserving" properties.

To address this problem, we propose a split-program model in which every function generation algorithm is composed of two algorithms, a random string generation algorithm RG that outputs a uniform ℓ-bit random string r, and a deterministic function index generation algorithm dKG that transforms the randomness r into a function index i. In this model, dKG is deterministic with a public input distribution (output distribution of RG). Following lemma 20 and the elaboration in section 4.3.1.1, we can consider it to be honestly implemented and we can focus on "cleaning up" the randomness generated by RG. [5]

Remark 12. *It is not hard to see, the split-program model is quite general and can be applied to most practical algorithms. To see this, the user gets the source code of the implementation, which makes calls to some API for generating randomness (e.g.,*

[5] The split-program model essentially forces the adversary to concentrate the parts which may potentially contain backdoors of a malicious implementation into RG. This gives us more flexibility to apply the sanitizing strategy. Furthermore, conceptually, the immunizing strategy itself (e.g., Fig 4) can be seen as a bigger piece of implementation in the split-program model, i.e., the hash function and the actual KG algorithm should be individually implemented and checked.

`rand()`*) whenever necessary. The user can hook up the interface with the calls to the API with the separate program* RG *provided by the big brother. In principle, one can always augment a randomized* KG *algorithm to output the function index* i *together with the randomness* r *used to generate* i*, thus the* RG *can be implemented as this augmented* KG*, and discards the function index* i *from the output.*

We first rephrase the standard OWF/TDOWF definitions in the split-program model.

Definition 11. *A function family* $\mathcal{F}$ *is one way in the split-program model if there exist a pair of algorithms* $(\mathsf{RG}, \mathsf{dKG})$ *where (i.)* RG*, given a security parameter* λ*, outputs a uniform* $\ell(\lambda)$*-bit string* r*; (ii.)* dKG *is deterministic: given the randomness* r *it outputs a function index* $i \in I_\lambda$*; and (iii.)* $\mathcal{F}$ *is one-way under this procedure for generating* i*.*

Similarly, we can define a TDOWF family in the split program model (In this case, dKG outputs a function index together with a trapdoor). For the ease of presentation, we often use the pair of algorithms $(\mathsf{RG}, \mathsf{dKG})$ to represent the OWF/TDOWF family $\mathcal{F}$ in the split-program model. Next we define δ-forgeable OWF in the split-program model by modifying Definition 10; we immediately have the complemented security notion of strongly-unforgeable OWF in the split-program model. [6]

Definition 12. *A one-way function family* $\mathcal{F} = \{f_i : X_i \to Y_i\}_{i \in I}$ *(with the specification version of function generation algorithm* $\mathsf{RG}_{\text{SPEC}}, \mathsf{dKG}_{\text{SPEC}}$*) is* δ***-forgeable*** *if there exist* PPT *algorithms* $(\mathsf{RG}, \mathsf{dKG}, \mathsf{Inv})$ *such that:*

[6] Note that it is easy to provide a more general definition to capture the fact dKG is also implemented by the adversary, we can simply require the indistinguishability condition hold for the joint output distribution of $(\mathsf{RG}, \mathsf{dKG})$. However as pointed out above, it is fine for us to consider the deterministic dKG as honestly implemented for simplicity, because it has the public input distribution from $h \circ \mathsf{RG}$.

1. $\Pr[x' = x \mid (r,z) \leftarrow \mathsf{RG}(\lambda); i \leftarrow \mathsf{dKG}_{\text{SPEC}}(\lambda, r, z); x \leftarrow X_i; y := f_i(x); x' \leftarrow \mathsf{Inv}(z, i, y)] \geq \delta$.

2. *The distribution of r is indistinguishable from the output distribution of $\mathsf{RG}_{\text{SPEC}}$.*

Correspondingly, we say a one-way function family is ***strongly-unforgeable*** *in the split-program model if it is not δ-forgeable in the split-program model for any non-negligible function δ.*

Strongly-unforgeable OWF in the split-program model.

Given a OWF family $\mathcal{F} := (\mathsf{RG}_{\mathcal{F}}, \mathsf{dKG}_{\mathcal{F}}, \mathsf{Eval}_{\mathcal{F}})$ (in the split program model) whose $\mathsf{RG}_{\text{SPEC}}$ outputs uniform bits, and a public hash function $h(\cdot)$ randomly selected from a hash family $H : \{0,1\}^{\ell(\lambda)} \to \{0,1\}^{\ell(\lambda)}$ which is modeled as a random oracle, we construct an strongly-unforgeable OWF family $\mathcal{G}$. $\mathcal{G} := (\mathsf{RG}_G, \mathsf{dKG}_{\mathcal{G}}, \mathsf{Eval}_{\mathcal{G}})$ can be described as $(\mathsf{RG}_{\mathcal{F}}, \mathsf{dKG}_{\mathcal{F}} \circ h, \mathsf{Eval}_{\mathcal{F}})$, i.e., $\mathsf{dKG}_{\mathcal{F}} \circ h(r) = \mathsf{dKG}_{\mathcal{F}}(h(r))$ the function index i is generated by $\mathsf{dKG}_{\mathcal{F}}$ using $h(r)$ as randomness, where $r \leftarrow \mathsf{RG}_{\mathcal{F}}$. See Figure 5 below:[7]

$$\boxed{\mathsf{RG}_{\mathcal{F}}} \to r \qquad r \to \boxed{h} \to \tilde{r} \qquad \tilde{r} \to \boxed{\mathsf{dKG}_{\mathcal{F}}} \to \tilde{i} \qquad x \to \boxed{\mathsf{Eval}_{\mathcal{F}}(\tilde{i}, \cdot)} \to y$$

Figure 5: Immunization strategy for OWF in the split-program model.

The intuition is analogous to that captured in the proof of Theorem 24: For a backdoor z that frequently appears in the output of $\mathcal{A}_{\mathsf{RG}}$, the "bad" set of randomness (i.e., for which the dKG algorithm outputs a function that can be efficiently inverted using z) should be sparse in $\{0,1\}^{\ell(\lambda)}$. (Otherwise, one can break the one way security

[7] We emphasize that in the split-program model, we can use any regular hash function such as SHA-256. Furthermore, this can be implemented by the adversary, as h has a public input distribution (the output distribution of RG). This in turn guarantees dKG has a public input distribution (output distribution of $h \circ \mathsf{RG}$).

of $\mathcal{F}$ by simply running $\mathcal{A}_{\mathsf{KG}}$ to get z.) In this case, the probability that a uniform random string falls into the "bad" set that z is useful for inverting is negligible. Hashing the random bits will then break the delicate connection between the backdoor and the function index that will be generated using the cleaned randomness. Specifically, it will be challenging for the adversary to design an efficient connection between a backdoor z and the "scrambled" sets of functions backdoored by z.

Theorem 25. *The OWF family $\mathcal{G}$ described above is strongly unforgeable in the split-program model if H is modeled as a random oracle.*

Proof. First, the hash function h takes the output distribution of RG as input distribution; thus following corollary 21 we can still think it as a random oracle.

Next, we will argue the following: assume $\mathcal{G}$ is δ-forgeable in the split program model, i.e., there exist PPT adversaries $(\mathcal{A}_{\mathsf{RG}}, \mathcal{A}_{\mathsf{dKG}}, \mathcal{A}_{\mathsf{Inv}})$ satisfying Definition 12. Then we can construct a simulator $\mathcal{S}$ that breaks the one-way security of $\mathcal{F}$ in the split-program model.

Suppose r^* is the randomness and $y = f_i(x)$ is the challenge value (for a randomly chosen x) received from the one-way security challenger $\mathcal{C}$, where $i = \mathsf{dKG}(r^*)$.

$\mathcal{S}$ first chooses a random bit b and runs $\mathcal{A}_{\mathsf{RG}}$, which asks random oracle queries $\{r_1^0, \ldots, r_{q_0}^0\} := Q_0$. If $b = 0$, $\mathcal{S}$ randomly selects $t_0 \in \{1, \ldots, q_0\}$, and answers $h(r_{t_0}^0) = r^*$; all other queries are answered with uniform ℓ-bit strings. If $b \neq 0$, $\mathcal{S}$ answers all queries with random strings. $\mathcal{S}$ maintains a list for the query-answer pairs.

If $[b = 0 \wedge r \neq r_{t_0}^0]$, $\mathcal{S}$ aborts; otherwise, it sets $h(r) = r^*$ and it runs $\mathcal{A}_{\mathsf{KG}}$ with inputs r^*, and expects $\mathcal{A}_{\mathsf{KG}}$ to output i. $\mathcal{S}$ then runs $\mathcal{A}_{\mathsf{Inv}}$ with inputs (i, y, z) and receives the response x'.

If $b = 1 \wedge r \in Q_0$, $\mathcal{S}$ aborts; otherwise, it sends x' to the challenger $\mathcal{C}$ as his answer.

Similar to the proof of Theorem 24, let W denote the event that $\mathcal{S}$ aborts: we can bound $\Pr[\overline{W}] \leq 1 - \frac{1}{q_0}$. Thus,

$$\Pr[x' = x] \geq \Pr\left[x' = x \,|\, \overline{W}\right] \Pr\left[\overline{W}\right] \geq \frac{\Pr\left[x' = x \,|\, \overline{W}\right]}{2q_0}.$$

While following Definition 12,

$$\Pr[x' = x|\overline{W}] = \Pr[\mathcal{A}_{\mathsf{Inv}}(z, i, f_i(x)) = x] \geq \delta.$$

To summarize, in the split-program model, if $\mathcal{G}$ is δ-forgeable, $\mathcal{S}$ will break the one way security of $\mathcal{F}$ with probability at least $\delta/(2q_0)$. □

Constructing unforgeable TDOWFs in the split-program model. More interestingly, we can apply the same method to immunize a TDOWF in the split-program model, whose $\mathsf{RG}_{\text{SPEC}}$ outputs uniform bits r and whose (deterministic) $\mathsf{dKG}_{\text{SPEC}}$, given r, outputs a function index and a trapdoor pair. However, in this case, we have to assume the implementation RG to be stateless. The detection condition can guarantee that the output of RG is unpredictable to the adversary. Then, similar to the OWF case, hashing the randomness yields a uniform bitstring, and it ensures that the resulting function index together with the corresponding trapdoor will be "safe." [8] As pointed out in Section 4.3.1, it is preferable to consider strong forgeability for TDOWF (since a forgeable TDOWF can be seen as itself), thus we adapt Definition 9 to the split-program model: [9]

[8] It is easy to see that if RG is stateful and it maintains a counter state, ctr, there is a simple attack that $\mathsf{RG}(z, ctr)$ runs $\mathsf{PRF}_z(ctr)$, where z is the backdoor and used as the seed for the PRF. In this case, the output of RG is completely known to the adversary, and also the trapdoor corresponding to the outputted function index.

[9] Again, since dKG is deterministic with public input distributions, for simplicity we consider they are honestly implemented as $\mathsf{dKG}_{\text{SPEC}}$, even though they are implemented by the adversary.

Definition 13. *A trapdoor one-way function family $\mathcal{F} = \{f_i : X_i \to Y_i\}_{i \in I}$ (with the specification algorithms $\mathsf{RG}_{\text{SPEC}}, \mathsf{dKG}_{\text{SPEC}}$) is δ-**strongly forgeable** if there exist* PPT *adversarial algorithms $\mathcal{A} = (\mathsf{BG}, \mathsf{RG}, \mathsf{dKG}, \mathsf{Inv})$ so that given a backdoor z produced by backdoor generation algorithm BG, the random string generation algorithm RG generates an $\ell(\lambda)$-bit random string r, and based on such r the deterministic function generation algorithm dKG generates a function index i and the corresponding trapdoor t_i, with the following properties: (1) f_i is invertible given z without providing t_i, and (2) the output distribution of RG is computationally indistinguishable from that generated by the specification algorithms $\mathsf{RG}_{\text{SPEC}}$. That is, for every $z \leftarrow \mathsf{BG}(\lambda)$, it holds that:*

(1) $\Pr[x' = x \mid z \leftarrow \mathsf{BG}; r \leftarrow \mathsf{RG}(\lambda, z); (i, t_i) \leftarrow \mathsf{dKG}_{\text{SPEC}}(\lambda, r); x \leftarrow X_i; y := f_i(x); x' \leftarrow \mathsf{Inv}(i, y, z)] \geq \delta$

(2) $\{r \mid z \leftarrow \mathsf{BG}; r \leftarrow \mathsf{RG}(\lambda, z)\} \overset{c}{\approx} \{r \mid r \leftarrow \mathsf{RG}_{\text{SPEC}}(\lambda)\}$

Correspondingly, we say a TDOWF family is **unforgeable** *if it is not δ-strongly forgeable for any non-negligible δ.*

The immunization methodology for OWF discussed before can also be used for immunizing stateless TDOWF in the split program model. That is, given any TDOWF family $\mathcal{F}$ whose $\mathsf{RG}_{\text{SPEC}}$ outputs uniform bits r, a hash function h randomly selected from a hash family $H_\lambda : \{0,1\}^\lambda \to \{0,1\}^\lambda$ can be used to determine an unforgeable TDOWF family $\mathcal{G}$, see Figure 6 below:

$$\boxed{\mathsf{RG}(z)} \to r \qquad r \to \boxed{h} \to \tilde{r} \qquad \tilde{r} \to \boxed{\mathsf{dKG}} \to (i, t_i) \qquad x \to \boxed{\mathsf{Eval}(i, \cdot)} \to y$$

Figure 6: Immunization strategy for TDOWF in the split-program model.

Remark 13. *We can consider* (h, dKG) *as one single deterministic algorithm* $\mathsf{dKG} \circ h$ *which first run* h *on inputs* r *to get* $\tilde{r}$ *and run* dKG *on* $\tilde{r}$*. The new specification will be* $(\mathsf{RG}_{\text{SPEC}}, \mathsf{dKG}_{\text{SPEC}} \circ h_{\text{SPEC}})$.

Theorem 26. *The TDOWF family* $\mathcal{G}$ *described above is unforgeable in the split-program model if* H *is modeled as a random oracle, and the adversarial implementations are stateless.*

Proof. First, since the output distribution of a stateless RG is pseudorandom, thus for any PPT adversary $\mathcal{A}$, including the one that produces RG, $\Pr[r' = r : r \leftarrow \mathsf{RG}(z), r' \leftarrow \mathcal{A}(z)] \leq \epsilon$, where $\epsilon = \mathsf{negl}(\lambda)$.

The rest proof is similar to the proof of Theorem 25 that if one can break the strong unforgeability of $\mathcal{G}$, then we can construct a simulator that breaks the specification of $\mathcal{F}$ in the split-program model. □

4.4 Pseudorandom Generator in the Complete Subversion Model

As mentioned in the Introduction, our goal is to stimulate a *systematic* study of Cliptography. Having studied the fundamental backdoor-free building blocks, unforgeable OWFs, and unforgeable TDOWFs, we intend to mimic the classic footprints of constructing cryptographic primitives from OWF/TDOWF, and provide solutions to other important backdoor-free building blocks. As our first example, we will show an interesting connection between our notion of unforgeable OWF and the notion of backdoor-free PRG, recently studied by Dodis et al. [47]. Next we first review the basic notions of PRG under subversion attacks. We then provide a specific solution based on the Blum-Micali PRG; this result can be viewed as a mimic of the classic result of Blum-Micali construction in cliptography. Furthermore, we examine how to extend the

applicability of our general sanitizing strategy for OWF/TDOWF to more settings. We will demonstrate a general method of public immunizing strategy for PRG. We remark that, all algorithms in our backdoor-free PRG construction, including the sanitizing function (which can be part of the KG algorithm in the specification), can be subverted. Thus we provide the first PRG constructions secure in the complete subversion model.

4.4.1 Preliminaries: backdoored and backdoor-free PRGs

We adopt the definition from [47], that a pseudorandom generator consists of a pair of algorithms $(\mathsf{KG}, \mathsf{PRG})$, where KG outputs a public parameter pk and $\mathsf{PRG} : \{0,1\}^* \times \{0,1\}^\ell \to \{0,1\}^\ell \times \{0,1\}^{\ell'}$ takes the public parameter pk and an ℓ-bit random seed s as input; it returns a state $s_1 \in \{0,1\}^\ell$ and an output string $r_1 \in \{0,1\}^{\ell'}$. PRG may be iteratively executed; in the i-th iteration, it takes the state from the previous iteration s_{i-1} as the seed and generates the current state s_i and output r_i. We use PRG^q to denote the result of q iterations of PRG with outputs $r_1, \ldots, r_q$ (each $r_i \in \{0,1\}^{\ell'}$).

In a backdoored PRG, the algorithms (in particular KG) are implemented by the adversary (represented by $\mathcal{A}_{\text{INITIAL}}$), outputs a public parameter pk together with a backdoor sk. The output distribution $\mathsf{PRG}(pk,\mathcal{U})$ is still pseudorandom, where $\mathcal{U}$ is the uniform distribution; however, with the corresponding backdoor sk, the adversary (represented by $\mathcal{A}_{\text{DIST}}$) is able to break the PRG security (e.g., the adversary can distinguish the output from a uniform string).

We will rephrase the definition for backdoored PRG in our setting—as in the definition of a forgeable OWF—there exist "specification" versions of the algorithms. In particular, the parameter generation algorithm $\mathsf{KG}_{\text{SPEC}}$ is with the requirement that the distribution of the adversarially generated public parameter must be indistinguishable from the output distribution of $\mathsf{KG}_{\text{SPEC}}$. It is easy to see that the output distribution of

$\mathsf{PRG}(pk, s)$ for a uniformly chosen s is pseudorandom even pk is generated by $\mathcal{A}_{\text{INITIAL}}$. (Otherwise, one can easily distinguish the output distribution of $\mathcal{A}_{\text{INITIAL}}$ from $\mathsf{KG}_{\text{SPEC}}$.) Additionally, as the PRG algorithm is deterministic, and its input distribution is public, we may assume that the adversary implements it honestly as the $\mathsf{PRG}_{\text{SPEC}}$ to avoid easy detection so that we can focus on the KG algorithm. The formal definitions are presented as follows:

Backdoored PRG. We re-phrase the definition of (q, δ)-**backdoored PRG** [10] as follows: We define a backdoored PRG game (see Figure 7) with a PPT adversary $\mathcal{A} = (\mathcal{A}_{\text{INITIAL}}, \mathcal{A}_{\text{DIST}})$ such that (i) the pk distribution is indistinguishable from that generated by $\mathsf{KG}_{\text{SPEC}}$; and (ii) the adversary wins the backdoored PRG game with probability δ, i.e., $\Pr[b = b'] - \frac{1}{2} \geq \delta$.

$(pk, sk) \leftarrow \mathcal{A}_{\text{INITIAL}}$
$s \leftarrow \{0,1\}^{\ell}$
$r_1^0, \ldots, r_q^0 \leftarrow \mathsf{PRG}^q(pk, s)$
$r_1^1, \ldots, r_q^1 \leftarrow \{0,1\}^{\ell' \cdot q}$
$b \leftarrow \{0,1\}$
$b' \leftarrow \mathcal{A}_{\text{DIST}}(pk, sk, r_1^b, \ldots, r_q^b)$

Figure 7: The backdoored PRG game

Backdoor-free PRG. Then we say that a PRG is q-**backdoor free** if, in the above backdoored PRG game, for all PPT adversaries $\mathcal{A} = (\mathcal{A}_{\text{INITIAL}}, \mathcal{A}_{\text{DIST}})$, whenever pk is indistinguishable from the specification distribution, the advantage is negligible, i.e., $\left|\Pr[b = b'] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$.

[10] We ignore the running time of the adversary here for simplicity. Also, according to the detection principle in section 4.2, lemma 20, the PRG algorithm is deterministic and with a uniform input distribution, thus we can treat it as honestly implemented.

Remark 14. *The generation of the seed s is out of the scope of this paper (same as [47]), since even the specification of pseudorandom generators do not cover this part. Our techniques can guarantee a received implementation is as good as the specification. Of course, it would be an important open question to consider the random seed generation for practice.*

4.4.2 Constructing backdoor-free PRG from strongly unforgeable OWP

In this subsection, we provide constructions for backdoor-free PRG based on strongly unforgeable one-way permutation. We start with a basic solution based on a (simplified) Blum-Micali PRG, and then extend it to a full-fledged solution. Before going to the details of our constructions, we recall the classic generic construction of Goldreich-Levin (GL), yielding a hardcore predicate [61] for any OWF f. We suppose the input x of f is divided into two halves $x = (x_1, x_2)$ and define the bit $B(x) = \langle x_1, x_2 \rangle$; $B(x)$ is hard to predict given $x_1, f(x_2)$, assuming that f is one-way. Moreover, if there is a PPT algorithm that predicts $B(x)$ with significant advantage δ given $x_1, f(x_2)$, then there is a PPT algorithm I that inverts f with probability $\text{poly}(\delta)$.

Basic construction. We will show that given a strongly unforgeable one-way permutation (OWP) family $\mathcal{F}$ with algorithms $(\mathsf{KG}_{\mathcal{F}}, \mathsf{Eval}_{\mathcal{F}})$, the classic Blum-Micali PRG [**?**] (using the GL hardcore predicate) is 1-backdoor free. Our basic construction $(\mathsf{KG}, \mathsf{PRG})$ is as follows:

- Parameter generation algorithm $pk \leftarrow \mathsf{KG}(\lambda)$:

  compute $i \leftarrow \mathsf{KG}_{\mathcal{F}}(\lambda)$ and set $pk := i$;

- Bit string generation algorithm $(s', b) \leftarrow \mathsf{PRG}(pk, s)$:

upon receiving s and pk, where $pk = i$, $s = s_1||s_2$ and $|s_1| = |s_2| = \ell$, compute the following: $s'_1 := s_1$, $s'_2 := f_i(s_2)$ (or $s'_2 := \mathsf{Eval}_{\mathcal{F}}(i, s_2)$), and $s' = s'_1||s'_2$, $b := \langle s_1, s_2 \rangle$.

We can show in the lemma below that the basic construction above is a 1-backdoor free PRG. The intuition is that in the (simplified) Blum-Micali PRG, a distinguisher can be transformed into an OWF inverter (following the GL proof), thus an adversary who can build a backdoor for this PRG implies that she has the ability to make $\mathcal{F}$ (forgeable), which violates the strong unforgeability of $\mathcal{F}$.

Lemma 27. *Given a strongly unforgeable one way permutation family $\mathcal{F}$, the basic construction above is 1-backdoor free.*

Proof. The specification $\mathsf{KG}_{\text{SPEC}}$ of the simplified Blum-Micali PRG outputs a random function index from the corresponding index set (by simply running the key generation specification of $\mathcal{F}$).

First, it is easy to see that the OWF function family $\mathcal{G} = \{g_i\}$, where $g_i(x_1||x_2) := x_1||f_i(x_2)$, is strongly unforgeable if $\mathcal{F}$ is strongly unforgeable.

If the above basic construction is a $(1, \delta)$-backdoored PRG for some non-negligible δ, there exist PPT adversaries $(\mathcal{A}_{\text{INITIAL}}, \mathcal{A}_{\text{DIST}})$ such that (i.) $\mathcal{A}_{\text{INITIAL}}$ can output a pair (pk, sk), where pk is indistinguishable from a randomly sampled public parameter; and (ii.) $\mathcal{A}_{\text{DIST}}$, with the backdoor sk, can distinguish $s_1||f_i(s_2)||B(s)$ (where $B(s) = \langle s_1, s_2 \rangle$) from a uniform $(2\ell + 1)$-bit string. It is not hard to see that we can then construct algorithms $(\mathsf{KG}, \mathsf{Inv})$ that break the strong unforgeability of $\mathcal{G}$.

In particular, KG runs $\mathcal{A}_{\text{INITIAL}}$ and outputs the received key pair (pk, sk), where pk here corresponds to a function index i that is indistinguishable from a random index, and sk corresponds to the backdoor z. Inv receives a challenge $y = g_i(x)$ and the

backdoor z; it first constructs an algorithm $\mathcal{A}_P$. $\mathcal{A}_P$ selects a random bit b, and runs $\mathcal{A}_{\text{DIST}}(pk, sk, y||b)$. By the definition of $\mathcal{A}_{\text{DIST}}$, if $b = B(s)$, $\mathcal{A}_{\text{DIST}}$ (with sk) will output 0 with probability $1/2 + \delta$. It is easy to see that $\mathcal{A}_P$ can predict the GL hardcore predicate B with advantage $\delta/2$, following the GL proof [61], there exists another algorithm $I^{\mathcal{A}_P}(pk, sk, \cdot)$ that can invert y with probability $\delta' = \text{poly}(\delta/2)$. Inv runs $I^{\mathcal{A}_P}(pk, sk, i, y)$ and recovers x'; Inv then outputs x' if it is a valid pre-image of y. It follows that $\Pr[x' = x] \geq \text{poly}(\delta/2) = \delta'$ and $\mathcal{G}$ will be δ'-forgeable for a non-negligible δ', thus contradicts our assumption. □

Full-fledged construction. We now extend our basic construction via iterations to show that the full fledged Blum-Micali PRG construction, using our strongly unforgeable OWF, achieves a q-backdoor free PRG for any $q = \text{poly}(\lambda)$. Our full-fledged construction $(\mathsf{KG}, \mathsf{PRG})$ [11] is as follows:

- Parameter generation algorithm $pk \leftarrow \mathsf{KG}(\lambda)$:

  compute $i \leftarrow \mathsf{KG}_{\mathcal{F}}(\lambda)$ and set $pk := i$;

- Bit string generation algorithm $(s', r) \leftarrow \mathsf{PRG}(pk, s)$:

  upon receiving s and pk where $pk = i$, $s = s_1||s_2$, and $|s_1| = |s_2| = \ell$, compute the following:

  - let $s_1^0 := s_1$ and $s_2^0 := s_2$;
  - for $j = 1, \ldots \ell'$,

    $b_j := \langle s_1^{j-1}, s_2^{j-1} \rangle$;

    $s_1^j := s_1^{j-1}$; $s_2^j := f_i(s_2^{j-1})$; $s^j := s_1^j||s_2^j$;

[11] PRG^q can be defined in a straightforward manner that runs PRG for q iterations, each iteration outputs ℓ' bits and updates the state for next iteration.

– $s' = s^{\ell'} = s_1||f_i^{\ell'}(s_2)$; and $r = b_1 \dots b_{\ell'}$.

Please see Figure 8 for pictorial illustration.

$$\begin{array}{ccccccc} s_1||s_2 & \to & s_1||f_i(s_2) & \to & s_1||f_i^2(s_2) & \to \dots \to & s_1||f_i^{\ell'}(s_2) \to \dots \\ & & \downarrow & & \downarrow & & \downarrow \\ & & \langle s_1, s_2 \rangle & & \langle s_1, f_i(s_2) \rangle & \dots & \langle s_1, f_i^{\ell'-1}(s_2) \rangle \end{array}$$

Figure 8: One iteration of BM-PRG

Theorem 28. *The full fledged construction above is q-backdoor free (for any polynomially large q), if the underlying OWP family $\mathcal{F}$ is strongly unforgeable.*

Proof sketch. Following Lemma 27, $s_1||f_i^j(s_2)||b_j$ is pseudorandom, i.e., it is indistinguishable from "$u_1, \dots, u_\ell, v_1$", even to the adversaries $(\mathcal{A}_{\text{INITIAL}}, \mathcal{A}_{\text{DIST}})$ who set i, where $\{u_i\}, v_1$ are all random bits. While b_{j+1} is only related to $s_1, f_i^j(s_2)$, it follows that to the adversary $\mathcal{A}_{\text{DIST}}$ (who has the backdoor), $b_1 \dots b_{\ell'}$ satisfies the next-bit unpredictability. Following the classic reduction from pseudorandomness to next-bit unpredictability, we can conclude that $b_1 \dots b_{\ell'}$ is indistinguishable from uniform bits from $\{0,1\}^{\ell'}$, even to $\mathcal{A}_{\text{DIST}}$. (This can be shown via the hybrid argument.) Then, inductively, we can conclude that $r_1, \dots, r_q$ are indistinguishable from $\ell' \cdot q$ uniform bits. □

Remark 15. *It is easy to see that if starting from an OWF(not necessarily unforgeables), the full fledged construction can be easily modified by replacing $f_i(\cdot)$ with $f_{h(i)}(\cdot)$.*

4.4.3 General public immunization strategy for PRG

An impossibility result about public immunization of a backdoored PRG was presented in [47]. However, we observe that this impossibility result only applies to an

immunization procedure that operates on the *output* of the backdoored PRG . The application of strongly unforgeable OWF to backdoor-free PRG shown above inspires us to consider a new, general immunizing strategy for backdoored PRG . We suggest that—similar to the procedure above for eliminating backdoors in OWFs—one can randomize the public parameters to sanitize the PRG. [12] The intuition for this strategy to be effective in the setting of PRG is similar: if a specification $\mathsf{KG}_{\text{SPEC}}$ that outputs a uniform pk from its domain, no single backdoor can be used to break the security for large amount of public parameters; otherwise, one can use this trapdoor to break the PRG security of the specification.

Consider a PRG implementation $(\mathsf{KG}, \mathsf{PRG})$ (in which the KG algorithm might be backdoored), which is proven secure if the $\mathsf{KG}_{\text{SPEC}}$ outputs uniformly from its range PP. Let h be randomly selected from a hash family $H : PP \to PP$ which is modeled as a random oracle. Then we can construct a backdoor-free PRG, $(\mathsf{KG}, \mathsf{PRG} * h)$, i.e., applying the hash to the given public parameter to derive the actual pk that will be fed into the PRG algorithm (the new deterministic pseudo randomness generation algorithm $\mathsf{PRG} * h$ defined as $\mathsf{PRG} * h(pk, s) = \mathsf{PRG}(h(pk), s)$). Note that in order for this method to work, we must insist that pk can not be null, and is indeed used by the PRG algorithm, as in the case of, e.g., Dual_EC PRG. Also see the pictorial illustration in Fig 9.

Theorem 29. *Assume* $(\mathsf{KG}_{\text{SPEC}}, \mathsf{PRG}_{\text{SPEC}})$ *is a pseudorandom generator if* $\mathsf{KG}_{\text{SPEC}}$ *outputs pk randomly from its domain. Given any implementation* $(\mathsf{KG}, \mathsf{PRG})$*, hashing the*

[12] To interpret this results, since the solution of [47] requires a trusted seed/key generation and apply the function to the PRG output, thus part of the PRG algorithm can not be subverted. It follows that the construction of PRG in the complete subversion model was still open until our solution. In contrast, our sanitizing strategy does not require any secret, and even the deterministic hash function can be implemented by the adversary as part of the KG algorithm.

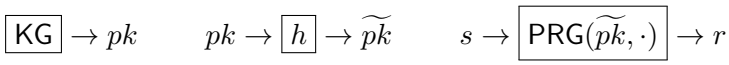

Figure 9: Public immunization strategy for PRG.

public parameters as described above, i.e., $(h \circ \mathsf{KG}, \mathsf{PRG}_{\text{SPEC}})$ *yields a q-backdoor-free pseudorandom generator in the random oracle model for any polynomially large* q*.*

Proof sketch. Suppose $(h \circ \mathsf{KG}, \mathsf{PRG}_{\text{SPEC}})$ is a (q, δ)- backdoored PRG . Then there exist a pair of adversaries $(\mathcal{A}_{\text{INITIAL}}, \mathcal{A}_{\text{DIST}})$ that can win the backdoored PRG game defined in Figure 7 with advantage δ. We will transform these adversaries into an adversary $\mathcal{S}$ that breaks the security of $(\mathsf{KG}_{\text{SPEC}}, \mathsf{PRG}_{\text{SPEC}})$.

Suppose the challenger of the specification version of PRG selects the parameter pk^* and the challenge string r^* is either $\mathsf{PRG}(pk^*, s)$ (for a uniform $s \in \{0,1\}^\ell$) or a uniform string from $\{0,1\}^{\ell' \cdot q}$ for some ℓ'.

The reduction follows the proof of Theorem 24: $\mathcal{S}$ attempts to embed pk^* into the answers to the random oracle queries. In particular, if $\mathcal{A}_{\text{INITIAL}}$ outputs pk, $\mathcal{S}$ wishes to answer the random oracle query about pk using pk^*, i.e., $h(pk) = pk^*$. We then proceed with a similar probabilistic analysis.

It is easy to see that if $\mathcal{A}_{\text{DIST}}$ can distinguish r^* from an uniform string, then $\mathcal{S}$ will be able to distinguish the output of the specification version from random which violates the PRG security of the specification. □

Remark 16. *There are several points we would like to stress:*

- *If the public parameter contains only random elements from a group, e.g., the Dual_EC PRG, we may simply encode them into bits and use the regular hash functions like SHA-256, directly and convert the resulting bits back to a group element;*

- *If the public parameters are structured elements, or the* $\mathsf{KG}_{\text{SPEC}}$ *does not output a uniform distribution, we can work on the split-program model that forces the adversarial implementation to explicitly isolate its generation of randomness, and make the randomness public as part of the public parameter.*

- *If we treat the immunizing method as part of the* KG *algorithms, i.e.,* $h \circ \mathsf{KG}$ *is a single algorithm, we can let the adversary sets both* $(pk, \widetilde{pk})$ *as the public parameter, regular user simply uses* $\widetilde{pk}$ *as the actual parameter and the crypto experts can check the validity of it. Similarly, in the split-program model, we can let the big brother set the original randomness* r *and sanitized randomness* $\tilde{r}$*, together with* $\widetilde{pk}$ *as the public parameters.*

4.5 Signatures in the Complete Subversion Model

As an immediate application, in this section, we will demonstrate how to use unforgeable OWFs and unforgeable TDOWFs as fundamental building blocks to construct backdoor-free digital signature schemes in the complete subversion model.

We build on recent works of Bellare et al. [7, 11], who studied a family of *algorithm-substitution attacks* (ASAs) on cryptographic primitives. In their ASAs, an encryption algorithm is implemented by a saboteur who may have backdoors embedded; then the sabotaged implementation judiciously samples randomness in order to generate the outputs (e.g., ciphertexts) in a way that leaks the secret bit-by-bit via a steganographic channel.

To defend against the ASAs, Bellare et al provided successful mechanisms of using deterministic algorithms (e.g., signing) with unique outputs. However, we note that the ASAs considered in [7, 11] are restricted in the sense that the adversary is not allowed

to launch the ASAs on *all* algorithms; In particular, the key generation algorithm is assumed to be implemented honestly and the adversary is not allowed to attack on it.

We are interested in securing cryptographic schemes in the *complete subversion model* where the adversary is allowed to launch algorithm-substitution attacks on *all* components of the schemes. We demonstrate below how our unforgeable TDOWF can be applied to defend against the ASAs, even if the key generation algorithm has been sabotaged. Together with the results of [7, 11], we then show a concrete example of a digital signature scheme, achieving security in the case that *all* the algorithms are sabotaged.

Subversion-secure signature scheme. We can define the existential unforgeability in the setting of complete subversion [13] in a way that the public key is generated using the adversarial implementation, and the signing/verification algorithm can also be implemented by the adversary, while all the other steps are the same as the standard definition.

Definition 14. *We say a signature scheme* $(\mathsf{KG}, \mathsf{Sign}, \mathsf{Verify})$ *(with the specification version of key generation algorithm* $\mathsf{KG}_{\text{SPEC}}$*, signing algorithm* $\mathsf{Sign}_{\text{SPEC}}$ *and verification algorithm* $\mathsf{Verify}_{\text{SPEC}}$*) is* ***existentially unforgeable in the complete subversion setting****, if for all* PPT *adversaries* $\mathcal{A} = (\mathcal{A}_{\text{INITIAL}}, \mathcal{A}_{\text{FORGE}})$*, the following properties hold:*

- *The adversary* $\mathcal{A}$ *wins the unforgeability game defined below, with no more than a negligible probability.*

[13] This definition may be weaker than the general definition of surveillance security defined in [11] that the output distribution of subverted algorithms looks the same as that of the specification even to the adversary who has the backdoor. However, preserving its standard security in this complete subversion model (we think) is good enough for each primitive to be used as normal.

UNFORGEABILITY GAME *between a* PPT *adversary* $\mathcal{A} = (\mathcal{A}_{\text{INITIAL}}, \mathcal{A}_{\text{FORGE}})$ *and a challenger* $\mathcal{C}$*:*

1. $\mathcal{A}_{\text{INITIAL}}$ *provides an implementation of* KG, Sign, Verify *(and an implementation of the hash function if needed) to the challenger (which may contain backdoor information* z*).*
2. $\mathcal{C}$ *queries* KG *to learn the key pair* (pk, sk)*, and sends the public key* pk *to* $\mathcal{A}_{\text{FORGE}}$*.*
3. $\mathcal{A}_{\text{FORGE}}$*, having the backdoor* z *from* $\mathcal{A}_{\text{INITIAL}}$*, asks signing queries for arbitrarily chosen messages* $m_1, \ldots, m_q$*; for each* m_i *from* $\mathcal{A}_{\text{FORGE}}$*, the challenger* $\mathcal{C}$ *queries* Sign *with input* (sk, m_i) *to learn the corresponding signature* σ_i*, and return* σ_i *to* $\mathcal{A}_{\text{FORGE}}$*.*
4. $\mathcal{A}_{\text{FORGE}}$ *returns a message-signature pair* (m^*, σ^*) *to the challenger* $\mathcal{C}$*; now* $\mathcal{C}$ *queries* Verify *with input* (pk, m^*, σ^*) *to learn output* b*.*
5. $\mathcal{A} = (\mathcal{A}_{\text{INITIAL}}, \mathcal{A}_{\text{FORGE}})$ *wins the game if* $m^* \notin \{m_1, \ldots, m_q\}$ *and* $b = 1$*.*

- *In the above game, the subversion of* KG, Sign *and* Verify *(and the hash function if there is any) are strongly undetectable. (See section 4.2 for details.)*

Construction. Following the result of [11], if the key generation algorithm is honest, a unique signature scheme [**?**, 97] remains existentially unforgeable against the algorithm substitution attack. In this case, the signing and verification algorithms are both deterministic and the detection condition "forces" the adversary to honestly implement these two. To obtain a *complete* subversion-secure solution from a unique signature scheme, we still need to "upgrade" the key generation algorithm so that it is secure against subversion attack.

We may consider the following approach to obtain a subversion-secure key generation algorithm: initially generate an unforgeable OWF f, and then randomly select a secret key sk and compute the public key $pk = f(sk)$. Since the OWF is unforgeable, it seems that we may be able to force the adversarial key generation algorithm to output a "safe" OWF. However, it is not clear how to show that the key pairs are well distributed.

We here consider an alternative approach; we use an unforgeable TDOWF, $(\mathsf{KG}_{\mathcal{F}}, \mathsf{Inv}_{\mathcal{F}})$. More concretely, we use the key generation algorithm of unforgeable TDOWF to generate a function index i together with the corresponding trapdoor t_i, and set the index i as the public key, and the trapdoor t_i as the secret key. To be compatible with the unique signature scheme, we choose to instantiate the unique signature scheme from the full domain hash construction [13, 40]. Details of the construction are as follows:

- Key generation $(pk, sk) \leftarrow \mathsf{KG}(\lambda)$:

  compute $(i, t_i) \leftarrow \mathsf{KG}_{\mathcal{F}}(\lambda)$, and set $pk := i$ and $sk := t_i$;

- Signature generation $\sigma \leftarrow \mathsf{Sign}(sk, m)$:

  upon receiving message m, compute $\sigma = \mathsf{Inv}_{\mathcal{F}}(sk, i, h(m))$, where $sk = t_i$.

- Signature verification $b := \mathsf{Verify}(pk, m, \sigma)$:

  upon receiving message-signature pair (m, σ), if $f_i(\sigma_1) = h(m)$ then set $b := 1$, otherwise set $b := 0$; here $pk = i$.

Lemma 30. *The algorithms* $\mathsf{Sign}, \mathsf{Verify}$ *and the hash* h *in the full domain hash signature scheme are subversion resistant (in the strong detection model).*

Proof. Suppose $(m_1, y_1), \ldots, (m_q, y_q)$ and $(m_1, \sigma_1), \ldots, (m_{q'}, \sigma_{q'})$ form the transcript of the adversary in the subversion game, trying to distinguish $(\mathsf{Sign}, \mathsf{Verify}, h)$ from

$(\mathsf{Sign}_{\text{SPEC}}, \mathsf{Verify}_{\text{SPEC}}, h_{\text{SPEC}})$. The transcript includes all the signing queries and the final forgery.

First, it is easy to see that if there is any $y_i \neq h_{\text{SPEC}}(m_i)$, then there exists a detector simply queries h_{SPEC} and noticed this inconsistency with probability 1.

If there exists $(m_i, (\sigma_i)$ such that $\mathsf{Sign}_{\text{SPEC}}(sk, m_i) \neq \mathsf{Sign}(sk, m_i) = \sigma_i$, then there exists a detecter algorithm $\mathcal{D}$ can always detect it with probability 1. $\mathcal{D}$ uses this pair as an input (part of the transcript) and runs the specification $\mathsf{Verify}_{\text{SPEC}}$ on this message-signature pair. Since the full domain hash is a unique signature, thus if they are not the same as that using the $\mathsf{Sign}_{\text{SPEC}}$, it will not pass the verification.

If there exists a pair (m_i, σ_i) such that $\mathsf{Verify}_{\text{SPEC}}(pk, m_i, \sigma_i) \neq \mathsf{Verify}(pk, m_i, \sigma_i)$, similarly, there exists a detecter $\mathcal{D}$ can always notice it.

It follows that every hash evaluation is consistent with that using the hash specification; every signature generated by the Sign algorithm is consistent with that generated by the specification; every signature pass the Verify algorithm is indeed a valid signature. If the adversary gains any advantage in the subversion game, there must be a pair that is inconsistent with the specifications, then it can be detected with probability 1. We can conclude that $\delta_s \leq \delta_d$. □

Remark 17. *The strong detection condition in the signature setting in practice is more reasonable than that in the symmetric key encryption setting as [11, 79]. Since the nature of signature schemes, the transcripts are for public verification anyway, anyone can verify the validity, including the lab. While in the symmetric key encryption, the detection requires the detecter to have access to the secret key. This essentially states that the every user is a detecter who has access to the specification, which weakens the motivation for considering the implementation subversion (since they have the specification anyway).*

Theorem 31. *Given an unforgeable TDOWF $\mathcal{F}$, the full domain hash signature scheme is existentially unforgeable (under complete subversion) in the random oracle model, and all the algorithms are stateless.*

Proof. First, following lemma 30, the $h, \mathsf{Sign}, \mathsf{Verify}$ algorithms we use are subversion resistant. It follows that the adversary behaves the same as sending honest implementations for those algorithms (otherwise got detected), i.e., consistent with the specifications. In particular, the hash function implementation can still be modeled as a random oracle for queries of the adversary.

We will show that if there is a subversion attack, then one can break the unforgeability of $\mathcal{F}$. Suppose $(\mathcal{A}_{\text{INITIAL}}, \mathcal{A}_{\text{FORGE}})$ are the adversaries who break the existential unforgeability of the full domain hash signature scheme in the complete subversion model with a non-negligible probability δ, we will construct $\mathcal{S}$ to simulate the adversaries $(\mathsf{BG}, \mathsf{KG}, \mathsf{Inv})$ as in Definition 9.

BG first runs $\mathcal{A}_{\text{INITIAL}}$ to receive the backdoor z and the implementation KG_0 for the signature key generation. KG simply runs KG_0 and outputs a function index $i := pk$ and the corresponding trapdoor t_i. $\mathcal{S}$ discards t_i and sends pk to $\mathcal{A}_{\text{FORGE}}$. Suppose $y = f_i(x)$ is the challenge that Inv receives from the unforgeable TDOWF challenger, Inv (constructed by $\mathcal{S}$) will feed $\mathcal{A}_{\text{FORGE}}$ with z and runs it as follows to try to invert.

W.l.o.g, we assume $\mathcal{A}_{\text{FORGE}}$ asks a random oracle query for a message before she asks for the signing query, (if not, the simulator can ask instead of her), and $\mathcal{A}_{\text{FORGE}}$ asks the random oracle query for her final forgery m^*.

Suppose $\mathcal{A}_{\text{FORGE}}$ asks q random oracle queries $m_1, \ldots, m_q$. Inv randomly choose $j \in \{1, \ldots, q\}$, and answers the query with $y = h(m_j)$; He then chooses $q - 1$ random elements $\sigma_1, \ldots, \sigma_{j-1}, \sigma_{j+1}, \ldots, \sigma_q$, and answers the random oracle queries as $h(m_k) = f_i(\sigma_k)$ for $k \neq j$. Inv maintains a list.

When $\mathcal{A}_{\text{FORGE}}$ asks a signing query m_k, if $m_k = m_j$, Inv aborts, otherwise, Inv checks the list and returns the corresponding σ_k. (Note that Sign algorithm can only have one output for each message, thus the above procedure perfectly simulates the signing oracle implemented by the subverted Sign algorithm). $\mathcal{A}_{\text{FORGE}}$ outputs m^*, σ^*. If $m^* \neq m_j$, Inv aborts, otherwise, Inv outputs σ^*.

Let us use W to denote the event that Inv aborts. Following the classic proof of security of the full domain hash signature scheme (e.g., [40]), $\Pr[W] \leq 1 - \text{poly}(\frac{1}{q})$, thus Inv successfully inverts with probability at least $\delta_0 = \text{poly}(\frac{1}{q})\delta$ (as we assume $\mathcal{A}_{\text{FORGE}}$ has δ advantage, and the Verify has to be honestly implemented to avoid detection). This means $\mathcal{F}$ is δ_0-forgeable, which contradicts our condition. □

4.6 Conclusion and Open Problems

We initiate the systematic study of defending mechanism against kleptographic attacks of cryptographic primitives when *all* algorithms are subject to the subversion, we call **cliptography**.

We start from the fundamental primitives of (trapdoor) one way functions. We formalize the notions of forgeable (trapdoor) OWF to capture the possibility of embedding backdoors, in particular, into the function generation algorithms, and show how to launch such attacks. More interestingly, we suggest a general sanitization method employing randomization of the function index to destroy the possibility of embedding backdoor information. To instantiate our method in practice, we propose a split-program model in which the function generation algorithm consists of two components, a randomized component RG and a deterministic component dKG; here, the first component RG generates a uniform random string for the second component dKG, and then the second component generates an index based on such random string. In such split-program

model, we can directly apply our general method of immunizing one way function generation to the randomness generated by RG.

We then pursue the possibility of building cliptography from unforgeable (trapdoor) OWFs. In particular, we show how to construct a signature scheme and a pseudorandom generator that preserves its security in the complete subversion model. These are done by using our unforgeable trapdoor OWF and strongly unforgeable OWF as the key/parameter generation algorithm. Finally, we show how to apply our immunizing technique directly to the setting of PRG and present a general public immunizing strategy for PRG.

Many important problems remain open about defending against kleptogaphic attacks, and in general against mass surveillance. The immediate open questions left by our paper would be: to what extend the results and techniques developed in this paper can be used to build cliptography. In particular, can we construct other cryptographic primitives that preserves their security in the setting that all algorithms are subject to kleptographic attacks? There are also many other related open questions, to name a few: (1) how to sanitize the OWF generation in the standard model? (2) how to eliminate the subliminal channel in general? (3). Can we show some results in the complete subversion model so that we can preserve the security for multiparty protocols? and finally, (4) how to design an accountable mechanism for surveillance, such that key escrow is provided but not abused?

Chapter 5

Rate Optimal Asymmetric Fingerprinting from Tardos Code

5.1 Introduction

In a fingerprinting scheme, cf. [24], a server (or service provider SP) distributes a file to a set of users. The server has the flexibility to furnish a different version of the file to each user. This is done by splitting the file into segments and offering at least two variations per segment. Given these segments, the file can be assembled in a *fingerprinted* fashion: at each segment the variation obtained corresponds to a symbol over an alphabet. Therefore, each user's file determines a string over that alphabet - the user's fingerprint (e.g., the data M is divided into n blocks, for each block i, there are two versions m_i^0, m_i^1, a user assigned with a binary codeword $b_1, \ldots, b_n$ will receive his versions as $m_1^{b_1}||\ldots||m_n^{b_n}$). The objective here is that if the users collude to produce

a "pirate" version of the file by combining their segments, the server is still capable of discovering (at least some) of the identities of the colluding users.

If the SP alone generates the fingerprints for users and directly transmits them the fingerprinted files, we have what is known as a symmetric fingerprinting scheme. As the server is fully trusted in this setting, the security requirement is that malicious users cannot collude to frame any innocent user or evade the tracing algorithm. The subtle issue is that the server and the user are both able to produce a pirate file so when a pirate copy is brought to light, an honest SP cannot provide an "undeniable" proof that a user is at fault and symmetrically an honest user cannot defend herself against a malicious SP that frames her (say, due to e.g., an insider attack on the SP side).

In order to resolve the above issue, [111] introduced asymmetric fingerprinting schemes in which no one (even the server) should be capable to implicate an innocent user. Thus when a dispute happens, the server can provide a convincing proof that a guilty user is at fault. It follows that the server should not be fully aware of the fingerprint of each user (otherwise it is capable of impersonating them) and hence this suggests that the download of the fingerprinted file should be performed in an oblivious manner from the servers' point of view. Now in this case, the Judge could resolve the dispute between the server and a user (i.e., guilty users will be found guilty by the judge while the server will not be able to implicate an innocent user in the eyes of the judge).

In the original papers [111–113] the file transfer stage was treated generically as an instance of secure two party computation. Unfortunately, even with "communication efficient" secure two-party computation [44, 46, 75, 104] the communication overhead of the resulting protocol is prohibitively high (e.g., even with the most communication efficient generic protocols, [46, 75], the communication rate — the size of the file over total number of bits transmitted — will be at most 0.5 and their use will impose the

additional cost of a prohibitively large CRS which needs to be known a-priori to both client and server). With the discovery of optimal length binary fingerprinting codes by Tardos [127], Charpentier et al. [34] observed that oblivious transfer could be used as a building block for a Tardos-based asymmetric fingerprinting. Their proposed solution however is sub-optimal (it has a rate still at most 0.5) and in order to achieve the fingerprint generation it relies on *commutative encryption*, a primitive not known to be constructible in a way that the resulting scheme can be shown provably secure. Furthermore no complete security analysis is provided in [34] which leaves a number of important security issues unaddressed (specifically "accusation withdrawals" and "selective aborts" – see below).

Achieving rate close to 1 is the most critical open question in the context of asymmetric fingerprinting from an efficiency point of view. Indeed, any asymmetric fingerprinting is particularly sensitive to its communication overhead: the file to be transmitted by the sender can be quite large (e.g., a movie file) and thus any scheme whose communication rate is not close to 1 is likely to be useless in a practical setting. We note that efficient asymmetric fingerprinting schemes can enable more complex applications; e.g., as building blocks for "anonymous buyer-seller watermarking" [116, 117]; these systems rely on asymmetric fingerprinting schemes to enable copyright protection (but they do not consider the implementation of such fingerprinting schemes explicitly).

Furthermore, analyzing the security of an asymmetric fingerprinting scheme is involved as the security requirements require that the SP cannot frame an innocent user, while at the same time the malicious users should still not be able to escape from tracing. The analysis should rely both on the security of the protocol and on the property of the code, specifically, no user should be able to produce a pirate file that makes the SP and the judge disagree. Given that Tardos tracing accuses a subset of the users

(based on a threshold condition) it is possible for the judge and the SP to disagree on some users. This type of attack has not been considered before; we call it accusation withdrawal as it forces the SP to withdraw an originally made accusation since the judge cannot support it. Ensuring that no accusation withdrawal happens protects the SP from starting accusation procedures that are not going to succeed with high probability. Finally, during the file transfer stage the user may abort. Given that these file transfer procedures can be lengthy (due to the large size of the files to be downloaded) the possibility of an adversary exploiting aborting and restarting as an attack strategy is important to be included in the security model (and in fact we show an explicit attack if many aborts are permitted — see below).

5.1.1 Our contributions.

Rate-optimality. We propose the first rate-optimal (rate is defined as the size of the actual data over the size of total communication) asymmetric fingerprinting scheme. Our scheme is based on Tardos codes [127]. To achieve this property, we use a rate optimal 1-out-of-2 oblivious transfer ((2,1)-OT), and a new rate-optimal 1-out-of-2 strong conditional oblivious transfer ($(2,1)$-SCOT, [14]). Both are constructed in [82], and they are built on the rate-optimal homomorphic encryption scheme developed in the same paper. Based on these rate optimal protocols, we propose a rate-optimal fingerprinted data transfer protocol (tailored specifically for bias-based codes including Tardos codes).

More precisely, in a fingerprinted data transfer protocol, the sender has as private input two pairs of messages and biases. The sender and the receiver simulate two private biased coin tosses using SCOT and the receiver obtains one message from each pair (which one of the two it receives, is determined by the outcome of the biased coin flip).

The actual message transmission is based on the rate optimal OT protocol. Furthermore the sender selects randomly one of the two SCOT-s and revokes its receiver security, i.e., the sender will learn which one of the two versions the receiver has obtained in this SCOT. This partial revocation of receiver-security will enable the sender to correlate "pirate" files that are generated by coalitions of malicious users. Our final scheme inherits the communication efficiency of the underlying SCOT and OT protocols and thus it is communication-optimal: the rate of each data transfer is asymptotically 1.

A complete security analysis in the (extended) Pfitzmann-Schunter model: we analyze the security of our construction in an *extended* version of the Pfitzmann-Schunder model [112]. The extension we present is two-fold: first we extend the model to capture the setting of multiple accusations. In the original modeling only a single colluder was required to be accused. In the extended model we allow a set of users to be accused. This accommodates accusation algorithms that are based on Tardos fingerprinting [127] that have this capability. Group accusation in asymmetric schemes needs special care from a security point of view: it makes the system prone to *accusation withdrawal*, the setting where the server will have to withdraw an accusation because the judge is unable to verify it. We demonstrate (through actual implementation experiments, see Fig 11) that the straightforward application of Tardos identification (as may naively be inferred from the description of [34]) does not preclude accusation withdrawal. We subsequently show how to modify the accusation algorithm between judge and server so that no accusation withdrawal can take place. Our second model extension concerns the explicit treatment of the abort operation within the security model: all known asymmetric fingerprinting schemes rely on two-party coin tossing. Given that *fair* coin tossing is known to be unattainable [39] it follows that it may be possible for an adversarial set of users to exploit this weakness and utilize a transmission abort strategy with the

purpose of evading detection. We demonstrate that an explicit treatment of this in the security model is essential as if one enables users to restart after an abort, it is possible to completely break server security! (this fact went entirely unnoticed before). By properly controlling aborts and restarts we show how security can be maintained.

5.2 Rate-Optimal OT and SCOT Protocols

We recall that an oblivious transfer (OT) protocol and a strong conditional oblivious transfer (SCOT, [14]) protocol for predicate Q (s.t. $Q(x,y) \in \{0,1\}$) implement securely the following functionalities respectively (W.l.o.g., assume $|m_0| = |m_1|$):

$$f_{\mathsf{OT}}(b,(m_0,m_1)) = (m_b, \perp),\ \ f_{Q-\mathsf{SCOT}}(x,(y,m_0,m_1)) = (m_{Q(x,y)}, \perp)\ \ .$$

Here, we will use the rate optimal OT and SCOT protocols derived in [82] from our recently developed rate optimal large-output branching program homomorphic encryption (LHE) scheme. Our LHE scheme enables the receiver to compute on ciphertexts the value $f(x,y)$, where x is his input, y is sender input, and f is an arbitrary function that can be evaluated by a polynomial-size (integer-valued) branching program. In the LHE scheme of [82], the receiver encrypts (by using a variant of the Damgård-Jurik cryptosystem [45]) his input x, and sends the ciphertext $\mathsf{Enc}(x)$ to the sender. The sender evaluates privately large-output branching programs like in [76, 94], but does it in a communication-preserving manner. Let the output of the evaluation be denoted as $\mathsf{Eval}(P, \mathsf{Enc}(x))$, where P is a branching program that evaluates $f(\cdot, y)$ on input x. The sender returns a single "ciphertext" to the receiver, who then (multiple-)decrypts it as in [76, 94]. The rate of the LHE scheme is defined as $r = (|x| + |P(x)|)/(|\mathsf{Enc}_r(x)| + |\mathsf{Enc}_r(P(x))|)$. Assuming $|f(x,y)|$ is large, [82] showed

by using an intricate analysis how to achieve a rate $1 - o(1)$. We refer to [82] for more information.

Rate-optimal OT. As shown in [82], one can define a rate-optimal $(2,1)$-oblivious transfer protocol as follows. Let the server have a database (m_0, m_1) and assume that $P[x, (m_0, m_1)] = m_x$ for $x \in \{0,1\}$. Thus, the size of P is 1. Since rate-optimal $(2,1)$-OT has many applications, we will call it *oblivious download* (OD). Let $\mathsf{OD}_s[\mathsf{Enc}_r(x), (m_0, m_1)]$ denote the server side computation in this protocol, given client ciphertext $\mathsf{Enc}_r(x)$ and server input (m_0, m_1).

Rate-optimal SCOT. Also, as shown in [82], one can use the LHE of to construct an efficient SCOT protocol for the functionality $f_{Q-\mathsf{SCOT}}(x, (y, m_0, m_1))$, where Q has a polynomial-size branching program (i.e., $Q \in \mathbf{L/poly}$), as follows. Let P' be an efficient branching program that evaluates the predicate $Q(x, y)$. Let P be a large-value branching program, obtained from P' by just replacing the leaf value 0 with m_0 and 1 with m_1. The LHE scheme (and thus also the resulting SCOT protocol) will have computation, linear in the size of P', and communication $(1 + o(1))(|x| + |m_0|)$ and thus rate $1 - o(1)$. In the rest of the paper we will need the next instantiation of a new rate-optimal SCOT protocol.

Rate-optimal SCOT for the LEQ predicate. Denote $\mathsf{LEQ}(x, y) := [x \leq y]$. It is easy to see that LEQ can be evaluated by a branching program of size and length $\ell := \max(|x|, |y|)$. Thus, one can implement $f_{\mathsf{LEQ}-\mathsf{SCOT}}$ securely in time $\Theta(\ell)$ and rate $1 - o(1)$. Let us denote server computation in this protocol as $\mathsf{LEQ}_s[\mathsf{Enc}_r(x), (y, m_0, m_1)]$.

Remark. The security of the $\mathsf{OD}, \mathsf{SCOT}$ protocols are simple corollaries of the security proofs from [76, 82]. Also, one can also use an arbitrary efficient — with communication $o(|m_i|)$ — millionaire's protocol, like the one in [14] to find out $b = [x < y]$, and then use the oblivious download protocol to implement an optimal-rate SCOT protocol for the

LEQ predicate. However, we think that the use of optimal-rate LHE from [82] (instead of composing a millionaire's protocol and an OD protocol) is more elegant.

5.3 Fingerprinted Data Transfer for Bias-Based Codes

In this section, we will introduce the main building block of our Tardos-based asymmetric fingerprinting scheme, which we call fingerprinted data transfer.

As our fingerprinting scheme relies on the properties of fingerprinting codes (we only focus on binary codes here), let us first recall the basics about fingerprinting codes. A *binary fingerprinting code* [84] is a pair of algorithms $(\mathsf{Gen}, \mathsf{Trace})$, where Gen is a probabilistic algorithm taking a number N, an optional number (upper-bound on the detected coalition size) $t \in [N] = \{1, \ldots, N\}$ and security parameter ϵ as input and outputs N bit-strings $\mathcal{C} = \{C_1, \ldots, C_N\}$ (called codewords), where $C_i = \mathsf{c}_1^i \ldots \mathsf{c}_n^i$ for $i \in [N]$ and a tracing key tk. Trace is a deterministic algorithm inputting the tracing key tk and a "pirate" codeword C^*, and outputting a subset $\mathcal{U}_{acc} \subseteq [N]$ of accused users. A code is called *bias-based* [3] if each codeword $C_j = \mathsf{c}_1^j \ldots \mathsf{c}_n^j$ is sampled according to a vector of biases $\langle p_1, \ldots, p_n \rangle$, where $\forall j \in [N] \, \forall i \in [n], \Pr[\mathsf{c}_i^j = 1] = p_i$, and $p_i \in [0, 1]$.

A fingerprinting code is called $t-$*collusion resistant* (*fully collusion resistant* if $t = N$) if for any adversary $\mathcal{A}$ who corrupts up to t users (whose indices form a set $\mathcal{U}_{cor} \subset \{1, \cdots, N\}$), and outputs a pirate codeword $C^* = \mathsf{c}_1^* \ldots \mathsf{c}_n^*$ (which satisfies the marking assumption, i.e., for each $i \in [n], \mathsf{c}_i^* = \mathsf{c}_i^j$ for some $j \in \mathcal{U}_{cor}$), $\Pr[\mathcal{U}_{acc} = \emptyset$ or $\mathcal{U}_{acc} \not\subseteq \mathcal{U}_{cor} : \mathcal{U}_{acc} \leftarrow \mathsf{Trace}(tk, C^*)] \leq \epsilon$ (i.e., the probability that no users are accused or an innocent user is accused is bounded by ε).

We also recall the Tardos code [127] $F_{nt\epsilon}$ here, it has length $n = 100t^2k$, with $k = \log \frac{1}{\epsilon}$. The Gen algorithm generates a codeword as follows. For each segment index $j \in [n]$, it chooses a bias $p_j \in [0, 1]$ according to a distribution μ (see [127] for the

definition of μ). Each bias satisfies $\frac{1}{300t} \leq p_j \leq 1 - \frac{1}{300t}$, where t is the collusion size. For each codeword $C = \mathsf{c}_1 \ldots \mathsf{c}_n$ outputted by Gen, $\Pr[\mathsf{c}_j = 1] = p_j$, and $\Pr[\mathsf{c}_j = 0] = 1 - p_j$ for all $j \in [n]$. Regarding security, there is a Trace algorithm such that, for any coalition of size at most t, with probability at least $1 - \epsilon^{t/4}$ accuses a member of the coalition, while any non-member is accused with probability at most ϵ.

5.3.1 Definitions of fingerprinted data transfer

Now we define our main building block of fingerprinted data transfer (FDT for short). Recall that each user should receive a fingerprinted copy of the file according to his codeword. In the case of asymmetric fingerprinting, the segments of the file will be transferred in an oblivious fashion so that the server should be aware of only half of the user fingerprinting code. To be more specific, all segments are transmitted using oblivious transfer to enable the user to receive one of the versions, and for each pair of segments $(2i - 1, 2i)$, where $i \in [n]$, the server will know one of the segments, the version that the user receives.

Intuitively, if we double the length of the fingerprinting code (dividing the file into $2n$ segments), each user is essentially assigned two codewords, one is known to the server, thus the Trace algorithm can be executed to identify malicious users; the other one is unknown to the server, and will be revealed to the judge only when dispute happens. A user will be accused only when both codewords are considered contributing to a pirate file. In this way, a malicious SP $\mathcal{S}$ frames an honest user unless innocent users may be accused in the fingerprinting code.

We also need to be careful that if malicious users know which half of the codeword is known to the server, they may collude in a way that every codeword in the collusion only contribute to half of the file thus no one will be accused on both fingerprints. Thus

for the segments $(2i-1, 2i)$ for $i \in [n]$, the index that the segment version is revealed to the server is also oblivious to the user.

The asymmetric fingerprinting scheme will essentially be running FDT (defined below) in parallel for all pairs of the segments $(2i-1, 2i)$, thus it is enough for us to illustrate the idea by considering only the sub-protocol for one pair of segments. As Tardos code is binary, there are only two versions for each segment. Consider two parties, a sender $\mathcal{S}$ and a receiver $\mathcal{R}$. The sender has two pairs of messages, two rational valued "biases" in $[0, 1]$ and one bit c as inputs. The receiver has no input. After the execution of the FDT protocol, $\mathcal{R}$ will sample one message from each of the two pairs following the binary probability distribution defined by the two biases and $\mathcal{S}$ will learn the output of the receiver for the c-th pair. This describes the ideal operation of the primitive for the case of one pair of segments. It is straightforward to extend to an arbitrary number of pairs. The following is the formal definition of the fingerprinted data transfer for bias-based codes including our main target Tardos code [127].

Definition 15. *A* fingerprinted data transfer functionality Π *involves two parties, a sender* $\mathcal{S}$ *and a receiver* $\mathcal{R}$*. The sender inputs two biases* $p_0, p_1 \in [0, 1]$*, four messages* $(m_0^0, m_0^1), (m_1^0, m_1^1)$*, and a bit* $c \in \{0, 1\}$*; at the end of the protocol,* $\mathcal{R}$ *outputs* $\{m_i^{b_i}\}$ *for* $i, b_i \in \{0, 1\}$ *such that* $\Pr[b_i = 1] = p_i$*; while* $\mathcal{S}$ *outputs* b_c*. We can express this (probabilistic) functionality as:*

$$\Pi[\perp, ((p_0, p_1), (m_0^0, m_0^1, m_1^0, m_1^1), c)] = [(m_0^{b_0}, m_1^{b_1}), b_c], \textit{where } \Pr[b_i = 1] = p_i$$

Somewhat similar functionalities have been used for completely different applications, see, e.g. [2, 48]. The FDT protocol of Sect. 5.3.2 might be modified so as to be used in these applications; we omit further discussions.

Following the standard definitions in secure computation [59], we first define security in the case that both sender and receiver are semi-honest (i.e., they follow the protocol, but try to learn more by performing additional local computation). Recall that the view of a party is composed of random variables of his inputs, coins and the messages received from the other party.

Definition 16. *If a protocol satisfies the following properties, we say that it securely implements fingerprinted data transfer.*

Correctness: *The receiver will obtain* $(m_0^{b_0}, m_1^{b_1})$*, satisfying that* $\Pr[b_i = 1] = p_i$ *for* $i = 0, 1$*. The sender will receive* b_c *with probability 1.*

Receiver Security: *The joint distribution of sender's view and the outputs in a real the protocol can be simulated by the inputs and outputs of the sender alone together with the ideal outputs of the functionality. That is,* $\forall PPT$ *semi-honest sender* $\mathcal{S}$*,* $\exists$ PPT $\mathcal{S}'$*, s.t.,* $\mathcal{S}'([(p_0, p_1), (m_0^0, m_0^1, m_1^0, m_1^1), c], b_c) \circ (m_0^{b_0}, m_1^{b_1}, b_c)$ *is computationally indistinguishable from* $VIEW_{\mathcal{S}} \circ OUTPUT$*.*

Sender Security: *The joint distribution of receiver's view and the outputs in a real protocol can be simulated by the inputs and outputs of the receiver alone, together with the ideal outputs. That is,* $\forall PPT$ *semi-honest receiver* $\mathcal{R}$*,* $\exists$ PPT $\mathcal{R}'$*, s.t.,* $VIEW_{\mathcal{R}} \circ OUTPUT$ *is computationally indistinguishable from* $\mathcal{R}'(m_0^{b_0}, m_1^{b_1}) \circ (m_0^{b_0}, m_1^{b_1}, b_c)$*.*

Note that the bits of the codeword $\{b_i\}$ are publicly recoverable from $\{m_i^{b_i}\}$.

5.3.2 A communication-optimal fingerprinted data transfer protocol

On the receiver $\mathcal{R}$ side, for each pair of messages, say pair 0, FDT will enable an oblivious sampling of one message from (m_0^0, m_0^1) according to the bias p_0, i.e., $\mathcal{R}$ receives m_0^1 with probability p_0. To enable efficient oblivious sampling, suppose $p_0 \approx t_0/T$ for

some t_0, where $T = 2^\ell$ and ℓ is the precision level (this provides an exponentially good approximation). To run a coin tossing protocol to generate a random coin u, $\mathcal{R}$ and the SP $\mathcal{S}$ can utilize a SCOT protocol (e.g., [14]) to transmit the data in a way that the user receives m_0^1 iff $u \leq t_0$. Doing this will allow the receiver to get m_0^1 with probability close to $p_0 = t_0/T$. Furthermore, they can run such procedure twice for the two pairs, and then run a $(2,1)$-OT protocol to reveal one of the bit to the SP.

Unfortunately, directly applying the SCOT protocol from, e.g, [14] will result in a communication rate as low as $1/\ell$, as the sender has to send ℓ ciphertexts with similar size to m_i^0. Moreover, a malicious user may abort after receiving the file without revealing half of his bits. To deal with these concerns, our protocol will be divided into two phases, the first (the handshake phase) samples only the codewords according to the biases that are specified by the sender; the second (the content-transfer phase) transfers the actual content according to the codewords that have been drawn. In our implementation, we will only use the SCOT protocol to sample the distribution (transfer only short messages) and then employ a rate-optimal OT protocol (we call oblivious download, OD for short)to execute the content-transfer after the OT protocol is run in which the SP is the receiver and the SP sees one of the bits. We assume that during the hand-shake phase, the sender and receiver exchange their public keys with the corresponding certificates.

Now we proceed to construct the new fingerprinted data transfer protocol. Suppose the sender has $p_0 = \frac{t_0}{T}$, $p_1 = \frac{t_1}{T}$; these determine two distributions over $\{0,1\}$. The sender also has two pairs of messages as inputs $(m_0^0, m_0^1), (m_1^0, m_1^1)$, and prepares another two pairs of $(h_0^0, h_0^1), (h_1^0, h_1^1)$, where $h_i^b = H(m_i^b)|i|b$ for $i, b \in \{0,1\}$. We assume that H is a collision resistant hash function shared by the sender and the receiver, and Com is a secure (binding and hiding) commitment scheme. We choose Enc_r and Enc_s to be good rate additive homomorphic encryption schemes (e.g. using the Damgard-Jurik [45]

encryption to encrypt the message bit by bit as in [82]). Here, $\mathcal{R}$ knows the secret key of Enc_r and $\mathcal{S}$ knows the secret key of Enc_s. Recall that $\mathsf{LEQ}_s[\mathsf{Enc}_r(x), (y, m_0, m_1)]$ denotes the sender computation in a concrete SCOT protocol that implements $f_{\mathsf{LEQ-SCOT}}$, and $\mathsf{OD}_s[\mathsf{Enc}_r(x), (m_0, m_1)]$ denote the computation of the server in this protocol, given client ciphertext $\mathsf{Enc}_r(x)$ and server input (m_0, m_1) (defined in Section 5.2). The full protocol of FDT is presented in Fig 10.

We can estimate $1/\alpha$ of our FDT protocol as follows:

$$\begin{aligned} & \frac{2(|\mathsf{Com}(r_0)| + |\mathsf{Enc}_r(s_0)| + |\mathsf{Enc}_r(h_0^{b_0})|) + |\mathsf{Enc}_s(c)| + |\mathsf{Enc}_s(h_c^{b_c})| + 2|\mathsf{Enc}_r(m_0^{b_0})|}{2|m_0^{b_0}| + 1} \\ \approx \quad & \frac{o(|m_0^{b_0}|)}{2|m_0^{b_0}|} + \frac{|\mathsf{Enc}_r(b_0)| + |\mathsf{Enc}_r(m_0^b)|}{|b_0| + |P[b_0, (m_0^0, m_0^1)]|} \to \frac{1}{r}, \text{when } m \to \infty \ , \end{aligned}$$

where m is the message size and r is the rate as defined in Sect. 5.2. We can group several terms into $o(|m_0^{b_0}|)$ as all those are encryptions (or commitments) of fixed size short messages. Thus, when the LHE scheme is rate optimal as [82], our FDT protocol (and further our asymmetric fingerprinting scheme, see next section) is also rate optimal.

Security analysis. We briefly explain the properties of our protocol in the semi-honest model. Correctness follows from the coin tossing and the property of the LHE [82]. For instance, $\overline{\mu_0} = \mathsf{Enc}_r(h_0^1), C_0 = \mathsf{Enc}_r(m_0^1)$, if $r_0 \oplus s_0 \leq t_0$, in this case, $\Pr[b_0 = 1] = t_0/T = p_0$. For security, as we are working in the semi-honest model for now, the sender and receiver views can be simulated easily to preserve the consistency with the output.

Lemma 32. *Our protocol shown in Fig. 10 securely implements the fingerprinted data transfer functionality. Specifically, it is correct; and it satisfies receiver security if the underlying encryption Enc_r is IND-CPA secure; it satisfies sender security if the underlying commitment scheme $\mathsf{Com}(\cdot)$ is computationally hiding, and the encryption Enc_s is IND-CPA secure.*

Receiver $\mathcal{R}$		Sender $\mathcal{S}$
		$\mathcal{S}$ selects $r_0, r_1 \leftarrow_r Z_T$, and
	$\xleftarrow{\mathsf{Com}(r_0),\mathsf{Com}(r_1)}$	computes commitments $\mathsf{Com}(r_0), \mathsf{Com}(r_1)$
$s_0, s_1 \leftarrow_r Z_T$	$\xrightarrow{\mathsf{Enc}_r(s_0),\mathsf{Enc}_r(s_1)}$	
		$\mathcal{S}$ computes $c_0 = \mathsf{Enc}_r(r_0 \oplus s_0)$
		$\mathcal{S}$ computes $c_1 = \mathsf{Enc}_r(r_1 \oplus s_1)$
		$\mathcal{S}$ computes $\overline{\mu_0} = \mathsf{LEQ}_s[c_0, (t_0, h_0^1, h_0^0)]$
	$\xleftarrow{\overline{\mu_0},\overline{\mu_1},\mathsf{Enc}_s(c)}$	$\mathcal{S}$ computes $\overline{\mu_1} = \mathsf{LEQ}_s[c_1, (t_1, h_1^1, h_1^0)]$
$\mathcal{R}$ retrieves $h_0^{b_0}, h_1^{b_1}$		
and computes $u_c =$		
$\mathsf{OD}_s[\mathsf{Enc}_s(c), (h_0^{b_0}, h_1^{b_1})]$	$\xrightarrow{u_c}$	
		$\mathcal{S}$ decrypts u_c and checks the validity
		$\mathsf{Enc}_r(b_0) = \mathsf{LEQ}_s[c_0, (t_0, 1, 0)]$
		$\mathsf{Enc}_r(b_1) = \mathsf{LEQ}_s[(c_1, (t_1, 1, 0)]$
		$C_0 = \mathsf{OD}_{\mathbf{s}}[\mathsf{Enc}_r(b_0), (m_0^0, m_0^1)]$
	$\xleftarrow{C_0,C_1}$	$C_1 = \mathsf{OD}_{\mathbf{s}}[\mathsf{Enc}_r(b_1), (m_1^0, m_1^1)]$
$\mathcal{R}$ checks the validity,		
outputs: $m_0^{b_0}, m_1^{b_1}$		$\mathcal{S}$ outputs b_c (as inferred by $h_c^{b_c}$)

Figure 10: Fingerprinted Data Transfer. $\{(m_b^0, m_b^1), p_b = \frac{t_b}{T})\}_{b=0,1}, c$ are inputs of $\mathcal{S}$

Proof. Correctness: it is easy to see. First, the exchange of random values enables both parties to obtain uniform values $\{r_i + s_i\}_{i\in\{0,1\}}$; then, the SCOT protocol will enable the receiver to receive h_i^1 (thus $b_i = 1$) with probability p_i as $\Pr[r_i + s_i \leq t_i] = t_i/T = p_i$, the rest simply follows from the correctness of the OD protocol.

Receiver Security: This property shows that the sender has no advantage predicting b_{1-c} given the corresponding value p_{1-c}. In more detail, $VIEW_{\mathcal{S}} \circ OUTPUT$ is composed of $[(r_0, r_1, \bar{r}), (p_0, p_1, m_0^0, m_0^1, m_1^0, m_1^1, c), (E(s_0), E(s_1), u_c)] \circ [(m_0^{b_0}, m_1^{b_1}), b_c]$, where r_0, r_1 are local coins for the commitments and $\bar{r}$ for other coins for encryptions; $(p_0, p_1), (m_0^0, m_0^1), (m_1^0, m_1^1), c$ are inputs, and $E(s_0), E(s_1), u_c$ are received messages, and $OUTPUT$ are $(m_0^{b_0}, m_1^{b_1})$, and a bit b_c which are consistent with the coin tossing and the values of the biases.

The simulation strategy is as follows: the simulator $\mathcal{S}'$ first samples $r_0, r_1, \bar{r}$ uniformly, and then he samples b_{1-c} according to the bias p_{1-c}, then with input b_c, $\mathcal{S}'$ chooses s_0, s_1 according to the value of b_0, b_1, i.e., $s_i + r_i \leq t_i$ if $b_i = 1$. And u_c is generated by encrypting $h_c^{b_c}$ which can be derived from $m_c^{b_c}$. Ideal output as simply two samples of messages $m_0^{b_0}, m_1^{b_1}$ satisfying $\Pr[b_i = 1] = p_i$, and b_c. It is easy to see that the simulated view preserves the correlation with outputs. And the simulated view is different with the real view only at $E(s_{1-b_c})$. We can conclude that these two views are computationally indistinguishable as the distinguisher only has public key of E, otherwise the distinguisher can break the semantic security of E.

Sender Security: This property guarantees that the receiver learns nothing beyond what he can learn from the messages he receives. The view of a receiver is composed of local coins and received messages $(s_0, s_1, \bar{s}), (c(r_0), c(r_1), \overline{\mu_1}, \overline{\mu_2}, E_1(c), E(m_0^{b_0}), E(m_1^{b_1}))$, and $OUTPUT$ is the same as above. The simulation can be done similarly as follows, the simulator $\mathcal{R}'$ first simulates the local coins using random samples. From the ideal output

$m_0^{b_0}, m_1^{b_1}$, the simulator $\mathcal{R}'$ calculates the values of $b_0, b_1, h_0^{b_0}, h_1^{b_1}$, and then simulates the transcript accordingly. Specifically, $\mathcal{R}'$ commits to two values r_0, r_1 in a way that consistent with b_0, b_1 without knowing the biases, if $b_i = 1$, $r_i + s_i = 0$, if $b_i = 1$ $r_i + s_i = 2^\ell$, doing these will preserve the consistency of $r_i + s_i \leq t_i$ if $b_i = 1$ without knowing the value of t_i. $\mathcal{R}'$ then generates $\overline{u_i}$ using $E(h_i^{b_i})$, and simulates $E_1(c)$ as $E_1(c')$ using a random bit c'; and finally, $\mathcal{S}'$ simulates C_0, C_1 using $E(m_0^{b_0}), E(m_1^{b_1})$.

We can see that the simulated view are consistent with the ideal output, the only difference are at the committed value $c(r_0), c(r_1)$ and $E_1(c)$. No efficient algorithm can distinguish the two views, otherwise, it can break either the binding property of the commitment scheme or break the semantic security of E_1 which can be shown through simple game transitions. □

Work in the malicious model. Ultimately, we would like to design protocols to defend against malicious adversaries who may arbitrarily deviate from the protocol. The general method that in every step, both parties deploy zero-knowledge proofs to show that they follow the protocol, could be inefficient. Note that our protocol is highly structured, user misbehaviors can be easily detected by the SP with some routine checks about the consistency in the transcripts. In the 2nd round in coin tossing phase, the user could gain nothing by not following protocol, as simulation for malicious user is not influenced by the choices of s_0, s_1. While in the 4th round, the SP checks the validity of $h_c^{b_c} = H(m_c^{b_c})||c||b_c$, if u_c is not calculated as in the protocol and passes the checking, it means the malicious user finds a value equal to $H(m_c^{1-b_c})$. As the message segment $m_c^{1-b_c}$ has sufficient entropy thus $h_c^{1-b_c}$ is also unpredictable, otherwise, the user could easily find a collision by randomly sample messages from the distribution of $m_c^{1-b_c}$. To be more specific, suppose M is the space of $m_c^{1-b_c}$ and $\mathcal{D}$ is its distribution, and $H(M) = \{H(m) : m \in M\}$, $\mathcal{D}$ will induce a distribution $\mathcal{H}$ on $H(M)$. Suppose

the sender can predict $h(m_c^{1-b_c})$ with probability δ, then the maximum probability of $\mathcal{H}$ is no less than δ. Let us use h_0 to denote the most probable value in $H(M)$. The adversary $\mathcal{A}$ simply sample m_i randomly according to $\mathcal{D}$, and computes the hash value. Following the Chernoff bound, using $O(\frac{1}{\delta^2})$ many samples, $\mathcal{A}$ will almost certainly reach h_0 twice. At the same time, the probability that there are two same messages appear in the sampled messages is exponentially small, as the most probable message from $\mathcal{D}$ appears with negligible probability. Based on these two facts, $\mathcal{A}$ found a collision.

Regarding malicious SP, the user can also do some simple checks of the consistency of the hash values. Note that there is a trusted judge that makes the final decision about the set of accused users. We will show in next section (as the judge is not involved with the FDT protocol) how we can take advantage of this third-party to "force" the SP to follow the protocol, by adding some simple and efficient "proofs of behavior". We require the SP signs on each round of messages she sends together with the user identity, and the user also signs on each round of messages he sends. We also let user store part of the transcripts and reveal them to the judge in case of dispute. Through a careful analysis of Tardos code property together with these simple mechanisms, we can argue security of our asymmetric fingerprinting scheme in the malicious model.

5.4 An Optimal Asymmetric Fingerprinting Scheme Based on Tardos Code

Pfitzmann and Schunter [112] define an asymmetric fingerprinting scheme to be a collection of four protocols $\langle \mathsf{key\_gen}, \mathsf{fing}, \mathsf{identify}, \mathsf{dispute} \rangle$. The algorithm $\mathsf{key\_gen}$ can be used by a user to produce a public and a secret-key. The protocol fing is a two-party protocol between a user and an SP that will result in the user obtaining the fingerprinted copy of the file and the SP receiving some state of the user codeword. The algorithm $\mathsf{identify}$ is an algorithm that, given a pirate copy and the state of the SP,

outputs a non-empty set of public keys (corresponding to the accused user identities). Finally the algorithm dispute is a 3-party protocol between the judge (or arbiter as it is called in [112]), the user and the SP that either accepts the SP's accusation or rejects it (depending on the evidence presented by the involved parties). For brevity we refer to [112] for the full syntax of the scheme. Regarding the security model, an asymmetric fingerprinting scheme should satisfy two security properties: (i) *security for the SP*, that states that no malicious coalition of less than t users can escape the accusation of one of its members from the identify algorithm as well as the validation of this accusation by the dispute protocol, and (ii) *security for the user*, that states that an innocent user cannot be implicated by a malicious SP (who can also corrupt other users) in being responsible for a certain pirate copy. Formally definitions are presented below:

Definition 17. *A fingerprinting scheme satisfies* t-collusion resilient SP security*, if no* PPT *adversary* $\mathcal{A}$ *wins the following game with a non-negligible probability:*

- $\mathcal{A}$ *sends corruption queries for up to* t *different users with public keys* $S = \{pk_{i_1}, \ldots, pk_{i_t}\}$. $\mathcal{A}$ *then receives* t *copies of fingerprinted file each file has* n *segments, i.e., for* $i \in S, F_i = f_i^1 || \ldots || f_i^n$*, where* n *is the length of the codeword, and all secret states of those users.*

- $\mathcal{A}$ *outputs a pirate copy* F_0 *satisfying the marking lemma that for each segment* f_0^i *of* F_0*, there exists an index* $j \in S$*, such that* $f_0^i = f_j^i$.

SP runs the identify *algorithm on* F^**, and returns a set of public keys* $\mathcal{PK}$*. The judge, SP, and* $\mathcal{A}$ *(or corresponding users) then run the* dispute *protocol and suppose* $\mathcal{PK}_{accused}$ *is the set of public keys accepted for accusation.* $\mathcal{A}$ *wins if,* $\mathcal{PK}_{accused} \not\subseteq S$ *or* $\mathcal{PK}_{accused} = \emptyset$.

Definition 18. *A fingerprinting scheme satisfies* c-collusion resilient SP security*, if no* PPT *adversary* $\mathcal{A}$ *wins the following game with a non-negligible probability:*

- $\mathcal{A}$ *runs* key_gen *for multiple times and runs the* fing *protocol with the SP using at most* t *different public keys* $S = \{pk_{i_1}, \ldots, pk_{i_t}\}$. *These* t fing *protocols are run independently.* $\mathcal{A}$ *receives* t *copies of fingerprinted file each file has* n *segments, i.e., for* $i \in S, F_i = f_i^1 || \ldots || f_i^n$, *where* n *is the length of the codeword.*

- $\mathcal{A}$ *outputs a pirate copy* F_0 *satisfying the marking lemma that for each segment* f_0^i *of* F_0, *there exists an index* $j \in S$, *such that* $f_0^i = f_j^i$.

SP runs the identify *algorithm on* F^*, *and returns a set of public keys* $\mathcal{PK}$. *The judge, SP, and* $\mathcal{A}$ *(or corresponding users) then run the* dispute *protocol and suppose* $\mathcal{PK}_{accused}$ *is the set of public keys accepted for accusation.* $\mathcal{A}$ *wins if,* $\mathcal{PK}_{accused} \not\subseteq S$ *or* $\mathcal{PK}_{accused} = \emptyset$.

Definition 19. *A fingerprinting scheme satisfies* innocent user security, *if for any user* U, *any PPT adversary* $\mathcal{A}$ *wins the following game with a negligible (in security parameter) probability:*

- U *runs* key_gen *and the* fing *protocol with* $\mathcal{A}$ *using public key* pk_u.

- $\mathcal{A}$ *arbitrarily corrupts or generate codewords for other users, and* $\mathcal{A}$ *outputs a pirate file* F_0.

$\mathcal{A}$ *wins if* $\mathcal{A}$ *submits* pk_u *to the judge as output of* identify *algorithm and* pk_u *is accepted for accusation in the* dispute *protocol.*

Definition 20. *A* PPT *adversary* $\mathcal{A}$ *plays the same game with the SP as in Definition 18, the conditional that* $\mathcal{A}$ *wins the game is changed to* $\mathcal{PK} \neq \mathcal{PK}_{accused}$ *or* $\mathcal{PK} = \emptyset$, *where* $\mathcal{PK}$ *is the set of public keys identified by the SP and* $\mathcal{PK}_{accused}$ *is the set of public keys outputted after* dispute. *A fingerprinting scheme satisfies* t-collusion resilient SP security under group accusations, *if for any PPT adversary* $\mathcal{A}$ *can win this game with a negligible (in security parameter) probability.*

In addition to the above basic requirements, we put forth two additional properties that will be of interest.

Communication efficiency. The communication rate of an asymmetric fingerprinting scheme is measured as the ratio of the length of the file that is distributed to the users over the total communication complexity of the fing protocol. In a communication optimal asymmetric fingerprinting scheme it holds that the rate approaches 1 as the length of the file becomes larger. All known schemes in the literature [34, 111–113] have rate at most 0.5.

Security for the SP under group accusations. In [112] the algorithm identify is responsible for producing a single colluder whose implication is guaranteed to be validated by the dispute algorithm. In [34] this is extended to group accusation, i.e., the identify algorithm produces a set of accused users as output (this is possible given that the underlying fingerprinting code enables such group accusation). For SP security to be preserved under group accusations however, it should be the case that for each accused user, its implication to the construction of the pirate copy is validated by the dispute protocol. In the other case, the SP will have to withdraw at least one accusation (something that may lead to problems in a practical deployment). Therefore a protocol solution should guarantee in the setting of group accusation no accusation withdrawal can occur with non-negligible probability. We refer the formal definitions to the full version.

5.4.1 Our construction.

We next describe our construction which satisfies the original security requirements of [112] as well as the two properties that we described above. Specifically it is the first asymmetric fingerprinting scheme with both optimal communication complexity

and code length. And one can easily adapt our construction to other asymmetric fingerprinting scheme from any bias-based code.

Recall the definition of Tardos code as explained in section 5.3, the main task is the fing protocol, which will be constructed from our fingerprinted data transfer protocol (see Fig 10) with some extra checks to achieve security in the malicious model in which the adversary may not follow the protocol. To describe the generation of the fingerprinted copy of each user in more detail, let us abstract each variant of a segment m_i^b with a value in $\{0,1\}$,where $i \in [2n], b \in \{0,1\}$ and $2n$ is the length of the fingerprint. Thus, the fingerprinted file of each user is a $2n$-bit string, where each bit signifies which variant of the corresponding segment the user received. It will be generated so that n bits from a set $L \subseteq [2n]$ will be known to the SP, while the other n bits (from $[2n] \setminus L$) will only be known by the user. The user, however, will not know if a given location belongs to L or not. Each of the parts L and $[2n] \setminus L$ is an instance of the Tardos code [127]. The two parts are produced by generating two segments at a time, using the functionality achieved by the protocol in Figure 10, i.e., for the i-th pair of segments $(2i-1, 2i)$, where $i \in [n]$, the user and the server runs the fingerprinted data transfer with the SP taking $[(p_{2i-1}, p_{2i}), (m^0_{2i-1}, m^1_{2i-1}), (m^0_{2i}, m^1_{2i}), c_i]$ as inputs. Based on the security properties of this protocol, the user receives two variants of two different segments, while the SP does not know one of them and the user is oblivious regarding which one the SP knows. Our asymmetric fingerprinting scheme proceeds as follows:

Key generation. The key_gen for the user is simply the generation of two public-secret key pairs $(pk_1, sk_1), (pk_2, sk_2)$. The first is for a digital signature scheme (which we use as black-box), and the second is for the additively homomorphic encryption Enc_r used in the fingerprinted data transfer.

The fing protocol. The user inputs his public and secret keys while the SP inputs the SP public keys, and system parameters, e.g., the level of precision ℓ. Furthermore, the protocol is stateful from the point of view of the SP. The state of the SP contains the definition of the Tardos code parameters (e.g., probabilities $\{p_i\}$). Also, the SP has as private inputs a set $L = \{c_1, \ldots, c_n\}$, and a file that is divided in $2n$ segments for each one of which there are two variations. The i-th segment, b-th variant is denoted by m_i^b.

The fing protocol proceeds as follows: the SP and the user first carry out a handshake protocol to prepare the system parameters including the exchange of the public keys of each other; then for each i-th pair of segments with indices $(2i - 1, 2i)$ where $i \in [n]$, the user and the server runs the FDT with the SP taking $[(p_{2i-1}, p_{2i}), (m_{2i-1}^0, m_{2i-1}^1), (m_{2i}^0, m_{2i}^1), c_i]$ as inputs, and these n protocols are run in parallel. Also in each round, if the SP sends out a message, she signs on the message together with the user's identity; if the user sends a message, he signs it as well. During protocol execution each party verifies that the messages they receive are proper and if they are not they will abort the protocol.

Furthermore some additional checks are in place to detect the malicious behavior within fing as explained at the end of section 5.3.2. These are as follows: The user checks after receiving the actual data segments (in the last round) whether they are consistent with the hash values (see Remark in section 5.3.2) he received in the 3-rd round. The SP, checks the validity of the hash value she received in the 4-th round. Also, both parties will store some information for input to the dispute protocol. The user keeps the commitments received from the first round and the hashed values and the encrypted bits of $\{\mathsf{Enc}_s c_i\}$ for $i \in L$, received in the 4-th round; the SP keeps the encrypted random coins of the user received in the 2nd round. Note that these checks do not enforce semi-honest behavior - nevertheless we will show (see Theorem 34,36)

they are sufficient for security against malicious parties in the presence of the judge (which is assumed honest).

We see that our fing protocol essentially runs out FDT protocol in parallel with only some extra signatures, thus it inherits the rate optimality from FDT.

The identify algorithm. This algorithm takes a pirate file M, and all users' half codeword $X_1, \ldots, X_N$ together with the location indices $L_1, \ldots, L_N$ and the vector of biases $\langle p_1, \ldots, p_{2n} \rangle$ as inputs. It first extracts a codeword $Y = y_1 \ldots y_{2n} \in \{0,1\}^{2n}$ from M (as we assume each bit is publicly computable from the segment). For the ease of presentation, we describe the algorithm for one user with stored codeword $X = x_{c_1} \ldots x_{c_n}$ and $L = \{c_1, \ldots, c_n\}$. For each $j \in L$, it computes:

$$U_j = \begin{cases} \sqrt{\frac{1-p_j}{p_j}}, & \text{if } x_j = 1; \\ -\sqrt{\frac{p_j}{1-p_j}}, & \text{if } x_j = 0 \end{cases}$$

as in [127]. The service provider calculates the score of the user over the locations in L; $S = \sum_{j \in L} y_j U_j$, and if $S > Z$, where $Z = 20tk$, the SP reports this user to the judge. This is repeated for every user and a list of accused users is compiled and reported to the judge.

The dispute protocol. This is a protocol among the SP, the judge and a user. The two parties first submit to the judge the protocol transcript they stored. In more detail, the judge requires the user to submit SP's commitments sent in the 1st round and the hash values; also, the judge requires the SP to submit the biases and the encryptions from the user in the 2nd round as well as openings of her commitments. The judge first verifies the code parameters, then does the following checks, (1). the validity of the segments, i.e., they should be one of the versions. (2). the validity of all signatures, if any signature is invalid, accuse the party who submits it. (3). Otherwise, the judge

checks whether the user codeword is generated properly, i.e., each bit of the codeword is consistent with the coin-tosses – whether $b_i = [r_i + s_i \leq t_i]$ where b_i is the i-th bit, r_i, s_i, t_i are as in the FDT (the notation $[\cdot]$ here denotes a predicate). To finish this check, the judge requires the SP to open her commitments, and the user to reveal his coins in the ciphertext $\{\mathsf{Enc}_r(s_i)\}$and prove their validity. (4). Furthermore, the judge requests from the user to submit the encrypted locations $\{\mathsf{Enc}_s(c_i)\}$ and requests the SP to decrypt it and prove a correct decryption, so that the judge calculates the set of locations L. Any party failed to prove the correct decryption will be accused.

If all the checks pass, the judge will recover user's fingerprint x' from the bits $\{b_i\}$, also he inspects the pirate content and extracts the fingerprint y'. Then he computes the U' as in the *identify* algorithm for locations $L' = [2n] \backslash L$ using x', y' as inputs. Finally, for any user reported, the judge calculates his score over the locations in L'; $S' = \sum_{j \in L'} y'_j U'_j$, and make decisions if $S' > Z'$, where $Z' = Z/2 = 10tk$, he validates the accusation; otherwise, the user is acquitted.

Note that we are using a lower threshold on the judge-side, to counter-balance the probability that a user is accused over L, but not over $[2n] \setminus L$. In fact this is an essential feature of our scheme to ensure security for the SP under group accusations. We in fact show that if $Z' = Z$ it can happen with high probability that the SP will have to withdraw an accusation; in Fig 11, we explore this experimentally by having a coalition of 40 users where the pirate strategy is as follows: the pirate content is formed via a majority strategy by the segments available to the coalition of size t. For each segment with probability p (a parameter of the strategy) the pirate segment is determined with probability p to be the majority of the segments of all t users or with probability $1-p$ the segment of the first user. We variate the parameter p of the strategy and we demonstrate from experimental data that for suitable choice of p the number

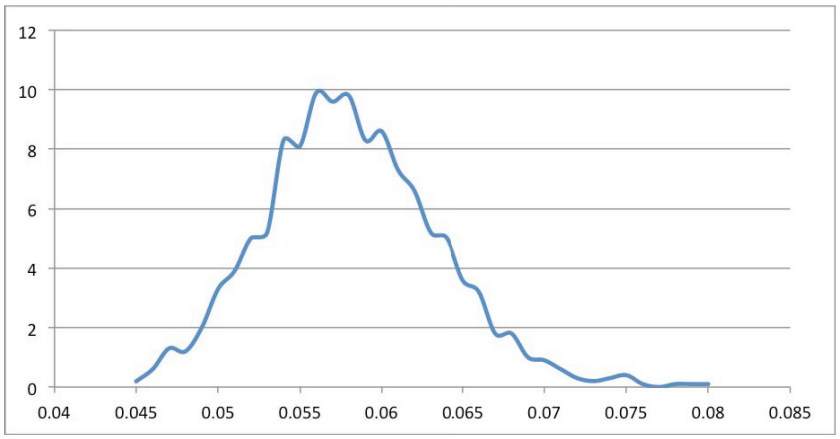

Figure 11: The vertical axis represents the number of accusation withdrawals (i.e., for how many users the service provider has to abandon its accusation); the total number of colluding users is 40. The horizontal access is the parameter of the colluding strategy p; for a suitably choice of p the accusation withdrawals reach 25% of colluders.

of accusation withdrawals can be as high as as a quarter of the coalition. One would expect that in practice, such high level of accusation withdrawal would impact seriously the credibility of the SP. In our construction, by appropriately differentiating the tracing algorithm between the judge and the SP we circumvent this problem entirely. It should be noted that this issue was not addressed in [34] where the Tardos tracing algorithm was also used for determining the implication of users.

Remark: There are two phases when the judge requests one of the two parties to prove a valid decryption of a bit. As we are using a variant of DJ encryption [45], the prover can simply reveal the message together with the random coins used in encryption as it decodes uniquely in this form. Specifically, if a message m is encrypted with random coin r, it results in a ciphertext $c = (n+1)^m r^{n^s} \mod n^{s+1}$, the prover can decrypts

c to recover m and then obtains $r = d^{n^{-s} \mod \phi(n)} \mod n$, where $d = r^{n^s} \mod n$ is computed from $c \cdot (n+1)^{-m} \mod (n^{s+1})$.

5.4.2 Security analysis.

We first give here explain the intuitions about the security analysis.

Security for the innocent user. We will show that no innocent user will be framed. In the case of a semi-honest SP, she follows the fingerprinted data transfer protocol and the accusation procedure, but will try to make a pirate copy to frame some innocent user. As the FDT protocol satisfies the simulation based definition, from the composition lemma [29], A semi-honest SP will have the similar behavior interacting with only an oracle which returns her half of the codeword. In this case, the SP wins only when she is able to break the innocent user security of Tardos code as shown in Theorem 2.1 in [127] that an innocent user will be framed with a probability no bigger than ϵ regardless of what biases are used and what is the pirate copy.

Lemma 33. *An innocent user will be accused with negligible probability by a semi-honest service provider assuming that the encryption* Enc_r *used is IND-CPA secure.*

Now we consider a malicious SP who may arbitrarily deviate from the protocol. With the simple checks, there is only one class of deviations left (which is not yet clear whether always detectable): the malicious SP submits different biases to the judge with those used during fing; This includes many subtle attacks, e.g., in one instance of FDT, the malicious SP uses the same messages in each pair to do the transmission, i.e., SP inputs $(m_0^{b_0}, m_0^{b_0}), (m_1^{b_1}, m_1^{b_1})$ (same for the hash values). Doing this the malicious SP will know the complete codeword of the user. Similarly, the SP could swap the messages in each pair, i.e., transmit version $1 - b_i$ if the code is b_i. Both of these behavior can

be seen as special case of the above deviation. In the first case, the user codeword is essentially generated using a vector of probabilities $\langle p_1, \ldots, p_{2n} \rangle$, where each $p_i \in \{0, 1\}$, while the latter case is that each $p_i = 1 - p'_i$ where p'_i is the reported bias. As the judge will check the constancy of the codeword with the coin tossing, the more indices the SP reports different biases, the hight the probability she got caught. Through a careful analysis, we manage to show that the probability of accusing an innocent user and the probability of reporting different biases without being detected can never be non-negligible simultaneously, which further implies that either the malicious SP deviates without hurting the innocent user security, or the deviation will be almost always detected by the judge.

Theorem 34. *A malicious SP can frame an innocent user without being detected by the judge with negligible probability if the underlying encryption Enc_r is IND-CPA secure, the commitment scheme is computationally binding, the digital signature scheme we use as black-box is existentially unforgeable, and the hash function is collision resistant.*

Proof. First, the only way a malicious SP can deviate the FDT protocol without being detected is via reporting a bias vector to the judge different with the one used in the fing protocol. This includes the case that the malicious SP always use the same $h_i^{b_i}, m_i^{b_i}$ for both $b_i = 0, 1$ which is equivalent to use $p_i = 1$ or 0, and reports to an arbitrary p'_i without being detected by the judge. We call this event F_1 and denote $\Pr[F_1] = \eta_1$.

With the checks in each round, the deviations could be (1). The malicious SP does not chooses a uniform r for the commitments. However in this case, as the user chooses random s, the coin tossing still returns a random value. Changing r does not influence the simulation of the sender view. (2). She submits different ciphertext for the 2nd and 4th round trying to frame the user, however, this implies a forgery of the signature scheme. (3). The SP uses different hash values and segments values to transmit. She

could only use the other values in the input, otherwise, it will be detected by the user and the judge. (4). The SP uses different segments to transmit, however, this can be easily identified by the hash function property. The first half of case (3) is belong to the deviation above which we analyze below.

Now, we examine the relation of η_1 and the error probability ϵ of accusing an innocent user (ϵ is given by the Tardos' code). We show that the malicious SP can not make both η_1 and ϵ non-negligible. Fix a pirate codeword $\bar{y}$ and suppose the codeword $\bar{x}$ of an innocent user is generated with biases $p_1, \ldots, p_{2n}$ (but we focus on the $L' = [2n] \setminus L$ coordinates here). Notice that the SP may only benefit by reporting a bias $p' < p$ at certain locations $j \in L'$ such that $y_j = 1$. Suppose the SP deviates from the true vector by reporting, for each $j \in L'$, $p'_j = p_j - \delta_j$. We assume that the judge knows the parameters n, k, t of the code, and thus p' is forced to satisfy $p_j - \delta_j \geq 1/(300t)$, otherwise the judge will accuse the sender. Observe here that each δ_j is non-negative, as reporting a $p'_j > p_j$, would decrease the score of the user. The probability the judge does not realize the deviation is $\eta_1 = \prod_{j \in L'}(1 - \delta_j)$. This holds because the judge also checks the coin tossing transcripts, the malicious SP can cheat only when she breaks the binding property of the commitment scheme. When the judge has the coin tossing transcript of the i-th commitment, he is able to detect the misbehavior of the SP with probability $1 - (p_j - p'_j)$.

Further, we will show that an innocent user accused by the SP will be convicted by the judge with probability at most $\epsilon^{1/4}/\eta_1^{2/\sqrt{t}}$, where t is the size of the coalition the code is designed to work against. For each $j \in L'$, let

$$U'_j = \begin{cases} \sqrt{\frac{1-p_j+\delta_j}{p_j-\delta_j}}, & \text{if } x_j = 1; \\ -\sqrt{\frac{p_j-\delta_j}{1-p_j+\delta_j}}, & \text{if } x_j = 0. \end{cases}$$

Let also S' denote the score of the innocent user under the reported biases, that is $S' = \sum_{j \in L'} y_j U_{ij}$. Our first task is to upper bound the expectation of the exponential $e^{\alpha U'_j}$, for an arbitrary $j \in L'$. We drop the subscript j in p, δ, U' for convenience. As in the proof of Theorem 2.1 in [127], we'll employ the inequality $e^x \leq 1 + x + x^2$, valid for $x \leq 1$. Observe that $U' \leq \sqrt{300t}$ and choosing $\alpha \leq 1/\sqrt{300t}$ will suffice. Thus,

$$\mathrm{E}[e^{\alpha U'}] \leq 1 + \alpha \mathrm{E}[U'] + \alpha^2 \mathrm{E}[U'^2].$$

We now upper-bound $\mathrm{E}[U']$ and $\mathrm{E}[U'^2]$ in terms of δ.

$$\begin{aligned}
\mathrm{E}[U'] &= p\sqrt{\tfrac{1-p+\delta}{p-\delta}} - (1-p)\sqrt{\tfrac{p-\delta}{1-p+\delta}} \\
&= (p-\delta)\sqrt{\tfrac{1-p+\delta}{p-\delta}} - (1-p+\delta)\sqrt{\tfrac{p-\delta}{1-p+\delta}} + \delta\sqrt{\tfrac{1-p+\delta}{p-\delta}} + \delta\sqrt{\tfrac{p-\delta}{1-p+\delta}} \\
&= \delta\big(\sqrt{\tfrac{1-p+\delta}{p-\delta}} + \sqrt{\tfrac{p-\delta}{1-p+\delta}}\big) \leq \sqrt{600t} \cdot \delta
\end{aligned}$$

and similarly

$$\begin{aligned}
\mathrm{E}[U'^2] &= p\tfrac{1-p+\delta}{p-\delta} + (1-p)\tfrac{p-\delta}{1-p+\delta} \\
&= (p-\delta)\tfrac{1-p+\delta}{p-\delta} + (1-p+\delta)\tfrac{p-\delta}{1-p+\delta} + \delta\tfrac{1-p+\delta}{p-\delta} - \delta\tfrac{p-\delta}{1-p+\delta} \\
&= 1 + \delta\big(\tfrac{1-p+\delta}{p-\delta} - \tfrac{p-\delta}{1-p+\delta}\big) \leq 1 + 300t \cdot \delta.
\end{aligned}$$

We now proceed by providing an upper bound for the expectation of the exponential $e^{\alpha U'}$, using $\alpha = 1/(20t)$, and the inequality $1 + x \leq e^x$, valid for all real x.

$$\begin{aligned}
\mathrm{E}[e^{\alpha U'}] &\leq 1 + \alpha \mathrm{E}[U'] + \alpha^2 \mathrm{E}[U'^2] \leq 1 + \alpha^2 + \big(\alpha\sqrt{600t} + \alpha^2 300t\big) \cdot \delta \\
&\leq 1 + \alpha^2 + 2\delta/\sqrt{t} \leq e^{\alpha^2 + 2\delta/\sqrt{t}} = e^{\alpha^2}/e^{-2\delta/\sqrt{t}} \leq e^{\alpha^2}/(1-\delta)^{2/\sqrt{t}}.
\end{aligned}$$

Next we upper bound the expectation of $e^{\alpha S'}$;

$$\mathrm{E}[e^{\alpha S'}] = \mathrm{E}\Big[\prod e^{\alpha y_j U'_j}\Big] \leq \prod_{j:y_j=1} \mathrm{E}[e^{\alpha U'_j}] \leq \prod_{j:y_j=1} \frac{e^{\alpha^2}}{(1-\delta_j)^{2/\sqrt{t}}}$$

$$\leq \prod_{j:y_j=1} e^{\alpha^2} \prod_{j:y_j=1} \frac{1}{(1-\delta_j)^{2/\sqrt{t}}} \leq e^{\alpha^2 n} \eta_1^{2/\sqrt{t}},$$

where for the last inequality we assumed, without loss of generality, that $\delta_j = 0$ when $y_j = 0$. Finally, by the Markov inequality,

$$\Pr[S' > Z/2] = \Pr[e^{\alpha S'} > e^{\alpha Z/2}] \leq \frac{\mathrm{E}[e^{\alpha S'}]}{e^{\alpha Z/2}} \leq \frac{e^{\alpha^2 n}/\eta_1^{2/\sqrt{t}}}{e^{\alpha Z/2}} = e^{-k/4}/\eta_1^{2/\sqrt{t}} \leq \epsilon^{1/4}/\eta_1^{2/\sqrt{t}}.$$

To summarize, the probability an innocent user accused by a malicious SP is convicted by judge, is non-negligible unless η_1 is negligible; In this case, the SP is semi-honest. □

Security for the SP under group accusations. The analysis for the effectiveness of the accusation procedure will also proceed in two steps. We first deal with semi-honest users who will follow the fing protocol and the accusation procedure, but they will try to make a pirate copy and avoid being accused. From the half fingerprint known to the SP, the SP can always identify colluders. As the FDT satisfies the simulation based security, the behavior of the adversary is essentially the same as the one interacting with only an oracle returning the codewords of corrupted users, while no information about which half of the codewords are known to the SP is leaked. Further we can show that by relaxing the threshold on the judge side, whoever accused by the SP using the half fingerprint will also be accused by the judge using another half fingerprint.

Lemma 35. *Suppose $\mathcal{U}_{cor}$, with $|\mathcal{U}_{cor}| \leq t$, is a coalition of users. If all users are semi-honest during the* fing *protocol execution. The probability that no user is accused or an accused user is acquitted by the judge is $\epsilon^{1/16} + \epsilon^{t/4} + \epsilon_0$, where ϵ is the parameter from the Tardos code, ϵ_0 is negligible (to the security parameter), if the underlying commitment scheme $\mathsf{Com}(\cdot)$ is computationally hiding, and the encryption Enc_s is IND-CPA secure.*

Proof. In the semi-honest model, as we show in the previous section, what the users can do is essentially the same as in the ideal model where they just receive the fingerprinted copies according to the biases from an oracle. In this case, the codeword of the user is a mix of two codewords, the one revealed to the SP while the other is hidden. In particular, for each pair of locations $(2j-1, 2j)$, for $j \in [n]$, the SP knows the codeword of each user at exactly on of these locations. Following the analysis of [127], in the ideal model, the accusation algorithm run by the SP (on the locations known to him) will return at least one of the users participating in the coalition that produced the pirate copy. Next, we will show that the user accused by the SP will also be accused by the with an overwhelming probability. In particular, if ϵ is the error parameter of the instantiation of Tardos' code, we show that an accused pirate by the sender is acquitted by the judge with probability at most $\epsilon^{1/16}$.

Let Y denote a pirate codeword generated by an arbitrary strategy. Define a sequence of random variables $D = (D_1, \ldots, D_m)$, where, for $j \in [n]$, D_j is a binary random variable taking the value 1 when $2j \in S$ (recall that $S \subseteq [2n]$ is the set of location known to the SP) and $2j - 1 \notin S$ and the value -1 when it is the other way around. If X denotes the codeword of a user u, then, for a location $j \in [2n]$ with bias p_j, the score at location j of user u is $Y_j U_j$, where

$$U_j = \begin{cases} \sqrt{\frac{1-p_j}{p_j}}, & \text{if } X_j = 1; \\ -\sqrt{\frac{p_j}{1-p_j}}, & \text{if } X_j = 0. \end{cases}$$

For each $j \in [n]$, let $\Delta_j = Y_{2j} U_{2j} - Y_{2j-1} U_{2j-1}$, and define $\Delta = \sum_{j \in [m]} \Delta_j D_j$, so that Δ is the difference between the score at the sender-side minus the score at the user-side. Since the judge will accuse a user only if his score is at least $Z/2$ (where Z is the threshold used by the Tardos code, it suffices to show that $\Pr[\Delta > Z/2] \leq \epsilon^{1/16}$.

From the inequality $1+x \leq e^x \leq 1+x+x^2$, valid for $x \leq 1$. For $\alpha = \frac{1}{80c}$, $\alpha\Delta_j D_j \leq 1$, we have

$$\begin{aligned}\mathrm{E}[e^{\alpha\Delta}] = \mathrm{E}\Big[\prod e^{\alpha\Delta_j D_j}\Big] = \prod \mathrm{E}[e^{\alpha\Delta_j D_j}] \leq \prod \mathrm{E}[1 + \alpha\Delta_j D_j + \alpha^2\Delta_j^2 D_j^2] \\ = \prod (1 + \alpha\mathrm{E}[\Delta_j]\mathrm{E}[D_j] + \alpha^2\mathrm{E}[\Delta_j^2]) = \prod (1 + \alpha^2\mathrm{E}[\Delta_j^2]) \leq e^{\alpha^2 \sum \mathrm{E}[\Delta_j^2]},\end{aligned}$$

where we used the independence of Δ_j and D_j, and that $\mathrm{E}[D_j] = 0$, for all $j \in [n]$. Furthermore,

$$\mathrm{E}[\Delta_j^2] \leq \mathrm{E}[U_{2j-1}^2] + \mathrm{E}[U_{2j}^2] + 2\mathrm{E}[|U_{2j-1}||U_{2j}|].$$

For any $j \in [2n]$ we compute $\mathrm{E}[|U_j|] = 2\sqrt{p_j(1-p_j)} \leq 1$. Since U_{2j-1} and U_{2j} are independent for all $j \in [n]$, $\mathrm{E}[|U_{2j-1}||U_{2j}|] = \mathrm{E}[|U_{2j-1}|]\mathrm{E}[|U_{2j}|] \leq 1$. Putting everything together we obtain $\mathrm{E}[e^{\alpha\Delta}] \leq e^{4n\alpha^2}$. Recall that $n = 100t^2k$ and $Z = 20tk$. By the Markov inequality,

$$\Pr[\Delta > Z/2] = \Pr[e^{\alpha\Delta} > e^{\alpha Z/2}] \leq \frac{\mathrm{E}[e^{\alpha\Delta}]}{e^{\alpha Z/2}} \leq e^{4n\alpha^2 - \alpha Z/2} = e^{-k/16} \leq \epsilon^{1/16}.$$

□

The case that malicious users can arbitrarily deviate from the protocol are easier to analyze than Theorem 34 due to the simple checks. It is easy to see that in each round, the user is forced to be behave honestly, otherwise the deviation will be detected with overwhelming probability.

Theorem 36. *Suppose $\mathcal{U}_{cor}$, with $|\mathcal{U}_{cor}| \leq t$, is a coalition of users. Assuming Enc_s is IND-CPA secure, the commitment scheme $\mathsf{Com}(\cdot)$ is computationally hiding and the signature scheme used is existentially unforgeable, and the hash function is collision resistent. Then the probability that no user is accused or an accused user is acquitted by the judge is $\epsilon^{1/16} + \epsilon^{t/4} + \epsilon_0$, where ϵ is the error probability from the Tardos code, ϵ_0 is negligible (to the security parameter).*

Proof. We will bound the probability ρ that a malicious user deviates the protocol but passes the checks, and we denote this probability as ρ_i when it happens in the i-th step.

In the 1st step, 3rd step, and 5th step, the malicious user needs to forge a signature of the SP if he wants to replace the commitments, hash values and encrypted bits, or the messages to frame the SP, this can happen with only a negligible probability otherwise we can directly break the underlying signature scheme used by the SP with this forgery.

In the 2nd step, the malicious user can not open to different numbers as his coin tosses, due to the soundness of the proofs.

In the 4-th step, the malicious user deviates the protocol without being noticed by the SP, if she transmits $h_c^{1-b_c}$. However, she succeeds with only a negligible probability as $m_c^{1-b_c}$ has sufficient min-entropy, $h(m_c^{1-b_c})$ should be also unpredictable (as elaborated at the end of Section 5.3.2). It follows that $\rho \leq \sum_i \rho_i$, which is negligible. □

5.5 Security Implications of Protocol Restarts

In the following, we consider the original Tardos code with length $m = 100t^2k$ and threshold $Z = 20tk$, where c is the number of colluders and $k = \log \frac{1}{\epsilon}$ the security parameter. For simplicity, we take t equal to the number of users n.

If the colluders are allowed restarts, they can act as follows. They do $(\mu - 1)$ restarts each to receive a total of μmn bits. For the pirate codeword, they output a zero whenever they can. Formally, for any $j \in [m]$, let x be the number of ones the pirates have received collectively at location j. They set y_j, the bit of the pirate copy at j, as follows.

$$y_j = \begin{cases} 1 & \text{if } x = \mu n; \\ 0 & \textit{otherwise}. \end{cases}$$

We are going to show that with this simple strategy, each pirate escapes with high probability. To that end, let p denote the bias-vector, X denote the codeword of an arbitrary pirate, Y the pirate copy generated by the aforementioned strategy, and

$$U_j = \begin{cases} \sqrt{\frac{1-p_j}{p_j}}, & if X_j = 1; \\ -\sqrt{\frac{p_j}{1-p_j}}, & if X_j = 0. \end{cases}$$

The score of the pirate can be expressed as $S = \sum_{j\in[m]} Y_j U_j$. Our task is to upper-bound $\Pr[S > Z]$. We'll use $e^x \leq 1 + x + x^2$, valid for $x \leq 1$. Since $|U_j| < \sqrt{300n}$, choosing $\alpha < \frac{1}{10n}$ we have

$$\mathrm{E}[e^{\alpha S}] = \mathrm{E}[\prod e^{\alpha Y_j U_j}] = \prod \mathrm{E}[e^{\alpha Y_j U_j}] \leq \prod \mathrm{E}[1 + \alpha Y_j U_j + \alpha^2 Y_j^2 U_j^2]$$

$$\leq \prod (1 + \alpha \mathrm{E}[Y_j U_j] + \alpha^2 \mathrm{E}[U_j^2]) = \prod (1 + \alpha \mathrm{E}[Y_j U_j] + \alpha^2).$$

For any $j \in [m]$ we have

$$\mathrm{E}[Y_j U_j] = \mathrm{E}_{p_j}\left[p_j^{\mu n}\sqrt{\tfrac{1-p_j}{p_j}}\right] = \frac{1}{\pi/2 - 2t'} \int_{t'}^{\pi/2-t'} \sin^{2\mu n-1} r \cos r \; dr$$

$$= \frac{1}{\pi/2 - 2t'} \cdot \frac{1}{2\mu n} \cdot \sin^{2\mu n} r \Big|_{t'}^{\pi/2-t'} = \frac{(1-t)^{\mu n} - t^{\mu n}}{(\pi - 4t')\mu n} \leq \frac{1}{3\mu n}.$$

Putting things together, and using $1+x \leq e^x$, we obtain $\mathrm{E}[e^{\alpha S}] \leq \prod(1+\alpha/(3\mu n)+\alpha^2) \leq e^{\alpha^2 m + \alpha m/(3\mu n)}$. An application of Markov's inequality now gives that:

$$\Pr[S > Z] < \frac{\mathrm{E}[e^{\alpha S}]}{e^{\alpha Z}} \leq e^{\alpha^2 m + \alpha m/(3\mu n) - \alpha Z}.$$

For $m = 100n^2 k$, $Z = 20nk$, $k = \log(1/\epsilon)$, $\alpha = \frac{1}{10n}(1 - \frac{5}{3\mu})$, $\mu > 1$,

$$\Pr[S > Z] < \epsilon^{(1-\frac{5}{3\mu})^2}.$$

Thus, even allowing a single restart per user, is sufficient for the pirates to escape with high probability. Another way to view this, is that an instantiation of Tardos code

that can handle a coalition of size t, is not secure against a coalition of size $2t$. The simple way around this, is to instantiate the code so as to handle coalition size μt, and allow each user at most $\mu - 1$ restarts.

5.6 Conclusion and Open Problems

In this chapter, we constructed the first communication optimal asymmetric fingerprinting scheme, (i.e., the total number of bits transmitted in the protocol is almost the same as the length of the files), based on Tardos code. This is an appealing feature, especially for fingerprinting schemes in which large data (like movies) are transmitted. Besides rate optimality, we also considered two properties: security against accusation withdrawal and security under adversarial aborts, which are overlooked in previous asymmetric fingerprinting schemes.

Since we are the first to study communication rate in this setting, it would be interesting to examine other applications that requires a private preserving transmission of a large size data. In general, it would be an interesting theoretic problem to consider rate optimal secure protocols.

Chapter 6

Making any Identity Based Encryption Accountable, Efficiently

6.1 Introduction

Identity-Based Encryption (IBE) was introduced by Shamir [123], to remove the need for maintaining a certificate based public-key infrastructure (PKI). Long time after the concept was proposed, Boneh and Franklin constructed the first practical IBE [19] in the random oracle model [12]. Since then, IBE has gotten more attention and a lot of alternative schemes have emerged with an extended set of properties, cf. [16, 17, 23, 57, 69, 119, 128, 129].

Although significant progress has been made in constructing secure and efficient IBE schemes, a critical problem of IBE is that a trusted authority, called PKG, is required to generate secret keys for all users. The possibility of the corruption of this authority (or temporary misbehavior due to an insider attack) is considered one of the most important

reasons hindering the deployment of IBE systems in practice [**?**, 1, 68]. The problem is inherent since there is no user-side secret that is used when generating the secret key corresponding to an arbitrarily formed identity; it follows that there is no *built-in* incentive for the PKG in a standard IBE system to protect the users' secret information.

Beyond the obvious privacy problem (the unavoidable fact that the PKG can decrypt all users' ciphertexts) there is also a more serious attack that can take place: the PKG may share the users' secret keys. One may address this by arguing that such malicious behavior can be detectable by the user: for instance, a decryption program B leaked to the public can be noticed by the user. In such case, the user could conceivably bring the program to court and sue the PKG, thus the PKG would be deterred from such behavior. However, notice that both user and PKG are capable of producing B thus the device itself can not be used as conclusive proof about who is at fault.

In order to make the above detect-then-punish mechanism effective, Goyal introduced the concept of accountable authority IBE, (A-IBE in short) [67], where a convincing proof can be provided from which a judge can make a decision about who is at fault. In order to achieve this characteristic, every identity must be corresponded with super-polynomially many secret keys, and the PKG and the user jointly generate a secret key for the user so that the PKG does not know which key is chosen by the user. Using the secret key received by the user, any third party, a judge for example, can tell whether the decryption device is made from the user secret key or not, thus the judge (and the public) can identify unequivocally the creator of the device. A number of works followed up this seminal result, [68, 91, 93, 118, 132], further refining the notion of A-IBE.

Still, the adoption of A-IBE in practice is hindered by a couple of facts. First, many constructions are inefficient (in the sense that they require linear in the security parameter number of group elements, cf. Figure 12) or that the designs are incompatible

with existing practical deployments such as RFC 5091 [26]. Second, when a user accidentally loses his key, in all existing A-IBE schemes, the user and the PKG have to discard this identity and generate a new key for the user using a different identity (otherwise, it enables malicious users to frame the PKG). This artifact brings users annoying inconvenience. These put forth the main motivations in our work: is it possible to add accountability to *any* existing (that is potentially already deployed, e.g., RFC 5091) IBE system, with a minimum cost? furthermore, we ask whether such generic transformation can be extended to allow identity reuse, without losing efficiency? If such transformation exists, users may choose to "upgrade" their IBE scheme to be accountable without requiring a modification to the basic algorithms of the underlying IBE.

Our contributions. In this work, we address both problems listed above. First, we propose a generic construction of an A-IBE (in the so-called weak black-box model with full security against malicious users, see definition in Appendix 6.2) that uses any existing IBE in a black-box way. And this generic construction has ciphertext size only 2 times the underlying IBE ciphertext size. (we call this construction S-I). The key observation behind our construction is that users can choose from a set of secret-keys that are based on an extended form of their identity. When encrypting messages it is possible for the sender to use only two ciphertexts to guarantee an honest user to decrypt. However, it is also possible to generate a set of tracing ciphertexts that can reveal part of the "fingerprint" of the secret-key that was assigned obliviously to the user by the PKG. The presence of the partial fingerprint in a user decoder that is found publicly incriminates the user, otherwise, incriminates the PKG.

We then consider how to allow identity reuse. This property is not known whether achievable before, even with specifically tailored constructions. We achieve it while maintaining the generic nature and the small size ciphertext. The main challenge for

achieving identity reuse in A-IBE setting is that a malicious user can obtain multiple secret keys corresponding to the same identity by claiming to the PKG that she lost the key. Such malicious user could then implement a pirate box B using one key, and reveal another key to the judge. A secret key tracing algorithm may erroneously accuse the PKG, as, by definition, the key used to implement B is different to the key that the user is currently using.

Our strategy is to add public traceability to our generic construction that will enable the judge to differentiate among all the secret keys that were ever obtained by a user for the same identity. Note that in S-I, part of the user fingerprint is recovered, if there is a public reference for the user fingerprint, it might be possible for the judge to check whether the recovered string matches. In order to implement this idea, we improve the generic construction S-I to allow the tracing algorithm to recover the *whole* "fingerprint" while maintaining the ciphertext size still to be small (at most logarithmic overhead, and we call it S-II). With this new feature of S-II we developed, it is possible to deposit the fingerprint (using a one way function) that the user chooses for selecting the secret key to the PKG in a secure way so that: (i) it enables the judge to use a public tracing key T to determine whether a recovered fingerprint matches the fingerprint, and (ii) it prevents a malicious PKG from producing a pirate box without being traced with the help of T. The main technical part is to design a proper one way function for the secure deposit of the bitstring, together with an efficient zero-knowledge proof for the consistency between the privately deposited fingerprint and that used in the OT protocol, bit by bit.

The intuition for S-II follows from the observation that if the "fingerprint" is generated from an error correcting code, a linear fraction of it could be enough to reveal the whole string. With a careful probabilistic analysis, we see that with slightly longer ciphertexts, one is able to retrieve a larger fraction of the fingerprint from a pirate box. (this new

mechanism also allows the length of the fingerprint to be reduced asymptotically, so as the secret key size). This feature of S-II makes it a steppingstone for further allowing identity re-use and public traceability. Our A-IBE tracing mechanisms are inspired by previous works related to traitor tracing [21] and leakage-deterring cryptosystems [85].

With such public traceability, the scheme can be further extended to support identity reuse. Each identity now will have multiple extended forms (instead of one in S-II), and for each extended form indexed by a state, the user can use an independent string as a fingerprint to request one secret key. During the i-th key generation protocol for an identity, the PKG will store a public tracing key T_i and the updated state about the current version of the extended form for each identity in a public directory. The encryption algorithm will use the current version of the extended form of identity. The tracing algorithm will run on all versions of the extended form of the identity, extract (potentially multiple) fingerprints; subsequently, it will check whether they match the public tracing keys. In this way, the tracing algorithm can decide that the key inside the pirate box is the one the user is currently using or whether it is one of the keys claimed to be lost, or is a key originating from the PKG. A malicious user can never frame the PKG using a key claimed to be lost, and a PKG can not evade the tracing algorithm if she ever leaks a decryption box for the user identity (even for previous versions of extended form of identity).

Note that after adding public traceability and id-reuse to our generic construction, the ciphertext efficiency and the generic nature are still the same as in S-II. The model that T has to be stored for each user is the same as the only existing paper [91](that was based on Gentry IBE [57]) providing public traceability. [1] Finally it is worth to

[1] In fact, it is not hard to see (explained in section 6.4.1) that the size of the public tracing key has to grow linearly with the number of users.

	G-I	G-II	[68]	[93]	[118]	[91]	[132]	S-I	S-III
Generic	no	no	no	no	no	no	no	yes	yes
Ciphertext size	$O(\lambda)$	$O(\lambda)$	$O(\lambda)$	O(1)	$O(\lambda)$	O(1)	O(1)	O(1)	O(1)
Malicious User	s	s	s	a	a	a	a	a	a
Malicious PKG	w	bb_0	bb_1	bb_0	bb_1	bb_0	bb_1	bb_0	bb_0
Public Traceable	no	no	no	no	no	yes	no	no	yes
ID Reuse	no	no	no	no	no	no	no	no	yes

Figure 12: Comparisons of all existing A-IBEs, ciphertext size means the number of group elements; 's' means selective, 'a' means adaptive; w, bb_0, bb_1 mean white box, weak black-box and full black-box traceability respectively; S-I, S-III are our constructions.

point out that our construction allows these two properties to be optional services by the PKG and the user may opt-in or opt-out to such properties at will when she requests a key from the PKG. We remark that our generic transformations can go beyond IBE and can be easily adapted to apply to more advanced systems like attribute based encryption [69, 119]. The performance comparison of all A-IBE schemes (including ours) is summarized in Figure 12.

Related work. In [67], Goyal proposed the notion of A-IBE and gave two constructions. The first one is traceable only in the white-box model (requires the key material of the pirate box) while the second one is in the weak black-box model. We call those constructions G-I, G-II and both have ciphertext size that includes a linear number of group elements. In the following work of [68], Goyal et al. proposed a construction having traceability in the full black-box model, but at the price of having (i) secret key

and ciphertext size that has linear in the security parameter number of group elements, (ii) security against malicious users only in a selective model (where the adversary needs to commit to its move ahead at the beginning of the game). Libert and Vergnaud [93] made an improvement on G-I, and they gave an A-IBE with constant group elements in the ciphertext that is proven traceable in the weak black box model. Sahai and Seyalioglu [118] improved the security against dishonest users, and achieved full security against dishonest users, but their construction still has a linear size ciphertext. Lai et al. [91] proposed the first scheme with public traceability that the authority is required to store a public tracing key for each user which is later used to generate tracing ciphertext, and it is also traceable in the weak black-box model. Our public traceability can be based on any IBE and uses a different tracing technique that we can directly compare whether a recovered fingerprint matches the one contained in the public tracing key. Concurrent to our work an E-print technical report [132] proposed an A-IBE with traceability in the full black-box model, adaptive security against malicious user and constant size ciphertext under non-standard assumptions. All these works rely on a highly specific structure, specifically, Gentry IBE [57] as in [67, 91, 93, 132] or fuzzy IBE [119] as in [67, 68, 118]; their techniques for accountability do not adapt in other settings straightforwardly (and specifically none can be applied to current real world IBE's such as those of RFC5091 directly). Also, none of those works allows public traceability (except [91]) or identity reuse.

There are also other proposals to deal with the key escrow problem in IBE. In [19], Boneh and Franklin suggested that multiple authorities distributively produce the master secret key. However, in principle, those PKGs still may collude to leak user's secret leaving the user defenseless; Al-Riyami and Paterson proposed the concept of certificateless public key cryptography [1], and attempted to combine both the advantages

of certificate-based PKI and IBE. The authority only has a partial secret key k_1, and it jointly generates secret key together with the user who has her own secret k_2. However, part of the public key must be in a special form corresponding to k_2 and thus it can not be as expressive as IBE. Hence such systems may be of more narrow applicability compared to proper IBE schemes. Au et al [5] proposed the notion of retrievability that from two secret keys of a user, one can compute the master secret key. The notion of retrievability is interesting but it is achieved only in the white box model. Chow [38] considers the notion of anonymous ciphertext indistinguishability, which requires that the PKG cannot learn the recipient identity from a ciphertext, thus hoping that the authority can not figure out which secret key to use to decrypt. This is an interesting notion as well, but only meaningful in an IBE system with an extremely large number of users; furthermore it does not protect against a PKG that targets a specific user and publishes the decryption algorithm (which is the main defense objective of A-IBE).

6.2 Preliminaries

1-out-of-2 oblivious transfer protocol Briefly speaking, a 1-out-of-2 OT protocol [106] is between a sender S and a receiver R. S has two messages (m_0, m_1) as input, and R chooses one of them according to a bit b. S should not know b, while R should not have any knowledge of m_{1-b}.

We only provide a half simulation type of definition (cf. [106]). For `sender security`, we make a comparison to the ideal implementation in which there is a trusted party receiving m_0, m_1 from S and b from R, and sends R the message m_b. We require that $\forall m_0, m_1$ and any efficient adversary $\mathcal{A}$ as the receiver, there is a simulator plays as the receiver in the ideal world that, the output distribution of the simulator and $\mathcal{A}$ are computationally indistinguishable. For `receiver security`, suppose t_0, t_1 represent the

trascript sent by the receiver w.r.t input 0 and 1 respectively, we require that the sender can not distinguish the distribution of t_0 and t_1.

Accountable authority identity-based encryption. Here we provide a general definition for an A-IBE scheme, it is composed of the following algorithms:

- **Setup**(λ, δ) This algorithm takes the security parameter λ and the correctness parameter δ as input and outputs master key pair (mpk, msk) and the system parameters $t(\delta), \ell(\delta)$.
- **KeyGen** This is a stateful protocol between a user and the PKG in which the user has an identity ID and mpk, and the PKG has mpk, msk, ID as inputs respectively. It ends with the user outputting her secret key sk_{ID} or $\perp$ if the secret-keys are malformed, and the PKG output a tracing key T_{ID} and a current state st_{ID}.
- **Enc**(ID, mpk, m, st_{ID}) This algorithm inputs a receiver identity ID, master public key mpk, the message m and potentially a public state st_{ID}, and outputs the ciphertext C.
- **Dec**(C, sk_{ID}) This algorithm takes ciphertext C and user secret key sk_{ID} as input, and outputs the plaintext m.
- **Trace**$^B(ID, \delta, T_{ID})$**.** This algorithm inputs a pirate decryption box B for ID, correctness parameter δ and a tracing key T_{ID} as input, it outputs "user", "PKG" or "$\perp''$.

Note that the algorithms can be stateless as usual if identity reuse is not required. When T_{ID} is public, then the A-IBE scheme has public traceability.

δ-correctness of a decryption device B, for regular A-IBE schemes, it is defined as $\Pr[B(C) = m : C = \textbf{Enc}(ID, mpk, m)] \geq \delta$; while for A-IBE schemes allowing identity

re-use, the box might contain a couple of keys for one identity corresponding to different states, we require that for a randomly selected state, it works with δ correctness, thus the δ-correctness in this case is defined as $\Pr[B(C) = m : C = \mathbf{Enc}(ID, mpk, m, i) \wedge i \leftarrow \{1, \ldots, st_{ID}\}] \geq \delta$. Note that according to the pigeonhole principle, there exists at least one state j, $\Pr[B(C) = m : C = \mathbf{Enc}(ID, mpk, m, j)] \geq \delta / st_{ID}$, and this is important for the tracing algorithm of S-III.

IND-ID-CPA security. This is similar to the standard semantic security for IBE schemes. Consider the following game between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$:

- **Setup** $\mathcal{C}$ runs **Setup**, and sends $\mathcal{A}$ the system public key: mpk.
- **Phase 1** $\mathcal{A}$ runs the **KeyGen** protocol with the challenger for several distinct adaptively chosen identities $ID_1, .., ID_q$ and gets the decryption keys $sk_{ID_1}, .., sk_{ID_q}$.
- **Challenge** $\mathcal{A}$ submits two equal length messages m_0, m_1 and an identity ID that is not appearing in the queries of Phase 1. $\mathcal{C}$ flips a random coin b and encrypts m_b with ID. The ciphertext C is passed on to $\mathcal{A}$.
- **Phase 2** This is identical to Phase 1 and $\mathcal{A}$ is not allowed to query for ID.
- **Guess** The adversary outputs a guess b' of b.

The advantage of the adversary $\mathcal{A}$ is defined as $|\Pr[b' = b] - 1/2|$; we say an A-IBE is IND-ID-CPA secure if $\mathcal{A}$'s advantage is negligible.

Note that for A-IBE schemes with public traceability, the adversary also gets the public tracing key T.

Besides standard semantic security, for an A-IBE scheme, there are two additional security properties that have to be considered. The first is security against a malicious PKG. Any A-IBE scheme, should prevent the PKG from learning useful information which can help her to leak a decryption program B (we will also call it a decryption

"box") on behalf of a certain identity and evade the tracing algorithm. The second is security against malicious users. In this perspective, a group of colluding users should not be able to make a working box B that frames the PKG. Depending of the form of B, one may consider various models for the tracing algorithm. Specifically, if the tracing algorithm only needs oracle access to B, we call it traceable in the black-box model. Common variants of the black-box model exist depending on whether the PKG is given access to the decryption oracle that corresponds to the secret key the user gets (called the "fully black-box model" if yes, and the "weak black-box model" otherwise).

Weak (black-box) dishonest-PKG game. Consider the following game between a PPT adversary $\mathcal{A}$ and a PPT challenger $\mathcal{C}$:

- **Setup**: The adversary acts as a malicious PKG, generates system public keys and sends $\mathcal{C}$ mpk. Also $\mathcal{A}$ specifies an identity ID.
- **KeyGen**: $\mathcal{C}$ and $\mathcal{A}$ then engage in the KeyGen protocols of A-IBE acting as a user and PKG respectively. In each run, they jointly generate a decryption key and a tracing key T_{ID} and state st_{ID} for the identity ID. If neither party aborts, then $\mathcal{C}$ gets a decryption key sk_{ID} for user ID as output.
- **Create Decryption Box**: The adversary outputs a decryption box B.

The adversary $\mathcal{A}$ wins the game if the following conditions hold true:

$$B \text{ has } \delta\text{-correctness} \wedge \mathbf{Trace}^{B}(ID, sk_{ID}) \neq \text{"PKG"}.$$

In a *full* dishonest-PKG game, $\mathcal{A}$ is also allowed to ask decryption queries. In other weaker (non-black-box) models, the tracing algorithm might have non-black-box access to the pirate box B.

Adaptive dishonest-user game. In this game, a set of malicious users collude to create a decoder box, trying to frame the PKG.

- **Setup** $\mathcal{C}$ runs the A-IBE **Setup** algorithm, and sends $\mathcal{A}$ mpk;
- **Secret Key Queries** The adversary runs the KeyGen protocols with $\mathcal{C}$, playing the role of different users and PKG respectively, for adaptively chosen identities $ID_1, .., ID_q$ for different times. $\mathcal{A}$ gets the corresponding secret keys $\{sk_{ID_1}\}, .., \{sk_{ID_q}\}$ and $\mathcal{C}$ outputs the corresponding tracing keys $T_{ID_1}, \ldots, T_{ID_q}$ and the states $st_{ID_1}, \ldots, st_{ID_q}$.
- **Create Decryption Box** The adversary outputs an identity ID together with a decryption box B for ID.

The adversary wins if the followings hold true:

$$B \text{ has } \delta\text{-correctness} \wedge \mathbf{Trace}^B(ID, sk_{ID}) = \text{“PKG”}.$$

Weaker model also exists, i.e., in the selective dishonest-user game, the adversary is required to declare the ID to be attacked at the beginning.

6.3 Generic Construction of A-IBE with Constant Size Ciphertext

In this section, we give a generic construction of A-IBE secure in the weak dishonest-PKG model from any IBE scheme using 1-out-of-2 OT, and it only has ciphertext size two times the underlying IBE scheme.

The intuition behind this generic transformation is that for each identity ID, there are exponentially many secret keys, each of which has a unique “fingerprint”. Each user will select his key with a random “fingerprint” using an OT protocol. Given only an oracle access to a decryption box B implemented using one key, part of the fingerprint can be retrieved. When a decryption box is found, the recovered partial “fingerprint” is able to reveal the source of the box.

Specifically, 2ℓ identities $(ID||1||0, ID||1||1), \ldots, (ID||\ell||0, ID||\ell||1)$ are all considered as the same user with identity ID. [2] During KeyGen, for each pair of secret keys corresponding to identities $ID||i||0$, and $ID||i||1$, user randomly selects one of them using a 1-out-of-2 OT. [3] The "fingerprint" of the user selected key corresponds to the bit string of length ℓ he uses in the OT protocols. Enc randomly selects an index r, and simply encrypts the same message under both $ID||r||0, ID||r||1$, thus sender does not need to know the fingerprint of the user with ID. Note the user has one key corresponding to the identity $ID||r||0$ or $ID||r||1$ for each r, thus he can decrypt. Also Trace can attempt to recover each bit of the fingerprint from a decryption box by feeding ciphertexts containing different messages for the location, i.e., for an index i, c_0, c_1 are fed, where $c_b = \mathsf{Enc}(ID||i||b, mpk, m_b)$. The semantic security of the underlying IBE suggests that the box will not distinguish these tracing ciphertexts from regular ciphertexts, and the answer m_b reveals the $i-$th bit of the user fingerprint. Whenever λ bits are recovered, and all of them equal to the corresponding bits in the user "fingerprint", the user will be accused, otherwise the PKG will be accused. Essentially, a malicious PKG can evade the tracing algorithm only if she guesses correctly λ random bits.

One may notice that a malicious user may put as few keys as possible, e.g., only one key corresponding to $ID||t||b_t$ for some t, into a pirate box B and thus for the other indices, there is no hope to recover the fingerprint bits. However, since B has to provide some minimum functionality, i.e., answering correctly with some noticeable probability δ, (formally, $\Pr[B(\mathsf{Enc}(m, ID, mpk)) = m] \geq \delta$), if we choose ℓ large enough (λ/δ through our probabilistic analysis), there must be at least λ keys contained in the pirate box to

[2] Doing above may reduce the original identity space, however, this problem can be easily addressed by extending the identity string $O(\log \ell)$ bits longer.

[3] Unlike ABE schemes, our generic construction does not have to provide collusion-resistance, as for each index, a user can obtain only one key.

maintain the δ-correctness. In particular, we can argue that there exist at least λ indices, the box decrypts ciphertext generated using those indices, with probability at least δ/λ. Then as elaborated above, once a key is used, we can recover the corresponding bit.

6.3.1 Detailed construction.

We call this generic construction S-I, for an IBE scheme (`Setup, KeyGen, Enc, Dec`), the details of S-I are as follows:

- **Setup**(λ, δ): This algorithm inputs the security parameter λ and the correctness parameter δ, it runs the `Setup` algorithm of the underlying IBE and outputs master key pair (mpk, msk), and a parameter $\ell = \lambda/\delta$.

- KeyGen This is a protocol between PKG and a user A with identity ID,

  1. PKG generates 2ℓ secret keys $\{k_{i,b}\}_{i=1,\ldots,\ell,b=0,1}$, using `KeyGen` of the underlying IBE, where $k_{i,b} = \texttt{KeyGen}(msk, ID||i||b)$.
  2. User A randomly chooses a bit string $\bar{b} = b_1, \ldots, b_\ell$ with length ℓ.
  3. A executes ℓ (1,2)-OT protocols with the PKG in parallel. In the i-th execution, A inputs b_i, PKG inputs $k_{i,0}, k_{i,1}$ and A receives k_{i,b_i}.

  The protocol ends with A outputting $sk_{ID} = \{sk_i = (b_i, k_{i,b_i})\}_{i=1,\ldots,\ell}$.

- Enc(ID, mpk, m): To encrypt a message m for user A, the algorithm randomly chooses an index $r \in \{1, \ldots, \ell\}$ and outputs ciphertext $C = (r, c_{r,0}, c_{r,1})$, where for $b \in \{0, 1\}$, $c_{r,b} = \texttt{Enc}(ID||r||b, mpk, m)$.

- Dec(C, sk_{ID}): On input ciphertext C and the secret keys of user A, the decryption algorithm parses the ciphertext and runs the underlying IBE decryption algorithm, it returns $m = \texttt{Dec}(c_{r,b_r}, sk_r)$.

- $\mathsf{Trace}^B(ID, \delta, \{b_i\})$ This is a two stage protocol. In the first stage , the judge $\mathcal{J}$ interacts with user A [4] to get his secret string and verify its validity.

  1. A sends $\bar{b}$ and a pirate decryption box B to $\mathcal{J}$.
  2. $\mathcal{J}$ parses $\bar{b}$, and then randomly selects 2ℓ messages $\{r_{i,0}, r_{i,1}\}_{i=1,\ldots,\ell}$, and asks A to decrypt one of the ciphertext $\{c_{i,0}, c_{i,1}\}$, where $c_{i,b} = \mathtt{Enc}(ID||i||b, mpk, r_{i,b})$ for $i = 1, \ldots, \ell$. A decrypts $\{c_{i,b_i}\}$ and sends back $\{r'_{i,b_i}\}$, $\mathcal{J}$ then checks $r_{i,b_i} \stackrel{?}{=} r'_{i,b_i}$ for all $i \in \{1, \ldots, \ell\}$.

  If not, $\mathcal{J}$ outputs "user"; otherwise, $\mathcal{J}$ runs the following algorithm:

  1. For each $i \in \{1, \ldots, \ell\}$, $\mathcal{J}$ repeats the following N times (the exact number of N will be specified in the analysis) to define a bit s_i. In each run, $\mathcal{J}$ randomly selects m_0, m_1, and feeds B with $(i, c_{i,0}, c_{i,1})$, where $c_{i,b} = \mathtt{Enc}(ID||i||b, mpk, m_b)$ for $b = 0, 1$. $\mathcal{J}$ records a b for s_i if B returns m_b, otherwise, $\mathcal{J}$ records a $\perp$.
  2. After the repetitions for each i, $\mathcal{J}$ takes the majority of the non-$\perp$ records as the value for s_i; if all records are $\perp$, then s_i is undefined.
  3. Suppose $s_{i_1}, \ldots, s_{i_t}$ are the defined bits. If $s_{i_j} = b_{i_j}$ for all $j \in \{1, \ldots, t\}$ and $t \geq \lambda$, $\mathcal{J}$ returns "user"; otherwise, $\mathcal{J}$ returns "PKG".

Remark. our tracing algorithm is conditioned on the fact that the box has a noticeable correctness δ for random messages, and the box is resettable.

A note about fully black-box traceability. We can see from the tracing algorithm of S-I that given access to a decryption oracle, the PKG learns the bit string that the user chose to select the secret keys, thus further learns the chosen secret keys of the

[4] It can be easily made non-interactive if the user proves that he has the right keys.

user. One possible remedy is to introduce a mechanism that only the judge can create a valid tracing ciphertext, i.e., regular ciphertext pair is augmented with a ZK proofs of the statement that "either they contain equal plaintexts or I am the judge". This prevents the PKG from learning information about the user fingerprint via access to a decryption oracle, but also at the same time enables the judge to trace. One downside of this mechanism is that the judge needs to keep some private state. We omit the details of achieving fully black-box recoverability in this model and focus on the other advanced properties, e.g., identity reuse, which is not known whether achievable before in the standard model of A-IBE in this thesis.

6.3.2 Security analysis.

We will give intuitions about the security properties of S-I and for the proof, we mainly focused on the most involved part dealing with malicious users.

IND-ID-CPA security. $ID||i||0, ID||i||1$ are considered two different identities and thus our generic construction S-I is simply a double encryption of a same message using two different identities. It follows easily that a double encryption is as secure as the underlying IBE.

Security in the weak dishonest-PKG game. Note that the Trace algorithm always outputs "PKG" except when the recovered string is composed of two parts: an all-$\perp$ part, and a bitstring which is at least λ bits long and matches the corresponding substring of the user secret string. All other cases, including an all-$\perp$ string is recovered, or any single bit recovered is different with that of the user "fingerprint", the PKG is accused.

The receiver security of the OT protocol executed in KeyGen guarantees that a malicious PKG can only guess each bit of the secret string, thus she can fools the Trace algorithm with probability negligibly close to $2^{-\lambda}$. Specifically, in the execution of the

i-th OT protocol, the malicious PKG can not distinguish the transcript created by an user inputting a random bit r from the transcript created using the selected bit b_i. We can do a sequence of game changes and end up with a game that all OT transcripts are created using independently selected random bits $\bar{r} = r_1, \ldots, r_\ell$. In the last game, since $\bar{b} = b_1, \ldots, b_\ell$ are independent of the transcripts, we can let the malicious PKG output a box and the judge recovers a substring with length at least λ first, and then select $\bar{b}$. It follows easily that the corresponding substring of $\bar{b}$ matches the recovered substring of $\bar{r}$, with probability at most $2^{-\lambda}$.

Security in the adaptive dishonest-user game. Our main observation that if the box is leaked by a user, the judge will always be able to accuse her, relies on the following reasons. First, since the user has only one key for each location, due to the semantic security of the underlying IBE (and the OT sender security), the user has to report to the judge honestly her secret string. Furthermore, the box B is not able to tell a tracing ciphertext (the pair of the ciphertext encrypting different messages) from a normal ciphertext, thus B will have δ-correctness during tracing. We will analyze that the box has to decrypt using the keys with probability δ/λ for at least λ indices to maintain such correctness. Again, for each index i, B can never succeed in decrypting m_{1-b} if only $k_{i,b}$ is inside, thus for the indices it responds, it has to reveal the correct bits after enough repetitions.

Theorem 37. *(1). S-I is IND-ID-CPA secure if the underlying IBE scheme is IND-ID-CPA secure; (2). S-I is secure in the* `weak dishonest-PKG` *game if the underlying 1-out-of-2-OT protocol satisfies the receiver security; (3). S-I is secure in the adaptive* `dishonest-user` *game if the underlying IBE is IND-ID-CPA secure, and the 1-out-of-2-OT protocol satisfies the (simulatable) sender security.*

Proof. The security properties (1) and (2) follow easily from the explanation above, we will focus on property (3).

First, we see that in the first phase of the Trace protocol, the user has to submit the same string she selected. This can be shown via a sequence of game changes. In the original game, the adversary $\mathcal{A}$ runs the OT protocols one by one for ℓ times (or in parallel), during the KeyGen protocol, and answers the decryption queries during the first phase of the Trace algorithm. In the modified ℓ games, the OT protocols are replaced with an oracle (one by one) that on inputting a bit, outputting the corresponding secret key. The indistinguishability of these game changes are ensured by the (simulatable) sender security of the OT protocol (see the composition lemma of Canetti [29]).

In the last game, during KeyGen, mcA has only oracle access to the OT instances, which can be "controlled" by a simulator. Now suppose the adversary answers correctly for the decryption request $c_{i,1-b_i}$ at some index i with probability Δ_i, there exists a simulator $\mathcal{S}$ playing the role of PKG with $\mathcal{A}$ as a user, can break the IND-ID-CPA security of the underlying IBE. $\mathcal{S}$ can answer all the OT queries perfectly with the corresponding secret keys, (which can be asked to the IND-ID-CPA game challenger directly). $\mathcal{S}$ simply uses $ID||i||1-b_i$ as the challenge identity. $\mathcal{S}$ selects m_0, m_1 as the challenge message, and forwards the challenge ciphertexts to the adversary. If $\mathcal{A}$ answers m_b, $\mathcal{S}$ answers b, otherwise, a random bit. It is straightforward that $\mathcal{S}$ breaks the IND-ID-CPA security with advantage $\frac{\Delta_i}{2}$ (which can be derived as follows: $\Delta_i \cdot 1 + (1-\Delta_i)\frac{1}{2} - \frac{1}{2}$).

Let $\delta_i = \Pr[B \text{ decrypts correctly} \mid i \text{ is selected}]$. We divide the indices $i \in \{1, \ldots, \ell\}$ in two sets, Bad and Good, we define $i \in$ Good if and only if $\delta_i \geq \delta_0$, where $\delta_0 = \delta/\lambda$.

Next, we lower bound $n = |\mathsf{Good}|$. If $n < \lambda$, then:

$$\begin{aligned} & \Pr[B \text{ works correctly}] = \sum_{i=1}^{\ell} \Pr[B \text{ works correctly} | i \text{ is selected}] \Pr[i \text{ is selected}] \\ \leq \quad & [1 \cdot (\lambda - 1) + \delta_0 \cdot (\ell - n + 1)] \frac{1}{\ell} = \frac{\lambda - 1}{\ell} + \frac{\delta(\ell - n + 1)}{\ell\lambda} \leq \frac{\lambda - 1}{\ell} + \frac{\delta}{\lambda} = \frac{\lambda}{\ell} = \delta \end{aligned}$$

thus, we can conclude that for at least λ indices, the box will answer correctly with probability at least δ/λ.

Next, similar to the analysis for the first stage of the protocol, we can show that the probability that B decrypts to the other message selected in the Trace algorithm (m_{1-b_i}, which is with high entropy) will be a negligible function. Following the standard Chernoff bound, we can see that if we run the Trace algorithm with the a number of $N = O(\delta_0^{-2} \log^2 \lambda)$ repetitions, the correct value of b_i would form a majority of the non-$\perp$ records for s_i.

Summarizing the above facts, if a box B implemented using one key from the user and it has δ-correctness, there will be at least λ indices that the Trace algorithm recovers the correct bits, ($\perp$ for all other indices), it returns "user". □

6.4 Generic Construction of A-IBE Allowing Public Traceability and Identity Reuse

In this section, we consider how to add advanced properties of A-IBE generically, without influencing the ciphertext efficiency much. And for a general definition and security models capturing the advanced properties, see section 6.2.

6.4.1 A general framework allowing identity re-use.

As elaborated in the introduction, a user may accidentally lose his secret key, in all previous works, the user has to change a different identity to request a new key. Allowing

identity re-use in such cases is highly desirable. The main difficulty for achieving id reuse lies in the fact that a malicious user can obtain multiple keys (for a same ID) by claiming to the PKG that she lost her key. Then she will implement a pirate box using one key and reveal a different key to the judge for the tracing algorithm, trying to frame the PKG.

Necessity of public traceability and linear size tracing key. To defend against the above attack, a correct tracing algorithm on inputting two keys requested using the same identity should not always output "PKG". It follows that the judge has to be able to identify a "lost" key using some public information, which in turn "implies" public traceability.

Note that in S-I, each user chooses a "fingerprint" $b_1 \dots b_\ell$ when requesting a key. If the Trace algorithm is able to recover the whole "fingerprint" from the pirate box, and there is a public reference, e.g., a value $T = f(b_1 \dots b_\ell)$ for a one way function f, then the judge can publicly check whether the pirate box is from the user or not. In particular, T is generated by the user during the key generation, and he proved in zero-knowledge that the bits of the pre-image of T are consistent with those used in the OT protocols. We will first revise S-I to enable the tracing algorithm to recover the whole fingerprint, and explain in detail in the next section about the one way function and the ZK proofs.

Before we go into technical details of constructions, we first argue that the public tracing key has to grow linear to the number of the identities. To see this, suppose there are N different identities, d_i is the binary random variable that denotes the judge output when seeing a key k_{ID_i} for identity ID_i, and T is the public tracing key. It is obvious that without the tracing key, each d_i is a uniformly random bit (and they are mutually independent), thus $H(d_1, \dots, d_N) = N$; while given T, all

$\{d_i\}$ will be determined, thus $H(d_1, \ldots, d_N|T) = 0$, from the chain rule, we can see $H(T) = H(d_1, \ldots, d_N, T) \geq H(d_1, \ldots, d_N) = N$. Thus the length of T grows linearly to the number of identities used in the system.

Recovering all bits of each user fingerprint. As one may notice, the Trace algorithm of S-I can recover only λ bits, thus for the above public tracing strategy to work, we have to improve the construction of S-I so that one can publicly recover the user "fingerprint" perfectly. A simple observation is that if one can recover a larger fraction of bits, e.g., a linear fraction of ℓ, one may use an error correcting code to generate the fingerprint and recover the whole string by decoding a string having a linear fraction of correct bits. However, the probabilistic analysis of S-I will not hold if we set $n = |\mathsf{Good}|$ to be $O(\ell)$. We further observe that if we use slightly more indices for encryption, (splitting the message, and using the S-I encryption algorithm at each index for the shares), the pirate box has to contain more keys to maintain the δ-correctness. Through a careful analysis, if we use $t = 5 \ln \frac{2}{\delta}$ pairs of identities for encryption, B has to include at least $\frac{4}{5}$ fraction of the keys to maintain δ-correctness. Interestingly, the secret key length of is reduced to $O(\log \frac{1}{\delta})$. We present here the modified generic construction, (named S-II) with only the difference with S-I. We will show how to augment S-II to allow id-reuse and analyze the security in the next sections.

- $\mathsf{Setup}(\lambda, \delta)$: Same as S-I, except $\ell = O(\log \frac{1}{\delta})$, and it also generates an error correcting code $\mathrm{ECC} : \{0,1\}^{\ell_0} \to \{0,1\}^{\ell}$, (e.g., [70].) which corrects at least $\frac{\ell}{5}$-bit errors.

- KeyGen: Same as S-I, except that the bitstring of user A is generated by first selecting a random bitstring $\bar{r}$ with length ℓ_0, then applying the ECC to $\bar{r}$ and produces $\bar{b} = b_1, \ldots, b_\ell$.

- $\mathsf{Enc}(ID, mpk, m, \delta)$: To encrypt a message m for user A, the algorithm first randomly chooses a subset $S = \{s_1, \ldots, s_t\} \subset \{1, \ldots, \ell\}$ with size $t(\delta) = 5 \ln \frac{2}{\delta}$. It then chooses $t-1$ random messages $m_2, \ldots, m_t$ and computes $m_1 = m - \sum_{i=2}^{t} m_i$ and uses the $\mathtt{Enc}$ algorithm of the underlying IBE to encrypt each m_i. The algorithm outputs ciphertext $C = \{(s_i, c_{i,0}, c_{i,1})\}_{i=1,\ldots,t}$, where for $b \in \{0,1\}$, $c_{i,b} = \mathtt{Enc}(ID||s_i||b, mpk, m_i)$.

- $\mathsf{Dec}(C, sk_{ID})$: On input ciphertext C and the secret key of user A, the decryption algorithm parses the ciphertext and then runs the underlying IBE decryption algorithm, and it selects the secrect keys corresponding to s_i and returns $m = \sum_{i=1}^{t} m_i$, where $m_i = \mathtt{Dec}(sk_{s_i}, c_{i,b_i})$.

- $\mathsf{Trace}^B(ID, \delta, \{b_i\})$ The first stage is the same as that of S-I except that the user submits $\bar{r}$ and the judge $\mathcal{J}$ applies the ECC to get $\bar{b}$ himself. If $\mathcal{J}$ does not output "user" in the first stage, it runs the following:

  1. For each $i \in \{1, \ldots, \ell\}$, $\mathcal{J}$ randomly selects a subset $S \subset \{1, \ldots, \ell\}$ of size t until $i \in S$, and let us denote $S = \{s_1, \ldots, s_t\}$ and $i = s_k$; $\mathcal{J}$ randomly samples m, m' and other $t-1$ messages $m_1, \ldots, m_{k-1}, m_{k+1}, \ldots, m_t$ uniformly, and he computes $m_{k,0} = m - \sum_{j \neq k} m_j, m_{k,1} = m' - \sum_{j \neq k} m_j$. For $j = 1, \ldots, t$, $\mathcal{J}$ feeds the box B with $\{(s_j, c_{j,0}, c_j^1)\}$, where for $j \neq k$, $c_{j,b} = \mathtt{Enc}(ID||s_j||b, m_j)$, and $c_{k,b}$ is encryption of $m_{k,b}$, i.e., $c_{k,b} = \mathtt{Enc}(ID||s_k||b, m_{k,b})$ for both $b = 0, 1$. $\mathcal{J}$ records a 0 for b_i if the box returns m, 1 if the box returns m' and $\perp$ otherwise.

  2. After repeating the above N times (the exact number of N will be specified in the analysis), $\mathcal{J}$ takes the majority of the non-$\perp$ symbols in the records as the value for b_i. If b_i is not defined, let $b_i = 0$.

3. $\mathcal{J}$ runs the decoding algorithm of ECC on $\bar{b}$, and gets a bitstring $\bar{r}'$ or $\perp$. If $\bar{r} = \bar{r}'$, $\mathcal{J}$ returns "user", otherwise, it returns "PKG".

Allowing identity re-use. Now with the above briefly explained intuition of public traceability, a user can use different secret string $\{b_1^k, \ldots, b_\ell^k\}$ to choose the $k-$th secret key. The PKG keeps different public tracing key for each string, and the judge can indeed differentiate among the keys of the same identity and the PKG as long as he can extract the "fingerprints" correctly. (For detailed construction, see section 6.4.3). To provide some collision resilience to the generic construction S-II, we extend it further to keep a state st_{ID} for each identity, so that each secret key request for a same identity can actually correspond to different extended identities. In more detail, in S-II, an identity ID is represented using a group of identities $\{ID||i||b_i\}_{i=1,\ldots,\ell,b_i=0,1}$. With a state st_{ID} denoting the number of key requested for ID, the modified extended identities would be $\{ID||st_{ID}||i||b_i^k\}_{i=1,\ldots,\ell;b_i^k=0,1;k=st_{ID}}$.

For the k-th time the user requests a key using $b_1^k, \ldots, b_\ell^k$, the PKG adds a new public tracing key $T_k = f_k(b_1^k, \ldots, b_\ell^k)$ to the public directory, and also updates st_{ID} to be $k+1$.[5] The sender first figures out the state, then he can simply run Enc of S-II using $ID||st_{ID}$ as identity. The Trace algorithm runs the S-II tracing algorithm on all $ID||1, \ldots, ID||st_{ID}$, with a smaller correctness parameter δ/st_{ID}, and extracts fingerprints (potentially more than one). If all of the fingerprints match the corresponding public tracing keys (except the $st_{ID}-$th one), they are considered as lost keys then no

[5] A malicious PKG may put different public tracing keys, however this is trivially detectable by the user and proves to the judge.

one will be accused; If the one that the user is using (the st_{ID}-th key) matches $T_{st_{ID}}$, the user would be accused, otherwise the PKG will be accused. [6]

We can see that we use the underlying IBE as a black-box, thus this improved construction (named S-III) is still a general transformation from IBE to A-IBE.

6.4.2 Building blocks for public traceability.

OT instantiation. We choose the Bellare-Micali OT [8] as an example, and construct efficent zero-knowledge proofs for the consistency. (In principle any OT is applicable if we do not insist on efficient ZK proofs). The sender S (the PKG in our setting) sets up the system parameters (including a prime q, group G_q with a random generator g, and a random value $C \in Z_q$). The receiver R(with input b) randomly chooses $PK_b = g^x$ and computes $PK_{1-b} = C/PK_b$, then R sends PK_0 to S; the sender computes $PK_1 = C/PK_0$ and encrypts the messages m_0, m_1 to be transmitted, using ElGamal encryption [54] with PK_0, PK_1 as public keys respectively, i.e., $\{(g^{r_b}, H(PK_b^{r_b}) \oplus m_b)\}_{b=0,1}$ are returned to R, where H is modeled as a random oracle. It is well-known that this OT protocol satisfies information theoretic receiver security, and simulatable sender security under the CDH assumption [106].

Public tracing key generation. We first describe the one way function tailored for our A-IBE scheme. Suppose $\bar{g} = (g_{1,0}, g_{1,1}), \ldots, (g_{\ell,0}, g_{\ell,1}) \in G^{2\ell}$, for each i, $g_{i,0} \cdot g_{i,1} = C$ for a random group element C, and $\bar{b} = b_1 \ldots b_\ell \in \{0,1\}^\ell$, we define $f_{\bar{g}}(b_1 \ldots b_\ell) = \prod_{i=1}^{\ell} g_{i,b_i}$. We will show that $f_{\bar{g}}(\cdot)$ is one way. Let us first look at a related one way function, suppose $\tilde{g} = (g_1, \ldots, g_\ell) \in G^\ell$ and for $b_1 \ldots b_\ell \in \{0,1\}^\ell$, $\tilde{f}_{\tilde{g}}(b_1 \ldots b_\ell)$ is defined by $\prod_{i=1}^{\ell} g_i^{b_i}$. It is

[6] Note that if the recovered fingerprint corresponds to one of the lost keys, it is impossible to decide whether it is from the user or from someone else who gets the lost key, not erroneously accusing the PKG is the best possible security in this case.

implicit that $\tilde{f}_{\tilde{g}}(\cdot)$ is one way in a couple of papers, e.g., in [**?**], $b_1 \dots b_\ell$ is the secret key and $\tilde{g}, h = \tilde{f}_{\tilde{g}}(b_1 \dots b_\ell)$ are the public keys for their circular secure encryption scheme. We will omit the proof of one-wayness for $\tilde{f}$, and we prove the one-wayness of our function f in the following lemma.

Lemma 38. *If there exists a* PPT *adversary $\mathcal{A}$ breaks the one way security of f with advantage δ, then there exists another PPT adversary $\mathcal{B}$ breaks the one way security of $\tilde{f}$ with advantage δ/ℓ.*

Proof. When $\mathcal{B}$ receives the public keys $\tilde{g} = g_1, \dots, g_\ell$ from the $\tilde{f}$ challenger $\mathcal{C}$, $\mathcal{B}$ selects a random C and prepares $g_{1,0}, \dots, g_{\ell,0}$ such that for each i, $g_{i,0} = g_i$, he also prepares $g_{1,1}, \dots, g_{\ell,1}$ in a way that $g_{i,1} = C/g_{i,0}$ for all i. $\mathcal{B}$ sends $\mathcal{A}$ $C, \bar{g} = (g_{1,0}, g_{1,1}), \dots, (g_{\ell,0}, g_{\ell,1})$ as public keys.

Once $\mathcal{B}$ receives the challenge $X = \tilde{f}_{\tilde{g}}(b_1 \dots b_\ell)$ for some $b_1 \dots b_\ell$, $\mathcal{B}$ selects a random $t \in \{1, \dots, \ell\}$, computes $Y = C^{\ell-t} \cdot \prod_{i=1}^{\ell} g_i \cdot X^{-2}$ and sends Y to $\mathcal{A}$. $\mathcal{B}$ forwards the bit string $b'_1 \dots b'_\ell$ returned by $\mathcal{A}$ as her answer to the challenger $\mathcal{C}$.

Note that if the bitstring $b_1 \dots b_\ell$ has Hamming weight $\ell - t$, i.e., t of them are 0, then $Y = \prod_{i=1}^{\ell} g_{i,b_i}$. To see this, suppose $S = \{i | b_i = 0\}$, and $|S| = t$, $Y = C^{\ell-t} \cdot \prod_{i=1}^{\ell} g_i / \prod_{i=1}^{\ell} g_i^{2b_i} = \prod_{i \in S} g_i \cdot \prod_{i \notin S} (C/g_i)$. Thus with probability $1/\ell$, $\mathcal{B}$ guesses t correctly, and in turn, $\mathcal{B}$ produces a valid value of $f_{\bar{g}}(b_1 \dots b_\ell)$. In this case under our assumption, $\mathcal{A}$ will invert correctly with probability δ. We can conclude that $\mathcal{B}$ breaks the one way security of $\tilde{f}$ with probability δ/ℓ. □

The public tracing key T will be $h = f_{\overline{PK}}(b_1 \dots b_\ell)$, together with $\overline{PK}$ which are $\{(PK_{1,0}, PK_{1,1}) \dots, (PK_{\ell,0}, PK_{\ell,1})\}$ used in the OT protocols.

Efficient zero-knowledge proof for consistency. Next, we provide an efficient zero-knowledge proof protocol for the consistency between the public tracing key and the bit

string selected by the user in the OT protocol. Essentially, we need to prove that each bit of the pre-image of the public tracing key is used for selecting one secret key in each call of the OT protocol. For the public tracing key h, the user first commits $\{PK_{i,b_i}\}$ to be $\{c_i\}$, and proves in zero-knowledge for the following statements, $\exists g_1, \ldots, g_\ell$:

$$h = \prod_{i=1}^{\ell} g_i \wedge_{i=1}^{\ell} [c_i \text{ opens to } g_i \wedge (g_i = PK_{i,0} \vee g_i = PK_{i,1})] \wedge \text{PoK for } \log_g h.$$

Before we describe the detailed ZK proofs, we first explain how we can prove a commitment opens to a value. We will use a homomorphic commitment scheme from the BBS encryption [18]. It has the public keys in the form of (g, u, v, w), where $u^x = v^y = w$, and x, y are private keys. The ciphertext (which is a commitment as well) for m is $\bar{C} = (C_1, C_2, C_3)$ where $C_1 = u^{r_1}, C_2 = v^{r_2}, C_3 = w^{r_1+r_2} m$. One can easily prove a BBS commitment $\bar{C}$ opens to a message m in zero-knowledge using the following $\Sigma-$protocol: the proof is in the form of (a_1, a_2, c, z_1, z_2), where $a_1 = C_1^{t_1}, a_2 = C_2^{t_2}$ are the first round messages sent by the prover, a random value c is returned by the verifier and $z_1 = t_1 + cx, z_2 = t_2 + cy$ are calculated by the prover, The verifier checks $C_1^{z_1} \cdot C_2^{z_2} = a_1 \cdot a_2 \cdot (C_3/m)^c$.

Now we are ready to construct the efficient ZK proofs. (1). Prove the first clause, which is equivalent to prove $\prod_{i=1}^{\ell} c_i$ opens to h. (2). Prove c_i opens to either $PK_{i,0}$ or $PK_{i,1}$. This can be done easily using the OR proof [41] of the two $\Sigma-$protocol. More specifically, suppose $b_i = 1$, the proof is in the form of $(a_{1,0}, a_{2,0}, a_{1,1}, a_{2,1}, c, z_{1,0}, z_{2,0}, z_{1,1}, z_{2,1})$, where $(a_{1,0}, a_{2,0}, c_0, z_{1,0}, z_{2,0})$ is simulated and using $c_1 = c - c_0$, and generates the proof of $(a_{1,1}, a_{2,1}, c_1, z_{1,1}, z_{2,1})$. The verifier checks $C_1^{z_{1,0}+z_{1,1}} \cdot C_2^{z_{2,0}+z_{2,1}} = a_{1,0} \cdot a_{1,1} \cdot a_{2,0} \cdot a_{2,1} \cdot (C_3/m)^c$. (3) Repeat step (2) for each commitment c_i to do an "And" proof. (4) Do a regular proof of knowledge about the exponent of h using e.g., Schnorr proof [121].

All these $\Sigma-$protocols can be made zero-knowledge following the standard technique, e.g., let the verifier commits to the challenge value c first, and they can be made non-interactive by applying the FS heuristic [53].

Finally, let us check whether the soundness is enough for ensuring h to be generated honestly, i.e. $h = \prod_{i=1}^{\ell} PK_{i,b_i}$. Suppose there is an adversary $\mathcal{A}$ convinces the verifier and uses one $PK_{i,1-b_i}$ when generating h. We can see that $\mathcal{A}$ can be separated into two independent parts $(\mathcal{A}_1, \mathcal{A}_2)$. $\mathcal{A}_1$ prepares $\{PK_{i,0}, PK_{i,1}\}$ and the corresponding exponents, and $\mathcal{A}_2$ finishes the ZK proofs. It follows that if we replace $\mathcal{A}_1$ with another algorithm $\mathcal{A}_1'$ which simply receives $\{PK_{i,0}, PK_{i,1}\}$ and the corresponding exponents from an oracle, the modified adversary $\mathcal{A}' = (\mathcal{A}_1', \mathcal{A}_2)$ behaves identically as $\mathcal{A}$.

According to the special soundness of the proof of knowledge part, a simulator can run $\mathcal{A}'$ ($\mathcal{A}_2$ part) to extract $\log_g h = \sum_{j \neq i} \alpha_{j,b_j} + \alpha_{i,1-b_i}$, where $\alpha_{j,b} = \log_g PK_{j,b}$ for $j \in \{1, \ldots, \ell\}$ and $b = 0, 1$. As the simulator can "control" the oracle of $\mathcal{A}_1'$, and prepare $\{PK_j\}_{j \neq i}$ accordingly for $\mathcal{A}_1'$, thus he knows the exponents $\{\alpha_{j,b_j}\}$ and recovers $\alpha_{i,1-b_i}$ and further $\log_g C = \alpha_{i,b_i} + \alpha_{i,1-b_i}$ thus breaks the discrete log assumption, where C is the system parameter in the OT protocol. (for the case that more than one PK_{i,b_i} are used in generating h, a similar argument can be made to recover $\log_g C$).

6.4.3 Concrete construction and security analysis.

With the building blocks we developed above, we now describe the concrete algorithms of our generic A-IBE construction allowing public traceability and identity reuse (named S-III). We only describe the difference with S-II here.

- **Setup**(λ, δ): Same as S-II.

- KeyGen: For the k-th key requests from user A for an identity ID, the KeyGen protocol of S-II is run for identity $ID||k$, and user returns $sk_{ID,k}$. During the KeyGen, the OT described above [8] is utilized to transmit secret keys. Suppose $\overline{PK}_k = \{(PK^k_{i,0}, PK^k_{i,1})\}$ are the first round messages of the user. After the OT protocols are done, A sends the PKG his public tracing key $h_k = \prod_{i=1}^{\ell} PK^k_{i,b_i}$ and proves in zero-knowledge (we call this proof π_k) for the consistency using protocol described in section 6.4.2. The PKG outputs a new public tracing key $T_k = (h_k, \overline{PK}_k)$, adds them to the list of public tracing keys T_{ID} for ID and updates the st_{ID} to be k. The PKG outputs (T_{ID}, st_{ID}) and the user outputs secret key $sk_{ID,k}$.

- $\mathsf{Enc}(ID, mpk, m, st_{ID}, \delta)$: It runs the Enc of S-II with identity $ID||st_{ID}$.

- $\mathsf{Dec}(C, sk_{ID,st_{ID}})$: It runs the Dec of S-II with identity $ID||st_{ID}$.

- $\mathsf{Trace}^B(ID, \delta/st_{ID}, T_{ID})$: The first stage is the same as S-II using $ID||st_{ID}$. If the judge does not output "user", the following is run. The second stage of the Trace algorithm of S-II is repeated for all identities from $ID||1$ to $ID||st_{ID}$. For $ID||k$, the algorithm recovers a bitstring $b^k_1, \ldots, b^k_\ell$ or $\perp$, and it records a flag t_k for this run. For $k = 1, \ldots, st_{ID} - 1$, if the recovered string is $\perp$ or $f_{\overline{PK}_k}(b^k_1, \ldots, b^k_\ell) = h_k$, where $h_k, \overline{PK}_k$ are from T_k, then $t_k = 0$; otherwise $t_k = 2$. For $k = st_{ID}$, if no string is extracted, $t_{st_{ID}} = 0$; if $f_{\overline{PK}_{st_{ID}}}(b^{st_{ID}}_1, \ldots, b^{st_{ID}}_\ell) = h_{st_{ID}}$, then $t_{st_{ID}} = 1$; otherwise, $t_{st_{ID}} = 2$.

  The algorithm returns $\perp$ if for all $k = 1, \ldots, st_{ID}$, $t_k = 0$; it returns "user" if $t_{st_{ID}} = 1$; it returns "PKG", otherwise.

Remark that using δ/st_{ID} for tracing is necessary, as from our definition of $\delta-$correctness in this case is only for a random state (see section 6.2).

Security analysis of S-III. Due to lack of space, we provide here only some high-level intuition for S-III, and mainly on the difference with S-I.

IND-ID-CPA security. This is very similar to that of S-I, except that there are extra public tracing keys T_{ID}, while they are only related with the bit strings for selecting the keys, thus independent with the real secret keys. Also S-III uses multiple extended form of identities, but all of them can be seen as different identities of the underlying IBE scheme. The semantic security is not influenced.

Security in the weak dishonest-PKG game. Note that a malicious PKG can evade the Trace algorithm only when the recovered string matches one of the fingerprints contained in the public tracing key. The difference with S-I is that the malicious PKG receives extra public tracing keys $\{T_i = (h_i, \overline{PK_i})\}$, and ZK proof transcripts $\{\pi_i\}$. If an adversary $\mathcal{A}$ (malicious PKG) is able to produce a pirate box which fools the Trace algorithm, it can be easily turned to an algorithm that breaks the OT receiver security or the one-wayness of f.

In more detail, we can argue the security via a sequence of game changes by first replacing each OT transcript with one generated using a random bit r. The indistinguishability can be guaranteed by the information theoretic receiver security of the Bellare-Micali OT. In the next game changes, the ZK proofs will be replaced with simulated transcripts, and the indistinguishability can be guaranteed by the zero-knowledge property of the proofs. Now in the last game, what the adversary sees are only simulated transcripts (OT and ZK proofs) which are independent with the actual fingerprints, there exists a simulator $\mathcal{S}$ who can use $\mathcal{A}$ to break the one-way security of f. In particular, $\mathcal{S}$ randomly picks an one way function instance, i.e., $\mathcal{S}$ embeds the public keys and a value h received from the one way security challenger and sets it to be the i-th public tracing key, and sends them (together with a simulated proof) to $\mathcal{A}$.

Then from the pirate box outputted by $\mathcal{A}$, with probability $1/st_{ID}$, the recovered string is the pre-image of h, thus $\mathcal{S}$ breaks the one way security.

Security in the adaptive dishonest user game. A malicious user may try to frame the PKG by outputting a box with recovered fingerprint not matching any of the public tracing keys for the target identity, and it is possible unless one of the following events happens:for at least one index i, the adversary $\mathcal{A}$, (1). learns the secret key of $ID||i||1-b_i$ during the OT protocol; (2). is able to decrypt ciphertext under $ID||i||1-b_i$ for which she does not have the secret key; (3). cheats in the ZK proof of consistency during KeyGen. We can similarly do a sequence of game changes that first replace the OT instance to be oracle, the indistinguishability is guaranteed by the simulatable sender security of OT. We then argue from a box, the tracing algorithm must extract one of the whole fingerprints of the keys. This is similar to the proof of theorem 37, we will focused on the main difference about the probabilistic argument. We can see that if the sender splits the message into $t(\delta) = 5 \ln \frac{2}{\delta}$ pieces, the user has to put at least $\frac{4}{5}$ fraction of keys for each state into the box B to ensure δ-correctness, and this fraction is enough for the ECC decoding to recover the whole original fingerprint.

The probabilistic argument. Let $\delta_i = \Pr[B \text{ decrypts correctly} \mid i \in S]$. We divide the indices $i \in \{1, \ldots, \ell\}$ in two sets, Bad and Good, we define $i \in \mathsf{Good}$ if and only if $\delta_i \geq \delta_0$, where $\delta_0 = \delta/\ell^2$. In order to upper bound the size of Bad consider the following. Let D be the event of correct decryption,

$$\Pr[\mathsf{D}] = \Pr[\mathsf{D} \mid S \cap \mathsf{Bad} = \emptyset] \cdot \Pr[S \cap \mathsf{Bad} = \emptyset] + \Pr[\mathsf{D} \mid S \cap \mathsf{Bad} \neq \emptyset] \cdot \Pr[S \cap \mathsf{Bad} \neq \emptyset],$$

Regarding $\Pr[S \cap \mathsf{Bad} = \emptyset]$ observe that if $k = |\mathsf{Bad}|$, this probability is bounded by $p(k,t) = C^t_{\ell-k}/C^t_\ell = \prod_{i=0}^{t-1}(1 - \frac{k}{\ell-i}) \leq (1 - \frac{k}{\ell})^t$. From inequality $e^x \geq 1 + x$, we can get $p(k,t) \leq e^{-kt\ell}$. Regarding $\Pr[\mathsf{D} \mid S \cap \mathsf{Bad} \neq \emptyset]$, note that it is bounded by

$\sum_{i \in \mathsf{Bad}} \delta_i \leq \ell\delta_0 = \delta/\ell$ (This follows from the fact that $\Pr[F| \cup_{i=1}^{n} A_i] \leq \sum_{i=1}^{n} \Pr[F|A_i]$, for any event F, A_i). We can now derive the following, $\delta \leq \Pr[\mathsf{D}] \leq e^{-tk/\ell} + \delta/\ell$, from which we obtain the upper bound $k \leq \frac{\ell}{t} \cdot \ln(\delta - \delta/\ell)^{-1}$, since $\delta - \delta/\ell \geq \delta/2$, when we set $t = 5\ln(2\delta^{-1})$ into the above bound for k, and in this case $k \leq \ell/5$.

Now in the last game, the adversary has only oracle access to OT which can be controlled by the simulator if from the outputted box, the simulator recovers a different fingerprint, the simulator can break the one way security using the extractor as explained at the end of section 6.4.2.

Public traceability is obvious, and identity reuse follows also straightforwardly as for each state, the identities are considered as independent "user", the above argument implicitly captures this property. We summarize the security properties of S-III in the following theorem:

Theorem 39. *(1). S-III is secure in the* `IND-ID-CPA` *model if the underlying IBE is IND-ID-CPA secure. (2). S-III is secure in the* `weak dishonest PKG` *game if the proof π is zero-knowledge, f is one way and the OT has receiver security. (3). S-III is secure in the* `adaptive dishonest user` *game, if the underlying IBE is IND-ID-CPA secure, the proof π is sound, and the CDH assumption holds.*

Proof. IND-ID-CPA security is the same as that in S-I, with only difference that in S-II, the adversary also receives the public tracing key. The simulator simply choose a random bitstring $\bar{b} \in \{0,1\}^{\ell}$, and generates the public tracing key according to $\bar{b}$. As the real fingerprint is uniformly random to adversary $\mathcal{A}$, thus this simulation is perfect. Suppose game G_0 is the original IND-ID-CPA game in the public traceability case, game G_1 is the same as G_0 except the public tracing key is generated as above. G_2 is the same as G_1 except the adversary does the above to generate the public tracing key for target ID. G_3 is the same as in G_2 except that there is no public tracing key available

to $\mathcal{A}$. It is easy to see that the advantage of adversary in all those games are equivalent. The rest follows the proof of Theorem 37.

Regarding security against malicious PKG, the difference with S-I is that the malicious PKG receives the extra public tracing key $T = (h, \overline{PK})$, and a zero-knowledge proof transcript π. The simulator $\mathcal{S}$ who plays the role of user in the KeyGen with adversary $\mathcal{A}$ and also the role of inverter with a challenger $\mathcal{C}$ of the one way function f. After receiving public keys from $\mathcal{C}$, $\mathcal{S}$ randomly picks one of $(PK_{i,0}, PK_{i,1})$ for each pair and sends them to $\mathcal{A}$ as the first round message in the OT protocol. As the OT protocol satisfies information theoretic receiver security, this simulation is as good as in a real execution. After receiving the challenge value h from $\mathcal{C}$, $\mathcal{S}$ rewinds the adversary $\mathcal{A}$ and simulates the zero-knowledge proof π. Once $\mathcal{A}$ outputs a pirate box B, $\mathcal{S}$ runs the Trace algorithm and recovers a bit string $\bar{b} = b_1 \dots b_\ell$, then returns $\bar{b}$ to $\mathcal{C}$. It is clear that if $\mathcal{A}$ can produce a pirate box which will be traced to the user, this means applying the one way function on the bit string recovered from the box will yield the value equal to h, thus $\bar{b}$ is a pre-image of h. We can conclude that at this moment, $\mathcal{S}$ breaks the one-wayness of f which contradicts Lemma 38.

For security against malicious user, as we see in the analysis of Theorem 37, the judge is able to recover the "fingerprint" of the keys contained in any pirate box. Besides the same factors as in S-I that for at least one index i, the adversary $\mathcal{A}$ might (1). learn the secret key of $ID||i||1-b_i$ during the OT protocol; (2). be able to decrypt ciphertext under $ID||i||1-b_i$ for which she does not have the secret key. There in only one extra possibilities for adversary to frame the honest PKG in S-II that (3). $\mathcal{A}$ cheats in the zero-knowledge proof of consistency during KeyGen. We will provide the game sequence here to show the security. Suppose game G_0 is the same of the original weak malicious user game, G_1 is the same as G_0 except that the public tracing key sent by the adversary

is not consistent with the bits used in the OT protocols. The advantage gap between G_0, G_1 is the same as the soundness error of the proof π, thus it is a negligible amount. The rest follows from the proof of Theorem 37. □

6.5 Conclusions and Open Problems

We presented a generic transformation from IBE to A-IBE, with ciphertext size to be only twice large as the underlying IBE. We further refine the generic construction, and for the first time achieve identity reuse. We believe that the efficient generic transformations with preferable advanced properties can be an important step towards a wider deployment of A-IBE thus may potentially stimulate the adoption of IBE schemes in practice.

There are still several interesting open problems relating to the authority accountability in IBE schemes. One is to consider efficient generic construction of A-IBE with fully blackbox traceability directly, the other is to do a systematic study about proactive deterring mechanisms for IBE schemes. For the latter, our ongoing work suggests to follow the leakage-deterring framework so that a master secret is embedded to each user, it can be unlocked only when the user has two decryption devices implemented using two different keys.

Chapter 7

Conclusions and Future Directions

Since we already summarized our results and described some open problems in each chapter, here we would like the end the dissertation with a conceptual overview. Research in computer security in general is to reach a good balance between usability and security. The more information revealed, the easier for users to operations on the data. Accountability is no exception.

One notable example is as follows: delegation and outsourcing are prominent functionalities provided by the emerging new technology of cloud computing. While by definition, our concept of leakage deterrence restricts the user from sharing his privileges of using secret keys in any form. They are at two opposite ends. How to find a meeting point, i.e., enable a fine-grained control of delegation, so that we can have the best of both would be one of the most natural directions to explore.

This philosophy is the same in the setting of cliptography. Of course, we will have to revisit cryptography in the new complete-subversion model; but eventually, we would like to build a framework that enables an accountable surveillance.

Bibliography

[1] S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In *ASIACRYPT*, pages 452–473, 2003. 194, 199

[2] A. Ambainis, M. Jakobsson, and H. Lipmaa. Cryptographic randomized response techniques. In *Public Key Cryptography - PKC 2004, 7th International Workshop on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004*, pages 425–438, 2004. 167

[3] E. Amiri and G. Tardos. High rate fingerprinting codes and the fingerprinting capacity. In *Proceedings of the Twentieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2009, New York, NY, USA, January 4-6, 2009*, pages 336–345, 2009. 165

[4] G. Ateniese, B. Magri, and D. Venturi. Subversion-resilient signature schemes. *IACR Cryptology ePrint Archive*, 2015:517, 2015. 116, 119, 122, 123

[5] M. H. Au, Q. Huang, J. K. Liu, W. Susilo, D. S. Wong, and G. Yang. Traceable and retrievable identity-based encryption. In *Applied Cryptography and Network Security, 6th International Conference, ACNS 2008, New York, NY, USA, June 3-6, 2008. Proceedings*, pages 94–110, 2008. 200

[6] M. Bellare and O. Goldreich. On defining proofs of knowledge. In *CRYPTO*, pages 390–420, 1992. 17

[7] M. Bellare and V. T. Hoang. Resisting randomness subversion: Fast deterministic and hedged public-key encryption in the standard model. pages 627–656, 2015. 114, 118, 150, 151

[8] M. Bellare and S. Micali. Non-interactive oblivious transfer and spplications. In *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, pages 547–557, 1989. 216, 220

[9] M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *ACM CCS*, pages 390–399, 2006. 18, 53

[10] M. Bellare and A. Palacio. Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *CRYPTO*, pages 162–177, 2002. 28

[11] M. Bellare, K. G. Paterson, and P. Rogaway. Security of symmetric encryption against mass surveillance. pages 1–19, 2014. 5, 114, 116, 118, 120, 123, 124, 150, 151, 152, 154

[12] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS*, pages 62–73, 1993. 18, 193

[13] M. Bellare and P. Rogaway. The exact security of digital signatures: How to sign with RSA and Rabin. pages 399–416, 1996. 153

[14] I. F. Blake and V. Kolesnikov. Strong conditional oblivious transfer and computing on intervals. In *Advances in Cryptology - ASIACRYPT 2004, 10th International*

Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004, Proceedings, pages 515–529, 2004. 161, 163, 164, 169

[15] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7):422–426, July 1970. 70, 75, 109, 110

[16] D. Boneh and X. Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, pages 223–238, 2004. 193

[17] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, pages 443–459, 2004. 193

[18] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *CRYPTO*, pages 41–55, 2004. 218

[19] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003. 9, 62, 193, 199

[20] D. Boneh and M. K. Franklin. An efficient public key traitor tracing scheme. In *Advances in Cryptology - CRYPTO '99*, pages 338–353, 1999. 66

[21] D. Boneh and M. Naor. Traitor tracing with constant size ciphertext. In *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, pages 501–510, 2008. 66, 83, 197

[22] D. Boneh, A. Sahai, and B. Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 573–592, 2006. 66, 69, 94

[23] D. Boneh, A. Sahai, and B. Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, pages 253–273, 2011. 193

[24] D. Boneh and J. Shaw. Collusion-secure fingerprinting for digital data. *IEEE Transactions on Information Theory*, 44(5):1897–1905, 1998. 158

[25] D. Boneh and B. Waters. A fully collusion resistant broadcast, trace, and revoke system. In *ACM CCS 2006*, pages 211–220, 2006. 66, 69, 94

[26] X. Boyen and L. Martin. Identity-Based Cryptography Standard (IBCS) #1: Supersingular Curve Implementations of the BF and BB1 Cryptosystems. RFC 5091 (Informational), Dec. 2007. 9, 195

[27] Z. Brakerski and V. Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In *FOCS 2011*, pages 97–106. 102

[28] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, pages 93–118, 2001. 14, 15

[29] R. Canetti. Security and composition of multiparty cryptographic protocols. *J. Cryptology*, 13(1):143–202, 2000. 183, 210

[30] R. Canetti, M. Charikar, S. R. Sridhar Rajagopalan, A. Sahai, and A. Tomkins. Non-transferrable anonymous credentials. US Patent 7,222,362., 2008. 14, 15, 55

[31] R. Canetti, S. Halevi, and J. Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, pages 207–222, 2004. 49

[32] L. Carter and M. N. Wegman. Universal classes of hash functions. *J. Comput. Syst. Sci.*, 18(2):143–154, 1979. 75, 88

[33] H. Chabanne, D. H. Phan, and D. Pointcheval. Public traceability in traitor tracing schemes. In *EUROCRYPT 2005*, pages 542–558, 2005. 66, 67, 72, 81

[34] A. Charpentier, C. Fontaine, T. Furon, and I. Cox. An asymmetric fingerprinting scheme based on tardos codes. In *Proceedings of the 13th international conference on Information hiding*, IH'11, pages 43–58, Berlin, Heidelberg, 2011. Springer-Verlag. 160, 162, 177, 182

[35] S. Checkoway, R. Niederhagen, A. Everspaugh, M. Green, T. Lange, T. Ristenpart, D. J. Bernstein, J. Maskiewicz, H. Shacham, and M. Fredrikson. On the practical exploitability of dual EC in TLS implementations. In *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, pages 319–335, 2014. 119, 130

[36] B. Chor, A. Fiat, and M. Naor. Tracing traitors. In *CRYPTO 94*, pages 257–270, 1994. 3, 66, 69, 82, 83, 88, 90

[37] B. Chor, A. Fiat, and M. Naor. Tracing traitors. In *CRYPTO*, pages 257–270, 1994. 16

[38] S. S. M. Chow. Removing escrow from identity-based encryption. In *Public Key Cryptography*, pages 256–276, 2009. 200

[39] R. Cleve. Limits on the security of coin flips when half the processors are faulty (extended abstract). In J. Hartmanis, editor, *STOC*, pages 364–369. ACM, 1986. 162

[40] J.-S. Coron. On the exact security of full domain hash. pages 229–235, 2000. 153, 156

[41] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO*, pages 174–187, 1994. 218

[42] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM J. Comput.*, 33(1):167–226, 2004. 32, 90

[43] I. Damgard. On σ- protocols. In *http://www.daimi.au.dk/ ivan/Sigma.pdf*, 2010. 18

[44] I. Damgård, S. Faust, and C. Hazay. Secure two-party computation with low communication. In *TCC*, pages 54–74, 2012. 159

[45] I. Damgård and M. Jurik. A generalisation, a simplification and some applications of paillier's probabilistic public-key system. In K. Kim, editor, *Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 119–136. Springer, 2001. 163, 169, 182

[46] I. Damgård and S. Zakarias. Constant-overhead secure computation of boolean circuits using preprocessing. In *TCC*, pages 621–641, 2013. 159

[47] Y. Dodis, C. Ganesh, A. Golovnev, A. Juels, and T. Ristenpart. A formal treatment of backdoored pseudorandom generators. pages 101–126, 2015. 6, 114, 115, 116, 118, 124, 141, 142, 144, 147, 148

[48] Y. Dodis, S. Halevi, and T. Rabin. A cryptographic solution to a game theoretic problem. In *Advances in Cryptology - CRYPTO 2000, 20th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2000, Proceedings*, pages 112–130, 2000. 167

[49] Y. Dodis, I. Mironov, and N. Stephens-Davidowitz. Message transmission with reverse firewalls—secure communication on corrupted machines. *IACR Cryptology ePrint Archive*, 2015:548, 2015. 119

[50] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *EUROCRYPT 2004*, pages 523–540, 2004. 69, 88, 89

[51] C. Dwork, J. B. Lotspiech, and M. Naor. Digital signets: Self-enforcing protection of digital information (preliminary version). In *STOC*, pages 489–498, 1996. 4, 13, 15, 60, 68, 70, 72, 105

[52] S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302, 2008. 17

[53] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986. 18, 219

[54] T. E. Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO*, pages 10–18, 1984. 216

[55] S. Garg, C. Gentry, S. Halevi, M. Raykova, A. Sahai, and B. Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *FOCS 2013*, pages 40–49, 2013. 69, 100

[56] S. Garg, A. Kumarasubramanian, A. Sahai, and B. Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In *ACM CCS 2010*, pages 121–130. 66

[57] C. Gentry. Practical identity-based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT 2006, 25th Annual International Conference on the Theory and Applications of Cryptographic Techniques, St. Petersburg, Russia, May 28 - June 1, 2006, Proceedings*, pages 445–464, 2006. 193, 197, 199

[58] C. Gentry and A. Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT*, pages 548–566, 2002. 62

[59] O. Goldreich. *The Foundations of Cryptography - Volume 2, Basic Applications*. Cambridge University Press, 2004. 168

[60] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. In *STOC*, pages 25–32, 1989. 58

[61] O. Goldreich and L. A. Levin. A hard-core predicate for all one-way functions. pages 25–32, 1989. 116, 144, 146

[62] O. Goldreich, B. Pfitzmann, and R. L. Rivest. Self-delegation with controlled propagation-or-what if you lose your laptop. In *CRYPTO*, pages 153–168, 1998. 16

[63] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC'13*, pages 555–564. 69, 76, 100, 101

[64] P. Golle, F. McSherry, and I. Mironov. Data collection with self-enforcing privacy. In *ACM CCS*, pages 69–78, 2006. 16

[65] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Functional encryption with bounded collusions via multi-party computation. In *CRYPTO 2012*, pages 162–179, 2012. 69, 76, 78, 101, 103, 104

[66] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Predicate encryption for circuits from lwe. *IACR Cryptology ePrint Archive*, 2015. 102

[67] V. Goyal. Reducing trust in the pkg in identity based cryptosystems. In *CRYPTO*, pages 430–447, 2007. 16, 194, 198, 199

[68] V. Goyal, S. Lu, A. Sahai, and B. Waters. Black-box accountable authority identity-based encryption. In *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, pages 427–436, 2008. 16, 194, 198, 199

[69] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM CCS*, pages 89–98, 2006. 193, 198

[70] V. Guruswami and P. Indyk. Expander-based constructions of efficiently decodable codes. In *FOCS*, pages 658–667, 2001. 41, 213

[71] V. Guruswami and M. Sudan. Improved decoding of reed-solomon and algebraic-geometry codes. *IEEE Trans on Information Theory*, 45(6):1757–1767, 1999. 69, 89

[72] S. Haber and B. Pinkas. Securely combining public-key cryptosystems. In *ACM CCS*, pages 215–224, 2001. 58

[73] N. J. Hopper, J. Langford, and L. von Ahn. Provably secure steganography. pages 77–92, 2002. 118

[74] C.-Y. Hsiao, C.-J. Lu, and L. Reyzin. Conditional computational entropy, or toward separating pseudoentropy from compressibility. In *EUROCRYPT*, pages 169–186, 2007. 58

[75] Y. Ishai, E. Kushilevitz, S. Meldgaard, C. Orlandi, and A. Paskin-Cherniavsky. On the power of correlated randomness in secure computation. In *TCC*, pages 600–620, 2013. 159

[76] Y. Ishai and A. Paskin. Evaluating branching programs on encrypted data. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pages 575–594, 2007. 163, 164

[77] Y. Ishai, A. Sahai, and D. Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, pages 463–481, 2003. 17

[78] M. Jakobsson, A. Juels, and P. Q. Nguyen. Proprietary certificates. In *CT-RSA*, pages 164–181, 2002. 14, 15

[79] P. F. Jean Paul Degabriele and B. Poettering. A more cautious approach to security against mass surveillance. In *Fast Software Encryption 2015.*, 2015. 122, 123, 124, 154

[80] A. Juels and J. Guajardo. RSA key generation with verifiable randomness. pages 357–374, 2002. 118

[81] A. Juels and M. Sudan. A fuzzy vault scheme. *Des. Codes Cryptography*, 38(2):237–257, 2006. 88

[82] A. Kiayias, N. Leonards, H. Lipmaa, K. Pavlyk, and Q. Tang. Optimal rate private information retrieval from homomorphic encryption. *Proceedings on Privacy Enhancing Technologies*, 2015(2):222–243, 2015. 161, 163, 164, 165, 170

[83] A. Kiayias and S. Pehlivanoglu. *Encryption for Digital Content*, volume 52 of *Advances in Information Security*. Springer, 2010. 69, 82

[84] A. Kiayias and S. Pehlivanoglu. *Encryption for Digital Content*, volume 52 of *Advances in Information Security*. Springer, 2010. 165

[85] A. Kiayias and Q. Tang. How to keep a secret: leakage deterring public-key cryptosystems. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 943–954, 2013. 72, 197

[86] A. Kiayias and Q. Tang. Making any identity based encryption accountable, efficiently. In *20th European Symposium on Research in Computer Security-ESORICS '15*, 2015. 16

[87] A. Kiayias and M. Yung. Traitor tracing with constant transmission rate. In *EUROCRYPT'02*, pages 450–465. 16, 66, 83

[88] A. Kiayias and M. Yung. Breaking and repairing asymmetric public-key traitor tracing. In *Digital Rights Management Workshop*, pages 32–50, 2002. 16

[89] H. Komaki, Y. Watanabe, G. Hanaoka, and H. Imai. Efficient asymmetric self-enforcement scheme with public traceability. In *Public Key Cryptography*, pages 225–239, 2001. 16

[90] K. Kurosawa and Y. Desmedt. Optimum traitor tracing and asymmetric schemes. In *Advances in Cryptology - EUROCRYPT '98*, pages 145–157, 1998. 66

[91] J. Lai, R. H. Deng, Y. Zhao, and J. Weng. Accountable authority identity-based encryption with public traceability. In *Topics in Cryptology - CT-RSA 2013 - The Cryptographers' Track at the RSA Conference 2013, San Francisco,CA, USA, February 25-March 1, 2013. Proceedings*, pages 326–342, 2013. 194, 197, 198, 199

[92] J. Larson, N. Perlroth, and S. Shane. Revealed: The NSA's secret campaign to crack, undermine internet security. Pro-Publica, 2013. http://www.propublica.org/article/the-nsas-secret-campaign-to-crack-undermine-internet-encryption. 113

[93] B. Libert and D. Vergnaud. Towards black-box accountable authority IBE with short ciphertexts and private keys. In *Public Key Cryptography - PKC 2009, 12th International Conference on Practice and Theory in Public Key Cryptography, Irvine, CA, USA, March 18-20, 2009. Proceedings*, pages 235–255, 2009. 16, 194, 198, 199

[94] H. Lipmaa. First CPIR protocol with data-dependent computation. In *Information, Security and Cryptology - ICISC 2009, 12th International Conference, Seoul, Korea, December 2-4, 2009, Revised Selected Papers*, pages 193–210, 2009. 163

[95] H. Lipmaa, G. Wang, and F. Bao. Designated verifier signature schemes: Attacks, new security notions and a new construction. In *ICALP*, pages 459–471, 2005. 16

[96] L.Lamport. Constructing digital signatures from a one-way function. In *Technical Reprot SRI-CSL-98*, 1979. 19

[97] A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, pages 597–612, 2002. 152

[98] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf. Pseudonym systems. In *International Workshop on Selected Areas in Cryptography*, SAC '99, pages 184–199, 2000. 14, 15, 55

[99] S. Micali, C. Peikert, M. Sudan, and D. A. Wilson. Optimal error correction against computationally bounded noise. In *TCC 2005*, pages 1–16, 2005. 89

[100] I. Mironov and N. Stephens-Davidowitz. Cryptographic reverse firewalls. pages 657–686, 2015. 114, 119

[101] D. Naccache, A. Shamir, and J. P. Stern. How to copyright a function? In *Public Key Cryptography*, pages 188–196, 1999. 16

[102] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. 2009. 111

[103] M. Naor. On cryptographic assumptions and challenges. In *CRYPTO 2003*, pages 96–109, 2003. 68

[104] M. Naor and K. Nissim. Communication preserving protocols for secure function evaluation. In J. S. Vitter, P. G. Spirakis, and M. Yannakakis, editors, *STOC*, pages 590–599. ACM, 2001. 159

[105] M. Naor and B. Pinkas. Efficient trace and revoke schemes. In *Financial Cryptography, 4th International Conference, FC 2000 Anguilla, British West Indies, February 20-24, 2000, Proceedings*, pages 1–20, 2000. 112

[106] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms, January 7-9, 2001, Washington, DC, USA.*, pages 448–457, 2001. 200, 216

[107] NIST. Special publication 800-90: Recommendation for random number generation using deterministic random bit generators. National Institute of Standards and Technology, 2012. http://csrc.nist.gov/publications/PubsSPs.html. 119

[108] A. Pagh, R. Pagh, and S. S. Rao. An optimal bloom filter replacement. In *SODA 2005*, pages 823–829. 110

[109] C. Peikert and B. Waters. Lossy trapdoor functions and their applications. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, Victoria, British Columbia, Canada, May 17-20, 2008*, pages 187–196, 2008. 48, 49

[110] N. Perlroth, J. Larson, and S. Shane. N.S.A. able to foil basic safeguards of privacy on web. The New York Times, 2013. http://www.nytimes.com/2013/09/06/us/nsa-foils-much-internet-encryption.html. 113

[111] B. Pfitzmann. Trials of traced traitors. In R. J. Anderson, editor, *Information Hiding*, volume 1174 of *Lecture Notes in Computer Science*, pages 49–64. Springer, 1996. 159, 177

[112] B. Pfitzmann and M. Schunter. Asymmetric fingerprinting (extended abstract). In U. M. Maurer, editor, *EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 84–95. Springer, 1996. 159, 162, 174, 175, 177

[113] B. Pfitzmann and M. Waidner. Asymmetric fingerprinting for larger collusions. In R. Graveman, P. A. Janson, C. Neumann, and L. Gong, editors, *ACM Conference on Computer and Communications Security*, pages 151–160. ACM, 1997. 159, 177

[114] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13(3):361–396, 2000. 18, 50, 51, 55

[115] O. Regev. On lattices, learning with errors, random linear codes, and cryptography. *J. ACM*, 56(6), 2009. 102

[116] A. Rial, J. Balasch, and B. Preneel. A privacy-preserving buyer-seller watermarking protocol based on priced oblivious transfer. *IEEE Transactions on Information Forensics and Security*, 6(1):202–212, 2011. 160

[117] A. Rial, M. Deng, T. Bianchi, A. Piva, and B. Preneel. A provably secure anonymous buyer-seller watermarking protocol. *IEEE Transactions on Information Forensics and Security*, 5(4):920–931, 2010. 160

[118] A. Sahai and H. Seyalioglu. Fully secure accountable-authority identity-based encryption. In *Public Key Cryptography*, pages 296–316, 2011. 16, 194, 198, 199

[119] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005. 193, 198, 199

[120] T. Sander and A. Ta-Shma. Auditable, anonymous electronic cash extended abstract. In *CRYPTO*, pages 555–572, 1999. 16

[121] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO*, pages 239–252, 1989. 218

[122] S. F. Shahandashti and R. Safavi-Naini. Construction of universal designated-verifier signatures and identity-based signatures from standard signatures. In *Public Key Cryptography*, pages 121–140, 2008. 16

[123] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984. 62, 193

[124] G. J. Simmons. The prisoners' problem and the subliminal channel. pages 51–67, 1983. 118

[125] G. J. Simmons. A secure subliminal channel (?). pages 33–41, 1986. 118

[126] M. Sudan. Decoding of reed solomon codes beyond the error-correction bound. *J. Complexity*, 13(1):180–193, 1997. 69, 89

[127] G. Tardos. Optimal probabilistic fingerprint codes. *J. ACM*, 55(2), 2008. 83, 112, 160, 161, 162, 165, 167, 178, 180, 183, 186, 188

[128] B. Waters. Efficient identity-based encryption without random oracles. In *Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings*, pages 114–127, 2005. 193

[129] B. Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *Advances in Cryptology - CRYPTO 2009, 29th Annual*

International Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2009. Proceedings, pages 619–636, 2009. 193

[130] A. Young and M. Yung. The dark side of "black-box" cryptography, or: Should we trust capstone? pages 89–103, 1996. 5, 113, 117, 124, 128

[131] A. Young and M. Yung. Kleptography: Using cryptography against cryptography. pages 62–74, 1997. 113, 117, 124, 128

[132] T. H. Yuen, S. S. M. Chow, C. Zhang, and S. Yiu. Exponent-inversion signatures and IBE under static assumptions. *IACR Cryptology ePrint Archive*, 2014:311, 2014. 194, 198, 199