Cryptographic Resilience to Continual Information Leakage

by

Daniel Wichs

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

New York University

September, 2011

Professor Yevgeniy Dodis

“It is a very sad thing that nowadays there is so little useless information.”

Oscar Wilde

“𐇑𐇛𐇜𐇐𐇡 𐇧𐇷𐇛 𐇬𐇼𐇠 𐇬𐇬𐇱
𐇑𐇛𐇔𐇷𐇰 𐇪𐇼𐇠𐇛.”

Minoan Proverb

To Elizabeth

Acknowledgements

Foremost, I would like to thank my advisor, Yevgeniy Dodis, for teaching me how to do research. I have always admired and tried to learn from Yevgeniy's enthusiasm in pursuing all kinds of new research problems, finding connections between different areas, and forging better abstractions and ways of understanding results. Most of all, I am grateful for the abundance of energy that Yevgeniy brought to our many collaborations. After long brainstorming sessions with Yevgeniy, I would usually emerge exhausted and slightly dazed, but also tremendously excited about the many new problems, ideas and directions that I couldn't wait to pursue. From Yevgeniy, I learned that research can be a lot of fun.

Next, I would like to thank my many other mentors and their respective institutions, who supported me both intellectually and financially during my summer internships: Ivan Damgård and Jesper Buus Nielsen at the University of Århus in Denmark, Juan Garay at Bell Laboratories (now at AT&T), Yael Tauman Kalai at Microsoft Research, and Tal Rabin and the crypto group at IBM Research T.J. Watson. Spending my summers away from NYU allowed me to get a much broader view of research areas, institutions, and approaches, than I would have had otherwise. In particular, I thank Ivan and Jesper for having the patience to work with me while I was still learning the most basic concepts of cryptography, Juan for much useful advice on research, espresso, single malt scotch, and many other aspects of life, Yael for her great enthusiasm and encouragement in the research topics relating to this thesis, and Tal for providing a truly fantastic and fun place to do research. I wish to thank Moni Naor at the Weizmann Institute of Science in Israel, Stefan Dziembowski at University of Rome La Sapienza, and Krzysztof Pietrzak at CWI Amsterdam for arranging research visits to their respective institutions. These visits turned out to be highly productive and great deal of fun.

I would also like to thank Joël Alwen, with whom I have collaborated on several results related to this thesis and discussed almost every conceivable topic in cryptography. I truly enjoyed and learned much from these enlightening conversations. I also thank Victor Shoup and the rest of the NYU cryptography group for all their support. The results in this thesis emerged from joint works with Yevgeniy Dodis, Kristiyan Haralambiev, Adriana Lòpez-Alt, Allison Lewko and Brent Waters. I thank them all for the many wonderful discussions and for all their hard work.

I thank my parents, Tomas and Adela, and my brother David, who first introduced me to the world and its many exciting mysteries. My parents made many sacrifices to ensure that I get a great education, and this thesis would not have been possible otherwise.

Last, but certainly not least, I thank Elizabeth for all her love, support, encouragement, patience and care.

Abstract

In this thesis, we study the question of achieving cryptographic security on devices that leak information about their internal secret state to an external attacker. This study is motivated by the prevalence of *side-channel attacks*, where the physical characteristics of a computation (e.g. timing, power-consumption, temperature, radiation, acoustics, etc.) can be measured, and may reveal useful information about the internal state of a device. Since some such leakage is inevitably present in almost any physical implementation, we believe that this problem cannot just be addressed by physical countermeasures alone. Instead, it should already be taken into account when designing the mathematical specification of cryptographic primitives and included in the formal study of their security.

In this thesis, we propose a new formal framework for modeling the leakage available to an attacker. This framework, called the *continual leakage model*, assumes that an attacker can continually learn arbitrary information about the internal secret state of a cryptographic scheme at any point in time, subject only to the constraint that the rate of leakage is bounded. More precisely, our model assumes some abstract notion of time periods. In each such period, the attacker can choose to learn arbitrary *functions* of the current secret state of the scheme, as long as the number of output bits leaked is not too large. In our solutions, cryptographic schemes will continually update their internal secret state at the end of each time period. This will ensure that leakage observed in different time periods cannot be meaningfully combined to break the security of the cryptosystem. Although these updates modify the secret state of the cryptosystem, the desired functionality of the scheme is preserved, and the users can remain oblivious to these updates. We construct *signatures*, *encryption*, and *secret sharing/storage* schemes in this model.

Contents

List of Figures

Chapter 1

Introduction

Information Leakage. One of the central tenets of computer science is that computation can be analyzed abstractly and independently of the physical processes that ultimately implement it. This is the paradigm usually followed in cryptography, where we analyze cryptosystems as abstract algorithms that get inputs and generate outputs using some internal *secret state*.[1] In particular, this paradigm assumes that the secret state of the cryptosystem stays perfectly hidden from the attacker, beyond what the algorithmic specification of the cryptosystem is designed to reveal via its outputs. Unfortunately, this abstraction may fail to properly model the real world, where various physical attributes of a computation executing on a physical device (e.g. its timing, power-consumption, temperature, radiation, acoustics, etc.) can be measured and may leak useful information about the internal state of the computation. Attacks that use such information to break security are called *side-channel attacks*, and they have been analyzed and exploited in many recent works, breaking real-world implementations of "provably secure" cryptosystems. For example, Kocher et al. [KJJ99] show how to retrieve the secret key of a smart-card running the DES cryptosystem, just by observing its power consumption. See e.g. [Koc96, KJJ99, QS01, AARR02, QK02, BE03, Rel, ECR] and the references therein for many other examples.

Moreover, there are many other scenarios, beyond side-channel attacks, where some information about the internal secret state of a device can leak to an attacker. For example, the device could be infected by a virus sending information about its internals to a remote adversary. Alternatively, some partial remnants of the secret state may remain on a device even after an attempt has been made to erase them or after the device is powered down and is made accessible to an attacker (e.g. see the "cold-boot attack" of Halderman et al. [HSH$^+$08]).

In all such scenarios, some unanticipated and unspecified information about

[1] Often, this is just a static *secret key*, but stateful cryptographic schemes (e.g. stream ciphers) have also been considered. The idea of an evolving state will be crucial to this thesis.

the secret state of a cryptosystem is leaked to an attacker and hence the standard theoretical security guarantees and proofs of security, which do not take leakage into account, become meaningless. Therefore, the prevalence of such attacks poses a major challenge to the applicability of cryptographic theory to practice.

Physical vs. Algorithmic Defenses. The obvious defense against leakage attacks is to design better physical implementations, which minimize the amount of leakage available to an attacker. Indeed, we must always require some security guarantees from the physical implementation, since it is clear that cryptographic security cannot be achieved if the implementation readily reveals its entire secret state to the outside world. Nevertheless, we believe that it is insufficient to rely on this approach alone since it seems *futile to expect that any physical implementation of a computational task will be completely free of all information leakage*. Moreover, being overprotective on the physical level may be unnecessarily expensive and cumbersome. An alternative and complementary approach is to design *resilient* algorithmic specifications of cryptosystems, which maintain security *despite* the availability of some leakage on the physical device on which they are ultimately implemented. This thesis will focus on the latter approach.

Ad-Hoc Countermeasures. There has been much prior research from the cryptographic hardware community, combining physical and algorithmic countermeasures against various specific attacks on various specific hardware architectures. See [ECR] for an overview of this general line of research. Unfortunately, most of the proposed countermeasures are ad-hoc and are only backed up by heuristic and informal security arguments. In particular, such countermeasures might potentially be broken by new side-channel attacks or even just small variants of the attack that they were designed to protect against. This cyclic "attack-fix" philosophy is fundamentally different from the *provable security* approach taken by modern cryptography, where a schemes should be *proven* secure against general adversaries and not only particular attacks.

Theory of Leakage Resilience. In this thesis, we take a methodical approach, by studying resilience to leakage within the framework of modern cryptography, with formal definitions and proofs of security. Our first contribution is to define a (new) general model of leakage, which is independent of any particular hardware architecture, and is likely to capture realistic examples of current and future side-channel attacks. This model, called the *continual leakage model*, essentially assumes that the attacker can *continually leak arbitrary information about the internal secret state of a cryptosystem, subject only to the constraint that there is a known upper bound on the rate of leakage*, measured as the number of leaked bits per unit time or per computation. We then present constructions of various cryptographic primitives and formally prove that they remain secure in this model, under well-studied computational hardness assumptions.

This thesis is based on our corresponding publications [DHLW10a, DLWW11]. Although the continual leakage model and the solutions that we propose are new, the idea of modeling leakage formally and constructing provably secure countermeasures has a rich history and has received much attention in recent years. We first give a survey of prior work in Chapter 2. Then, in Chapter 3, we present our continual leakage model in more detail and give an informal overview of our results for this model. Finally, we will begin the technical exposition in Chapter 4.

Chapter 2

Prior Models and Results

In this section, we give a survey of the prior work on leakage resilience and various prior models of leakage. Although this is useful for understanding the history and the choices that we made when defining the *continual-leakage model*, the contents of this survey are not needed in order to understand the technical content of future chapters. Therefore the impatient reader who immediately wants to see our model and results can safely (but without encouragement) skip to Chapter 3.

2.1 Models of Full Leakage

Many prior works consider the setting where a device can become completely compromised and its *entire* secret state is leaked to an attacker. This is an extremely pessimistic model of leakage, and it is clear that we cannot maintain security if the secret state of the device provides the full functionality which we are trying to protect (e.g. the ability to sign arbitrary messages or to decrypt arbitrary ciphertexts). Therefore, these works consider interesting settings where the compromised device may *not* need to be fully functional on its own.

2.1.1 Threshold Cryptography

One setting in which full leakage has been studied is the *distributed setting*, where multiple parties/devices are needed to jointly accomplish a sensitive task. For example, Shamir's *secret sharing* [Sha79] shows how to securely store a secret value across multiple devices so that it remains secret even if small subsets of the devices are fully compromised (but larger subsets can recover the value). Work on *secure multiparty computation* [Yao82, GMW87, BGW88] shows how to securely perform arbitrary computations in such distributed fashion, while work on *threshold cryptosystems* [DF89, SDFY94, Gem97] shows how to efficiently distribute specific cryptographic tasks like encryption or signatures. Some works also consider *proactive security* [HJKY95, FGMY97], where the "shares" of the devices are periodically updated so as to protect against a mobile adversary who corrupts different subsets of devices in different time periods.

2.1.2 Forward Security

Another setting in which the study of complete leakage is interesting is commonly referred to as *forward security* [DvOW92, And97, BM99, AR00, MMM02, CHK03]. In this setting, a cryptographic scheme is implemented on a single device, but this device has an *evolving state* which provides *evolving functionality* in different time periods. For example, in a forward-secure signature scheme, there is a static verification key but the signing key of the device evolves in each time period and the signatures that the device produces at some time T only verify with respect to the given period T. Forward security requires that, if the secret state of the scheme is fully leaked in some time period T, the functionality that was available in the *past* stays protected (e.g. the attacker will be unable to forge signatures for time-periods $1, \ldots, T-1$). In addition to signatures, we know how to do encryption, identification and key agreement in this setting.

2.1.3 Key-Insulated Cryptography

The work of Dodis et al. [DKXY02, DKXY03] on *key-insulated cryptography* merges threshold cryptography and forward security, inheriting some of the benefits and restrictions of each. In particular, this setting considers two distinct devices: a *secure* "base" who never gets compromised and an *insecure* client who may get compromised at different time periods. As in the setting of forward security, the state of the client and the functionality of the cryptosystem evolves over time (e.g. signatures are produced and verified with respect to some fixed time period T). Moreover, the client has the ability to perform the various cryptographic operations (e.g. signing) *on its own* in each time period, without the help of the base. However, unlike the setting of forward security, the client must now contact the base to evolve its secret state into the next time period. The attacker can fully compromises the client in some time periods, but security should be maintained for *all other time periods*, past as well as future. For example, in a key insulated signature scheme, the attacker should be unable to produce signatures that verify with respect to any time period in which the client was not compromised. An extended security notion called *intrusion-resilient cryptography* [IR02, DFK$^+$03, DFK$^+$04] allows the base to get compromised in some time periods as well, and if the base/client are ever compromised simultaneously then at least forward-security is maintained and *past* time-periods remain secure.

2.2 Models of Partial Leakage

If we want to consider leakage resilience for devices which already contain the very functionality that we are trying to protect (e.g. devices storing a fully functional secret key of an encryption or signature scheme), than we must consider models where the leakage is *incomplete*. Many such models appeared in the literature, differing on *which* components of a device can leak, the *type/complexity* of the allowed leakage, and the *amount* of allowed leakage.

2.2.1 Oblivious RAMs

Perhaps the first model to consider incomplete leakage is that of *oblivious RAMs* [Gol87, GO96, Ajt10, DMN11]. This model assumes an architecture where a (large) random-access memory is controlled by an adversary, and hence its contents and the access pattern to it can leak completely. On the other hand, The processor along with some (small) cache is assumed to be perfectly secure and *leakage free*. Assuming such hardware architecture, it is shown to implement any (possibly large) computation securely. Unfortunately, solutions in this model already assume that the leak-free processor can store small secret keys and perform small cryptographic computations securely, without any leakage. In particular, the RAM is encrypted and the processor securely decrypts each block of memory when it is accessed – the difficult aspect of the solutions is to ensure that the pattern of "which memory blocks are accessed and when" does not reveal useful information. The work of [Ajt11] shows how to construct oblivious RAMS that remain secure even if the *decrypted contents* of some small fraction of accessed locations can leak to the attacker. Still, all models of oblivious RAM assume fairly non-trivial secure hardware, which can at minimum perform encryption/decryption without producing any leakage on the secret key.

2.2.2 Exposure resilient cryptography

The study of *exposure resilient cryptography* [Dod00, CDH$^+$00, DSS01] considers partial leakage, where an attacker can leak an arbitrary but sufficiently small *subset of the bits* of the secret state of a device. These works propose a solution where the secret state is encoded using a tool called an *all or nothing transform (AONT)*, initially studied by [Riv97, Boy99]. An AONT can efficiently encode any secret value into a codeword so that leaking any (small enough) subset of the bits of the codeword does not reveal any information about the secret, but, given the entire codeword, the secret can be efficiently reconstructed. Alternatively, one can think of an AONT as a (gap) secret-sharing scheme where the shares are *bits*. Known constructions of statistically secure AONTs can maintain security even if a $(1-\epsilon)$ fraction of the bits of the encoding can leak, for any constant $\epsilon > 0$.

There are three main criticisms of exposure resilient cryptography, which later works attempt to address. **Firstly**, in using exposure-resilient cryptography to protect a cryptographic scheme against leakage, we can encode the secret state of the device to protect it – but what do we do if we want to perform some computation using the encoded state? The natural solution is to first decode the original state and then run the original computation on it. However, this is completely insecure if the attacker can get even some simple leakage during the computation, when the secret state is decoded and manipulated in the clear. In other words, exposure-resilient cryptography allows us to protect *storage, but not computation.* **Secondly**, if an attacker has prolonged access to a physical device and the ability to make many side-channel observations, it is not clear how to justify that the total number of bits leaked is *bounded overall.* This question is partially addressed by [CDH$^+$00], which considers a method for (deterministically) updating the encoding so that a different subset of the bits can leak from each updated version. However, the update is assumed to be leak free. **Thirdly**, exposure-resilient cryptography may be overly restrictive in the *type* of leakage, by only allowing an attacker to probe individual bits of the state, but not allowing her to learn any global properties. For example, AONTs do not provide any guarantees if the attacker can learn the hamming weight of the codeword or the XOR of all the bits of the codeword, even though the amount of leakage in such cases is small. Indeed, since the hamming weight of the state usually corresponds to the amount of power flowing through the device, learning the hamming weight is the basis of various practical differential power analysis (DPA) side-channel attacks (e.g. [KJJ99]). The work of [DDV10] can be seen as addressing exactly this criticism and extending AONTs to other (less) restricted types of leakage on a codeword, such as assuming that two halves of the codeword leak arbitrarily but independently and that the amount of leakage on each half is bounded.

2.2.3 Private circuits

The work of Ishai et al. on *private circuits* [ISW03] shows how to encode the secret state of a device and perform arbitrary computations on it privately, even if some sufficiently small *subset of the wires* of the circuit performing the computation can leak to an attacker *in each invocation.* That is, there is a general *compiler* which takes any computation and its secret state and outputs a compiled stateful circuit that operates on an encoded state and achieves the same functionality. The attacker can continually invoke the compiled circuit on arbitrary inputs, and learn the values of some subset of the wires during the computation of the compiled circuit, but will not learn anything useful beyond just the outputs of the original computation (formalized via the simulation paradigm). Since the subset of the wires that can leak may correspond to the encoded secret state, the encoding is also necessarily an AONT (albeit with sub-optimal parameters). Therefore, we can view work on private circuits as addressing the first two criticisms of exposure-resilient cryptography by: (1) showing how to securely run arbitrary computations on an AONT-encoded state without decoding it and (2) only bounding the number of wires that can leak per invocation (but not overall) by updating the encoded secret state in each invocation. Constructions of private circuits can be instantiated so as to meet any bound ℓ on the *number* of wires that can leak, by making the compiled circuit sufficiently large depending on ℓ. Unfortunately, the *fraction* of wires that can leak tends to 0 in this solution. The work of [Ajt11] gives a solution where the fraction of wires that leaks is a small constant, but the subset of wires that leaks is random and not adverasrial.

Private circuits already provide a way to protect general computations against a well-defined class of leakage attacks. The main criticism of private circuits is inherited from exposure-resilient cryptography – by restricting the *type/complexity* of allowed leakage to getting individual wire values, we may fail to capture many real-world side-channel attacks, such as DPA attacks that may leak the hamming weight of the wires. Another criticism is that, by focusing on leakage of wires in a circuit, we get tied down to a specific hardware architecture, which may be undesirable in practice.

2.2.4 Private Circuits for AC0 and Noisy Leakage

Faust et al. [FRR+10] extend the idea of private circuits to other restrictions on the type of allowed leakage, beyond just leaking some subset of the wire values. In particular, two possible distinct models are considered:

- In each invocation of the circuit, the attacker can leak an arbitrary **AC0** function[1] of *all* the wire values in the circuit, as long as the output length of this leakage function is bounded by some known upper bound. The size of the compiled circuit grows with the bound, but the *fraction* of information that can leak in this solution approaches 0.
- In each invocation of the circuit, the attacker learns a noisy reading of *every wire in the computation.* That is, for some fixed probability bound $p \in (1, \frac{1}{2}]$, the attacker independently gets the value of each wire with probability p and a flipped value with probability $(1-p)$. The size of the compiled circuit grows with $(1-p)^{-1}$, but the *fraction* of the total information in the circuit that leaks approaches 1 as p does.

[1]A function is in the class **AC0** if it can be represented by a polynomial-size, constant depth circuity composed only of NOT gates and unbounded fan-in AND and OR gates.

These models are interesting and useful since (1) **AC0** functions can reveal some *global* properties of the circuit, such as the *approximate hamming weight*, which may better capture realistic leakage, and (2) the amount of information that can leak in the second solution corresponds to almost the entire state of the circuit. Unfortunately, the solutions have a drawback as compared to [ISW03] in that they require some *simple leak-free gadgets*, which can sample a random value from a public distribution in a leak-free way. It is currently unknown how to get rid of such secure gadgets in these solutions. Moreover, the type of allowed leakage in these solutions is still significantly restricted.

2.2.5 Only Computation Leaks Information (OCLI)

Micali and Reyzin [MR04] introduce the axiom that "computation, and only computation, leaks information" (OCLI). That is, computation is somehow divided into steps and, during each computational step, only the portion of the secret state that is accessed in this step can (partially) leak. Any other part of the state which is not accessed will not leak during that step. The work of [MR04] does not specify the type/amount of leakage available during each step, but instead shows general reductions for building advanced leakage-resilient primitives from simpler leakage-resilient ones, for abstract classes of such leakage.

The work of Dziembowski and Pietrzak [DP08] proposes a concrete model of leakage under the OCLI axiom, where only the *amount* of information that leaks during each computational step is bounded by some (small) *leakage bound* ℓ. More concretely, in each computational step i, the attacker can choose any (polynomial time) function f with ℓ-bit output, and learn the value $f(\mathsf{state}_i)$ where state_i is the portion of the state that is accessed during step i. Under these conditions, the works of [DP08, Pie09] construct leakage-resilient *stream ciphers*, which output a random stream of bits. Even if an attacker can get leakage during various invocations of the stream cipher, the randomness produced by any invocation during which there isn't any leakage will look uniformly random. The tolerated leakage bound ℓ is *logarithmic in the security parameter* under standard assumptions, or possibly even a large fraction of the size of the accessed state under sufficiently strong exponential hardness assumptions. The work of [FKPR10] shows how to build signature schemes in this model, and [KP10] proposes a construction of a public-key encryption schemes (but only having heuristical security arguments). Several works [PSP$^+$08, SMY09, YSPY10] also consider different variants of the OCLI model, focusing on more practical approaches to modeling the limited side-channel attacks that are often used in real life (at the expense of theoretical generality) and on the symmetric-key primitives which are most commonly attacked.

The work of Goldwasser et al. [GKR08] on *one-time programs* can be seen as showing how to securely compile any computational task so that it can later be securely evaluated on a *single arbitrary input* (or some a-priori bounded number of inputs) under the OCLI axiom, even if the *entire* portion of the state that is accessed by the computation is leaked to the attacker. The works of Juma and Vahlis [JV10] and Goldwasser and Rothblum [GR10] show how to compile any computation into one which can be securely executed on *an arbitrary number of inputs* under the OCLI axiom, as long as the amount of information leaked during each computational step is bounded (and this is also shown to be necessary).[2] Unfortunately, both of these latter works also require some additional *leak-free gadgets*, with the ability to sample from a publicly known random distribution in a leak-free way. It remains an open problem whether it is possible to get rid of all leak-free gadgets in such constructions.

Limitations. Constructions of leakage-resilient primitives under the OCLI axiom already provide protection against a large and meaningful class of side-channel attacks. Still, it is debatable whether all natural leakage satisfies the OCLI axiom. For example, the *cold-boot attack* of Halderman et al. [HSH$^+$08] shows an example of a leakage attack where some remnants of the secret keys can remains in memory even after a device is powered down and the memory can be retrieved by an attacker. This occurs even if these areas of memory are not being accessed by any computation. Another disadvantage of relying on the OCLI axiom is that the definition of which portion of a state is "accessed by a computation" depends on the model of computation and on the specific hardware architecture. It may be hard to analyze this property on real systems where e.g. an entire page of memory is accessed even if a computation only needs to read a single bit on the page.

We note that the main technique for constructing schemes under the OCLI axiom, introduced by [DP08], is to split the secret state into a small number of components (often just 2), and have each computational step operate on only a single component at a time. The OCLI axiom is only used to argue that each component leaks *individually*, and the attacker cannot leak global functions of all components. As noted in e.g. [DP08, JV10], this can also be considered its own meaningful axiom, which may be reasonable even if the OCLI axiom does not hold. We will return to this idea when we discuss our results on secret-sharing in the continual-leakage model.

2.2.6 The Bounded-Leakage Model

The *bounded-leakage model* (also called the *memory-attack* model in some prior works) was introduced by Akavia et al. [AGV09] with the goal of removing the

[2]In all three works, security is defined via the simulation paradigm where leakage on the internals of the compiled computation can be simulated given only the "ideal" inputs and outputs of the original computation.

OCLI axiom. This model places absolutely *no* restrictions on which components leak or the type/complexity of the leakage, but only bounds the overall amount of observable leakage. That is, the attacker can learn arbitrary information about the entire internal secret state of a device, as long as the total *amount* of leaked information (measured in bits) is bounded by some leakage bound ℓ. This is formalized by allowing the attacker to specify an arbitrary poly-time *leakage function* $f : \{0,1\}^* \to \{0,1\}^\ell$ and learn the value $f(\mathsf{state})$, where state is the entire secret state of the device, including the secret key sk and all of the internal secret randomness that the scheme uses throughout its lifetime.[3]

It is relatively easy to see that one cannot perform general computations securely on such device. Indeed, it is impossible to even *store* a message with complete privacy on a leaky device, while ensuring that it remains efficiently retrievable from the device – a single bit of leakage of the internal state can always just reveal (say) the first bit of the message. However, it turns out to be possible to implement many specific cryptographic schemes on such leaky devices without sacrificing the security of the cryptosystem. In fact, *any* signature or public-key encryption scheme is secure in this model, as long as the amount of leaked information is *logarithmic* in the security parameter – otherwise it would be easy to break the scheme without any leakage just by guessing what the leakage should have been.[4] Prior work shows how to construct public-key (identity-based) encryption [AGV09, NS09, ADN$^+$10, CDRW10, BG10, HL11], signatures and key agreement schemes [ADW09a, KV09, DHLW10b, BSW11] and various related primitives in the bounded-leakage model, where the amount of leakage ℓ can be an arbitrarily large polynomial in the security parameter, and/or a $(1-\epsilon)$ fraction of the secret-key size, for any $\epsilon > 0$. That is, almost the entire secret key can leak without compromising security!

Benefits of General Leakage. The main benefit of the bounded-leakage model is its generality and simplicity, which make it highly applicable as well as easy to use. For example, if we analyze leakage on (just) the secret key of a deterministic cryptosystem, then it already implies security against leakage on the entire computation of the cryptosystem, including any intermediate values derived during the computation. This is because a poly-time leakage-function can just compute all of the intermediate values on its own given only the secret key.[5] Moreover, in contrast

[3]Simplified versions of this model, where state only includes the secret key but not all of the randomness, have also been considered.

[4]When defining security for encryption schemes, we must assume that the leakage only occurs *before* the attacker sees the challenge ciphertext, as otherwise a single bit of leakage could just output the first bit of the message by decrypting the ciphertext insdie the leakage function.

[5]For randomized cryptosystems, these intermediate values may also depend on the random coins of the cryptosystem and so we must analyze leakage on the secret key *and* the random coins.

to all of the other models we discussed so far, the bounded-leakage model is independent of the hardware architecture and the implementation. Leakage-resilience in this model is a property of only the algorithmic description of a cryptosystem and *not* its implementation. If a cryptosystem is shown to be resilient to bounded leakage, then *any* implementation of it on *any* hardware architecture is resilient.

Limitation of Bounded Leakage. The main criticism of the bounded-leakage model is that it may be unrealistic to assume that the *overall* amount of observed leakage is *bounded*. This may be a reasonable assumption about the leakage produced by a *single* side-channel measurement on a *single* operation of the device. But if an attacker can get prolonged access to a device, she may take many side-channel measurements over time, and therefore the overall amount of observed leakage will exceed any a-priori bound. Indeed, the vast majority of practical side-channel attacks work by fully recovering the entire secret key after sufficiently many measurements, and therefore will remain applicable even against "leakage-resilient" schemes in the bounded-leakage model.

In other words, the bounded-leakage model gives us the amazing *guarantee* that, as long as the leakage is "incomplete" (sufficiently bounded so as not to contain the entire key), then the cryptosystem remains secure – but it does not provide any *mechanism* for ensuring that continual side-channel leakage remains incomplete and does not recover the entire key. This is in contrast with many of the models we mentioned earlier (e.g. [ISW03, MR04, DP08]), which explicitly consider continual leakage (usually per invocation of the cryptosystem) and provide such a mechanism. In particular, these other models crucially rely on the idea of *evolving* the secret state of a cryptosystem over time, to ensure that the attacker cannot leak too much information about any single value. The bounded-leakage model provides no such mechanism.

Related Models and Applications

The generality of the bounded-leakage model has another benefit, in that this model appears to have many applications beyond side-channel attacks. In fact, there is much work (some preceding [AGV09]) which considers variants of the bounded-leakage model for various different applications.

Weak Randomness. The study of cryptography with *weakly random secrets*, considers a setting where the secret key can come from some arbitrary and unknown distribution, only guaranteed to have a sufficient level of (min-)entropy. Although, in the bounded-leakage model, the secret key is initially chosen honestly (usually uniformly random), it becomes *weakly random* if we condition on the leakage seen by the adversary. Therefore, the two settings are very related and results usually translate from one setting to another. Most of the prior work concerning weak secrets is specific to the *symmetric key setting* and much of this work is information-theoretic in nature. For example, the study of privacy-amplification [BBR88, Mau92, BBCM95, MW97, RW03, DKRS06, KR09, DW09] shows how two users who share a weakly random secret can use it to agree on a uniformly random key.

Virus Attacks. The bounded-leakage model was also considered earlier from the point of view of *virus attacks*, where a system is compromised by a virus leaking information to an external attacker. The main point of these work, which used the name *bounded retrieval model* [CLW06, Dzi06, CDD$^+$07, ADW09a, ADN$^+$10], is to construct *efficient* schemes with a *huge* secret key (say, many Gigabytes long) that maintain security even if a large amount of the secret key leaks. In other words, these works essentially consider the bounded-leakage model with an additional efficiency requirement – the operations of the cryptosystem must remain efficient, independently of how large of a secret key we may want to use. In particular, the cryptosystem cannot even read the entire key to perform its various operations such as signing or encrypting/decrypting.

Many of the ideas for the bounded-retrieval model evolved from the earlier *bounded-storage model* [Mau92, AR99, ADR02, Lu02, Vad04], where a huge random string is made publically available to all parties (honest and adversarial) but the parties have limited storage capacity and hence cannot store all of it. Indeed, the data stored by the attacker can be considered "leakage" on the huge random string.

Entropy Leakage and Auxiliary Inputs. Lastly, following [AGV09], several variants of the bounded-leakage model introduce different measures for the *amount of leakage* seen by the attacker. In all these variants the attacker can learn arbitrary poly-time computable functions of the secret state. The basic version of the model measures the amount of leakage by the output lengths of these functions. The work of [NS09] considers an *entropy* based approach, where the amount of leakage is defined as the amount of *entropy loss* that the leakage causes on the secret key. The main benefit is that some leakage attacks (e.g. power-analysis) produce a large output (e.g. the entire power-trace), but most of it is not useful and hence the entropy loss should still be small. Several variants of the entropy-based definition are also considered in [DHLW10a, BSW11]. The works of [DKL09, DGK$^+$10] consider a further generalization called the *auxiliary-input model*. In this version, the leakage amount is measured by the loss in the *computational hardness* of recovering the secret key given the leakage. This captures resilience against leakage that completely determines the secret key information theoretically, as long as it is hard to recover the secret key from the leakage computationally. Moreover, the auxiliary-input model may be useful if a secret-key is re-used for many different applications (signatures, encryption, etc.), each of whose security is analyzed individually and is known to not reveal the entire key.

Unfortunately, it is tricky to come up with formal definitions that match the intuition of the entropy and auxiliary-input based models. Current definitions are often confined to very specific schemes/scenarios and are hard to extend to others, lack robustness (for example, leaking ℓ bits twice in these models may not be the same as leaking 2ℓ bits), and are generally harder to work with. Therefore, in this thesis, we will always measure the amount of leakage just by its size, as in the basic version of the bounded-leakage model. Doing so may not be as restrictive as it first seems. The model already manages to capture attacks where large amounts of information about the secret key are leaked, as long as this leakage is efficiently compressible. Moreover, even if we do not know how to efficiently compress some form of leakage, resilience in this model will rule out any attacks that eventually only rely on some compressed version (e.g. attacks that use a long incompressible power trace but only extract some simple features from it). Lastly, it turns out that many of the results in the basic version of the bounded-leakage model can often also be translated to some of the more general variants.

2.3 Tampering and Active Physical Attacks

Leakage is one form of a *physical* attack, in which the attacker can *passively learn* more information than what is specified by the algorithmic description of a cryptosystem. Another form of physical attack is *tampering*, where the attacker can actively influence/modify the functioning of the cryptosystem, beyond what the algorithmic description allows. For example, by hitting a device with radiation, an attacker can introduce some random faults into the state and computation of a cryptosystem. This may already be enough to completely break an otherwise secure cryptosystem [BDL97, BS97], just by observing the outputs produced by the faulty computation.

Although we will not consider tampering in this thesis, we mention several prior works that consider formal models of resilience to tampering attacks. Firstly, the works of [GLM$^+$03, DPW10, CKM10] consider resilience against tampering on (just) the secret state of a device (but not on computation). These works propose a solution concept which is analogous to the use of AONTs in exposure-resilient cryptography – namely, the state of the device is encoded in some special form so that tampering with the encoding cannot meaningfully modify the encoded value. In particular, the work of [DPW10] introduces an abstract notion of "non-malleable codes" to capture this property. The work of Ishai et al. [IPSW06], on the other hand, considers a model where an attacker can tamper with *memory and computation* in a limited way. In particular, it is assumed that the attacker can modify the values of a small subset of the individual wires of a circuit performing an arbitrary computation. A similar setting with various interesting tradeoffs is considered in [FPV11]. Unfortunately, it seems much more difficult to come up with *general* models of tampering than it is for leakage.

Chapter 3

The Continual-Leakage Model and Overview of Results

In this thesis, we propose a new model of leakage resilience, called the *continual-leakage model* (CLM).[1] The main goal of the CLM is to get the "best of both worlds" – the generality of the bounded-leakage model together with a mechanism for evolving the secret state so as to defend against the reality of continual leakage. That is, we will allow the attacker to *continually* learn *arbitrary* information about the internal secret state of a cryptosystem, as long as the rate of leakage is bounded.

3.1 Overview of the CLM

Key updates. In addition to the usual functionality of a cryptographic primitive (e.g. signature, encryption,...), cryptosystems in the CLM come with an additional *randomized update procedure* for updating the secret key $sk' \leftarrow \mathsf{Update}(sk)$. This update process can be called an arbitrary number of times and should not have any visible effect on the functionality of the cryptosystem to the outside world. For example, signatures produced under each updated version of the secret key should always verify under the static verification key, using a fixed verification procedure. We leave it up to the implementation to decide when or how frequently to call the update procedure, and this may correspond to physical time (e.g. every second) or to the operations of the cryptosystem (e.g. after every signing operation). We will assume that when a device concludes running an update, it manages to perfectly erase/overwrite the old key and all of the random coins and intermediate values used during prior operations.[2] We define a *time period* to be the span of time that begins at the conclusion of one update and ends at the conclusion of the next one.

[1]This model was introduced concurrently by our publication [DHLW10a] and by the work of Brakerski et al. [BKKV10], which we discuss in further detail later.

[2]If these erasures are imperfect, we can just think of the unerased data as leakage.

Security in the CLM. The main security guarantee of the CLM is that the cryptosystem remains secure even if the attacker can leak up to ℓ bits of information about the (current) secret state of the cryptosystem *in each time period*, where ℓ is called the *leakage bound*. More precisely, in each time period, the attacker can adaptively specify an arbitrary (poly-time) *leakage function* $f : \{0,1\}^* \to \{0,1\}^\ell$, and learn the output $f(\mathsf{state})$, where state contains all information about the secret state of the cryptosystem in that time period. More precisely, in each period the value state includes the current secret key sk, all of the random coins (if any) used by the cryptographic operations during the time period, and the random coins used to update the secret key into the next time period. Therefore, this model captures continual leakage on the *entire state and computation* in each time period, including all the intermediate values created during a computation, since these are all poly-time computable functions of state. [3]

Remarks on the Model. The CLM captures the idea that we are only bounding the *rate* of leakage: if the device performs updates at fixed time intervals, then we are bounding the number or leaked bits per unit time, and if the device performs updates after each operation, then we are bounding the number of leaked bits per operation. We impose *no* a-priori bound on the overall number of leaked bits during the lifetime of the system.

The CLM is strictly more powerful than the bounded leakage model, which we can now think of as a restriction of the CLM to a single time period. Also, it is strictly more powerful than the OCLI model of [MR04, DP08], since, in each time period, the leakage is a global function of the *entire* state, and not only the portion of the state that is accessed by computation. In fact, essentially all of the prior models of partial leakage can be thought of as some *restrictions* of the CLM.

The main difficulty of constructing secure schemes in the CLM lies in showing how to perform updates so that leakage in different time periods cannot be meaningfully *combined* to reconstruct a full secret key that breaks the cryptosystem. In contrast to the bounded-leakage model, allowing even $\ell = 1$ bits of leakage is non-trivial here. Notice that the updates must *necessarily be randomized*; otherwise the leakage functions could always *pre-compute* the value of the secret key in some future time period i and leak it in full by learning one bit of it at a time in periods $1, 2, \ldots, i-1$.

We now give an overview of our results, showing how to construct secure prim-

[3]For simplicity, we will not consider leakage on the randomness of the key generation algorithm for our cryptosystems. We can justify this by assuming that this algorithm is run securely before the cryptosystem is "used in the field". Nevertheless, most of our results can be extended to allowing some small amount of leakage on the key generation algorithm as well (usually just logarithmic in the security parameter) and it remains an interesting open problem to design systems tolerating more such leakage.

itives that remain secure in the CLM. We call such primitives *continual-leakage resilient (CLR)*, or more precisely ℓ-CLR, if they allow ℓ bits of leakage in each time period.

3.2 CLR One-Way Relations and Signatures

We begin by considering a continual-leakage resilient version of *one-wayness*, which is the most basic security property in cryptography.

Defining Security. A *one-way relation (OWR)* consists of an efficient NP relation $R(pk, sk) = 1$ over public keys pk (the *statement*) and secret keys sk (the *witness*), along with an efficiently samplable distribution $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ over the relation R. The basic one-wayness security property only requires that, if we sample a random $(pk, \cdot) \leftarrow \mathsf{KeyGen}(1^\lambda)$, then no efficient attacker can find any matching sk^* such that $R(pk, sk^*) = 1$.[4]

To define a CLR OWR, we also introduce a *randomized update* $sk' \leftarrow \mathsf{Update}(sk)$ that updates the secret key while preserving the relation $R(pk, sk') = 1$. We consider a security game where we choose $(pk, sk_1) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and keep updating the secret key in each period i using randomness ω_i to get $sk_2 \leftarrow \mathsf{Update}(sk_1; \omega_1), sk_3 \leftarrow \mathsf{Update}(sk_2; \omega_2), \ldots$

We first consider a restricted security notion assuming *leak-free updates*, where the attacker only leaks on the keys but not on the update randomness. The attacker initially gets the public key pk. It then runs for arbitrary many time periods, where in each period i, it can adaptively choose a poly-time leakage function $f : \{0,1\}^* \to \{0,1\}^\ell$ and learn the answer $f(sk_i)$. We say that the scheme is secure if the attacker can never produce sk^* such that $R(pk, sk^*) = 1$. We also define the regular notion of CLR OWR, where the attacker gets to see $f(\mathsf{state}_i)$ for the full $\mathsf{state}_i = (sk_i, \omega_i)$ including the randomness of updates.

Results. We first show how to construct ℓ-CLR OWRs with leak-free updates, where the leakage bound ℓ can be made an arbitrarily large polynomial in the security parameter, and a $(\frac{1}{2} - \epsilon)$ fraction of the key size, for arbitrary $\epsilon > 0$. We do so by first specifying a general framework for reasoning about continual leakage, and then show hot to instantiate this framework using a new primitive called a *homomorphic NIZK*, which we finally instantiate concretely under the *linear assumption in prime-order bilinear groups* (see Section 4.3).

[4]Syntactically, a one-way relation is weaker than a one-way function since we do not require that $pk = f(sk)$ for some efficient function f but only that we can sample the tuple (pk, sk) jointly. However, if we disregard leakage, then one-way relations exist iff one-way functions do.

We then show that *any* ℓ-CLR-OWR with leak-free updates also satisfies *full* ℓ'-CLR OWR security for any $\ell' \leq \ell$, but the reduction suffers a security loss exponential in ℓ'. Therefore, we can get full ℓ' CLR security where the leakage ℓ' is logarithmic in the security parameter under standard assumptions, and would need exponential hardness assumptions to tolerate larger ℓ'. This essentially follows by "guessing" the value of the leakage on the updates. Lastly, we show how to generically convert CLR OWRs into CLR Signatures, relying on prior work for constructing leakage-resilient signatures in the bounded-leakage model [ADW09a, KV09, BSW11, JGS11]. We mention that, using CLR Signatures, it's possible to directly construct many other useful cryptographic primitives with CLR security, such as CLR identification schemes and key-agreement protocols.

3.3 CLR Public-Key Encryption

Definition. In addition to the usual functionality of *public-key encryption* (PKE), we again require the existence of an update procedure for updating the secret key. The security definition is similar to that of a one-way relation, and the attacker can adaptively leak up to ℓ bits of information on each secret key (and the randomness of the update) in each time period, for arbitrarily many time periods. After continually leaking such information, *semantic security* should be preserved and the attacker should be unable to distinguish encryptions of any two messages. Note that there is an important caveat here: the attacker cannot leak any more information on the secret key *after* seing the challenge ciphertext. This is a necessary restriction in this model: if the attacker could leak even one bits of information about the secret key that depends on the challenge ciphertext, she could leak (say) the first bit of the encrypted message, breaking semantic security.[5] Note that the relation between the public and secret keys of CLR PKE is necessarily also a CLR OWR. Therefore, constructions of CLR PKE will give us alternate constructions of fully secure CLR OWR (*with* leaky updates) where the leakage can be substantially larger than in the construction of the previous chapter. However, the construction is much less generic than the previous one.

Results. Concurrent to our work [DHLW10a] on CLR one-way relations and signatures, the work of Brakesrski et al. [BKKV10] gave a similar result for CLR

[5]As you may recall, this issue already comes up in the bounded leakage model. The work of [HL11] explores other security guarantees, weaker than semantic security, that PKE can satisfy in these models. Another approach toward getting rid of this restriction, suggested in [ADW09a], is the use *interactive* authenticated key-agreement protocols (e.g. based on leakage-resilient signatures) to securely communicate. This way, the message remains hidden as long as there is no leakage *during* the protocol execution in which it is sent, but continual leakage can occur before and after.

public-key encryption with leak-free updates, based on the linear assumption in bilinear groups. Following that, the work of Lewko et al. [LLW11] showed how to construct a CLR PKE with full security (including leakage on the updates) where the leakage bound ℓ can be an arbitrarily large polynomial in the security parameter (and a constant fraction of the key size). However, the resulting scheme and proof are fairly complicated and require strong assumptions on composite-order bilinear groups. In this thesis, we present a new scheme (from our corresponding work [DLWW11]), which is based on the ideas of [LLW11], but is vastly simpler and can be proven secure under the more standard linear assumption in prime-order bilinear groups. We also present the scheme in several stages, starting with simpler schemes that achieve weaker notions of security and progressively building up to the full scheme with full security. This presentation connects the main ideas behind several prior works on leakage-resilient encryption including [NS09], [BKKV10] and [LLW11]. Lastly, our final encryption scheme will be the starting point of our construction of CLR Secret Sharing, which we discuss next.

3.4 CLR Secret Sharing (Storage)

Storing Secrets on Leaky Devices. Finally, we ask a basic question of how to simply store a secret value (message) on continually leaky devices while preserving its secrecy. Unfortunately, in the bounded and continual leakage models, it is impossible to store a message secretly on a single leaky device from which it is efficiently retrievable, because a single leaked bit of the internal state of such device can reveal (say) the first bit of the message. There are two natural alternatives to overcoming this difficulty:

1. We can weaken the leakage model and restrict the attacker to only learning some *limited class of leakage function* of the internal state of the device. This class should capture realistic attacks but cannot be powerful enough to recover the stored secret, even though there is an efficient method for doing so.

2. We can consider a model where the secret is *shared* between two or more devices, each of which leaks *individually* in the continual leakage model. The attacker can continually learn *arbitrary* bounded-length functions of the internal state of each individual device, but *not* of the combined joint state of all the devices.

We will frame our discussion in terms of the *second* approach. However, this can also be naturally viewed as a concrete instantiation of the first approach, where we think of the state of a single device as divided into multiple *components*, and leakage is restricted to the limited class of functions that each depend on only a single component. This may be a natural and realistic class of leakage attacks if the components of the state are e.g. stored in different areas of memory and accessed separately by the device. In particular, this can be seen as a *strengthening* of the "only computation leaks information" (OCLI) model. In the OCLI model, the various components leak individually but only when accessed by a computation, while here they leak individually but all the time. We note that this strengthening was explicitly considered by prior works in the OCLI model, starting with Dziembowski and Pietrzak [DP08] in the case of stream ciphers. Although prior results in *various* models of continual leakage construct many basic and advanced cryptographic primitives, they do not address the simple question of storing a consistent value secretly on leaky devices. Indeed, they rely on the fact that one does not need to store a consistent secret key over time to e.g. decrypt, sign, or generate a random stream.

Defining CLR Sharing. More concretely, we will consider schemes for sharing a value between two devices, each of which is leaking information individually in the *continual leakage model.* We assume that each device has its own individual notion of *time periods*, but these notions can differ across devices and they need not be synchronized. At the end of each time period, a device updates its share using some local fresh randomness. This update is conducted individually, and the devices do *not* communicate during the update process. At any point in time, no matter how many updates occurred on each device, the shares of the devices can be efficiently combined to *reconstruct* the shared secret message.

For security, we allow the attacker to continually learn arbitrary (efficiently computable) functions of the internal state of *each* device .The attacker can choose the functions *adaptively* and can alternate leakage between the devices. The internal state of each device in each time period consists of the current version of its share and the randomness of the update process used to derive the next share. We only restrict the attacker to leaking at most ℓ bits of information from each device during each time period. For security, we require that the shared message remains *semantically secure* throughout the lifetime of the system. We call a scheme satisfying these criteria an *ℓ-continual-leakage-resilient sharing (ℓ-CLRS).*

Results. Our main result is to construct an ℓ-CLRS scheme between two devices, for any polynomial leakage-bound ℓ and where the share size scales linearly in ℓ, so that a *constant* fraction of each share can leak in each time period. The security of our scheme is based on the *linear* assumption in prime-order bilinear groups. In fact, the main tool of our construction is exactly our encryption scheme from the previous section; to share a message we simply encrypt it and make one share the secret key and the other share a ciphertext.Taking any of the recent results on CLR-PKE, we get a method for updating (just) the key share. We also get the guarantee that the message remains hidden even if the attacker continually leaks on the key share and later gets the ciphertext share in full. We then show an alternate way of creating ciphertexts, which allows us to define a natural procedure for updating them analogously to how secret keys are updated. Lastly, we prove a new and delicate security property, showing that even if an attacker interleaves leakage on the secret key and the ciphertext shares, the message remains hidden.

We also show that in contrast to standard secret sharing, ℓ-CLRS cannot be realized *information theoretically*, even for $\ell = 1$ bit of leakage.

Relation to Other Primitives. It is useful to compare CLRS schemes to other primitives from the literature. Most obviously, standard secret sharing schemes [Sha79] provide security when some subset of the shares are fully compromised while others are fully secure. In CLRS schemes, *all* shares leak and hence none are fully secure. The idea of updating shares to protect them against continual compromise was also considered in the context of *proactive secret sharing* [HJKY95]. However, the motivation there was to protect against a mobile adversary that corrupts different subsets of the shares in different time periods, while in our case *all* shares leak in all time periods. Another important connection is to the *leakage-resilient storage* scheme of [DDV10]. This gives an information-theoretic solution for sharing a secret securely on two leaky devices/components in the *bounded* leakage model, where the overall amount of leakage on each share is bounded. The work of [DF11] extends this information theoretic solution to the continual leakage model, but requires that devices have access to some correlated randomness generated in a leak-free way (e.g. using leak-free hardware) and update their shares interactively. In contrast, we do not assume any leak-free hardware. Also, our updates are performed individually, and we show that this comes at the necessary expense of having computational assumptions.

Related to the above model, prior (unpublished) work by [AGH10] was the first to propose the two processor distributed setting for public key decryption, where the systems secret state is shared by both processors, and is subject to continual memory leakage attacks, where the attacker is restricted to leak from each of the processors share of the secret state separately. Their ultimate goal was the security of the public key encryption scheme rather than the maintenance of a particular secret, which is addressed by an interactive secret state refresh protocol in their work.

Lastly, we mention the prior works [JV10, GR10], which consider general compilers for executing arbitrary computations privately on leaky devices. Both works provide solutions in variants of the "only computation leaks information model", but require some additional leak-free hardware. Implicitly, these works also address the question of storing a value secretly on leaky devices, since the state of the computation must be somehow stored consistently. However, the use of leak-free hardware in these solutions greatly simplifies the problem of storage and avoids virtually all of the challenges that we address in the current work. We believe that our work provides an important first step in the direction of building general compilers without any leak-free hardware, since the question of (just) securing storage must be addressed as a part of any solution to the larger question of securing computation. One elegant solution to the latter problem would be to design *secure computation protocols* where two (or more) continually leaky devices can securely compute functions of a secret value that is shared between them under our CLRS scheme, in such a way that leakage on the computation does not reveal anything more then the output value and some additional independent leakage on the input shares.

Chapter 4

Preliminaries

Notation and Conventions. If X is a probability distribution or a random variable then $x \leftarrow X$ denotes the process of sampling a value x at random according to X. If S is a set then $s \overset{\$}{\leftarrow} S$ denotes sampling s according to the *uniformly random* distribution over the set S. For a randomized algorithm or function f, we use the semicolon to make the randomness explicit i.e. $f(w; r)$ is the output of f with input w using randomness r. Otherwise, we let $f(x)$ denote a random variable for the output of $f(x; R)$ where R is uniformly random. As usual in computer science, all logarithms are *base 2* by convention so that $\log(x) \overset{\text{def}}{=} \log_2(x)$. We use *Kleene star* notation letting $\{0,1\}^* \overset{\text{def}}{=} \bigcup_{n\in\mathbb{N}}\{0,1\}^n$ denote all finite-length bit strings. For a bit string $s \in \{0,1\}^*$, we let $|s|$ denote the bit length of s.

Throughout the paper, we let λ denote the *security parameter* which determines the level of security that we are hoping to achieve. A function $\nu(\lambda)$ is called *negligible*, denoted $\nu(\lambda) = \mathsf{negl}(\lambda)$, if for every integer c there exists some integer N_c such that for all integers $\lambda \geq N_c$ we have $\nu(\lambda) \leq 1/\lambda^c$ (equivalently, $\nu(\lambda) = 1/\lambda^{\omega(1)}$). A function $\rho(\lambda)$ (usually associated with some probability) is called *overwhelming* if $\rho(\lambda) = 1 - \mathsf{negl}(\lambda)$. We will implicitly assume a uniform model of computation throughout the paper and identify *efficient* algorithms with Turing Machines that run in polynomial time in the security parameter λ. [1]

Computational Indistinguishability. Let $X = \{X_\lambda\}_{\lambda\in\mathbb{N}}$ and $Y = \{Y_\lambda\}_{\lambda\in\mathbb{N}}$ be two ensembles of random variables. We say that X, Y are (t, ϵ)-indistinguishable if for every distinguisher D that runs in time $t(\lambda)$ we have

$$|\Pr[D(X_\lambda) = 1] - \Pr[D(Y_\lambda) = 1]| \leq \frac{1}{2} + \epsilon(\lambda).$$

[1]This is just for concreteness. All results will also hold in the non-uniform setting under corresponding non-uniform hardness assumptions. In particular, all our reductions will be uniform.

We say that X, Y are *computationally indistinguishable*, denoted $X \overset{\text{comp}}{\approx} Y$, if for every polynomial $t(\cdot)$ there exists a negligible $\epsilon(\cdot)$ such that X, Y are (t, ϵ)-indistinguishable.

4.1 Statistical Distance, Entropy, Extractors

Statistical Indistinguishability. The *statistical distance* between two random variables X, Y is defined by

$$\mathbf{SD}(X, Y) = \frac{1}{2} \sum_x |\Pr[X = x] - \Pr[Y = x]| .$$

We write $X \overset{\text{stat}}{\approx}_\epsilon Y$ to denote $\mathbf{SD}(X, Y) \le \epsilon$ and just plain $X \overset{\text{stat}}{\approx} Y$ if the statistical distance is negligible in the security parameter. In the latter case, we say that X, Y are *statistically indistinguishable.*

Entropy and Extractors. The *min-entropy* of a random variable X is

$$\mathbf{H}_\infty(X) \overset{\text{def}}{=} -\log(\max_x \Pr[X = x]).$$

This is a standard notion of entropy used in cryptography, since it measures the worst-case predictability of X. We often find it useful to work with a generalized version of min-entropy, called *average conditional min-entropy*, defined by[DORS08] as

$$\widetilde{\mathbf{H}}_\infty(X|Z) \overset{\text{def}}{=} -\log\left(\mathop{\mathbb{E}}_{z \leftarrow Z}\left[\ \max_x \Pr[X = x|Z = z]\ \right]\right) = -\log\left(\mathop{\mathbb{E}}_{z \leftarrow Z}\left[2^{-\mathbf{H}_\infty(X|Z=z)}\right]\right).$$

This measures the best guess for X by an adversary that may observe an *average-case* correlated variable Z. That is, for all (inefficient) functions $\mathcal{A}$, we have $\Pr[\mathcal{A}(Z) = X] \le 2^{-\widetilde{\mathbf{H}}_\infty(X|Z)}$ and there exists some $\mathcal{A}$ which achieves equality. The following lemma says that conditioning on ℓ bits of information, the min-entropy drops by at most ℓ bits.

Lemma 4.1.1 ([DORS08])**.** *Let X, Y, Z be random variables where Y takes on values in a set of size at most 2^ℓ. Then $\widetilde{\mathbf{H}}_\infty(X|(Y, Z)) \ge \widetilde{\mathbf{H}}_\infty((X, Y)|Z) - \ell \ge \widetilde{\mathbf{H}}_\infty(X|Z) - \ell$ and, in particular, $\widetilde{\mathbf{H}}_\infty(X|Y) \ge \mathbf{H}_\infty(X) - \ell$.*

We now define the notion of an (average case) randomness extractor.

Definition 4.1.2 (Extractor [NZ96, DORS08])**.** *A* randomized *function* $\mathsf{Ext} : \mathcal{X} \to \mathcal{Y}$ *is an* (k, ϵ)-extractor *if for all r.v. X, Z such that X is distributed over $\mathcal{X}$ and $\widetilde{\mathbf{H}}_\infty(X|Z) \ge k$, we get $(Z, R, \mathsf{Ext}(X; R)) \overset{\text{stat}}{\approx}_\epsilon (Z, R, Y)$ where R is a random variable for the coins of* Ext *and Y is the uniform over $\mathcal{Y}$.*

Lemma 4.1.3 (Leftover-Hash Lemma [NZ96, DORS08])**.** *Assume that the family $\mathcal{H}$ of functions $h : \mathcal{X} \to \mathcal{Y}$, is a* universal hash family *so that for any $x \neq x' \in \mathcal{X}$ we have* $\Pr_{h \xleftarrow{\$} \mathcal{H}}[h(x) = h(x')] \leq 1/|\mathcal{Y}|$. *Then the randomized extractor* $\mathsf{Ext}(x; h) = h(x)$ *is a (k, ϵ)-extractor for any k, ϵ satisfying $k \geq \log(|\mathcal{Y}|) + 2\log(1/\epsilon)$.*

Two-Source Extractors. We will also rely on the notion of a two-source extractor [CG88]. Since we will only use the *inner product* extractor, we do not define the notion abstractly but simply present the concrete lemma that inner product is a good two-source extractor. For $\vec{x} = (x_1, \ldots, x_n) \in \mathbb{F}_q^n$ and $\vec{y} = (y_1, \ldots, y_n) \in \mathbb{F}_q^n$, the inner product is defined as $\langle \vec{x}, \vec{y} \rangle \overset{\text{def}}{=} \sum_{i=1}^n x_i y_i$. We will state the *average case* version of the lemma (analogous to the average case version of seeded extractors defined by [DORS08]).

Lemma 4.1.4 (Inner Product Two-Source Extractor)**.** *Let $\vec{X}, \vec{Y}, Z$ be correlated random variables, where $\vec{X}, \vec{Y}$ have their support in $\mathbb{F}_q^m$ and are independent conditioned on Z. Let U be uniform and independent over $\mathbb{F}_q$. Then*

$$\mathbf{SD}(\ (Z, \langle \vec{X}, \vec{Y} \rangle)\ ,\ (Z, U)\ ) \leq 2^{-s}$$

where $s \geq 1 + \frac{1}{2}(k_X + k_Y - (m+1)\log(q))$ for $k_X := \widetilde{\mathbf{H}}_\infty(\vec{X} \mid Z)$, $k_Y := \widetilde{\mathbf{H}}_\infty(\vec{Y} \mid Z)$.

The *worst-case* version of the lemma, where $Z = z$ is fixed, is proved in [CG88] (see [LLTT05] for a very simple proof giving the above parameters). We now prove the *average-case* version, where Z is arbitrary. The proof follows that of [DORS08] showing that leftover-hash is a good average-case extractor.

Proof. Let $(X_z, Y_z) = (X, Y | Z = z)$. Then

$$\begin{aligned}
&\mathbf{SD}(\ (Z, \langle X, Y \rangle)\ ,\ (Z, U)\ ) \\
&= \mathop{\mathbb{E}}_z [\mathbf{SD}(\ \langle X_z, Y_z \rangle\ ,\ U\ )] \\
&\leq \frac{1}{2} \mathop{\mathbb{E}}_z \left[\sqrt{2^{-(\mathbf{H}_\infty(X_z) + \mathbf{H}_\infty(Y_z))} q^{m+1}} \right] \\
&\leq \frac{1}{2} \sqrt{\mathop{\mathbb{E}}_z [2^{-(\mathbf{H}_\infty(X_z) + \mathbf{H}_\infty(Y_z))} q^{m+1}]} \\
&\leq \frac{1}{2} \sqrt{2^{-(\widetilde{\mathbf{H}}_\infty(X|Z) + \widetilde{\mathbf{H}}_\infty(Y|Z))} q^{m+1}}
\end{aligned}$$

where the first inequality follows from the *worst-case* version of the lemma and the second inequality is Jensen's inequality. This gives us the average case version of the lemma. □

4.2 Linear Algebra Notation

Linear Algebra. Let $\mathbb{F}$ be a field. We denote *row* vectors with $\vec{v} \in \mathbb{F}^n$. If $\vec{v}_1, \dots, \vec{v}_m \in \mathbb{F}^n$ are m vectors we let $\mathsf{span}(\vec{v}_1, \dots, \vec{v}_m) \subseteq \mathbb{F}^n$ denote the liner space spanned by these vectors. We let $\langle \vec{v}, \vec{w} \rangle \overset{\text{def}}{=} \vec{v} \cdot \vec{w}^\top$ be the *dot product* of $\vec{v}, \vec{w} \in \mathbb{F}_q^n$. If $A \in \mathbb{F}^{n \times m}$ is a $n \times m$ matrix of scalars, we let $\mathsf{colspan}(A), \mathsf{rowspan}(A)$ denote the subspaces spanned by the columns and rows of A respectively. If $\mathcal{V} \subseteq \mathbb{F}^n$ is a subspace, we let $\mathcal{V}^\perp$ denote the *orthogonal space* of $\mathcal{V}$, defined by $\mathcal{V}^\perp \overset{\text{def}}{=} \{ \vec{w} \in \mathbb{F}_q^n \mid \langle \vec{w}, \vec{v} \rangle = 0 \quad \forall \vec{v} \in \mathcal{V} \}$. We write $(\vec{v}_1, \dots, \vec{v}_m)^\perp$ as shorthand for $\mathsf{span}(\vec{v}_1, \dots, \vec{v}_m)^\perp$. We write $\mathcal{V} \perp \mathcal{W}$ if $\mathcal{V} \subseteq \mathcal{W}^\perp$ and therefore also $\mathcal{W} \subseteq \mathcal{V}^\perp$. We define the *kernel* of a matrix A to be $\mathsf{ker}(A) \overset{\text{def}}{=} \mathsf{rowspan}(A)^\perp$.

Matrix-in-the-Exponent Notation. Let $\mathbb{G}$ be a group of prime order q generated by an element $\mathbf{g} \in \mathbb{G}$. We write the group operation as multiplication and assume that it can be performed efficiently. and let $A \in \mathbb{F}_q^{n \times m}$ be a matrix. Then we use the notation $\mathbf{g}^A \in \mathbb{G}^{n \times m}$ to denote the matrix $\left(\mathbf{g}^A\right)_{i,j} \overset{\text{def}}{=} \mathbf{g}^{(A)_{i,j}}$ of group elements. Note that, given a matrix of group elements $\mathbf{g}^A \in \mathbb{G}^{n \times m}$ and a matrix $B \in \mathbb{F}_q^{m \times k}$ of "exponents", one can efficiently compute $\mathbf{g}^{AB}$. However, given $\mathbf{g}^A$ and $\mathbf{g}^B$ it is (generally) not feasible to efficiently compute $\mathbf{g}^{AB}$. On the other hand, assume $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are three groups of prime order q and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficient *bilinear map* satisfying $e(\mathbf{g}^a, \mathbf{h}^b) = e(\mathbf{g}, \mathbf{h})^{ab}$. Then, given $\mathbf{g}^A$ and $\mathbf{h}^B$ for generators $\mathbf{g} \in \mathbb{G}_1, \mathbf{h} \in \mathbb{G}_2$, one can efficiently compute $e(\mathbf{g}, \mathbf{h})^{AB}$ via $(e(\mathbf{g}, \mathbf{h})^{AB})_{i,j} = \prod_{k=1}^m e\left(\mathbf{g}^{A_{i,k}}, \mathbf{h}^{B_{k,j}}\right)$. We abuse notation and define $\mathbf{g}^A \mathbf{g}^B \overset{\text{def}}{=} \mathbf{g}^{A+B}$ and $e(\mathbf{g}^A, \mathbf{h}^B) \overset{\text{def}}{=} e(\mathbf{g}, \mathbf{h})^{AB}$ for any appropriately sized matrices A, B.

Random Matrices. For integers d, n, m with $1 \leq d \leq \min(n, m)$, we use the notation $\mathsf{Rk}_d(\mathbb{F}_q^{n \times m})$ to denote the set of all $n \times m$ matrices over $\mathbb{F}_q$ with rank d. When $\mathcal{W} \subseteq \mathbb{F}_q^m$ is a subspace, we also use the notation $\mathsf{Rk}_d(\mathbb{F}_q^{n \times m} \mid \mathsf{row} \in \mathcal{W})$ to denote the set of rank d matrices in $\mathbb{F}_q^{n \times m}$ whose rows come from the subspace $\mathcal{W}$. For $\mathcal{V} \subseteq \mathbb{F}_q^n$, we define $\mathsf{Rk}_d(\mathbb{F}_q^{n \times m} \mid \mathsf{col} \in \mathcal{V})$ analogously. We prove several simple properties of random matrices.

Lemma 4.2.1. *Assume q is super-polynomial in the security parameter.*
(I) *For $n \geq m$ the uniform distributions over $\mathsf{Rk}_m(\mathbb{F}_q^{n \times m})$ and $\mathbb{F}_q^{n \times m}$ are statistically indistinguishable.*
(II) *For $n, m \geq d$ and $\mathcal{W} \subseteq \mathbb{F}_q^m$ a subspace of dimension d, the uniform distributions over $\mathsf{Rk}_d(\mathbb{F}_q^{n \times m} \mid \mathsf{row} \in \mathcal{W})$ and $\mathcal{W}^n$ (seen as n rows) are statistically indistinguishable.*

Proof. Notice that (II) implies (I) with $d = m$ and $\mathcal{W} = \mathbb{F}_q^m$. The statistical distance between the uniform distributions over $\mathsf{Rk}_d(\mathbb{F}_q^{n \times m} \mid \mathsf{row} \in \mathcal{W})$ and $\mathcal{W}^n$ is just the probability that n samples from $\mathcal{W}$ span a sub-space of dimension $< d$. Think of choosing the rows from $\mathcal{W}$ one-by-one. Then the probability that the ith sample falls into the subspace of the previous ones is at most q^{i-1}/q^d. By union-bound, the probability of this happening in the first d samples is upper-bounded by $\sum_{i=1}^{d} q^{i-1}/q^d \le 2/q$. $\square$

Lemma 4.2.2. *Let d, n, m be integers with* $\min(n, m) \ge d \ge 1$.
(I) *The uniform distribution over* $\mathsf{Rk}_d(\mathbb{F}_q^{n\times m})$ *is equivalent to sampling* $C \xleftarrow{\$} \mathsf{Rk}_d(\mathbb{F}_q^{n\times d})$, $R \xleftarrow{\$} \mathsf{Rk}_d(\mathbb{F}_q^{d\times m})$ *and outputting* $A = CR$. *Notice*

$$\mathsf{colspan}(A) = \mathsf{colspan}(C) \quad , \quad \mathsf{rowspan}(A) = \mathsf{rowspan}(R).$$

(II) *If* $\mathcal{W} \subseteq \mathbb{F}_q^m$ *is a* fixed subspace *of dimension* $\ge d$, *the uniform distribution over* $\mathsf{Rk}_d(\mathbb{F}_q^{n\times m} \mid \mathsf{row} \in \mathcal{W})$ *is equivalent to sampling* $C \xleftarrow{\$} \mathsf{Rk}_d(\mathbb{F}_q^{n\times d})$, $R \xleftarrow{\$} \mathsf{Rk}_d(\mathbb{F}_q^{d\times m} \mid \mathsf{row} \in \mathcal{W})$ *and outputting* $A = CR$.
(III) *If* $\mathcal{W} \subseteq \mathbb{F}_q^m$ *a* uniformly random *subspace of a fixed dimension* $\ge d$, *the uniform distribution over* $\mathsf{Rk}_d(\mathbb{F}_q^{n\times m} \mid \mathsf{row} \in \mathcal{W})$ *is equivalent to the uniform distribution over* $\mathsf{Rk}_d(\mathbb{F}_q^{n\times m})$.

Proof. Notice that (II) implies (I) with $\mathcal{W} = \mathbb{F}_q^m$. For (II), it suffices to show that number of ways of writing $A = CR$ as a product of some C and R is the same for every A (for appropriate domains of A, C, R). In particular, it is equal the number of ways of choosing such R so that its rows form a basis of $\mathsf{rowspan}(A)$, which is $\prod_{i=0}^{d-1}(q^d - q^i)$. For (III), we notice that for every $A \in \mathsf{Rk}_d(\mathbb{F}_q^{n\times m})$ the number of spaces $\mathcal{W}$ (of any fixed dimension) such that $A \in \mathsf{Rk}_d(\mathbb{F}_q^{n\times m} \mid \mathsf{row} \in \mathcal{W})$ is just the number of spaces $\mathcal{W}$ that $\mathsf{rowspan}(A) \subseteq \mathcal{W}$ which is the same for every A. $\square$

4.3 Computational Hardness Assumptions

Our results will require us to rely on hardness assumptions in prime-order groups. We let such groups be defined via an abstract group generation algorithm $(\mathbb{G}, \mathbf{g}, q) \leftarrow \mathcal{G}(1^\lambda)$, where $\mathbb{G}$ is a (description of a) cyclic group of prime order q with generator $\mathbf{g}$. We assume that the group operation, denoted by multiplication, can be computed efficiently. We define several hardness assumptions on such groups.

Decisional Diffie-Hellman (DDH). The DDH assumption on $\mathcal{G}$ states that

$$(\mathbb{G}, \mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_0^r, \mathbf{g}_1^r) \overset{\text{comp}}{\approx} (\mathbb{G}, \mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_0^{r_0}, \mathbf{g}_1^{r_1})$$

where $(\mathbb{G}, \mathbf{g}, q) \leftarrow \mathcal{G}(1^\lambda)$, $\mathbf{g}_0, \mathbf{g}_1 \xleftarrow{\$} \mathbb{G}$, and $r, r_0, r_1 \xleftarrow{\$} \mathbb{F}_q$.

k-Linear. This assumption was proposed by [BBS04, HK07, Sha07] as a generalization/weakening of the DDH assumption. Let $k \geq 1$ be constant. The k-Linear assumption on $\mathcal{G}$ states that

$$(\mathbb{G}, \mathbf{g}_0, \mathbf{g}_1, \dots, \mathbf{g}_k, \mathbf{g}_1^{r_1}, \mathbf{g}_2^{r_2}, \dots, \mathbf{g}_K^{r_k}, \mathbf{g}_0^{\sum_{i=1}^k r_i}) \overset{\text{comp}}{\approx} (\mathbb{G}, \mathbf{g}_0, \mathbf{g}_1, \dots, \mathbf{g}_k, \mathbf{g}_1^{r_1}, \mathbf{g}_2^{r_2}, \dots, \mathbf{g}_K^{r_k}, \mathbf{g}_0^{r_0})$$

where $(\mathbb{G}, \mathbf{g}_0, q) \leftarrow \mathcal{G}(1^\lambda)$, $\mathbf{g}_1, \dots, \mathbf{g}_k \leftarrow \mathbb{G}$, and $r_0, r_1, \dots, r_k \leftarrow \mathbb{F}_q$. Notice that, when $k = 1$, the k-linear assumptions is exactly the DDH assumption. It turns out that, as k gets larger, the assumption becomes *weaker*. Therefore, even in groups in which the DDH assumption does not hold, the k-linear assumptions may still hold for $k \geq 2$.

k-linear (Matrix Form). It may be worth rewriting the k-linear assumption in matrix form. This equivalent form of the assumption states that

$$(\mathbb{G}, \mathbf{g}^X, \mathbf{g}^{\vec{r}X}) \overset{\text{comp}}{\approx} (\mathbb{G}, \mathbf{g}^X, \mathbf{g}^{\vec{u}})$$

where we sample $(\mathbb{G}, \mathbf{g}, q) \leftarrow \mathcal{G}(1^\lambda)$, $x_1, \dots, x_k \overset{\$}{\leftarrow} \mathbb{F}_q$, $\vec{r} \overset{\$}{\leftarrow} \mathbb{F}_q^k, \vec{u} \overset{\$}{\leftarrow} \mathbb{F}_q^{k+1}$ and set $X \in \mathbb{F}_q^{k \times (k+1)}$ be the matrix:

$$X := \begin{bmatrix} 1 & x_1 & 0 & \dots & 0 \\ 1 & 0 & x_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \dots & x_k \end{bmatrix}.$$

Rank Hiding. This assumption was introduced by [NS09] and shown to be implied by the *k-linear assumption*. The *k-rank hiding assumption* on $\mathcal{G}$ states that for any constants $k \leq i < j \leq \min\{m, n\}$ we cannot distinguish rank i and j matrices in the exponent of $\mathbf{g}$:

$$(\mathbb{G}, \mathbf{g}, \mathbf{g}^X) \overset{\text{comp}}{\approx} (\mathbb{G}, \mathbf{g}, \mathbf{g}^Y)$$

where $(\mathbb{G}, \mathbf{g}, q) \leftarrow \mathcal{G}(1^\lambda)$, $X \overset{\$}{\leftarrow} \mathsf{Rk}_i(\mathbb{F}_q^{n \times m})$, $Y \overset{\$}{\leftarrow} \mathsf{Rk}_j(\mathbb{F}_q^{n \times m})$.

To summarize, for any $k \geq 1$ and any group algorithm $\mathcal{G}$, the *k-linear assumption* is **equivalent** to the *matrix-form k-linear assumption*, it **implies** the *k-rank-hiding assumption*, and it **implies** the *$(k + 1)$-linear assumption*.

Bilinear Groups. Most of our results will rely on the above assumptions in *bilinear groups*. We let such groups be defined via an abstract *pairing generation algorithm*

$$(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e, \mathbf{g}, \mathbf{h}) \leftarrow \mathcal{G}_{pair}(1^\lambda)$$

where $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ are (descriptions of) cyclic group of prime order q with generators $\mathbf{g} \in \mathbb{G}_1, \mathbf{h} \in \mathbb{G}_2$, and e is a description of a *bilinear map* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. We require two properties:

Efficiency: The bilinear map $e(\cdot, \cdot)$ and the multiplication operation in all three groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ can be computed efficiently.

Non-Degenerate: The bilinear map is *non-degenerate* so that the element $e(\mathbf{g}, \mathbf{h})$ is a generator of $\mathbb{G}_T$.

We say that the pairing is *symmetric* if $\mathbb{G}_1 = \mathbb{G}_2$, $\mathbf{g} = \mathbf{h}$ and *asymmetric* otherwise.

We can define analogous versions of the DDH, k-linear and k-rank hiding assumptions in such groups. In all cases the attacker is also given the full description of the bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, q, e, \mathbf{g}, \mathbf{h})$ and we will assume that the corresponding hardness property holds in both groups $\mathbb{G}_1, \mathbb{G}_2$.

It turns out that the 1-linear (DDH) and 1-rank-hiding assumptions are *false* for symmetric pairings where $\mathbb{G}_1 = \mathbb{G}_2$. However, it is often reasonable to assume DDH holds in some *asymmetric pairings*, and this is also called the *external Diffie-Hellman assumption SXDH* [Sco02, BBS04, GR04, Ver04]. Since the SXDH assumption is fairly strong, it is sometimes preferable to rely on k-linear (or rank-hiding) assumptions for $k \geq 2$. The $(k = 2)$*-linear assumption*, also called *decisional linear*, is commonly believed to hold in symmetric and asymmetric pairings.

4.4 Public-Key Encryption

Recall that a public-key encryption scheme consists of three algorithms $\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}$ with the following syntax:

- $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$: Outputs a public/secret key pair.
  The public key defines some efficiently samplable message space $\mathcal{M}$.
- $c \leftarrow \mathsf{Encrypt}_{pk}(m)$: Given a message $m \in \mathcal{M}$ and a public key pk, outputs a ciphertext c.
- $m' \leftarrow \mathsf{Decrypt}_{sk}(c)$: Given a ciphertext c and a secret key sk, outputs a message $m' \in \mathcal{M}$ or a value $\perp$.

Often, instead of just defining a single algorithm KeyGen, we break it up into two separate procedures: $\mathsf{ParamGen}, \mathsf{KeyGen}$ with the syntax:

- $\mathsf{prms} \leftarrow \mathsf{ParamGen}(1^\lambda)$ generates *public parameters* prms which define some message space $\mathcal{M}$.
- $(pk, sk) \leftarrow \mathsf{KeyGen}(\mathsf{prms})$ generates the public/secret key.

The advantage of this definition, over just thinking of prms as part of the public key pk, is that the prms can be reused multiple times by different users and even different schemes. For example, the public parameters could include the description of a DDH group which can be reused many times. We will often just switch between these two syntactic definitions. That is, when discussing security, we can always just use the first definition but, when extra functionality requires it, we may specify a distinction between the public parameters prms and the public key pk.

The scheme should satisfy correctness and security defined as follows.

Perfect Correctness: For all $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ with pk defining a message space $\mathcal{M}$, for all $m \in \mathcal{M}$, for all $c \leftarrow \mathsf{Encrypt}_{pk}(m)$, we have $\mathsf{Decrypt}_{sk}(c) = m$.

Security: We will define two distinct notions of security. First, we start with a weak notion which we call *one-way security*, where we only require that an encryption of a random message makes the message hard to recover.

Definition 4.4.1 (One-Way Security)**.** *An encryption scheme* $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$*, is one-way secure if, for any PPT adversary* $\mathcal{A}$ *we have:*

$$\Pr\left[m^* = m \;\middle|\; \begin{array}{c} (pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda) m \overset{\$}{\leftarrow} \mathcal{M}_{pk} \\ c \leftarrow \mathsf{Encrypt}_{pk}(m), m^* \leftarrow \mathcal{A}(pk, c) \end{array} \right] \leq \mathsf{negl}(\lambda)$$

where the public-key pk defines the message-space $\mathcal{M}_{pk}$.

The stronger and more standard notion of security for encryption is *semantic security* where the attacker cannot distinguish between the encryptions of any two messages.

Definition 4.4.2 (Semantic Security)**.** *An encryption scheme* $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$*, is* semantically secure*, if for any PPT adversary* $\mathcal{A}$ *we have* $|\Pr[b^* = b] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$ *in the following game:*

- *Challenger samples $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and gives pk to $\mathcal{A}$.*
  Let $\mathcal{M}$ be the message space defined by pk.
- *$\mathcal{A}$ chooses two values $m_0, m_1 \in \mathcal{M}$.*
- *Challenger samples $b \overset{\$}{\leftarrow} \{0, 1\}$, $c \leftarrow \mathsf{Encrypt}_{pk}(m_b)$ and gives c to $\mathcal{A}$.*
- *$\mathcal{A}$ outputs a bit b^*.*

It is easy to see that the standard notion of semantic-security implies one-way security if the size of the message-space $\mathcal{M}_{pk}$ is super-polynomial in the security parameter.

4.5 Non-Interactive Zero Knowledge (NIZK)

Let $R \subseteq \mathcal{Y} \times \mathcal{X}$ be an NP relation on pairs (y, x) with corresponding language $L_R = \{y \mid \exists x \text{ s.t. } (y, x) \in R\}$. A *non-interactive zero-knowledge (NIZK) argument* for a relation R consists of PPT algorithms $\texttt{Setup}, \texttt{Prov}, \texttt{Ver}$ with syntax:

- $\text{CRS} \leftarrow \texttt{Setup}(1^\lambda)$: Creates a common reference string (CRS).
- $\pi \leftarrow \texttt{Prov}_{\text{CRS}}(y, x)$: Creates an argument that "$y \in L_R$" using witness x.
- $0/1 \leftarrow \texttt{Ver}_{\text{CRS}}(y, \pi)$: Verifies whether or not the argument π is correct.

For security, we also require the PPT algorithms $\texttt{SetupSim}, \texttt{Sim}$ with the syntax:

- $(\text{CRS}, \text{TK}) \leftarrow \texttt{SetupSim}(1^\lambda)$: Creates a simulated CRS with a trapdoor TK.
- $\pi \leftarrow \texttt{Sim}_{\text{CRS}}(y, \text{TK})$: Creates a simulated argument for $y \in \mathcal{Y}$.

For the sake of clarity, we write $\texttt{Prov}, \texttt{Ver}, \texttt{Sim}$ without the CRS in the subscript when the CRS can be inferred from the context.

Definition 4.5.1. *We say that* $\Pi = (\texttt{Setup}, \texttt{Prov}, \texttt{Ver}, \texttt{SetupSim}, \texttt{Sim})$ *is a NIZK argument system for the relation R if the following three properties hold.*

Completeness: *For any* $(y, x) \in R$, *if* $\text{CRS} \leftarrow \texttt{Setup}(1^\lambda)$, $\pi \leftarrow \texttt{Prov}(y, x)$, *then* $\texttt{Ver}(y, \pi) = 1$.

Soundness: *For any PPT adversary* $\mathcal{A}$,

$$\Pr\left[\begin{array}{c} \texttt{Ver}(y, \pi^*) = 1 \\ y \notin L_R \end{array} \middle| \begin{array}{c} \text{CRS}, \leftarrow \texttt{Setup}(1^\lambda) \\ (y, \pi^*) \leftarrow \mathcal{A}(\text{CRS}) \end{array}\right] \le \mathsf{negl}(\lambda).$$

Composable Zero-Knowledge: *We require two properties.*
(1) $\text{CRS} \overset{\text{comp}}{\approx} \text{CRS}'$ *where* $\text{CRS} \leftarrow \texttt{Setup}(1^\lambda)$ *and* $(\text{CRS}', \text{TK}) \leftarrow \texttt{SetupSim}(1^\lambda)$.
(2) For any PPT adversary $\mathcal{A}$ *we have* $\left|\Pr[\mathcal{A} \text{ wins }] - \frac{1}{2}\right| \le \mathsf{negl}(\lambda)$ *in the following game:*

- *Challenger samples* $(\text{CRS}, \text{TK}) \leftarrow \texttt{SetupSim}(1^\lambda)$, *gives* (CRS, TK) *to* $\mathcal{A}$.
- $\mathcal{A}$ *chooses* $(y, x) \in R$ *and gives these to the challenger.*
- *Challenger samples* $\pi_0 \leftarrow \texttt{Prov}(y, x), \pi_1 \leftarrow \texttt{Sim}(y, \text{TK}), b \leftarrow \{0, 1\}$, *gives* π_b *to* $\mathcal{A}$.

- $\mathcal{A}$ *outputs a bit* $\tilde{b}$*, and* wins *if* $\tilde{b} = b$.

We note that we include *composability* in our default notion of NIZK. By insisting that real and simulated proofs look the same even given TK, it is easy to see that they look the same even if an attacker can see many other simulated proofs. The reason that we define two algorithms `Setup` and `SetupSim`, instead of just always using `SetupSim`, is that soundness may not hold when the CRS is generated via `SetupSim`. In particular, there may be a strategy for picking statements depending on the CRS and generating valid arguments, such that the statements are false if the CRS is simulated but true otherwise. Therefore the attacker cannot distinguish which type of CRS is being used, but may manage to prove false statements only when the CRS is simulated.

Public Parameters. As with encryption, we can also talk about NIZK with *public parameters* by defining three algorithms:

- $\mathsf{prms} \leftarrow \mathsf{ParamGen}(1^\lambda)$: generates the public parameters.
- $\textsc{crs} \leftarrow \mathtt{Setup}(\mathsf{prms})$: creates the CRS.
- $(\textsc{crs}, \textsc{tk}) \leftarrow \mathtt{SetupSim}(\mathsf{prms})$ generates a simulated CRS with a trapdoor.

Again, the main advantage over just thinking about prms as part of the CRS is that the above definition makes it clear that prms can be reused by different schemes. We will alternate between the above two definitions freely. As with encryption, when discussing security we just think of the prms as part of the CRS (this is without loss of generality) to simplify notation. But we will explicitly talk about public parameters when the additional functionality requires it.

Chapter 5

CLR One-Way Relations, Signatures

5.1 Modeling Leakage.

We model leakage attacks (also called memory attacks in prior work) against a cryptosystem with some secret state state via the notion of a *leakage oracle*. The leakage oracle allows the attacker to learn information about the state of the cryptosystem by submitting arbitrary "questions" about the state to the oracle. Without loss of generality, we assume each question provides the attacker with one bit of information, but the attacker can ask many such questions. We model the attacker's questions as predicates $h : \{0,1\}^{|\mathsf{state}|} \to \{0,1\}$ and the leakage oracle answers each question by providing the attacker with the corresponding output $h(\mathsf{state})$. In our definitions of leakage-resilient primitives, the adversary can adaptively query the leakage oracle to learn information about the secret state during an attack, but we will restrict the number of allowed queries (i.e. the amount of leaked information) during various stages of the attack.

Definition 5.1.1 (Leakage Oracle)**.** *A leakage oracle $\mathcal{O}_{\mathsf{state}}(\cdot)$ is parameterized by the secret state state of publicly known length $m := |\mathsf{state}|$. A query to the oracle consists of a circuit computing some predicate $h : \{0,1\}^m \to \{0,1\}$. The oracle responds to the query by outputting the value $h(\mathsf{state})$.*

Note that a polynomial time attacker with access to a leakage oracle $\mathcal{O}_{\mathsf{state}}(\cdot)$ can only query the oracle on circuits of polynomial size (since it must write down the entire circuit explicitly) and hence the queries can all be evaluated in polynomial time as well.

5.2 Defining CLR-OWR

5.2.1 One-Way Relations

A one-way relation (OWR) consists of two PPT algorithms: a key-generation algorithm $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$, and a verification algorithm $b = \mathtt{Ver}(pk, sk)$ which outputs a bit $b = 1$ to indicate that a secret-key sk *is valid for* the public-key pk, and $b = 0$ otherwise. Implicitly, this gives us the relation $R = \{(pk, sk) : \mathtt{Ver}(pk, sk) = 1\}$.

Definition 5.2.1 (One-Way Relation (OWR)). *We say that* $(\mathsf{KeyGen}, \mathtt{Ver})$ *is a* one-way relation *if it satisfies:*

Correctness: *If* $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$*, then* $\mathtt{Ver}(pk, sk) = 1$.
Security: *For any PPT attacker* $\mathcal{A}$*, we have*

$$\Pr\left[\mathtt{Ver}(pk, sk^*) = 1 \;\middle|\; \begin{array}{c} (pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ sk^* \leftarrow \mathcal{A}(pk) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

A one-way relation generalizes the concept of a one-way function (OWF). Of course, we can always set sk to include all of the randomness of the KeyGen algorithm, so that $pk = \mathsf{KeyGen}(sk)$ *is* a OWF. However, when defining leakage-resilient one-wayness (which we do next), this equivalence might no longer hold – by putting more information into the secret key we would also have to give the adversary more information during key-leakage attacks. Therefore, we consider OWRs, rather than OWFs, as the basic cryptographic primitive for leakage-resilience.

5.2.2 OWRs in the Bounded-Leakage Model

We first define security of OWRs against *bounded leakage*, where the overall amount of leakage is bounded by ℓ bits. Although this definition will not be required for any of the results in the paper, it is here as a warm-up exercise used to to build up intuition towards our definition in the continual leakage model. We define an ℓ-leakage-resilient OWR (ℓ-LR-OWR) by modifying Definition 5.2.2 so that the adversary can learn up to ℓ bits of information about the secret sk after seing the public value pk and before outputting the forgery sk^*.

Definition 5.2.2 (ℓ-LR-OWR). *We say that* $(\mathsf{KeyGen}, \mathtt{Ver})$ *is an* ℓ*-leakage-resilient OWR if it satisfies the correctness property of OWR and the following leakage-resilient security property. For any polynomial* $p(\cdot)$ *and any PPT attacker* $\mathcal{A}$*, we have* $\Pr[\mathcal{A}$ *wins* $] \leq \mathsf{negl}(\lambda)$ *in the following game:*

- *Challenger chooses* $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$*, gives* pk *to* $\mathcal{A}$*, sets* $\mathsf{state} = sk$.

- *Attacker $\mathcal{A}$ can make up to ℓ queries to the oracle $\mathcal{O}_{\mathsf{state}}(\cdot)$ and* wins *if it outputs sk^* s.t.* $\mathtt{Ver}(pk, sk^*) = 1$.

We now show that any *second-preimage resistant (SPR)* function F is also an ℓ-LR-OWR, where ℓ is roughly the number of bits by which F shrinks its input. This theorem was first shown formally in our works [ADW09b, DHLW10a] but was also implicit in [ADW09a, KV09].

Definition 5.2.3 (Second Pre-Image Resistant Functions (SPR))**.** *Let*

$$\left\{F \ : \ \{0,1\}^{n(\lambda)} \to \{0,1\}^{m(\lambda)}\right\}_{\lambda \in \mathbb{N}}$$

be an ensemble of functions where $n(\cdot), m(\cdot)$ are some polynomials. We say that F is second-preimage resistant *(SPR) if $F(x)$ is efficient to compute, and for any PPT algorithm $\mathcal{A}$,*

$$\Pr[x' \neq x \wedge F(x') = F(x) \mid x \xleftarrow{\$} \{0,1\}^{n(\lambda)},\ x' \leftarrow \mathcal{A}(x)] \leq \mathsf{negl}(\lambda)$$

Theorem 5.2.4. *Assume that F is SPR. Let* $(\mathsf{KeyGen}, \mathtt{Ver})$ *be a relation where* $\mathsf{KeyGen}(1^\lambda)$ *samples $sk \xleftarrow{\$} \{0,1\}^{n(\lambda)}$ and sets $pk := F(sk)$, and* $\mathtt{Ver}(sk, pk)$ *outputs 1 iff $pk = F(sk)$. Then this relation is an ℓ-LR-OWR for any $\ell(\cdot)$ such that $\ell(\lambda) = n(\lambda) - m(\lambda) - \omega(\log(\lambda))$.*

Proof. Assume otherwise, that there exists an attacker $\mathcal{A}$ that break ℓ-LW-OWR security. That is, given $pk = F(sk)$ and ℓ bits of leakage on sk, $\mathcal{A}$ and outputs sk^* such that $F(sk^*) = pk$. Assume $\mathcal{A}$ succeeds with some non-negligible probability $\epsilon(\lambda)$. Then we construct an attacker $\mathcal{B}$ that breaks SPR security with non-negligible probability. The attacker $\mathcal{B}$ simply gets $sk \xleftarrow{\$} \{0,1\}^{n(\lambda)}$, computes $pk = F(sk)$ and runs the attacker $\mathcal{A}$ with pk. It simulates the leakage oracle and answers the leakage queries made by $\mathcal{A}$ honestly using its knowledge of sk. The attacker $\mathcal{B}$ *wins* as long as both of the following two events occur: (1) the attacker $\mathcal{A}$ *wins* and outputs a key sk^* such that $F(sk^*) = pk$, and (2) the keys are different $sk^* \neq sk$. Let us denote the two events by E_1, E_2 respectively. Then

$$\Pr[E_1 \wedge E_2] \geq \Pr[E_1] - \Pr[\neg E_2] \geq \epsilon(\lambda) - \Pr[\neg E_2] \geq \epsilon(\lambda) - \mathsf{negl}(\lambda). \qquad (5.1)$$

where, for the last inequality, we need to show $\Pr[\neg E_2] = \mathsf{negl}(\lambda)$. But this we can show information theoretically, relying on the fact that $\mathcal{A}$ just doesn't have enough information about sk to output it exactly. Let *leak* denote the leakage observed by $\mathcal{A}$ (as a random variable over its randomness and that of sk). Then

$$\widetilde{\mathbf{H}}_\infty(sk \mid pk, leak) \geq n(\lambda) - m(\lambda) - \ell(\lambda) \geq \omega(\log(\lambda))$$

where the first inequality follows from Lemma 4.1.1. Therefore the probability that $\mathcal{A}$ outputs sk after observing $pk, leak$ is bounded by $2^{-\omega(\log \lambda)} = \mathsf{negl}(\lambda)$ as we wanted to show. $\square$

Using the result of Rompel [Rom90] which shows how to construct SPR functions with arbitrarily large stretch from any one-way function, we get the following corollary.

Corollary 5.2.5. *Assuming the existence of one-way functions, there exists an ℓ-LR-OWR for any polynomial $\ell(\lambda)$. Moreover, for any constant $\epsilon > 0$, there exists an ℓ-LR-OWR where the ratio of leakage ℓ to the key size $|sk|$ is $(1 - \epsilon)$.*

5.2.3 OWRs in the Continual Leakage Model

A *continuous-leakage-resilient (CLR) one-way relation (OWR)* consists of the algorithms KeyGen and $\mathtt{Ver}$ as before, but also includes an *update* algorithm:

$sk' \leftarrow \mathsf{Update}_{pk}(sk)$ **:** Update the secret key sk to sk'. We will omit the subscript pk, when clear from context.

For convenience, we also implicitly define the algorithm that performs $i \geq 0$ consecutive updates via:

$sk' \leftarrow \mathsf{Update}^{i}(sk)$ **:** Let $sk_0 = sk$, $sk_1 \leftarrow \mathsf{Update}(sk_0), \ldots sk_i \leftarrow \mathsf{Update}(sk_{i-1})$. Output $sk' = sk_i$.

On a high level, a OWR is continuous-leakage-resilient if an adversary can observe ℓ bits of leakage on each of arbitrarily many secret keys and the randomness of the update, and still be unable to produce a valid secret key herself.

Definition 5.2.6 (CLR-OWR)**.** *We say that a scheme* $(\mathsf{KeyGen}, \mathsf{Update}, \mathtt{Ver})$ *is an* ℓ-continuous-leakage-resilient (ℓ-CLR) one-way relation *if it satisfies the following correctness and security properties:*

Correctness: *For any polynomial $i = i(\lambda) \geq 0$, if we sample $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $sk_i \leftarrow \mathsf{Update}^{i}(sk)$, then $\mathtt{Ver}(pk, sk_i) = 1$.*

Security: *For any PPT $\mathcal{A}$, we have $\Pr[\mathcal{A}$ wins $] \leq \mathsf{negl}(\lambda)$ in the following game:*

- *Challenger chooses $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and gives pk to $\mathcal{A}$. It samples randomness ω for the first update and sets $\mathsf{state} = (sk, \omega)$. It sets $L = 0$.*
- *$\mathcal{A}$ can adaptively make the following queries to the challenger:*
  - *If $L < \ell$ then $\mathcal{A}$ can query the leakage-oracle $\mathcal{O}_{\mathsf{state}}(\cdot)$ and get the appropriate response. The challenger sets $L := L + 1$.*
  - *$\mathcal{A}$ can make an update query. The challenger parses $\mathsf{state} = (sk, \omega)$, sets $sk' := \mathsf{Update}(sk; \omega)$ and samples fresh randomness ω'. It updates $\mathsf{state} := (sk', \omega')$ and sets $L := 0$.*
- $\mathcal{A}$ wins *if at any point it produces a value sk^* such that $\mathtt{Ver}(pk, sk^*) = 1$.*

We also define a weaker variant of the above definition, called ***security with leak free updates****, where the value* state only *includes the current secret key sk and* not *the randomness ω of the next update.*

Remarks on Definition. By including the randomness ω of the next update in the state that the attacker can leak on, we are modeling leakage that occurs *during* the update process itself. That is, consider a device that initially stores the value *sk* generated by KeyGen, and then continually updates it to a new key every time period (say, every couple of seconds). Then, at any point in time when an update is not running, the secret state of the device is fully described by the secret key *sk*. However, during the time that the update is running, the secret state of the device is fully described by $\mathsf{state} = (sk, \omega)$ which includes the randomness ω used to run the update. Therefore the weaker notion of security with leak-free updates models the case where the device is only leaking in between updates, but not during the update process itself. This is already an interesting notion of security which is (highly) non-trivial to achieve.

Why Prior LR Techniques Fail for CLR. All of the prior works on bounded-leakage memory-leakage attacks crucially relied on an entropy argument: given the leakage and the public-key, the secret-key *sk* still had some entropy left. For example, this was the main step of our argument for the leakage-resilience of SPR functions. However, it is unclear how to translate this type of argument to the setting of *continuous leakage-resilience*, where the total amount of information seen by the adversary is unbounded.

Organization. We begin by showing a novel strategy for reasoning about continuous leakage in the next section (Section 5.3). We then instantiate this strategy based on generic components in Section 5.4 and finally instantiate these components under the *linear* assumption in Section 5.5. In all these sections, we only focus on the weaker security notion of **CLR-OWR with leak-free updates**. Then, in Section 5.6, we show that this notion already generically implies the stronger security notion, where the randomness of the updates may leak as well, albeit with a security-loss exponential in the amount of leakage. Lastly, in Section 5.7, we discuss how to use CLR-OWR to construct CLR signatures.

5.3 Construction (Part I): Continuous Leakage from Bounded Leakage

We now define a new primitive, called a *leakage-indistinguishable re-randomizable relation (LIRR)*, and show that it can be used to construct a secure CLR-OWR with leak-free updates. Although the definition of the new primitive is fairly complex with several security requirements, its main advantage is that it reduces the problem of *continuous*-leakage resilience for OWR to a simpler *bounded-leakage-resilience* property, which we can reason about more easily.

A LIRR allows one to sample two types of secret-keys: "good" keys and "bad" keys. Both types of keys look valid and are acceptable by the verification procedure, but they are produced in very different ways. In fact, given the ability to produce good keys, it is hard to produce any bad key and vice-versa. On the other hand, even though the two types of keys are very different, they are hard to distinguish from each other. More precisely, given the ability to produce both types of keys, and ℓ bits of leakage on a "challenge" key of an unknown type (good or bad), it is hard to come up with a new key of the same type.

More formally, a LIRR consists of PPT algorithms ($\texttt{Setup}$, $\texttt{SampG}$, $\texttt{SampB}$, Update, $\texttt{Ver}$, $\texttt{isGood}$) with the following syntax:

- $(pk, \mathsf{sam}_G, \mathsf{sam}_B, dk) \leftarrow \texttt{Setup}(1^\lambda)$: Outputs a *public-key* pk, "good" and "bad" sampling-keys $\mathsf{sam}_G, \mathsf{sam}_B$, and a *distinguishing-key* dk.
- $sk_G \leftarrow \texttt{SampG}_{pk}(\mathsf{sam}_G)$, $sk_B \leftarrow \texttt{SampB}_{pk}(\mathsf{sam}_B)$: These algorithms sample good/bad secret-keys using good/bad sampling keys respectively. We omit the subscript pk when clear from context.
- $b = \texttt{isGood}(pk, sk, dk)$: Uses dk to distinguish good/bad secret-keys sk.
- $sk' \leftarrow \mathsf{Update}_{pk}(sk)$, $b = \texttt{Ver}(pk, sk)$. These have the same syntax as in the definition of CLR-OWR.

Definition 5.3.1 (LIRR). *We say that the scheme* ($\texttt{Setup}$, $\texttt{SampG}$, $\texttt{SampB}$, Update, $\texttt{Ver}$, $\texttt{isGood}$) *is an* ℓ-leakage-indistinguishable re-randomizable relation *(*ℓ-LIRR*) if it satisfies the following properties:*

Correctness: *If* $(pk, \mathsf{sam}_G, \mathsf{sam}_B, dk) \leftarrow \texttt{Setup}(1^\lambda)$, $sk_G \leftarrow \texttt{SampG}(\mathsf{sam}_G)$, $sk_B \leftarrow \texttt{SampB}(\mathsf{sam}_B)$ *then w.o.p. the following holds:*

$$\texttt{Ver}(pk, sk_G) = 1,\ \texttt{isGood}(pk, sk_G, dk) = 1$$

$$\texttt{Ver}(pk, sk_B) = 1,\ \texttt{isGood}(pk, sk_B, dk) = 0$$

Re-randomization: *We require that updates* re-randomize *the good keys:*

$$(pk, \mathsf{sam}_G, sk_0, sk_1) \overset{\text{comp}}{\approx} (pk, \mathsf{sam}_G, sk_0, sk_1'),$$

where

$$(pk, \mathsf{sam}_G, \mathsf{sam}_B, dk) \leftarrow \texttt{Setup}(1^\lambda)$$

$$sk_0 \leftarrow \texttt{SampG}(\mathsf{sam}_G), \quad sk_1 \leftarrow \texttt{SampG}(\mathsf{sam}_G), \quad sk_1' \leftarrow \mathsf{Update}(sk_0)$$

Hardness of Bad Keys: *Given* sam_G*, it's hard to produce a valid "bad key". Formally, for any PPT adversary* $\mathcal{A}$*:*

$$\Pr\left[\begin{array}{c}\mathtt{Ver}(pk, sk^*) = 1\\ \mathtt{isGood}(pk, sk^*, dk) = 0\end{array}\;\middle|\;\begin{array}{c}(pk, \mathsf{sam}_G, \mathsf{sam}_B, dk) \leftarrow \mathtt{Setup}(1^\lambda)\\ sk^* \leftarrow \mathcal{A}(pk, \mathsf{sam}_G)\end{array}\right] \le \mathsf{negl}(\lambda)$$

Hardness of Good Keys: *Given* sam_B*, it's hard to produce a "good key". Formally, for any PPT adversary* $\mathcal{A}$*:*

$$\Pr\left[\mathtt{isGood}(pk, sk^*, dk) = 1\;\middle|\;\begin{array}{c}(pk, \mathsf{sam}_G, \mathsf{sam}_B, dk) \leftarrow \mathtt{Setup}(1^\lambda)\\ sk^* \leftarrow \mathcal{A}(pk, \mathsf{sam}_B)\end{array}\right] \le \mathsf{negl}(\lambda)$$

ℓ**-Leakage-Indistinguishability:** *Informally, this property says that given both sampling keys* $\mathsf{sam}_G, \mathsf{sam}_B$*, and* ℓ *bits of leakage on a secret-key* sk *(which is either good or bad), it is hard to produce a secret-key* sk^* *which is in the same category as* sk*. Formally, for any PPT adversary* $\mathcal{A}$*, we have* $\left|\Pr[\mathcal{A}\ \textit{wins}\ ] - \frac{1}{2}\ \right| \le \mathsf{negl}(\lambda)$ *in the following game:*

- *The challenger chooses* $(pk, \mathsf{sam}_G, \mathsf{sam}_B, dk) \leftarrow \mathtt{Setup}(1^\lambda)$ *and gives* $pk, \mathsf{sam}_G, \mathsf{sam}_B$ *to* $\mathcal{A}$*. The challenger also chooses a random bit* $b \overset{\$}{\leftarrow} \{0, 1\}$*. If* $b = 1$ *then it samples* $sk \leftarrow \mathsf{SampG}(\mathsf{sam}_G)$*, else it samples* $sk \leftarrow \mathsf{SampB}(\mathsf{sam}_B)$*. It sets* $\mathsf{state} := sk$*.*
- $\mathcal{A}$ *can make up to* ℓ *queries* in total *to the leakage-oracle* $\mathcal{O}_{\mathsf{state}}(\cdot)$*.*
- $\mathcal{A}$ *outputs* sk^* *and* wins *if* $\mathtt{isGood}(pk, sk^*, dk) = b$*.*

An ℓ-LIRR can be used to construct an ℓ-CLR-OWR *with leak-free updates*, where the $\mathsf{Update}, \mathtt{Ver}$ algorithms are kept the same, while KeyGen samples pk and a "good" secret key sk_G (see Figure 5.1). Note that the CLR-OWR completely ignores the bad sampling algorithm SampB, the "bad" sampling key sam_B, the distinguishing algorithm $\mathtt{isGood}$, and the distinguishing key dk of the LIRR. These are only used in the argument of security. Moreover, the "good" sampling key sam_G is only used as an intermediate step during key-generation to sample the secret-key sk, but is never explicitly stored afterwards.

$\mathsf{KeyGen}(1^\lambda)$**:** Sample $(pk, \mathsf{sam}_G, \cdot, \cdot) \leftarrow \mathtt{Setup}(1^\lambda)$, $sk \leftarrow \mathsf{SampG}(\mathsf{sam}_G)$.
Output (pk, sk).
$\mathsf{Update}, \mathtt{Ver}$**:** Same as for LIRR.

Figure 5.1: Constructing CLR-OWR from a LIRR

Security Intuition. We argue that the above construction is secure. Assume an adversary attacks the construction and, after several leakage-rounds, produces a valid secret key sk^*. Since the adversary does not see any information related to the bad sampling key sam_B, we can use the *hardness of bad keys* property to argue that sk^* must be a "good" key. However, we then argue that the adversary cannot notice if we start switching good keys to bad keys in the leakage rounds. More precisely, we define several hybrid games, where each game differs from the previous by replacing a good key with a bad key in one additional leakage round. We argue that, by the *ℓ-leakage-indistinguishability* property, the probability that the adversary produces a good key sk^* as her forgery does not change between any two hybrids. Notice that this argument allows us to only analyze leakage in a single round at a time, and thus avoids the main difficulty of analyzing continuous leakage. In the last of the hybrids, the adversary only sees "bad keys", yet still manages to produce a good key sk^* as her forgery. But this contradicts the *hardness of good keys* property, and proves the security of the scheme.

Theorem 5.3.2. *Given any ℓ-LIRR scheme, the construction in Figure 5.1 is a secure ℓ-CLR-OWR.*

Proof. The correctness properties of CLR-OWR follow from the correctness/re-randomization of the LIRR. Let $\mathcal{A}$ be any polynomial-time attacker whose probability of winning the ℓ-CLR-OWR security game (with leak-free updates) is $\epsilon(\lambda)$. We use a series-of-games argument to argue that ϵ is negligible:

Game 0: This is the original ℓ-CLR Game from Definition 5.2.6. the challenger initially samples $(pk, \mathsf{sam}_G, \mathsf{sam}_B, dk) \leftarrow \mathtt{Setup}(1^\lambda)$, $sk_1 \leftarrow \mathtt{SampG}(\mathsf{sam}_G)$ and gives pk to $\mathcal{A}$. It sets $\mathsf{state} = sk_1$. At the end of each leakage round, it pareses $sk_i = \mathsf{state}$, updates $sk_{i+1} \leftarrow \mathsf{Update}(sk_i)$, and sets $\mathsf{state} = sk_{i+1}$.

By assumption, the probability that $\mathcal{A}$ wins in this game is $\epsilon(\lambda)$.

Game 1: In this game, the challenger initially samples $(pk, \mathsf{sam}_G, \mathsf{sam}_B, dk) \leftarrow \mathtt{Setup}(1^\lambda)$ $sk_1 \leftarrow \mathtt{SampG}(\mathsf{sam}_G)$ and gives pk to $\mathcal{A}$. The game them proceeds as before with many leakage rounds, *except that* the secret key used in each leakage-round i is chosen at fresh as $sk_i \leftarrow \mathtt{SampG}(\mathsf{sam}_G)$, and independently of all previous rounds.

Games 0 and 1 are indistinguishable by the *re-randomization property* (applied q times, where q is the total number of leakage-rounds). Therefore, $\Pr[\mathcal{A} \text{ wins Game 1}] \geq \epsilon(\lambda) - \mathsf{negl}(\lambda)$.

Game 2: Game 2 is the same as Game 1, except that we modify the winning-condition to say that the adversary only wins if, at the end, it outputs a "good" sk^* such that $\mathtt{isGood}(pk, sk^*, dk) = 1$. Notice that the winning condition now cannot be checked by the attacker (since it does not know dk).

Let E be the event that, at the end of Game 1, $\mathcal{A}$ ends up outputting a value sk^* which satisfies $\mathtt{Ver}(pk, sk^*) = 1 \wedge \mathtt{isGood}(pk, sk^*, dk) = 0$. Then

$$\Pr[\mathcal{A}\text{wins in Game 2}] \geq \Pr[\mathcal{A}\text{wins in Game 1}] - \Pr[E]$$

We argue that $\Pr[E]$ is negligible. This follows directly by the *hardness of bad keys* and the observation that the entire view of the attacker in Game 1 can be simulated from pk, sam_G alone. Therefore $\Pr[\mathcal{A}$ wins Game 2$] \geq \epsilon(\lambda) - \mathsf{negl}(\lambda)$.

Games 2.i - 3: Let q be the total number of leakage rounds for which $\mathcal{A}$ runs. We define the Games 2.i for $i = 0, 1 \ldots, q$ as follows. The challenger initially samples $(pk, \mathsf{sam}_G, \mathsf{sam}_B, dk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and gives pk to $\mathcal{A}$. The game then proceeds as before with many leakage rounds, *except that* the secret keys in rounds $j \leq i$ are chosen randomly and independently via $sk_j \leftarrow \mathtt{SampB}(\mathsf{sam}_B)$, and in the rounds $j > i$, they are chosen randomly and independently as $sk_j \leftarrow \mathtt{SampG}(\mathsf{sam}_G)$. Note that Game 2.0 is the same as Game 2, and we define Game 3 to be the same as Game 2.q.

We use the *ℓ-Leakage Indistinguishability* property to argue that, for $i = 1, \ldots, q$, the winning probability of $\mathcal{A}$ is the same in Game 2.$(i-1)$ as in Game 2.i, up to negligible factors:

$$\Pr[\mathcal{A} \text{ wins Game 2.}i] \geq \Pr[\mathcal{A} \text{ wins Game 2.}(i-1)] - \mathsf{negl}(\lambda)$$

Assuming otherwise, we construct a reduction $\mathcal{B}$ which breaks leakage indistinguishability. The reduction simulates $\mathcal{A}$s view in leakage-rounds $j < i$ using sam_B and rounds $j > i$ using sam_G. For round i, the reduction simulates leakage by calling its own leakage-oracle, on the challenge secret-key. At the end, $\mathcal{B}$ outputs the value sk^* output by $\mathcal{A}$. If $\mathcal{B}$'s challenger uses a good key then that corresponds to the view of $\mathcal{A}$ in game 2.$(i-1)$ and a bad key corresponds to game 2.i. Therefore, letting b be the bit used by $\mathcal{B}$s challenger:

$$\begin{aligned}
&\left|\Pr[\mathcal{B} \text{ wins }] - \frac{1}{2}\right| \\
&= \left|\Pr[\mathtt{isGood}(pk, sk^*, dk) = b] - \frac{1}{2}\right| \\
&= \frac{1}{2}\left|\begin{array}{l} \Pr[\mathtt{isGood}(pk, sk^*, dk) = 1 \mid b = 1] \\ - \Pr[\mathtt{isGood}(pk, sk^*, dk) = 1 \mid b = 0] \end{array}\right| \\
&= \frac{1}{2}\left| \Pr[\mathcal{A} \text{ wins in Game 2.}(i-1)] - \Pr[\mathcal{A} \text{ wins in Game 2.}i]\right|
\end{aligned}$$

Therefore, by applying the hybrid argument, we get $\Pr[\mathcal{A}$ wins in Game 3$] \geq \epsilon(\lambda) - \mathsf{negl}(\lambda)$.

Can't Win Game 3: We now argue that probability of $\mathcal{A}$ winning game 3 is negligible, by *the hardness of good keys*. Notice that $\mathcal{A}$'s view in Game 3 can be simulated entirely just given pk, sam_B. Therefore, there is a PPT algorithm which, given pk, sam_B as inputs, can run Game 3 with $\mathcal{A}$ and output sk^* such that $\mathtt{isGood}(pk, sk^*, dk) = 1$ whenever $\mathcal{A}$ wins.

So $\epsilon(\lambda) - \mathsf{negl}(\lambda)$ is itself negligible, which implies that $\epsilon(\lambda)$ must be negligible as well, as we wanted to show.

□

5.4 Construction (Part II): Generic Components

5.4.1 Syntax of Construction and Hardness Properties

We now instantiate an ℓ-LIRR using two public-key encryption schemes and a NIZK argument system. See Section 4.5 for a review of NIZKs.

Overview of Construction. We first start by describing the high level idea and the syntax of the construction. Let $\mathcal{E}_1 = (\mathsf{KeyGen}^1, \mathsf{Encrypt}^1, \mathsf{Decrypt}^1)$, $\mathcal{E}_2 = (\mathsf{KeyGen}^2, \mathsf{Encrypt}^2, \mathsf{Decrypt}^2)$ be two *public-key encryption schemes, with perfect correctness* (we will define the security properties that we need from them later on). We define the *plaintext equality* relation for the schemes $\mathcal{E}_1, \mathcal{E}_2$ by:

$$R_{eq} \stackrel{\text{def}}{=} \left\{ (y, x) \;\middle|\; \begin{array}{c} y = (pk_1, pk_2, c_1, c_2), \\ x = (m, r_1, r_2) \end{array} \text{ s.t. } \begin{array}{c} c_1 = \mathsf{Encrypt}^1_{pk_1}(m; r_1), \\ c_2 = \mathsf{Encrypt}^2_{pk_2}(m; r_2) \end{array} \right\}.$$

The corresponding language $L_{eq} := \{y \;:\; \exists x, (y, x) \in R_{eq}\}$, is the language of honestly generated ciphertext pairs that encrypt the same plaintext. Let $\Pi = (\mathtt{Setup}^\Pi, \mathtt{Prov}^\Pi, \mathtt{Ver}^\Pi, \mathtt{SetupSim}^\Pi, \mathtt{Sim}^\Pi)$ be a NIZK argument system for R_{eq}. We will often omit the public-keys pk_1, pk_2 from the descriptions of statements $y \in L_{eq}$, when clear from context.

We will assume that the schemes $\mathcal{E}_1, \mathcal{E}_2, \Pi$ can be "tied together" by sharing some common *system parameters* $\mathsf{prms} \leftarrow \mathsf{ParamGen}(1^\lambda)$ (e.g. the description of some group) which are implicitly used as inputs by all of the algorithms of each of the schemes. The parameters define a common message-space $\mathcal{M}$ for the schemes $\mathcal{E}_1, \mathcal{E}_2$. The basic syntax of our construction of LIRR, except for the re-randomization algorithm, is shown in Figure 5.2. The main idea is to encrypt a random message m using the scheme $\mathcal{E}_1$, and put the ciphertext c_1 in the public-key. The secret key consists of a ciphertext/proof pair (c_2, π). In a good secret-key,

c_2 is a new random encryption of m under $\mathcal{E}_2$, and π is a proof of plaintext-equality. In a bad secret-key, c_2 is just an encryption of some fixed message $0_{\mathcal{M}}$, and π is a simulated proof. The secret keys sk_1, sk_2 of the encryption scheme can be used to distinguish "good keys" from "bad keys". Verification just checks that the proof π verifies as a valid proof of ciphertext equality for (c_1, c_2).

$\mathtt{Setup}(1^\lambda)$**:** Output $pk = (\mathsf{prms}, \textsc{crs}, pk_1, pk_2, c_1)$, $\mathsf{sam}_G = (m, r_1)$, $\mathsf{sam}_B = \textsc{tk}$, $dk = (sk_1, sk_2)$ where:

$$\begin{gathered}\mathsf{prms} \leftarrow \mathsf{ParamGen}(1^\lambda),\\ (\textsc{crs}, \textsc{tk}) \leftarrow \mathtt{SetupSim}^\Pi(\mathsf{prms}),\\ (pk_1, sk_1) \leftarrow \mathsf{KeyGen}^1(\mathsf{prms}),\ (pk_2, sk_2) \leftarrow \mathsf{KeyGen}^2(\mathsf{prms})\\ m \leftarrow \mathcal{M},\quad c_1 \leftarrow \mathsf{Encrypt}^1_{pk_1}(m; r_1)\end{gathered}$$

$\mathtt{SampG}(\mathsf{sam}_G)$**:** Output $sk_G = (c_2, \pi)$ where: $c_2 \leftarrow \mathsf{Encrypt}^2_{pk_2}(m; r_2)$, $\pi \leftarrow \mathtt{Prov}^\Pi((c_1, c_2), (m, r_1, r_2))$.

$\mathtt{SampB}(\mathsf{sam}_B)$**:** Output $sk_B = (c_2, \pi)$ where: $c_2 \leftarrow \mathsf{Encrypt}^2_{pk_2}(0_{\mathcal{M}})$, $\pi \leftarrow \mathtt{Sim}^\Pi((c_1, c_2), \textsc{tk})$.

$\mathtt{Ver}(pk, sk)$**:** Parse $sk = (c_2, \pi)$ and output $\mathtt{Ver}^\Pi((c_1, c_2), \pi)$.

$\mathtt{isGood}(pk, sk, dk)$**:** Parse $sk = (c_2, \pi), dk = (sk_1, sk_2)$. Output 1 iff $\mathsf{Decrypt}^1_{sk_1}(c_1) = \mathsf{Decrypt}^2_{sk_2}(c_2)$.

Figure 5.2: Constructing LIRR

It is easy to see that the scheme satisfies the correctness property. The hardness of *bad keys*, follows directly from the soundness of the NIZK argument system. The hardness of *good keys*, on the other hand, follows is we assume that the encryption scheme $\mathcal{E}_1$ has *one-way security* (see), which is weaker than semantic-security and only requires that an encryption of a random message is hard to invert.

Lemma 5.4.1. *Assume that $\mathcal{E}_1$ is a* one-way secure *encryption scheme, $\mathcal{E}_2$ is* semantically secure*, and Π is a NIZK. Then the construction of LIRR in Figure 5.2 satisfies the* correctness*,* hardness of good keys *and* hardness of bad keys *properties from Definition 5.3.1.*

Proof. We prove *correctness*, *hardness of good keys* and *hardness of bad keys* one at a time.

Correctness. Assume that we sample $pk, \mathsf{sam}_G, \mathsf{sam}_B$, dk, sk_G, sk_B as specified. By the correctness of the encryption schemes, we have $\mathtt{isGood}(pk, sk_G, dk) = 1$.

On the other hand $\texttt{isGood}(pk, sk_B, dk) = 0$, unless the message m encrypted by the ciphertext c_1 is the value $m = 1$, which only occurs with negligible probability.[1] Finally, we show that $\texttt{Ver}(pk, sk_G) = \texttt{Ver}(pk, sk_B) = 1$ with overwhelming probability. Assume otherwise. Letting E be the event that $\texttt{Ver}(pk, sk_G) = 0$ or $\texttt{Ver}(pk, sk_B) = 0$, there is therefore a non-negligible probability of E occurring. Let us switch how we generate the public-key pk and the values sk_B. Firstly, let us choose the ciphertext c_2 in sk_B in the same way as sk_G, via $c_2 \leftarrow \mathsf{Encrypt}^2_{pk_2}(m; r_2)$, to be an encryption of the same message m as contained in c_1 instead of the value 0. Then the event E still has a non-negligible probability of occurring by semantic security of $\mathcal{E}^2$. Secondly, let us choose the proof π in the key sk_B to be an honestly generated proof of ciphertext equality $\pi \leftarrow \texttt{Prov}^{\Pi}((c_1, c_2); (m, r_1, r_2))$. Thirdly, let us choose the value CRS in the public key to be honestly generated via $\text{CRS} \leftarrow \texttt{Setup}^{\Pi}(\mathsf{prms})$. Then the event E still has a non-negligible probability of occurring by the zero-knowledge property of Π. However, now this contradicts the completeness of the NIZK since both keys sk_G, sk_B now contain honestly generated proofs of true statements under an honestly generated CRS.

Hardness of bad keys. Assume that there is a poly-time attacker $sk^* \leftarrow \mathcal{A}(pk, \mathsf{sam}_G)$ and that $\Pr[\texttt{Ver}(pk, sk^*) = 1, \texttt{isGood}(pk, sk^*, dk) = 0]$ is non-negligible. Firstly, we argue that this probability remains non-negligible even if we change the value CRS (contained in pk) to being chosen honestly via $\text{CRS} \leftarrow \texttt{Setup}(\mathsf{prms})$ instead of using the simulator. This just follows by the ZK property of the NIZK (and that the entire above experiment is independent of TK). But now $\mathcal{A}$ breaks the soundness of the NIZK argument system Π. That is, the value $sk^* = (c_2, \pi)$ defines the statement (c_1, c_2) such that:

1. $(c_1, c_2) \notin L_{eq}$ whenever $\texttt{isGood}(pk, sk^*, dk) = 0$ (by the correctness of encryption).
2. $\texttt{Ver}^{\Pi}((c_1, c_2), \pi) = 1$ whenever $\texttt{Ver}(pk, sk^*) = 1$.

Therefore $\mathcal{A}$ gives a poly-time attack against the soundness of the NIZK.

Hardness of good keys. This property follows by the one-wayness of $\mathcal{E}_1$. In particular, assume we are given a one-wayness challenge pk_1, c_1. Then we use an attacker $\mathcal{A}$ that breaks hardness of good keys to recover the message m contained in c_1. We simply choose everything else $\text{CRS}, \text{TK}, pk_2, sk_2$ ourselves (honestly) and use these to create the values pk, sam_B for $\mathcal{A}$, where the challenge ciphertext c_1 is embedded in pk. The attacker $\mathcal{A}$ outputs $sk^* = (c_2^*, \pi^*)$ which has a non-negligible probability of satisfying $\texttt{isGood}(pk, sk^*, dk) = 1$ (we cannot check this

[1] The value m is chosen at random from the message space $\mathcal{M}$, which must be super-polynomial for one-way security to hold.

since we do not know dk). Nevertheless, we can just use the key sk_2 to decrypt $m^* = \mathsf{Decrypt}^2_{sk_2}(c_2^*)$ and output it, thus breaking one-wayness with non-negligible probability. $\square$

We are left to show (1) how to re-randomize secret keys, and (2) that the leakage-indistinguishability property holds. We do so in the next two sections, by requiring additional properties from the building-blocks $\mathcal{E}_1, \mathcal{E}_2, \Pi$.

5.4.2 Re-randomization via Homomorphisms

We now show how to perfectly re-randomize the secret keys of our LIRR construction. Recall that a secret-key consists of a pair (c_2, π) where π is a proof of plaintext-equality for the statement (c_1, c_2). Therefore, to re-randomize the key, we need to first re-randomize the ciphertext c_2 into a new ciphertext c_2^* with the same plaintext. We then need to update the proof π to look like a *fresh new* proof of a *new* statement (c_1, c_2^*), for which we do not know the witness! We show that this is indeed possible if the encryption schemes $\mathcal{E}_1, \mathcal{E}_2$ and the argument-system Π are all homomorphic over some appropriate group. In particular, we define a new notion of *homomorphic NIZKs*, which is influenced by the notions of re-randomizable and malleable NIZKs from [BCC+09]. Throughout this section we assume that the schemes $\mathcal{E}^1, \mathcal{E}^2, \Pi$ share *system parameters* $\mathsf{prms} \leftarrow \mathsf{ParamGen}(1^\lambda)$ which define some group structure over the message/randomness/ciphertext spaces. Throughout, we denote abstract group operations as "+" (addition) and the identity elements as 0.

Definition 5.4.2 (Homomorphic Encryption). *We say that an encryption scheme* $(\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is* homomorphic *if the system-parameters of the scheme define groups* $\mathcal{M}, \mathcal{R}, \mathcal{C}$ *for the message-space, randomness-space, and ciphertext-space respectively, such that, for any* $m, m' \in \mathcal{M}$ *any* $r, r' \in \mathcal{R}$*:*

$$c = \mathsf{Encrypt}_{pk}(m; r),\ c' = \mathsf{Encrypt}_{pk}(m'; r') \quad \Rightarrow \quad c + c' = \mathsf{Encrypt}_{pk}(m + m'; r + r')$$

It is easy to see that for any homomorphic encryption scheme and any public-key pk, we have $\mathsf{Encrypt}_{pk}(0_{\mathcal{M}}; 0_{\mathcal{R}}) = 0_{\mathcal{C}}$ (i.e. encryption of the identity -message under identity-randomness gives an identity-ciphertext).

Definition 5.4.3 (Linear Relation). *We say that a relation* $R \subseteq \mathcal{Y} \times \mathcal{X}$ *is* linear *if* $\mathcal{Y}, \mathcal{X}$ *are groups and, for any* $(y, x), (y', x') \in R$*, we have* $(y + y', x + x') \in R$.

Definition 5.4.4 (Homomorphic NIZK). *We say that a NIZK argument-system* $(\mathtt{Setup}, \mathtt{Prov}, \mathtt{Ver}, \mathtt{Sim})$ *for a linear-relation* $R \subseteq \mathcal{Y} \times \mathcal{X}$ *is* homomorphic *if there are groups* $\mathcal{R}, \mathcal{P}$ *for the randomness of the prover and the proof respectively, such that for any* $(y, x), (y', x') \in R$*, any* $r, r' \in \mathcal{R}$*:*

$$\pi = \mathtt{Prov}(y, x; r),\ \pi' = \mathtt{Prov}(y', x'; r') \quad \Rightarrow \quad \pi + \pi' = \mathtt{Prov}(y + y', x + x'; r + r')$$

where $\pi, \pi' \in \mathcal{P}$.

We now connect the above definitions of homomorphic primitives to our construction in Figure 5.2, showing how to re-randomize the secret-keys if $\mathcal{E}_1, \mathcal{E}_2$, and Π are homomorphic. First, we show that the plaintext-equality linear if we *fix* the public-keys pk_1, pk_2. That is, for each pk_1, pk_2, define the relation $R_{eq}^{(pk_1,pk_2)} \subseteq R_{eq}$ where pk_1, pk_2 are fixed and the statements only consist of (c_1, c_2). It is easy to verify the following lemma.

Lemma 5.4.5. *If $\mathcal{E}_1, \mathcal{E}_2$ are homomorphic with a common message-group $\mathcal{M}$, and respective randomness-groups $\mathcal{R}_1, \mathcal{R}_2$, and ciphertext-groups $\mathcal{C}_1, \mathcal{C}_2$, then for any pk_1, pk_2, the relation $R_{eq}^{(pk_1,pk_2)}$ is a linear relation over $\mathcal{Y} = \mathcal{C}_1 \times \mathcal{C}_2$ and $\mathcal{X} = \mathcal{M} \times \mathcal{R}_1 \times \mathcal{R}_2$.*

Proof. Let $(c_1, c_2), (c_1', c_2') \in L_{eq}^{(pk_1,pk_2)}$ be two statements with respective witnesses (m, r_1, r_2) and (m', r_1', r_2') so that $c_1 = \mathsf{Encrypt}^1_{pk_1}(m; r_1), c_2 = \mathsf{Encrypt}^2_{pk_2}(m; r_2)$ and $c_1' = \mathsf{Encrypt}^1_{pk_1}(m'; r_1'), c_2' = \mathsf{Encrypt}^2_{pk_2}(m'; r_2')$.

Then the statement $(c_1 + c_1', c_2 + c_2') \in L_{eq}^{(pk_1,pk_2)}$ is a true statement having the corresponding witness $(m + m', r_1 + r_1', r_2 + r_2')$ since $c_1 + c_1' = \mathsf{Encrypt}^1_{pk_1}(m + m'; r_1 + r_1')$ and $c_2 + c_2' = \mathsf{Encrypt}^2_{pk_2}(m + m'; r_2 + r_2')$. □

To simplify the discussion, we will say that a proof-system Π for R_{eq} is homomorphic if, for *every fixed choice* of the public-keys pk_1, pk_2, it is homomorphic for the linear relations $R_{eq}^{(pk_1,pk_2)}$. Now assume that $\mathcal{E}_1, \mathcal{E}_2$ are two homomorphic encryption schemes satisfying the requirements of Lemma 5.4.5, and that Π is a homomorphic NIZK for R_{eq}, with randomness-group $\mathcal{R}_3$ and proof-group $\mathcal{P}$. In Figure 5.3, we show how to re-randomize the secret keys of our LIRR construction from Figure 5.2.

```
Update(sk): Parse $sk = (c_2, \pi)$. Choose $(r_2', r_3') \xleftarrow{\$} \mathcal{R}_2 \times \mathcal{R}_3$.
        Set $c_2' := \mathsf{Encrypt}^2_{pk_2}(0_{\mathcal{M}}; r_2')$, $\pi' := \mathtt{Prov}^{\Pi}((0_{\mathcal{C}_1}, c'), (0_{\mathcal{M}}, 0_{\mathcal{R}_1}, r_2'); r_3')$.
        Output $sk^* = (c + c', \pi + \pi')$.
```

Figure 5.3: Re-randomization

The main idea is to re-randomize the ciphertext c_2 in the secret key (without modifying the encrypted message), by adding in a random encryption c_2' of the message 0. We then need to update the proof π in the secret key to look like a *fresh new proof* of the *new true statement* $(c_1, c_2 + c_2') \in L_{eq}$. We do so by adding to π a randomly generated proof π' of the true statement $(0_{\mathcal{C}_2}, c_2') \in L_{eq}$ (recall that the 0 ciphertext encrypts the 0 message using the 0 randomness).

Lemma 5.4.6. *The re-randomization method in Figure 5.3 satisfies the* re-randomization of LIRR *(Definition 5.3.1).*

Proof. Fix any $pk = (pk_1, pk_2, \text{CRS}, c_1), \mathsf{sam}_G = (m, r_1)$ output by KeyGen and $sk_G = (c_2, \pi)$ output by $\mathtt{SampG}(\mathsf{sam}_G)$. Then there exist some r_2, r_3 for which

$$c_1 = \mathsf{Encrypt}^1_{pk_1}(m; r_1) \ , \ c_2 = \mathsf{Encrypt}^2_{pk_2}(m; r_2) \ , \ \pi = \mathtt{Prov}((c_1, c_2), (m, r_1, r_2); r_3).$$

If we sample $sk^* \leftarrow \mathsf{Update}(sk_G)$, we have $sk^* = (c_2^*, \pi^*)$ with $c_2^* = c_2 + c_2', \pi^* = \pi + \pi'$ where:

$$\begin{aligned} c_2^* &= c_2 + c_2' = c_2 + \mathsf{Encrypt}_{pk_2}(0_{\mathcal{M}}; r_2') \ = \ \mathsf{Encrypt}_{pk_2}(m; r_2 + r_2') \\ \pi^* &= \pi + \pi' = \pi + \mathtt{Prov}((0_{\mathcal{C}_1}, c_2'), (0_{\mathcal{M}}, 0_{\mathcal{R}_1}, r_2'); r_3') \\ &= \mathtt{Prov}((c_1, c_2^*), (m, r_1, r_2 + r_2'); r_3 + r_3') \end{aligned}$$

for some fresh random values r_2', r_3'. Letting $r_2^* = r_2 + r_2'$ and $r_3^* = r_3 + r_3'$, we see that c_2^* is a fresh new encryption of m under randomness r_2^* and π^* is a fresh proof of ciphertext equality under randomness r_3^*. Therefore, sampling $sk^* \leftarrow \mathsf{Update}(sk_G)$ is *exactly the same* as sampling $sk^* \leftarrow \mathtt{SampG}(\mathsf{sam}_G)$, even after fixing any choice of pk, sam_G, sk_G. This shows that the re-randomization property holds with *perfect* distributional equality.

□

5.4.3 Leakage-Indistinguishability

We are left to show the leakage-indistinguishability of our construction. To do so, we need to define a new security property, called *leakage-of-ciphertext non-malleability*, for the encryption scheme $\mathcal{E}_2$. Intuitively, this property says that, given ℓ bits of leakage on a *ciphertext* c, the adversary cannot produce a *related ciphertext* c^*.

Definition 5.4.7 (Leakage-of-Ciphertext Non-Malleable Encryption). *A public-key encryption scheme* $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is* ℓ-leakage-of-ciphertext non-malleable (ℓ-LoC NM) *if, for any PPT adversary* $\mathcal{A}$*, we have*
$\left|\Pr[\mathcal{A} \text{ wins }] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$ *in the following game:*

- *Challenger samples* $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and gives* pk *to* $\mathcal{A}$*.*
- $\mathcal{A}$ *chooses two messages* $m_0, m_1 \in \mathcal{M}_{pk}$ *and gives these to the challenger.*
- *Challenger samples* $b \xleftarrow{\$} \{0, 1\}$*,* $c \leftarrow \mathsf{Encrypt}_{pk}(m_b)$*, and sets* $\mathsf{state} := c$
  $\mathcal{A}$ *can make up to* ℓ *queries to the leakage oracle* $\mathcal{O}_{\mathsf{state}}(\cdot)$*.*
- $\mathcal{A}$ *chooses a single arbitrary value* c^* *and the challenger responds with* $m^* = \mathsf{Decrypt}_{sk}(c^*)$ *to* $\mathcal{A}$*.*
- $\mathcal{A}$ *outputs a bit* $\tilde{b}$ *and wins if* $\tilde{b} = b$*.*

Remarks on the Definition. Note that, in the definition, the leakage is only on the ciphertext c and *not* on the secret-key sk. It is easy to see that *even* 1-LoC-NM security (i.e. adv. gets only 1 bit of leakage) *already* implies semantic-security, since a 1-bit leakage function can run the distinguisher. On the other hand, the standard notion of non-malleable encryption from [DDN91] implies ℓ-LoC-NM security where ℓ approaches the message-size of the scheme. This is because an adversary that leaks less than ℓ bits about a ciphertext c is unlikely to be able to re-produce c exactly, and the decryption of any other ciphertext $c^* \neq c$ safely keeps the challenge message hidden. However, non-malleable encryption is inherently *not* homomorphic while, as we will soon see, Leakage-of-Cipherext Non-malleable encryption can indeed be homomorphic as well.

Application to Leakage-Indistinguishability. We now show that, if the scheme $\mathcal{E}_2$ in our construction is ℓ-LoC-NM, then the construction satisfies ℓ-leakage-indistinguishability (Definition 5.3.1). This is because the ability to create a "related secret-key" which is in the same category as a challenge key on which we can leak, requires the ability to create a "related ciphertext" which encrypts the same message as another ciphertext on which we can leak.

Lemma 5.4.8. *Assume that, in the construction of LIRR in Figure 5.2, the scheme $\mathcal{E}_2$ is ℓ-LoC-NM and Π is a secure NIZK. Then the construction satisfies the leakage-indistinguishability property of LIRR (Definition 5.3.1).*

Proof. Assume $\mathcal{A}$ is a PPT adversary which has a non-negligible advantage in the leakage-indistinguishability (LI) game of Definition 5.3.1.

Consider the following modification to the LI game. In the original LI game, if the challenger's bit is $b = 1$, then the challenger samples the challenge secret-key $\tilde{sk} = (\tilde{c}, \tilde{\pi}) \leftarrow \mathtt{SampG}(\mathsf{sam}_G)$. Therefore, $\tilde{c}$ is a random encryption of m, and π is an honestly generated proof for the statement $(c_1, c_2) \in L_{eq}^{(pk_1, pk_2)}$ under the witness (m, r_1, r_2). In the modified-LI game, if the challenger's bit is $b = 1$, then the challenger samples $\tilde{c}$ as before, but uses a simulated proof $\tilde{\pi} \leftarrow \mathtt{Sim}^{\Pi}((c_1, c_2), \text{TK})$.

The adversary's probability of winning in the modified LI game is the same as that of the original game, up to negligible factors, by the (composable) NIZK property of the proof system Π (note that we need composable NIZK, since the adversary sees sam_B which includes the trapdoor TK for the proof system Π).

We now show how to use any adversary $\mathcal{A}$ that attacks the modified LI game to construct a reduction $\mathcal{B}$ that attacks the LoC-NM game.

1. The reduction $\mathcal{B}$ initially gets a challenge public-key, which it sets as pk_2. In addition $\mathcal{B}$ samples and the values (pk_1, sk_1), (CRS, TK), $\mathsf{sam}_G = (m, r)$, $\mathsf{sam}_B = \text{TK}$ and $c_1 = \mathsf{Encrypt}_{pk_1}(m; r)$ just as the honest key-generation algorithm of the LIRR construction. It gives $pk = (\text{CRS}, pk_1, pk_2, c_1), \mathsf{sam}_G, \mathsf{sam}_B$ to $\mathcal{A}$.

2. The reduction $\mathcal{B}$ chooses challenge messages $m_0 = 0_{\mathcal{M}}$, $m_1 = m$ and gives these to its challenger.

3. The adv. $\mathcal{A}$ expects to make up to ℓ calls to a leakage-oracle (simulated by $\mathcal{B}$) on a secret key $\tilde{sk} = (\tilde{c}, \tilde{\pi})$. The reduction $\mathcal{B}$ initially chooses some randomness r for a NIZK simulator. Then, for each function h chosen by $\mathcal{A}$ (recall h expects a value $\tilde{sk} = (\tilde{c}, \tilde{\pi})$ as input), the adv. $\mathcal{B}$ creates a function h' (which expects only $\tilde{c}$ as input, and contains TK, r hard-coded) such that $h'(\tilde{c})$ computes $\tilde{\pi} = \texttt{Sim}((c_1, \tilde{c}), tk; r)$ and then outputs $h(\tilde{c}, \tilde{\pi})$. The reduction $\mathcal{B}$ then passes h' to its own leakage-oracle on the challenge ciphertext $\tilde{c}$. [2]

4. When $\mathcal{A}$ outputs $sk^* = (c^*, \pi^*)$, the reduction $\mathcal{B}$ gives c^* to its challenger and, if it gets back the message m, it outputs $\tilde{b} = 1$. Else it outputs $\tilde{b} = 0$.

We now argue that the reduction $\mathcal{B}$ wins the ℓ-LoC game with the exact same probability that $\mathcal{A}$ wins in the modified-LI game, which concludes the proof. This is because, no matter what bit b is chosen by the challenger in the ℓ-LoC-NM game, the simulated view of $\mathcal{A}$ above is *exactly* that of the modified LI game with challenge-bit b. Moreover, $\mathcal{B}$ wins (outputs $\tilde{b} = b$) whenever $\mathcal{A}$ wins (outputs sk^* such that $\texttt{isGood}(pk, sk^*, dk) = b$). □

5.4.4 Summary of Component Requirements

The following theorem follows directly from Lemmata 5.4.1, 5.4.6 and 5.4.8 and Theorem 5.3.2.

Theorem 5.4.9. *For any function $\ell(\cdot)$ such that $\ell(\lambda) \geq 1$, the construction in Figure 5.2, with the re-randomization procedure in Figure 5.3, is a secure ℓ-LIRR as long as the components $\mathcal{E}_1, \mathcal{E}_2, \Pi$ satisfy:*

- *$\mathcal{E}_1, \mathcal{E}_2$ are homomorphic encryption schemes with perfect correctness and a common message-space $\mathcal{M}$.*
- *$\mathcal{E}_1$ is* one-way secure *and $\mathcal{E}_2$ is ℓ-LoC-NM secure.*
- *Π is a homomorphic NIZK for the plaintext-equality relation R_{eq}.*

Therefore, the existence of such $\mathcal{E}_1, \mathcal{E}_2, \Pi$ implies the existence of a ℓ-CLR-OWR.

For clarity, we also provide a stripped-down construction of (just) the CLR-OWR from the schemes $\mathcal{E}_1, \mathcal{E}_2, \Pi$, without the additional layer of abstraction provided by LIRR. This construction is shown in Figure 5.4.

[2]Essentially, we are saying that $\mathcal{B}$ can correctly simulate ℓ bits of leakage on $\tilde{sk} = (\tilde{c}, \tilde{\pi})$ given ℓ bits of leakage on $\tilde{c}$ alone, by having the leakage-functions simulate $\tilde{\pi}$.

$\mathsf{KeyGen}(1^\lambda)$**:** Output $pk = (\mathsf{prms}, \textsc{crs}, pk_1, pk_2, c_1)$, $sk = (c_2, \pi)$ where:

$$\mathsf{prms} \leftarrow \mathsf{ParamGen}(1^\lambda), (\textsc{crs}, \cdot) \leftarrow \mathtt{Setup}^\Pi(\mathsf{prms})$$
$$(pk_1, \cdot) \leftarrow \mathsf{KeyGen}^1(\mathsf{prms}),\ (pk_2, \cdot) \leftarrow \mathsf{KeyGen}^2(\mathsf{prms})$$
$$m \leftarrow \mathcal{M},\quad c_1 \leftarrow \mathsf{Encrypt}^1_{pk_1}(m; r_1)$$
$$c_2 \leftarrow \mathsf{Encrypt}^2_{pk_2}(m; r_2), \pi \leftarrow \mathtt{Prov}^\Pi((c_1, c_2), (m, r_1, r_2))$$

$\mathtt{Ver}(pk, sk)$**:** Parse $sk = (c_2, \pi)$ and output $\mathtt{Ver}^\Pi((c_1, c_2), \pi)$.

$\mathsf{Update}(sk)$**:** Parse $sk = (c_2, \pi)$. Choose $(r_2', r_3') \overset{\$}{\leftarrow} \mathcal{R}_2 \times \mathcal{R}_3$.
Set $c_2' := \mathsf{Encrypt}^2_{pk_2}(0_\mathcal{M}; r_2')$, $\pi' := \mathtt{Prov}((0_{\mathcal{C}_1}, c_2'), (0_\mathcal{M}, 0_{\mathcal{R}_1}, r_2'); r_3')$.
Output $sk^* = (c_2 + c_2', \pi + \pi')$.

Figure 5.4: CLR-OWR from Components $\mathcal{E}_1, \mathcal{E}_2, \Pi$

5.5 Construction (Part III): Instantiating the Components

In this section, we now show how to instantiate the homomorphic encryption and NIZK schemes $\mathcal{E}_1, \mathcal{E}_2, \Pi$ so as to satisfy the requirements of Theorem 5.4.9. We will do so under the *k-linear assumption* in Bilinear Groups. The main tool here will be the Groth-Sahai (GS) NIZK argument system, which we notice to be homomorphic. Below, we present the concrete constructions of all three schemes. It is useful to review the *linear-algebra* and *matrix-in-the-exponent* notation from the Preliminaries cahapter.

5.5.1 The encryption scheme $\mathcal{E}_1$

Let $k \in \mathbb{Z}^+$. Let $\mathsf{prms} = (q, \mathbb{G}, \mathbf{g})$ where $(p, \mathbb{G}, \mathbf{g}) \leftarrow \mathcal{G}(1^\lambda)$.

$\mathsf{KeyGen}(\mathsf{prms})$**:** Choose $(x_1, x_2, \ldots, x_k) \leftarrow \mathbb{F}_q^k$. Set $\mathbf{f}_1 := \mathbf{g}^{x_1}, \ldots, \mathbf{f}_k := \mathbf{g}_0^{x_k}$.
Output $sk = (x_1, \ldots, x_k)$, $pk = (\mathbf{f}_1, \ldots, \mathbf{f}_k)$.

$\mathsf{Encrypt}_{pk}(\mathbf{m})$**:** Choose $(r_1, \ldots, r_k) \leftarrow \mathbb{F}_q^k$. Output $c := (\mathbf{m} \cdot \mathbf{g}^{\sum_{i=1}^k r_i}, \mathbf{f}_1^{r_1}, \ldots, \mathbf{f}_K^{r_k})$.

$\mathsf{Decrypt}_{sk}(c)$**:** Parse $c = (\mathbf{c}_0, \mathbf{c}_1, \ldots, \mathbf{c}_k)$. Output $\mathbf{c}_0 / \left(\prod_{i=1}^k \mathbf{c}_i{}^{1/x_i}\right)$.

Figure 5.5: Generalized k-linear ElGamal

For the encryption scheme $\mathcal{E}_1$, we can use any homomorphic one-way secure encryption. We simply choose to use a generalization of the ElGamal encryption scheme to the k-linear assumption, as shown in Figure 5.5.

Theorem 5.5.1. *For any $K \geq 1$, the generalized ElGamal scheme (Figure 5.5) is semantically-secure and one-way secure under the k-linear assumption. Furthermore, it is homomorphic over the message-group $\mathcal{M} = \mathbb{G}$, randomness-group $\mathcal{R} = \mathbb{F}_q^K$ and ciphertext-group $\mathcal{C} = \mathbb{G}^{K+1}$.*

Proof. Follows directly from the k-linear assumption, which shows that given $pk = (\mathbf{f}_1, \ldots, \mathbf{f}_K)$ and $\mathbf{f}_1^{r_1}, \ldots, \mathbf{f}_K^{r_K}$, the value $\mathbf{g}^{\sum_{i=1}^K r_i}$ is indistinguishable from uniformly random. □

5.5.2 The encryption scheme $\mathcal{E}_2$

Let $k, n \in \mathbb{Z}^+$. Let $\mathsf{prms} = (q, \mathbb{G}, \mathbf{g})$ where $(q, \mathbb{G}, \mathbf{g}) \leftarrow \mathcal{G}(1^\lambda)$.

$\mathsf{KeyGen}(\mathsf{prms})$: Choose $u_1, \ldots, u_k \xleftarrow{\$} \mathbb{F}_q$ and define

$$A := \begin{pmatrix} 1 & u_1 & 0 & \ldots & 0 \\ 1 & 0 & u_2 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \ldots & u_k \end{pmatrix}$$

Choose $\vec{x}_0 \xleftarrow{\$} \mathbb{F}_q^{k+1}$, $X \xleftarrow{\$} \mathbb{F}_q^{(k+1)\times n}$.
Output $pk = (\mathbf{g}^A, \mathbf{g}^{A\vec{x}_0^\top}, \mathbf{g}^{AX})$, $sk = (\vec{x}_0, X)$.

$\mathsf{Encrypt}_{pk}(\mathbf{m})$: Parse $pk = (\mathbf{g}^A, \mathbf{g}^{\vec{b}}, \mathbf{g}^B)$. To encrypt a message $\mathbf{m} \in \mathbb{G}$:
Choose $\vec{r} \xleftarrow{\$} \mathbb{F}_q^k$. Output $c := (\mathbf{g}^{\vec{r}A}, \mathbf{g}^{\vec{r}\cdot\vec{b}^T}\mathbf{m}, \mathbf{g}^{\vec{r}B})$.

$\mathsf{Decrypt}_{sk}(c)$: Parse $c = (\mathbf{g}^{\vec{y}}, \mathbf{h}, \mathbf{g}^{\vec{v}})$ (without knowing the exponents $\vec{y}, \vec{v}$).
Use component X of sk to verify $\mathbf{g}^{\vec{y}X} \overset{?}{=} \mathbf{g}^{\vec{v}}$ and output $\perp$ if this fails.
Else use $\vec{x}_0$ to compute $\mathbf{m} = \mathbf{h}\mathbf{g}^{-\vec{y}\cdot\vec{x}_0^\top}$.

Figure 5.6: Generalized CS-Lite Scheme

For the scheme $\mathcal{E}_2$, we need a homomorphic encryption satisfying ℓ-LoC-NM security. Our scheme is shown in Figure 5.6. For those familiar with Cramer-Shoup -encryption [CS98] and and hash-proof systems [CS02], we observe that our scheme is a generalization of the “Cramer-Shoup Lite” (CS-Lite) encryption scheme. There are two main differences in our scheme. Firstly, the scheme is generalized to be secure under the k-linear assumption for any $k \geq 1$ (we note that similar generalizations of Cramer-Shoup to the k-linear assumption were already shown in [Sha07, CCS09]). Secondly, we can make the “verification element” arbitrarily long, consisting of n group elements for any $n \geq 1$ (we recover CS-Lite by setting $k = 1, n = 1$). Implicitly, we will show that ℓ-LoC-NM security can be constructed from any "1-*universal hash proof system*" (of which CS-Lite is an example), with sufficiently long proofs, where the leakage ℓ is proportional to the size of the proof. But, since we will need a scheme based on k-linear to make it work with Groth-Sahai NIZKs, we restrict ourselves to the concrete scheme in Figure 5.6 and do not discuss the greater generalization to hash proof systems any further. The discussion here is self-contained and does not require any prior knowledge of CS-Lite or of hash-proof systems.

Proof intuition. The high level idea of the proof goes as follows. By the k-linear assumption (matrix form), the adversary cannot distinguish a correctly generated *challenge ciphertext* from one computed as $(\mathbf{g}^{\vec{y}}, \mathbf{g}^{\vec{y}\cdot\vec{x}_0^\top}\mathbf{m}, \mathbf{g}^{\vec{y}X})$ where $\mathbf{g}^{\vec{y}}$ is *uniformly random* and the other components are efficiently computable from it and the secret key. This is true even if the attacker gets the ciphertext *in full*, let alone if it can only leak on it. With the above modification, the second component of the challenge ciphertext is *uniformly random* over the randomness of the secret vector $\vec{x}_0$ (even conditioned on the public-key), and hence $\mathbf{m}$ is *statistically* hidden by the ciphertext and the public-key. We only have to argue that the decryption query does not reveal any further information on $\vec{x}_0$ and therefore keeps $\mathbf{m}$ hidden. Let the ciphertext c^* in the decryption query be denoted by $c^* = (\mathbf{g}^{\vec{y}^*}, \mathbf{h}^*, \mathbf{g}^{\vec{v}^*})$. If $\vec{y}^* = \vec{r}^*A$ for some $\vec{r}^*$ (i.e. is in the row-span of A), then the decrypted message m^* is completely determined by the public-key exponent $A\vec{x}_0^T$ alone, and does not reveal any additional information about $\vec{x}_0$. On the other hand, if $\vec{y}^*$ is not in the row-span of A, we argue that c^* decrypts to $\perp$ with overwhelming probability and hence the response to the query does not reveal any information either. Consider the value $\mathbf{g}^{\vec{y}^*X}$, used during decryption to check "well-formedness" of c^*. This value is uniformly random over the randomness of the secret matrix X, even conditioned on the public value $\mathbf{g}^{AX}$. Unfortunately, it is *not* independent of the challenge-ciphertext component $\mathbf{g}^{\vec{y}X}$. For example, if the attacker chooses $\mathbf{g}^{\vec{y}^*} = \mathbf{g}^{b\vec{y}}$ for some known constant b, then $\mathbf{g}^{\vec{y}^*X} = \mathbf{g}^{b\vec{y}X}$ is completely determined from $\mathbf{g}^{\vec{y}X}$. However, since the adversary only sees ℓ bits of leakage on the challenge ciphertext, the value $\mathbf{g}^{\vec{y}^*X}$ has sufficient entropy even conditioned on *everything* that the attacker sees, that the attacker cannot guess it with anything better than negligible probability. Therefore, the attacker will be unable to produce a valid ciphertext of this latter form.

Theorem 5.5.2. *For any $k \geq 1$, $n \geq 1$, the generalized CS-Lite scheme (Figure 5.6) is an ℓ-LoC-NM secure encryption under the k-linear assumption with $\ell = n\log(q) - \lambda$, where q is the group size. Furthermore it is a homomorphic over the message-group $\mathcal{M} = \mathbb{G}$, randomness-group $\mathcal{R} = \mathbb{F}_q^k$, and ciphertext-group $\mathcal{C} = \mathbb{G}^{n+k+2}$.*

Proof. The scheme satisfies perfect correctness since, for an honestly generated encryption $c := (\mathbf{g}^{\vec{y}}, \mathbf{h}, \mathbf{g}^{\vec{v}}) = (\mathbf{g}^{\vec{r}A}, \mathbf{g}^{\vec{r}A\vec{x}_0^\top}\mathbf{m}, \mathbf{g}^{\vec{r}AX})$ of the message $\mathbf{m}$ we have:

- The verification check passes since $g^{\vec{y}X} = \mathbf{g}^{\vec{r}AX} = \mathbf{g}^{\vec{v}}$.
- The decrypted message is $\mathbf{h}\mathbf{g}^{-\vec{y}\cdot\vec{x}_0^\top} = \mathbf{m}\mathbf{g}^{\vec{r}A\vec{x}_0^\top}\mathbf{g}^{-\vec{y}\cdot\vec{x}_0^\top} = \mathbf{m}$.

For security, we first prove the following simple claim.

Claim 5.5.3. *Fix any matrix A matching the form of Figure 5.6 and assume that $u_i \neq 0$ for all i. Fix* any *$\vec{y} \in \mathbb{F}_q^{k+1} \setminus \mathsf{rowspan}(A)$. Then, over a random $\vec{x} \xleftarrow{\$} \mathbb{F}_q^{k+1}$, the distribution of $(A\vec{x}^\top, \vec{y}\cdot\vec{x}^\top)$ uniformly random over $\mathbb{F}_q^{k+1}$.*

Proof. This follows from the fact that the rows of A, together with the vector $\vec{y}$, span all of $\mathbb{F}_q^{k+1}$ and so the linear map $f_{A,\vec{y}}(\vec{x}) \overset{\text{def}}{=} (A\vec{x}^\top, \vec{y}\cdot\vec{x}^\top)$ is bijective. □

Returning to the proof of Theorem 5.5.2, we now do a series-of-games argument to show that the scheme satisfies ℓ-LoC-NM security.

Game 0: Let Game 0 be the original ℓ-LoC-NM game from Definition 5.3.1.

Game 1: In this game, the challenge ciphertext is distributed incorrectly. Instead of honestly encrypting $\mathbf{m}_b$, the challenger chooses $\vec{y} \xleftarrow{\$} \mathbb{F}_q^{k+1}$ and sets

$$c := (\mathbf{g}^{\vec{y}}, \mathbf{g}^{\vec{y}\cdot\vec{x}_0^\top}\mathbf{m}, \mathbf{g}^{\vec{y}X})$$

Notice that c is computed efficiently using knowledge of the secret key $\vec{x}_0, X$. The only difference is that, instead of $\vec{y}$ being random over $\mathsf{rowspan}(A)$, it is now uniformly random over all of $\mathbb{F}_q^{k+1}$.

We argue that the probability of $\mathcal{A}$ winning in Games 0 and 1 differs by at most negligible factors or else $\mathcal{A}$ breaks the k-linear assumption (matrix form). Assume we are given a k-linear challenge $(\mathbf{g}, \mathbf{g}^A, \mathbf{g}^{\vec{y}})$, where either (1) $\vec{y} = \vec{r}A$ is uniform over $\mathsf{rowspan}(A)$ or (2) $\vec{y}$ is uniform over $\mathbb{F}_q^{k+1}$. Then we can choose $sk = (\vec{x}_0, X)$ honestly and use these to define the public-key pk which we give to $\mathcal{A}$. We define the ciphertext $c := (\mathbf{g}^{\vec{y}}, \mathbf{g}^{\vec{y}\cdot\vec{x}_0^\top}\mathbf{m}_b, \mathbf{g}^{\vec{y}X})$ for a random $b \xleftarrow{\$} \{0,1\}$ that we choose. This allows us to simulate the rest of the game to the attacker, including the responses to the leakage and decryption queries, so that if the challenge is of type (I) the distribution is as in Game 0, and otherwise it is as in Game 1. Since we can also test whether the attacker wins $\tilde{b} \overset{?}{=} b$, an attacker that has a non-negligibly different probability of winning in Games 0 and 1 can be used to break the k-linear assumption.

Game' 1: In this game, when choosing A, we choose $u_i \xleftarrow{\$} \mathbb{F}_q \setminus \{0\}$ instead of $\mathbb{F}_q$. This is statistically indistinguishable from the previous game.

Game 2: In this game, the challenger will run in exponential-time. When the adversary submits the "decryption query" $c^* = (\mathbf{g}^{\vec{y}^*}, \mathbf{h}^*, \mathbf{g}^{\vec{v}^*})$, the challenger first checks if there is a $\vec{r}^*$ such that $\vec{y}^* = \vec{r}^* A$ (by computing discrete logarithms in exponential time) and outputs $\perp$ if not. Otherwise, it runs the original decryption check $\mathbf{g}^{\vec{y}^* X} \overset{?}{=} \mathbf{g}^{\vec{v}^*}$ and, if this passes, it outputs $\mathbf{m} = \mathbf{h}^* \mathbf{g}^{-\vec{r}^*(A\vec{x}_0^\top)}$. Note that the decryption procedure does not depend on $\vec{x}_0$ beyond $A\vec{x}_0$.

We show that games 1 and 2 are *statistically indistinguishable*, even if the adversary $\mathcal{A}$ is computationally unbounded.

Case 1: If the decryption query has $\vec{y}^* \in \mathsf{rowspan}(A)$ then the challenger's response is identical in Games 1 and 2.

Case 2: If the decryption query has $\vec{y}^* \notin \mathsf{rowspan}(A)$, then we argue that the response in Game 1 is $\perp$ with overwhelming probability (over the choice of the matrix X). To see this, we just apply Claim 5.5.3 to each column $\vec{x}_i$ of the matrix X to see that the values AX, $\vec{y}^* X$ are mutually uniform. Therefore

$$\widetilde{\mathbf{H}}_\infty(\vec{y}^* X \mid A, AX) = n \log(q)$$

On the other hand, the challenge-ciphertext c (as computed in Game 1) can reveal additional information about X and therefore also $\vec{y}^* X$. However, the adversary only gets ℓ bits of information about the challenge ciphertext c. Let *leak* be a random variable for the leakage observed by the attacker. Then (by Lemma 4.1.1)

$$\widetilde{\mathbf{H}}_\infty(\vec{y}^* X \mid A, AX, \mathit{leak}) \geq n \log(q) - \ell$$

So, in Game 1, the probability of the attacker making a decryption query of this type and not getting a response $\perp$ is the probability of the attacker guessing $\vec{v}^* = \vec{y}^* X$ given the public key and the leakage on the challenge ciphertext, which is at most $2^{\ell - n\log(q)} = \mathsf{negl}(\lambda)$.

So the probability of $\mathcal{A}$ winning Game 2 differs at most negligibly from that of $\mathcal{A}$ winning Game 0.

We now argue that, in Game 2, the challenger's bit b is perfectly hidden (even for a computationally unbounded adversary $\mathcal{A}$). This follows by applying Claim 5.5.3 to the vector $\vec{x}_0$ to see that the $(A\vec{x}_0^\top, \vec{y} \cdot \vec{x}_0^\top)$ is mutually uniform. Therefore, even given $A\vec{x}_0^\top$ as part of pk, the challenge ciphertext hides the message the message $\mathbf{m}_b$ since it is "one-time-padded" with a fresh uniform value $\vec{y} \cdot \vec{x}_0^\top$. Moreover, the answer to the decryption query in Game 2 can now be (inefficiently) simulated given $A\vec{x}_0^\top$ and hence they do not reveal any more information about $\vec{x}_0$.

So the adversary's probability of winning in Game 2 is *exactly* $\frac{1}{2}$ and, by the hybrid argument, the probability of winning in Game 0 must therefore be at most negligibly close to $\frac{1}{2}$, as we wanted to show. □

We notice that, in the proof of Theorem 5.5.2, we never made use of the fact that the adversary only submits *one* decryption query at the end of the game. Indeed, we achieve a stronger notion then just ℓ-LoC-NM security (closer to ℓ-LoC CCA-2 security). Even if the adversary can adaptively access a *decryption oracle* arbitrarily many times during the game, before and after leaking up to ℓ bits on the challenge ciphertext, the challenger's bit stays hidden.

5.5.3 Homomorphic NIZKs

Linear Equations in Exponent. We consider the language of *satisfiable systems of linear equations*, over some group $\mathbb{G}$ of primer order q with a generator $\mathbf{g}$. A system of m equations over n variables consists of a matrix of coefficient $\mathbf{g}^B \in \mathbb{G}^{m\times n}$ and a vector of target values $\mathbf{g}^{\vec{c}} \in \mathbb{G}^m$. We say that the system $(\mathbf{g}^B, \mathbf{g}^{\vec{c}})$ is *satisfiable* if there exists a vector $\vec{x} \in \mathbb{F}_q^n$ such that $\mathbf{g}^{B\vec{x}^\top} = \mathbf{g}^{\vec{c}}$. We call the vector $\vec{x}$, the *satisfying assignment*. We define the language L_{linear} of satisfiable systems $(\mathbf{g}^B, \mathbf{g}^{\vec{c}})$. We define the relation R_{linear} as consisting of all pairs $((\mathbf{g}^B, \mathbf{g}^{\vec{c}}), \vec{x})$, where the system $(\mathbf{g}^B, \mathbf{g}^{\vec{c}})$ is satisfiable and $\vec{x}$ is satisfying assignment. The system acts as a *statement* and the assignment acts as a *witness* in the sense that we can efficiently check that the statement is true given the witness. If we *fix* the coefficients $\mathbf{g}^B$, and define the relation

$$R^B_{linear} = \{(\mathbf{g}^{\vec{c}}, \vec{x}) \ : \ ((\mathbf{g}^B, \mathbf{g}^{\vec{c}}), \vec{x}) \in R_{linear}\}$$

then R^B_{linear} is a *linear relation* with the group operation

$$(\mathbf{g}^{\vec{c}}, \vec{x}) + (\mathbf{g}^{\vec{c}'}, \vec{x}') = (\mathbf{g}^{\vec{c}+\vec{c}'}, \vec{x} + \vec{x}').$$

For simplicity, we will say that a NIZK argument system for R_{linear} is homomorphic if, for every fixed choice of B, it is homomorphic for the (linear) relations R^B_{linear}.

Groth-Sahai NIZKs. The Groth-Sahai (GS) [GS08] NIZK argument system is (among other things) an argument system for the relation R_{linear}. For completeness, we give simple and short but self-contained description of GS NIZKs for this relation. The construction is shown below and relies on the matrix-in-the-exponent notation introduced in the Preliminaries. Let $k \in \mathbb{Z}$ be a parameter of the system.

$\mathsf{prms} \leftarrow \mathsf{ParamGen}(1^\lambda)$**:** Choose $\mathsf{prms} = (\mathbb{G}_1, \mathbb{G}_2, \mathbf{g}, \mathbf{h}, e, q) \leftarrow \mathcal{G}_{pair}(1^\lambda)$.

$\texttt{Setup}(\mathsf{prms})$**:** Choose $u_1, \ldots, u_k \overset{\$}{\leftarrow} \mathbb{F}_q$ and define

$$U := \begin{pmatrix} 1 & u_1 & 0 & \ldots & 0 \\ 1 & 0 & u_2 & \ldots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \ldots & u_k \end{pmatrix}$$

Choose $\vec{v} \overset{\$}{\leftarrow} \mathbb{F}_q^{k+1}$ and output $\textsc{crs} = (\mathbf{h}^U, \mathbf{h}^{\vec{v}})$.

$\texttt{SetupSim}(1^\lambda)$**:** Chooses U as in honest $\texttt{Setup}$.
Choose $\vec{r} \overset{\$}{\leftarrow} \mathbb{F}_q^k$ and output $\textsc{crs} = (\mathbf{h}^U, \mathbf{h}^{\vec{r}U}), \textsc{tk} = \vec{r}$.

$\texttt{Prov}_{\mathbf{crs}}((\mathbf{g}^B, \mathbf{g}^{\vec{c}}), \vec{x})$**:** Assume $\mathbf{g}^B \in \mathbb{G}^{m \times n}, \mathbf{g}^{\vec{c}} \in \mathbb{G}^m, \vec{x} \in \mathbb{F}_q^n$ and that $\mathbf{g}^{B\vec{x}^\top} = \mathbf{g}^{\vec{c}^\top}$.
Select $R \overset{\$}{\leftarrow} \mathbb{F}_q^{n \times k}$ and output

$$\pi := \left(\mathbf{h}^{\vec{x}^\top \cdot \vec{v}} \mathbf{h}^{RU} \ , \ \mathbf{g}^{BR} \right)$$

Note that π can be efficiently computed using the inputs and R (without knowing any of the exponents $B, \vec{c}, U, \vec{v}$).

$\texttt{Sim}_{\mathbf{crs}}((\mathbf{g}^B, \mathbf{g}^{\vec{c}}), \mathbf{tk})$**:** Assume $\mathbf{g}^B \in \mathbb{G}^{m \times n}, \mathbf{g}^{\vec{c}} \in \mathbb{G}^m$. Parse $\textsc{tk} = \vec{t} \in \mathbb{F}_q^k$.
Select $R \overset{\$}{\leftarrow} \mathbb{F}_q^{n \times k}$ and output:

$$\pi := \left(\mathbf{h}^{RU} \ , \ \mathbf{g}^{BR} \mathbf{g}^{-(\ \vec{c}^\top \cdot \vec{t}\ )} \right)$$

$\texttt{Ver}_{\mathbf{crs}}((\mathbf{g}^B, \mathbf{g}^{\vec{c}}), \pi)$**:** Parse $\pi = (\mathbf{h}^D, \mathbf{g}^P)$ and output 1 iff

$$e(\mathbf{g}^B, \mathbf{h}^D) \overset{?}{=} e(\mathbf{g}^{\vec{c}^\top}, \mathbf{h}^{\vec{v}}) e(\mathbf{g}^P, \mathbf{h}^U)$$

Theorem 5.5.4. *Fix a constant $k \geq 1$, and assume that the k-linear assumption holds for $\mathcal{G}_{pair}$. Then the above construction is a* homomorphic *NIZK for the relation R_{linear}.*

Proof. We analyze the properties of homomorphic NIZKs one by one.

Correctness. Assume that $\textsc{crs} \leftarrow \texttt{Setup}(1^\lambda)$ and $\pi \leftarrow \texttt{Prov}_{\textsc{crs}}((\mathbf{g}^B, \mathbf{g}^{\vec{c}}), \vec{x})$ where $\mathbf{g}^{B\vec{x}^\top} = \mathbf{g}^{\vec{c}^\top}$. Then $\pi = (\mathbf{h}^D, \mathbf{g}^P)$ for $D = \vec{x}^\top \cdot \vec{v} + RU$, $P = BR$, where R is some matrix. Therefore

$$\begin{aligned} e(\mathbf{g}^B, \mathbf{h}^D) &= e(\mathbf{g}, \mathbf{h})^{B(\vec{x}^\top \cdot \vec{v} + RU)} \\ &= e(\mathbf{g}, \mathbf{h})^{\vec{c}^\top \cdot \vec{v} + PU} = e(\mathbf{g}^{\vec{c}^\top}, \mathbf{h}^{\vec{v}}) e(\mathbf{g}^P, \mathbf{h}^U) \end{aligned}$$

(Statistical) Soundness. Let CRS $= (\mathbf{h}^U, \mathbf{h}^{\vec{v}}) \leftarrow \mathtt{Setup}(1^\lambda)$. Let $(\mathbf{g}^B, \mathbf{g}^{\vec{c}})$ be some statement and let $\pi = (\mathbf{h}^D, \mathbf{g}^B)$ some proof such that $\mathtt{Ver}_{\text{CRS}}((\mathbf{g}^B, \mathbf{g}^{\vec{c}}), \pi) = 1$. Then, since the proof verifies, we have

$$BD = \vec{c}^\top \cdot \vec{v} + PU = \left[\vec{c}^\top \mid P\right] \left[\frac{\vec{v}}{U}\right]$$

It is easy to see that the matrix $\left[\frac{\vec{v}}{U}\right]$ is invertible with overwhelming probability over the choice of the CRS. Let $M = \left[\frac{\vec{v}}{U}\right]^{-1}$ be its inverse. Then $BDM = [\vec{c}^\top \mid P]$ and therefore, letting $\vec{x}$ be the first column of DM, we have $B\vec{x} = \vec{c}^\top$, which shows that $(\mathbf{g}^B, \mathbf{g}^{\vec{c}})$ is a true statement.

Zero Knowledge. Firstly, the fact that the distributions of the CRS produced by `Setup` and `SetupSim` are indistinguishable follows directly from the k-linear assumption (matrix form). Secondly, we show that for any fixed choice of (CRS, TK) $\leftarrow$ `SetupSim(prms)`, honest and simulated proofs are distributed *identically* (over the coins of the prover/simulator). Let CRS $= (\mathbf{h}^U, \mathbf{h}^{\vec{v}})$ and TK $= \vec{t}$ such that $\vec{v} = \vec{r}U$. Let $(\mathbf{g}^B, \mathbf{g}^{\vec{c}})$ be some true statement and $\vec{x}$ a witness so that $B\vec{x}^\top = \vec{c}^\top$. Let $\pi = (\mathbf{h}^D, \mathbf{g}^P)$ be an honestly generated proof. Then the distribution of π is defined by:

$$\begin{aligned} D &= \vec{x}^\top \cdot \vec{v} + RU = \vec{x}^\top(\vec{r}U) + RU = (\vec{x}^T\vec{r} + R)U \\ P &= BR \end{aligned}$$

where R is uniformly random. Let us define $R' := (\vec{x}^T\vec{r} + R)$. Then the distribution of R' is just uniformly random (over the choice of R). Therefore the joint distribution of D, P above can be re-written as

$$\begin{aligned} D &= R'U \\ P &= B(R' - \vec{x}^T\vec{r}) = BR' - \vec{c}^\top\vec{r} \end{aligned}$$

for a uniformly random R'. But this is exactly how simulated proofs are chosen!

Homomorphism. Fix any CRS $= (\mathbf{g}^U, \mathbf{g}^{\vec{v}})$ and any $\mathbf{g}^B \in \mathbb{G}^{m\times n}$. Let the pairs $(\mathbf{g}^{\vec{c}}, \vec{x})$ and $(\mathbf{g}^{\vec{c}'}, \vec{x}')$ be such that $\mathbf{g}^{B\vec{x}^\top} = \mathbf{g}^{\vec{c}^\top}$ and $\mathbf{g}^{B\vec{x}'^\top} = \mathbf{g}^{\vec{c}'^\top}$. Let

$$\begin{aligned} \pi &= \mathtt{Prov}_{\text{CRS}}((\mathbf{g}^B, \mathbf{g}^{\vec{c}}), \vec{x}; R) = \left(\mathbf{h}^{\vec{x}^\top \cdot \vec{v}}\mathbf{h}^{RU} \ , \ \mathbf{g}^{BR} \right) \\ \pi' &= \mathtt{Prov}_{\text{CRS}}((\mathbf{g}^B, \mathbf{g}^{\vec{c}'}), \vec{x}'; R') = \left(\mathbf{h}^{\vec{x}'^\top \cdot \vec{v}}\mathbf{h}^{R'U} \ , \ \mathbf{g}^{BR'} \right) \end{aligned}$$

Then we can define the group operation:

$$\begin{aligned}
\pi + \pi' &= \left(\mathbf{h}^{\vec{x}^\top \cdot \vec{v}} \mathbf{h}^{RU} \mathbf{h}^{\vec{x}'^\top \cdot \vec{v}} \mathbf{h}^{R'U} \ , \ \mathbf{g}^{BR} \mathbf{g}^{BR'} \right) \\
&= \left(\mathbf{h}^{(\vec{x}^\top + \vec{x}'^\top) \cdot \vec{v}} \mathbf{h}^{(R+R')U} \ , \ \mathbf{g}^{B(R+R')} \right) \\
&= \texttt{Prov}_{\text{CRS}}((\mathbf{g}^B, \mathbf{g}^{\vec{c}+\vec{c}'}), \vec{x} + \vec{x}'; R + R')
\end{aligned}$$

Therefore the NIZK is homomorphic for the relation R_{linear}. $\square$

Proving Plaintext Equality. We now show how to use GS NIZKs for proving satisfiability of linear equations to prove plaintext equality for the encryption schemes $\mathcal{E}_1, \mathcal{E}_2$ presented earlier in this section (see Figure 5.5, and Figure 5.6 respectively). That is, we now show that the corresponding language for plaintext-equality L_{eq} can be expressed in terms of a satisfiable set of linear equations.

Let $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathbf{g}, \mathbf{h}) \leftarrow \mathcal{G}_{pair}(1^\lambda)$ be the common system parameters. Let pk_1, c_1 be a public-key and ciphertext of $\mathcal{E}_1$ and pk_2, c_2 be a public-key and ciphertext of $\mathcal{E}_2$, which we can write as:

$$\begin{array}{ll}
pk_1 = (\mathbf{g}^{a_1}, \ldots, \mathbf{g}^{a_k}) & c_1 = \left(\mathbf{g}^c, \mathbf{g}^{\vec{y}}\right) \\
pk_2 = (\mathbf{g}^A, \mathbf{g}^{\vec{b}}, \mathbf{g}^B) & c_2 = \left(\mathbf{g}^{\vec{y}'}, \mathbf{g}^{c'}, \mathbf{g}^{\vec{v}}\right).
\end{array}$$

(where the exponents are unknown to us). Moreover, $(c_1, c_2) \in L_{eq}$ if and only if there exist vector $\vec{r} = (r_1, \ldots, r_k), \vec{r}' = (r'_1, \ldots, r'_k)$ such that:

$$\begin{aligned}
(\mathbf{g}^{a_1 r_1}, \ldots, \mathbf{g}^{a_k r_k}) &= \mathbf{g}^{\vec{y}} \\
\mathbf{g}^{\vec{r}' A} &= \mathbf{g}^{\vec{y}'} \\
\mathbf{g}^{\vec{r}' B} &= \mathbf{g}^{\vec{v}} \\
\mathbf{g}^{(\sum_{i=1}^k r_i) - \vec{r}' \cdot \vec{b}^\top} &= \mathbf{g}^{c-c'} \qquad (\Leftrightarrow \exists \mathbf{m} \ : \ \mathbf{g}^c = \mathbf{g}^{\sum_{i=1}^k r_i} \mathbf{m}, \mathbf{g}^{c'} = \mathbf{g}^{\vec{r}' \cdot \vec{b}^\top} \mathbf{m})
\end{aligned}$$

But the above is just a system of linear equations in the exponent, and therefore can be expressed as a statement in the language L_{linear}. Moreover, the coefficient on the left-hand side $(\mathbf{g}^{a_1}, \ldots, \mathbf{g}^{a_k}), \mathbf{g}^A, \mathbf{g}^B, \mathbf{g}^{\vec{b}}$ only depend on (and can be efficiently computed from) pk_1, pk_2, while the target values on the right only depend on (and can be efficiently computed from) the ciphertexts c_1, c_2. Lastly, the witness $(\vec{r}, \vec{r}')$ only depends on (and can be efficiently computed from) the randomness used to generate the two ciphertexts.

Theorem 5.5.5. *Let $\mathcal{E}_1, \mathcal{E}_2$ be the encryption schemes from Figure 5.5, and Figure 5.6 respectively. Then, GS NIZKs provide a* homomorphic NIZK Π *for the plaintext equality relation R_{eq}.*

Proof. From the above discussion, for every statement $st = (pk_1, pk_2, c_1, c_2)$ there is an efficiently computable statement $\tilde{st} = f(st) = (\mathbf{g}^B, \mathbf{g}^{\vec{c}})$ such that $st \in L_{eq}$ satisfies plaintext equality iff $\tilde{st} \in L_{linear}$ is a satisfiable set of linear equations. Moreover if $w = (m, r_1, r_2)$ is the witness for st then there is an efficiently computable witness $\tilde{w} = g(w) = \vec{x}$ for $\tilde{st}$. Therefore, we can use GS NIZKs for linear-equations in the exponent R_{linear} as NIZK for plainest equality R_{eq}.

Lastly, the coefficients $\mathbf{g}^B$ in st' only depend on pk_1, pk_2 and the target $\mathbf{g}^{\vec{c}}$ only depends on c_1, c_2. Furthermore, if we fix pk_1, pk_2 and the corresponding $\mathbf{g}^B$, the maps f and g become homomorphic so that, if $st = (c_1, c_2), st' = (c_1', c_2')$ then $f(st + st') = f(st) + f(st')$ and $f(w + w') = f(w) = f(w')$. Therefore, for any fixed choice of pk_1, pk_2, the resulting NIZK is also homomorphic for the relation $R_{eq}^{pk_1, pk_2}$ since GS NIZKs are homomorphic for R_{linear}^B. □

5.5.4 The Final CLR-OWR Scheme

Putting everything together we get the following theorem.

Theorem 5.5.6. *Fix any constant $k \geq 1$ and assume that the k-linear assumption holds for some pairing $\mathcal{G}_{pair}$. Then, for any polynomial $\ell = \ell(\lambda)$, there exists an ℓ-CLR-OWR* with secure updates. *Moreover, for any constant $\epsilon > 0$, there is such a scheme with* fractional leakage $\frac{\ell}{|sk|} \geq \frac{1}{k+1} - \epsilon$. *In particular, under the SXDH assumption ($k = 1$), the fractional leakage approaches $\frac{1}{2}$.*

Proof. Under Theorem 5.4.9 we get the existence of ℓ-CLR-OWR assuming the existence of encryption schemes $\mathcal{E}_1, \mathcal{E}_2$ and a NIZK Π with the specified additional properties. Then Theorem 5.5.1, Theorem 5.5.2, and Theorem 5.5.4 show how to construct these schemes for any polynomial $\ell(\cdot)$. As for the fractional leakage, we notice that the secret key $sk = (c_2, \pi)$ consists of a ciphertext and a proof. Let $n \geq 1$ be a parameter of $\mathcal{E}_2$. Then the ciphertext c_2 is of size $n + k + 2 = n + O(1)$ group elements. The proof π is a proof for a system of $m = n + 2k + 2$ equations in $n' = 2k$ unknowns, which results in a proof of size $n'(k + 1) + mk = kn + O(1)$ group elements. Therefore the size of sk is $(k + 1)n + O(1)$ group elements and the leakage bound is $\ell \geq n \log(q) - \omega(\log(\lambda)) \geq (n - 1) \log(q)$ bits. Assuming each group element can be represented optimally using $\log(q) + O(1)$ bits, we get a fractional leakage of

$$\frac{\ell}{|sk|} = \frac{(n-1)\log(q)}{[(k+1)n + O(1)][\log(q) + O(1)]}$$

Therefore, for any constant $\epsilon > 0$ there is some sufficiently large n and q such that $\frac{\ell}{|sk|} \geq \frac{1}{k+1} - \epsilon$. □

5.6 Security with *Leaky* Updates

We now show that, generically, the weaker notion of security with leak-free updates also implies some level of security when the updates may leak as well. Unfortunately, the reduction runs in time exponential with the amount of leakage on the update process, and therefore under standard assumptions, we can only get *logarithmic* leakage of the updates. A result along these lines was proved by Brakerski et al. [BKKV10], but only for their specific scheme. The idea that this should hold generically was pointed out to us by Brent Waters (personal communication [Wat]). However, getting all the details right turns out to be trickier than anticipated. Here, we provide a formal result and proof.

Theorem 5.6.1. *Assume that for some $\ell = \ell(\lambda)$ there is an ℓ-CLR-OWR* with leak-free updates. *Let $\mu = \mu(\lambda) = O(\log(\lambda))$. Then the* same *construction is also a $\min(\mu, \ell)$-CLR-OWR with full security. More generally, if the original construction is secure against attackers running in (super-polynomial) time $t(\lambda)\mathsf{poly}(\lambda)$ then the above holds for any $\mu = \mu(\lambda) = O(\log(t(\lambda)))$.*

Proof Intuition. The main idea behind allowing logarithmic leakage-of-updates is that we can simulate the $\mu(\lambda)$-bits of leakage on the internal state (sk_i, r_i) of the refresh operation $sk_{i+1} = \mathsf{Update}(sk_i; r_i)$, by leaking $\mu(\lambda)$-bits on the secret-key sk_{i+1} alone, without knowing r_i. In particular, our leakage function will get sk_{i+1} and try *all possible choices* of the $\mu(\lambda)$-bit leakage that the adversary $\mathcal{A}$ could have seen on the full state. For each of the $2^{\mu(\lambda)}$ possible choices, our leakage-function it will *gage $\mathcal{A}$'s success probability* given this choice, by running $\mathcal{A}$ on many random continuations of the leakage game using the key sk_{i+1} to simulate the challenger going forward. At the end, the leakage-function will output the optimal $\mu(\lambda)$-bit value that maximizes $\mathcal{A}$'s probability. So leakage on (sk_i, r_i) is simulated using sk_{i+1} alone. Note that the *actual* behavior of the challenger in future rounds depends *only* on sk_{i+1}, so the estimate that the leakage-function gets on success probability of $\mathcal{A}$ is (likely) accurate. Therefore, given this "simulated" leakage, $\mathcal{A}$'s success probability should not be much smaller then when given the "correct" leakage calculated using (sk_i, r_i). The only bottleneck of the approach is that the simulated-leakage-function runs in $2^{\mu(\lambda)}$ time. Therefore, using this approach, we are stuck with $\mu(\lambda) = O(\log(\lambda))$ or with making stronger hardness assumptions.

Proof of Theorem 5.6.1. Let $(\mathsf{KeyGen}, \mathtt{Ver}, \mathsf{Update})$ satisfy the syntax of a CLR-OWR. Assume that there is a poly-time adversary $\mathcal{A}$ that breaks the full $\mu(\lambda)$-CLR security of the scheme, for some $\mu(\lambda) \leq \ell(\lambda)$. Then there are some polynomials $q(\cdot), p(\cdot)$ such that $\mathcal{A}$ always runs in time at most $q(\lambda)$ and that its success probability is at least $\frac{1}{p(\lambda)}$ for infinitely many $\lambda \in \mathbb{N}$. We define an adversary $\mathcal{B}$ that runs in time $2^{\mu(\lambda)}\mathsf{poly}(\lambda)$ and wins the restricted $\ell(\lambda)$-CLR security game *with leak-free* *updates* against the scheme with probability $\frac{1}{2p(\lambda)} - \mathsf{negl}(\lambda)$ for infinitely many λ.

Description of Reduction $\mathcal{B}$. It is easiest to think of $\mathcal{A}$ as submitting a single deterministic leakage-query h_i on the state sk_i, r_i in each round with output-length μ. Recall that this is w.l.o.g. as any adversary that submits multiple adaptive randomized queries can always be converted into an adversary making a single deterministic query in each round. In particular, the adversary can choose the randomness for the query itself beforehand, and can combine several adaptive queries into one. The adversary $\mathcal{B}$ works as follows. It runs an internal copy of $\mathcal{A}$ and forwards the public-key *pk* from its challenger to $\mathcal{A}$. When $\mathcal{A}$ submits a leakage query $h_i : \{0,1\}^* \to \{0,1\}^\mu$ on the full state (sk_i, r_i), the reduction $\mathcal{B}$ moves on to the next leakage round and leaks μ bits on the secret key sk_{i+1}. In particular, $\mathcal{B}$ constructs a query $h'_i : \{0,1\}^* \to \{0,1\}^\mu$ on the key sk_{i+1}, as described below, and feeds the output $h'_i(sk_{i+1})$ to $\mathcal{A}$. At the end, $\mathcal{B}$ outputs whatever secret-key sk^* is output by $\mathcal{A}$.

Description of the function $h'_i : \{0,1\}^* \to \{0,1\}^\mu$.

The function h'_i contains a hard-coded copy of $\mathcal{A}$, in its current state after issuing the query h_i. Let $\epsilon(\lambda) = \frac{1}{2p(\lambda)}$.

Main Computation: Try all 2^μ possible values of $\psi \in \{0,1\}^\mu$. For each ψ, find an *estimate* $\tilde{\rho}_\psi \in [0,1]$ for the success probability of $\mathcal{A}$ with the leakage ψ, as described below. Output the choice of ψ that corresponds to the *maximal* estiamte $\tilde{\rho}_\psi$.

Estimating $\tilde{\rho}_\psi$ using sk_{i+1}: Perform the following random experiment independently for $k = 8q^2(\lambda)(\lambda+\mu)/\epsilon^2(\lambda)$ times:

Choose fresh random coins $r_{i+1}, \ldots, r_{q(\lambda)-1}$ and compute the keys

$$sk_{i+2} = \mathsf{Update}(sk_{i+1}; r_{i+1}), \ldots, sk_{q(\lambda)} = \mathsf{Update}(sk_{q(\lambda)-1}, r_{q(\lambda)-1}).$$

Then run the rest of the leakage game with $\mathcal{A}$ by feeding it ψ in response to h_i, and use the values $sk_{i+1}, r_{i+1}, sk_{i+2}, \ldots, r_{q(\lambda)-1}, sk_{q(\lambda)}$ to correctly respond to all of $\mathcal{A}$'s future queries h_j for $j > i$.
(Recall $\mathcal{A}$ makes at most $q(\lambda)$ queries.)

Set $\tilde{\rho}_\psi$ to be the fraction of the k experiments in which $\mathcal{A}$ "wins" at the end.

Analysis of Reduction $\mathcal{B}$. It is easy to see that $\mathcal{B}$ runs a legal strategy since the number of bits leaked in each round i is $\mu \leq \ell$. Furthermore, the run-time of B is at most $2^\mu\mathsf{poly}(\lambda)$; recall that $\mathcal{B}$ needs to run as long as all of its leakage functions. So we are left to analyze the success probability of $\mathcal{B}$. To do so, it is useful to consider the following hybrid experiments.

Define **Experiment** 0 to be the μ-CLR OWR game between the challenger and the adversary $\mathcal{A}$. Define experiments $1, \ldots, q(\lambda)$ analogously, but in **Experiment** i, the first i leakage queries $h_0, \ldots, h_{i-1}$ asked by $\mathcal{A}$ are answered via $h'_j(sk_{j+1})$ instead of $h_j(sk_j, r_j)$. Let W_i be the event that $\mathcal{A}$ wins in **Experiment** i. Then $\mathcal{B}$'s success probability is exactly $\Pr[W_{q(\lambda)}]$. Let $\mathsf{Pre}_{i,j}$ denote the random variable for the "preamble" of experiment i up to the "point" j, consisting of:

- The view of $\mathcal{A}$ right after making the leakage query h_j, but before seing the response. This includes the random coins of $\mathcal{A}$, the public-key pk seen by $\mathcal{A}$, and all of the responses to all of the prior leakage queries.

- The value of the secret key sk_j used in round j and the randomness r_j used to compute $sk_{j+1} = \mathsf{Update}(sk_j; r_j)$.

Conditioned on the preambles $\mathsf{Pre}_{i,i}$ and $\mathsf{Pre}_{i+1,i}$ of experiments i and $i+1$ taking on some concrete value σ, the only difference between experiments i and $i+1$ is how the leakage query h_i is answered. In the former case, it is answered via $h_i(sk_i, r_i)$, and in the latter case it is answered via $h'_i(sk_{i+1})$, where sk_i, r_i, sk_{i+1} are fixed by σ. The following claim intuitively says that, since h'_i chooses its response by trying all options and testing, its answer should not be much worse than that of h_i.

Claim 5.6.2. *For any σ in the support of $\mathsf{Pre}_{i+1,i}$ and any $i \in \{0, \ldots, q(\lambda)-1\}$, we have*

$$\Pr[W_{i+1} \mid \mathsf{Pre}_{i+1,i} = \sigma] \geq \Pr[W_i \mid \mathsf{Pre}_{i,i} = \sigma] - \epsilon(\lambda)/q(\lambda) - \mathsf{negl}(\lambda).$$

Proof of Claim 5.6.2. In the proof we fix σ and always condition all probabilities on the preamble of the experiment being σ; we will use the variables $W_{i,\sigma}$ and $W_{i+1,\sigma}$ to make this conditioning explicit. Recall that σ fixes sk_i, r_i, sk_{i+1}. Let $\psi^* = h_i(sk_i, r_i)$ and let $\rho^* = \Pr[W_{i,\sigma}]$. Define the random variable A to be the value of $h'_i(sk_{i+1})$ in experiment $i+1$, over the internal randomness of h'_i. Define $\rho_\psi = \Pr[W_{i+1,\sigma} \mid A = \psi]$. Then, it is easy to see that $\rho_{\psi^*} = \rho^*$ since, conditioned on the response to h_i being ψ, the continuation of the two experiments $i+1$ and i are exactly the same. Let us define the set $Good = \{\psi \ : \ \rho_\psi \geq \rho^* - \epsilon(\lambda)/q(\lambda)\}$. Then:

$$\begin{aligned}
\Pr[W_{i+1,\sigma}] &\geq \sum_{\psi \in Good} \Pr[W_{i+1,\sigma} \mid A = \psi] \Pr[A = \psi] \\
&\geq (\rho^* - \epsilon(\lambda)/q(\lambda)) \Pr[A \in Good] \qquad (5.2)
\end{aligned}$$

We now show that $\Pr[A \in \mathit{Good}]$ is high. To do so, recall that the function $h'_i(sk_{i+1})$ chooses the response ψ based on the estimated success probabilities $\tilde{\rho}_\psi$. Since $\tilde{\rho}_\psi$ is computed by averaging k experiments each of which succeeds with probability ρ_ψ, we can use the Chernoff bound to get:

$$\Pr[|\tilde{\rho}_\psi - \rho_\psi| \geq \epsilon(\lambda)/2q(\lambda)] \leq 2e^{-k\epsilon^2(\lambda)/8q^2(\lambda)} \leq 2e^{-\lambda}2^{-\mu} \tag{5.3}$$

Using the Union Bound over all $\psi \in \{0,1\}^\mu$, we see that *every estimate* $\tilde{\rho}_\psi$ computed internally by h'_i satisfies $|\tilde{\rho}_\psi - \rho_\psi| < \epsilon(\lambda)/2q(\lambda)$ with probability at least $(1 - 2e^{-\lambda})$ over the internal coins of h'_i . If that occurs then:

- For the "correct" $\psi^* = h_i(sk_i, r_i)$, we have $\tilde{\rho}_{\psi^*} > \rho_{\psi^*} - \epsilon(\lambda)/2q(\lambda) = \rho^* - \epsilon(\lambda)/2q(\lambda)$.
- For the "actual" ψ' output by h'_i, we have $\tilde{\rho}_{\psi'} \geq \tilde{\rho}_{\psi^*}$ and $\tilde{\rho}_{\psi'} < \rho_{\psi'} < \epsilon(\lambda)/2q(\lambda)$.
- Putting these together, we get: $\tilde{\rho}_{\psi'} > \rho^* - \epsilon(\lambda)/q(\lambda)$ and hence the output is $\psi' \in \mathit{Good}$.

So $\Pr[A \in \mathit{Good}] \geq (1 - 2e^{-\lambda}) = (1 - \mathsf{negl}(\lambda))$ and hence, plugging this into (5.2) and expanding out, we get

$$\Pr[W_{i+1,\sigma}] \geq \Pr[W_{i,\sigma}] - \epsilon(\lambda)/q(\lambda) - \mathsf{negl}(\lambda)$$

Using the above claim, we get that for each $i \in \{0, \ldots, q(\lambda)\}$:

$$\begin{aligned}
\Pr[W_{i+1}] &= \sum_{\sigma \in \{0,1\}^*} \Pr[W_{i+1} \mid \mathsf{Pre}_{i+1,i} = \sigma] \Pr[\mathsf{Pre}_{i+1,i} = \sigma] \\
&\geq \sum_{\sigma \in \{0,1\}^*} (\Pr[W_i \mid \mathsf{Pre}_{i,i} = \sigma] - \epsilon(\lambda)/q(\lambda) - \mathsf{negl}(\lambda)) \Pr[\mathsf{Pre}_{i,i} = \sigma] \\
&\geq \Pr[W_i] - \epsilon(\lambda)/q(\lambda) - \mathsf{negl}(\lambda)
\end{aligned}$$

where the second line follows from the claim and the fact that $\mathsf{Pre}_{i+1,i}$ and $\mathsf{Pre}_{i,i}$ have the exact same distribution (the preambles are the same in both experiments). Therefore

$$\Pr[\mathcal{B} \text{ wins }] = \Pr[W_{q(\lambda)}] \geq \Pr[W_0] - \epsilon(\lambda) - \mathsf{negl}(\lambda) = \Pr[\mathcal{A} \text{ wins }] - \epsilon(\lambda) - \mathsf{negl}(\lambda)$$

which proves the theorem.

It is easy to extend the above proof to other CLR primitives, such as encryption, signatures etc. as long as there is an efficient way to *test* whether the attacker wins.

Corollary 5.6.3. *Fix any constant $k \geq 1$, and assume the k-linear assumption holds for some pairing $\mathcal{G}_{pair}$. Then, for any $\mu(\lambda) = O(\log(\lambda))$ there exists $\mu(\lambda)$-CLR OWR with full security. Further, if we assume that the k-linear assumption is secure against adversaries running in time $t(\lambda)\mathsf{poly}(\lambda)$, then the above also holds for $\mu(\lambda) = O(\log(t(\lambda)))$.*

5.7 Signatures

Given a CLR-OWR, its relatively easy to build signatures using a variety of techniques from several previous works. Let us begin with a definition.

Definition 5.7.1. *We say that a signature scheme* $(\mathsf{KeyGen}, \mathtt{Sign}, \mathtt{SigVer}, \mathsf{Update})$ *is* ℓ-continuous-leakage-resilient (ℓ-CLR) *if for any PPT adversary* $\mathcal{A}$*, we have* $\Pr[\mathcal{A}$ *wins* $] \leq \mathsf{negl}(\lambda)$ *in the following game:*

- *The challenger chooses* $(vk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and gives* vk *to* $\mathcal{A}$*. It chooses randomness* ω *for the update and sets* $\mathsf{state} = (sk, \omega)$ *and* $L := 0$*.*
- *The adversary* $\mathcal{A}$ *can adaptively interleave the following queries:*
  - *If* $L < \ell$*, then* $\mathcal{A}$ *can make a query to the leakage-oracle* $\mathcal{O}_{\mathsf{state}}(\cdot)$ *and get the appropriate response. The challenger sets* $L := L + 1$*.*
  - *The adversary can makes signing queries. Given a message* m *the challenger parses* $\mathsf{state} = (sk, \ldots)$*, chooses randomness* r_{sign}*, computes* $\sigma = \mathtt{Sign}_{sk}(m; r_{sign})$ *and sets* $\mathsf{state} := (\mathsf{state}, r_{sign})$*.*
  - *The adversary can makes update queries. The challenger parses* $\mathsf{state} = (sk, \omega, \ldots)$ *and computes* $sk' = \mathsf{Update}(sk; \omega)$*. It chooses new update randomness* ω' *and sets* $\mathsf{state} := (sk', \omega')$ *and* $L := 0$*.*
- *The adversary* wins *if it produces a message/signature pair* (m^*, σ^*) *such that* $\mathtt{SigVer}_{vk}(m^*, \sigma^*) = 1$ *and* m^* *was never contained in any signing query.*

We can also define **security with leak-free updates** *where* ω *is not made part of the* state*. We can also define* **security with leak-free signing** *where* r_{sign} *is not concatenated to the state.*

$\mathsf{KeyGen}(1^\lambda)$**:** Run $(pk, sk) \leftarrow \mathsf{KeyGen}^{\mathcal{C}}(1^\lambda)$, $(\textsc{crs}, \cdot) \leftarrow \mathtt{Setup}^{\Psi}(1^\lambda)$. Output: $vk := (pk, \textsc{crs})$, sk.
$\mathtt{Sign}_{sk}(m)$**:** Output $\sigma \leftarrow \mathtt{Prov}^{\Psi}((pk, m), sk)$.
$\mathtt{SigVer}_{vk}(m, \sigma)$**:** Output $\mathtt{Ver}^{\Psi}((pk, m), \sigma)$.
$\mathsf{Update}(sk)$**:** Run the re-randomization procedure of the CLR-OWR $\mathcal{C}$.

Figure 5.7: A CLR-Signature Scheme from a CLR-OWR and a tSE NIZK

Our construction is based on the leakage-resilient signature scheme of [DHLW10a], which slightly generalizes the original scheme of [KV09]. We start with a CLR-OWR $\mathcal{C} = (\mathsf{KeyGen}^{\mathcal{C}}, \mathsf{Update}, \mathtt{Ver}^{\mathcal{C}})$, and a NIZK $\Psi = (\mathtt{Setup}^{\Psi}, \mathtt{Prov}^{\Psi}, \mathtt{Ver}^{\Psi})$ for the relation

$$R_{\mathcal{C}} = \{(y, x) \mid y = (pk, m), \quad x = sk \quad \text{s.t.} \quad \mathtt{Ver}^{\mathcal{C}}(pk, sk) = 1\}.$$

The signature scheme construction is shown in Figure 5.7. The main idea is to re-use the key pair (sk, pk) from a CLR-OWR for the Signature scheme. To sign a message m, the signer produces an "extractable NIZK" certifying that she knows a valid secret key sk for the public key pk, where the message m is made a part of the statement.

For security, we need the NIZK system Ψ to satisfy a new notion of security called *true-simulation extractable (tSE)* (defined in our work [DHLW10b]); even if an adversary sees simulated proofs of arbitrary true statements $y \in L_R$, if she produces a valid proof ψ^* for a new statement y^*, then there is a way to *extract* a valid witness x^* from ψ^*, so that $(y^*, x^*) \in R$. Notice that this explains the (seemingly useless) role of m in the relation $R_{\mathcal{C}}$. Even if the adversary sees simulated proofs for statements that contain many different values m, any proof she produces for a new m^* is extractable. W

Definition 5.7.2 (True-Simulation Extractability [DHLW10b])**.** *Let*

$$\Psi = (\texttt{Setup}, \texttt{Prov}, \texttt{Ver}, \texttt{Sim})$$

be an NIZK argument for an NP relation R, satisfying the completeness, soundness and zero-knowledge properties. We say that Ψ is true-simulation extractable *(tSE) if:*

- *Apart from outputting a CRS and a trapdoor key,* `SetupSim` *also outputs an extraction key:* $(\text{CRS}, \text{TK}, \mathsf{ek}) \leftarrow \texttt{SetupSim}(1^\lambda)$.
- *There exists a PPT algorithm* $\mathsf{Ext}_{\mathsf{ek}}$ *such that for all $\mathcal{A}$ we have* $\Pr[\mathcal{A}\ wins] \leq negl(\lambda)$ *in the following game:*
    1. *Challenger runs* $(\text{CRS}, \text{TK}, \text{EK}) \leftarrow \texttt{SetupSim}(1^\lambda)$ *and gives* CRS *to $\mathcal{A}$.*
    2. $\mathcal{A}^{\mathcal{SIM}_{\text{TK}}(\cdot)}$ *is given access to a simulation oracle $\mathcal{SIM}_{\text{TK}}(\cdot)$, which it can adaptively access. A query to the simulation oracle consists of a pair (y, x). The oracle checks if $(y, x) \in R$. If true, it ignores x and outputs a simulated argument* $\texttt{Sim}_{\text{TK}}(y)$. *Otherwise, the oracle outputs $\perp$.*
    3. *$\mathcal{A}$ outputs a pair (y^*, ψ^*), and the challenger runs* $x^* \leftarrow \mathsf{Ext}_{\mathsf{ek}}(y^*, \psi^*)$.

  $\mathcal{A}$ wins if $(y, x^*) \notin R$, $\texttt{Ver}(y^*, \psi^*) = 1$*, and y^* was not part of a query to the simulation oracle.*

We note that, as observed in [DHLW10a], tSE NIZKs can be constructed by either (1) composing a *simulation-sound* NIZK with a CPA-secure encryption, yielding the scheme of [KV09], or (2) composing a standard NIZK with a CCA-secure encryption, yielding a (possibly) more efficient scheme. See [DHLW10b] for more details of the construction. For now it suffices to note that such NIZKs are shown to exist based on the k-linear assumption.

We are now ready to prove that the CLR signature construction in Figure 5.7 achieves *security with leak-free signing.*

Theorem 5.7.3. *If* $\mathcal{C} = (\mathsf{KeyGen}^{\mathcal{C}}, \mathsf{Update}, \mathtt{Ver}^{\mathcal{C}})$ *is an* ℓ*-CLR-OWR, and* $\Psi = (\mathtt{Setup}^{\Psi}, \mathtt{Prov}, \mathtt{Ver}^{\Psi}, \mathtt{SetupSim}, \mathtt{Sim}, \mathsf{Ext})$ *is a tSE-NIZK argument for relation*

$$R_{\mathcal{C}} = \{(y, x) \mid y = (pk, m), \quad x = sk \quad s.t. \quad \mathtt{Ver}^{\mathcal{C}}(pk, sk) = 1\}$$

then the signature scheme in Figure 5.7 is ℓ*-CLR secure* with leak-free signing. *Furthermore if* $\mathcal{C}$ *is only* ℓ*-CLR OWR* with leak-free updates *then the signature scheme is* ℓ*-CLR secure with* leak-free signing and updates*.*

Proof. We use a series-of-games argument to prove the above theorem.

Game 0: This is the original ℓ-CLR attack game described in Definition 5.7.1.

Game 1: In this game, the CRS is chosen as a simulated one $(\text{CRS}, \text{TK}, \mathsf{ek}) \leftarrow \mathtt{SetupSim}(1^{\lambda})$. The queries to the signing oracle are answered with simulated arguments: $\sigma \leftarrow \mathtt{Sim}_{\text{TK}}(pk, m)$. Games 0 and 1 are indistinguishable by the *zero-knowledge* property of the argument Ψ. Notice that the simulated arguments given to $\mathcal{A}$ as answers to signing queries are always of true statements.

Game 2: In this game, we modify the winning condition so that the adversary only wins if it produces a valid forgery (m^*, σ^*) *and* the challenger is able to extract a valid secret key sk^* for pk from (m^*, σ^*). That is, $\mathcal{A}$ wins if $\mathtt{SigVer}(m^*, \sigma^*) = 1$ *and* $\mathtt{Ver}^{\mathcal{C}}(pk, sk^*) = 1$, where $sk^* \leftarrow \mathsf{Ext}_{\mathsf{ek}}((pk, m^*), \sigma^*)$. The winning probability of $\mathcal{A}$ in Game 2 is at least that of Game 1 minus the probability that $\mathtt{Ver}^{\Psi}((pk, m^*), \sigma^*) = 1 \wedge \mathtt{Ver}^{\mathcal{C}}(pk, sk^*) = 0$. By the *true-simulation extractability* of the argument Ψ we know that this probability is negligible. Therefore, the winning probability of $\mathcal{A}$ in Game 2 differs from that in Game 1 by a negligible amount.

We have shown that the probability that $\mathcal{A}$ wins in Game 0 is the same as that in Game 2, up to negligible factors. We now argue that the probability that $\mathcal{A}$ wins in Game 2 is negligible, which proves that the probability that $\mathcal{A}$ wins in Game 0 is negligible as well. To prove that the probability that $\mathcal{A}$ wins in Game 2 is negligible, we assume otherwise and show that there exists a PPT algorithm $\mathcal{B}$ that breaks the ℓ*-continuous-leakage resilience* of $\mathcal{C}$. On input pk, $\mathcal{B}$ generates $(\text{CRS}, \text{TK}, \mathsf{ek}) \leftarrow \mathtt{Setup}^{\Psi}(1^{\lambda})$ and emulates $\mathcal{A}$ on input $vk = (\text{CRS}, pk)$. In each leakage round, $\mathcal{B}$ answers $\mathcal{A}$'s leakage queries using the leakage oracle $\mathcal{O}_{\mathsf{state}}(\cdot)$ and answers signing queries m_i by creating simulated arguments $\mathtt{Sim}_{\text{TK}}(pk, m_i)$. When $\mathcal{A}$ outputs her forgery (m^*, σ^*), $\mathcal{B}$ runs $sk^* \leftarrow \mathsf{Ext}_{\mathsf{ek}}((pk, m^*), \sigma^*)$ and outputs sk^*. Notice that $\Pr[\mathcal{B} \text{ wins}] = \Pr[\mathcal{A} \text{ wins}]$, so that if $\Pr[\mathcal{A} \text{ wins}]$ is non-negligible then

$\mathcal{B}$ breaks the ℓ-continuous-leakage resilience of $\mathcal{C}$. We therefore conclude that the probability that $\mathcal{A}$ wins in Game 2 is negligible. This concludes the proof of the theorem. □

The following corollary follows from Theorem 5.7.3 together with the work of [DHLW10b] showing the existence of tSE NIZKs for NP under the k-linear assumption.

Corollary 5.7.4. *Under the k-linear assumption on a pairing $\mathcal{G}_{pair}$, for any polynomial $\ell = \ell(\lambda)$ and any constant ϵ there exist ℓ-CLR signature schemes with* leak-free signing and updates *where the relative leakage is $\frac{\ell}{|sk|} \geq \frac{1}{k+1} - \epsilon$. Moreover, for any $\ell(\lambda) = O(\log(\lambda))$ there exist ℓ-CLR signatures with* leak-free signing *(but not updates).*

Signatures with *Leaky* Signing. Allowing leakage on the randomness of the signing operations is addressed in several recent works. We briefly describe some of the techniques without providing formal theorems and proofs. The main difficulty is that, although a tSE NIZK allows us to simulate signing queries and extract from a forgery, this only holds if the attacker does not see any information about the randomness of the NIZK. Otherwise, the attacker may distinguish simulated and real proofs. There are several strategies for getting around this.

Firstly, in [DHLW10a], we show how to get a scheme in the *random-oracle model* that provides full security. This essentially uses a Σ-protocol for proving knowledge of a secret key for a CLR-OWR and then compiles it into a signature scheme using the *Fiat-Shamir* heuristic [FS86] (this approach was originally suggested for getting such signatures in the bounded-leakage model by [ADW09a, KV09]). Since Σ-protocols are statistically witness indistinguishable, leaking on the secret-key (witness) and the random-coins of the protocol does not provide any *more* information than leaking on the secret key itself (in an information theoretic sense). The above approach was generalized in [BSW11] to relying on NIZKs which are simultaneously *extractable* and *statistically witness indistinguishable (WI)*. Unfortunately, this can only occur in the Random-Oracle model. However, [BSW11] showed that a variant of this approach can be instantiated in the standard model by relying on NIZKs whose parameters partition the message-space into messages for which the corresponding NIZK is extractable and those for which it is statistically WI. By using a careful partitioning strategy this approach can be made to work. The concurrent work of [MTVY11] can also be seen as providing an instantiation of the above strategy for a particular number-theoretic scheme. Lastly, the work of [JGS11] shows that a variant of tSE NIZKs with a strong version of *adaptive security* can also be used to get leakage on the signing randomness.

5.8 Advanced Primitives

Now that we constructed CLR OWR and Signatures, we can use these to build more advanced CLR primitives. Perhaps the two most useful ones are identification schemes (ID) and authenticated key agreement (AKA). We describe these primitives, their security properties and construction only at a high level since all the details are relatively straight-forward (and tediously boring). However, the interested reader can see [DHLW10a] for details.

5.8.1 Identification Schemes (ID)

In an (interactive) identification scheme a *prover* $P(pk, sk)$ owning a public/secret key pair (pk, sk) attempts to prove his identity to a *verifier* $V(pk)$ knowing only the public key pk. Security requires that even if an attacker gets to interact with the honest prover (in an arbitrarily malicious manner, arbitrarily many times), she will later be unable to *impersonate* the prover by running an *accepting* interaction with $V(pk)$. In a CLR variant of the definition, the honest prover also periodically updates the secret key sk (without modifying pk). Security is strengthened by allowing the adversary to leak up to ℓ bits on the state of the honest prover (including the randomness of the updates and the proof operations) in each time period between updates. The formal definition is analogous to that of signatures (Definition 5.7.1) and we can define variants with leak-free updates and proof operations.

We can immediately get CLR ID schemes from CLR Signatures. In the protocol, the verifier just chooses a random message m from a sufficiently large (super-polynomial) message space and the prover signs the message using its secret key sk. However, in order to get full security with leakage of the proof operations, we need the signature scheme to be secure for leakage of the signing randomness, and we saw that this was relatively difficult to achieve. An alternative construction of CLR ID schemes directly from CLR OWR is also possible. The prover simply runs a Σ-protocol to prove that it knows the corresponding secret key sk for a public key pk of a CLR OWR.

5.8.2 Authenticated Key Agreement(AKA)

In an AKA protocol, all parties have individual (long-term) public/secret keys. Any two parties (Alice and Bob) can use these to run a protocol in which they agree on a shared *ephemeral session key.* The basic security of the protocol considers a man-in-the-middle attacker who can arbitrarily modify/insert/delete protocol messages. Security dictates that if Alice is attempting to run such a protocol with an honest party Bob then the resulting session key that Alice generates will look uniformly random to the attacker. Furthermore, Bob will generate the same exact session key or none at all . Lastly, such protocols are forward secure if the session key looks uniformly random even if the attacker learns the full secret key of Alice and Bob *after* the protocol execution. The definition of Canetti and Krawczyk [CK01] makes the above formal. In the CLR setting, parties can also periodically update their secret keys and the attacker can learn up to ℓ bits of information on the internal state of each party in each time period. However, the standard security properties will hold for all such sessions during which no leakage occurs (even if leakage occurs in many prior executions and the secret keys are fully revealed to the attacker later).

To construct CLR AKA, the parties use a CLR signature scheme to set up a public-key infrastructure (PKI). Then, any pair of parties can agree on a fresh ephemeral session keys, by running a passively-secure key agreement protocol (such as the Diffie-Hellman key agreement), and authenticating the flows of the protocol by signing them.

Chapter 6

CLR Encryption

We now show how to construct Continuous Leakage Resilient (CLR) Public-Key Encryption (PKE) schemes tolerating arbitrary polynomial amounts of leakage on the secret key and the update randomness (full security). As mentioned earlier, this also yields an alternate construction of a CLR-OWR which achieves full security. The main drawback is that the construction is less generic than the previous one, relying on number theoretic assumptions from the beginning rather than providing simpler intermediate abstractions.

As mentioned in Chapter 3, the first CLR-PKE scheme was constructed by Brakesrski et al. [BKKV10] achieving security with leak-free updates and arbitrarily large leakage or full security with logarithmic leakage (e.g. similarly to our results for CLR-OWR). The work of Lewko et al. [LLW11] provides new techniques to deal with arbitrarily large leakage on the updates, but at the cost of a (conceptually) more complicated scheme in composite-order bilinear groups and relying on less standard assumptions in such groups. In this thesis, we present a new scheme which gets the best of both worlds: the simplicity of the [BKKV10] scheme (indeed, our scheme looks quite similar) together with the stronger security guarantees [LLW11] under (only) the k-linear assumption.

For simplicity, we begin by describing and analyzing a scheme under the SXDH assumption (1-linear). We then briefly discuss how to extend it to the k-linear assumption for arbitrary $k \geq 1$.

6.1 Definition

A CLR *public key encryption* (PKE) scheme consists of the efficient algorithms $(\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt}, \mathsf{Update})$ with the syntax:

$(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ **:** Outputs the key pair (pk, sk). The public key pk defines a message space $\mathcal{M}_{pk}$.

$c \leftarrow \mathsf{Encrypt}_{pk}(m)$ **:** Produces a ciphertext c for the plaintext message $m \in \mathcal{M}_{pk}$ using the public key pk.

$m \leftarrow \mathsf{Decrypt}_{sk}(c)$ **:** Decrypts the ciphertext c and outputs the corresponding plaintext m using the secret key sk.

$sk' \leftarrow \mathsf{Update}(sk)$ **:** Updates the secret key sk into a new key sk'.

For convenience, we also implicitly define the algorithm that performs $i \geq 0$ consecutive updates via:

$sk' \leftarrow \mathsf{Update}^i(sk)$ **:** Let $sk_0 = sk$, $sk_1 \leftarrow \mathsf{Update}(sk_0), \ldots sk_i \leftarrow \mathsf{Update}(sk_{i-1})$. Output $sk' = sk_i$.

Definition 6.1.1 (CLR PKE)**.** *A scheme with the given syntax is an ℓ-CLR PKE if it satisfies the following correctness and security properties:*

Perfect Correctness: *For all polynomial $i = i(\lambda) \geq 0$, all $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $sk' \leftarrow \mathsf{Update}^i(sk)$, $m \in \mathcal{M}_{pk}$, $c \leftarrow \mathsf{Encrypt}_{pk}(m)$, we get $\mathsf{Decrypt}_{sk'}(c) = m$.*

Security: *For any poly-time attacker $\mathcal{A}$ we have $\Pr[\mathcal{A} \text{ wins }] \leq \mathsf{negl}(\lambda)$ in the following game:*

- *Challenger samples $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$, gives pk to $\mathcal{A}$. It samples update randomness ω and sets $\mathsf{state} := (sk, \omega)$, $L := 0$.*
- *$\mathcal{A}$ can adaptively interleave any number of the following queries:*

  Leakage Queries: *If $L < \ell$ then $\mathcal{A}$ can make a query to the oracle $\mathcal{O}_{\mathsf{state}}(\cdot)$ and gets a response. The challenger sets $L := L + 1$.*

  Update Queries: *On an update query, the challenger parses $\mathsf{state} = (sk, \omega)$. It computes $sk' := \mathsf{Update}(sk; \omega)$ and samples new randomness ω'. Finally it sets $\mathsf{state} = (sk', \omega')$ and $L := 0$.*

- *$\mathcal{A}$ chooses two messages $m_0, m_1 \in \mathcal{M}_{pk}$.*
- *Challenger chooses $b \xleftarrow{\$} \{0, 1\}$, $c \leftarrow \mathsf{Encrypt}_{pk}(m_b)$ and gives c to $\mathcal{A}$.*
- *$\mathcal{A}$ outputs a bit $\tilde{b}$ and* wins *if $\tilde{b} \stackrel{?}{=} b$.*

We can also define a weaker variant of the above definition, called **ℓ-CLR PKE with Leak-Free Updates**, *where the variable* state *only contains the secret key sk and* not *the randomness ω.*

Remarks. Note that in the above definition, the attacker can only get leakage prior to seeing the challenge ciphertext. This is a necessary restriction in the CLR setting since, after seing the challenge ciphertext, the attacker can always simply leak the first bit of the decrypted message to break security. Also note that the above definition provides full security where the attacker can leak on the entire secret state of a decryption device, including its secret key and update randomness. Since decryption operations are deterministic, the above components are sufficient to describe the entire secret state of the decryption device at any point in time.

6.2 Hiding Ranks and Subspaces

In this section we prove various abstract indistinguishability lemmas about (statistically) hiding subspaces given leakage on some of their vectors and (computationally) hiding the rank of matrices "in the exponent". The reader may wish to skip this section on first reading and look at these lemmas on an "as needed" basis.

Hiding Subspaces. The following lemma says that, given some sufficiently small leakage on a random matrix A, it is hard to distinguish random vectors from $\mathsf{colspan}(A)$ from uniformly random and independent vectors. A similar lemma was shown in [BKKV10, LLW11]. Here we give a significantly simpler proof using *leftover-hash* (Lemma 4.1.3).

Lemma 6.2.1 (Subspace Hiding with Leakage)**.** *Let $n \geq d \geq u, s$ be integers, $S \in \mathbb{F}_q^{d \times s}$ be an arbitrary (fixed and public) matrix and $\mathsf{Leak} : \{0,1\}^* \to \{0,1\}^\ell$ be an arbitrary function with ℓ-bit output. For randomly sampled $A \xleftarrow{\$} \mathbb{F}_q^{n \times d}$, $V \xleftarrow{\$} \mathbb{F}_q^{d \times u}, U \xleftarrow{\$} \mathbb{F}_q^{n \times u}$, we have:*

$$(\mathsf{Leak}(A), AS, V, AV) \overset{\text{stat}}{\approx} (\ \mathsf{Leak}(A), AS, V, U)$$

as long as $(d - s - u)\log(q) - \ell = \omega(\log(\lambda))$ and $n = \mathsf{poly}(\lambda)$.

Proof. The lemma follows by applying leftover-hash (see Lemma 4.1.3) to each row of A independently. In particular, take any row $\vec{a}_i$ of A and think of it as a random source (while all the other rows of A are arbitrarily fixed) whose conditional min-entropy is

$$\widetilde{\mathbf{H}}_\infty(\vec{a}_i \mid AS, \mathsf{Leak}(A)) \geq d\log(q) - (s\log(q) + \ell).$$

Think of V as the seed of the universal hash function $h_V(\vec{a}_i) = \vec{a}_i \cdot V$ whose output size is $u\log(q)$ bits. The leftover-hash lemma tells us that the ith row of AV looks uniform. By using the hybrid argument over all n rows, the first part of the lemma follows.

□

We also show a dual version of Lemma 6.2.1, where a random matrix A is chosen and the attacker either leaks on random vectors in $\mathsf{colspan}(A)$ or uniformly random vectors. Even if the attacker is later given A in full, it cannot distinguish which case occurred. This version of "subspace hiding" was first formulated by [BKKV10], but here we present a significantly simplified proof and improved parameters.

Corollary 6.2.2 (Dual Subspace Hiding)**.** *Let $n \geq d \geq u$ be integers, and let $\mathsf{Leak} : \{0,1\}^* \to \{0,1\}^\ell$ be some arbitrary function. For randomly sampled $A \overset{\$}{\leftarrow} \mathbb{F}_q^{n \times d}$, $V \overset{\$}{\leftarrow} \mathbb{F}_q^{d \times u}$, $U \overset{\$}{\leftarrow} \mathbb{F}_q^{n \times u}$, we have:*

$$(\mathsf{Leak}(AV), A) \overset{\text{stat}}{\approx} (\ \mathsf{Leak}(U), A)$$

as long as $(d-u)\log(q) - \ell = \omega(\log(\lambda))$, $n = \mathsf{poly}(\lambda)$, and $q = \lambda^{\omega(1)}$.

Proof. We will actually prove the above assuming that A, V, U are random full-rank matrices, which is statistically close to the given statement since q is super-polynomial (see Lemma 4.2.1). We then "reduce" to Lemma 6.2.1.

Given A and C such that $C = AV$ or $C = U$, we can probabilistically choose a $n \times d'$ matrix A' depending only on C and a $n \times u'$ matrix C' depending only on A such that the following holds:

- If $C = AV$ for a random (full rank) $d \times u$ matrix V, then $C' = A'V'$ for a random (full rank) $d' \times u'$ matrix V'.
- If $C = U$ is random (full rank) and independent of A, then $C' = U'$ is random (full rank) and independent of A'.

and where $d' = n - u$, $u' = n - d$. To do so, simply choose A' to be a random $n \times d'$ matrix whose columns form a basis of $\mathsf{colspan}(C)^\perp$ and choose C' to be a random $n \times u'$ matrix whose columns form a basis of $\mathsf{colspan}(A)^\perp$. If $C = U$ is independent of A, then $C' = U'$ is a random full-rank matrix independent of A'. On the other hand, if $C = AV$, then $\mathsf{colspan}(A)^\perp \subseteq \mathsf{colspan}(C)^\perp$ is a random subspace. Therefore $C' = A'V'$ for some uniformly random V'.

Now assume that our lemma does not hold and that there is some function Leak and an (unbounded) distinguisher D that has a non-negligible distinguishing advantage for our problem. Then we can define a function Leak' and a distinguished D' which breaks the problem of Lemma 6.2.1 (without even looking at AS, V). The function $\mathsf{Leak}'(A)$ samples C' as above and outputs $Leak = \mathsf{Leak}(C')$. The distinguisher D', given $(Leak, C)$ samples A' using C as above and outputs $D(Leak, A')$. The distinguisher D' has the same advantage as D. Therefore, by Lemma 6.2.1, indistinguishability holds as long as

$$(d' - u')\log(q) - \ell = \omega(\log(\lambda)) \Leftrightarrow (d-u)\log(q) - \ell = \omega(\log(\lambda))$$

□

It is also easy to extend the above corollary to the case where (the column space of) A is a subspace of some larger public space $\mathcal{W}$.

Corollary 6.2.3. *Let $n \geq m \geq d \geq u$. Let $\mathcal{W} \subseteq \mathbb{F}_q^n$ be a fixed subspace of dimension m and let $\mathsf{Leak} : \{0,1\}^* \to \{0,1\}^\ell$ be some arbitrary function. For randomly sampled $A \overset{\$}{\leftarrow} \mathcal{W}^d$ (interpreted as an $n \times d$ matrix), $V \overset{\$}{\leftarrow} \mathbb{F}_q^{d\times u}, U \overset{\$}{\leftarrow} \mathcal{W}^u$ (interpreted as an $n \times u$ matrix), we have:*

$$(\mathsf{Leak}(AV), A) \overset{\text{stat}}{\approx} (\ \mathsf{Leak}(U), A)$$

as long as $(d-u)\log(q) - \ell = \omega(\log(\lambda))$, $n = \mathsf{poly}(\lambda)$, and $q = \lambda^{\omega(1)}$.

Proof. Let W be some $n \times m$ matrix whose columns span $\mathcal{W}$. Then we can uniquely write $A = WA'$, where $A' \in \mathbb{F}_q^{m\times d}$ is uniformly random. Now we just apply Corollary 6.2.2 to A'. $\square$

Lastly, we show that a variant of Lemma 6.2.1 where, if one is given leakage on a matrix A, one cannot distinguish random vectors from $\mathsf{colspan}(A)$ from uniformly random vectors, even if A is a random matrix of some (non-full) rank d and the row-space of A is fixed and known.

Corollary 6.2.4. *Let the integers n, d, s, u be polynomial in the security parameter λ, with $n \geq d \geq 1$. Let $S \in \mathbb{F}_q^{n\times s}$ be a fixed matrix, $\mathcal{W} \subseteq \mathbb{F}_q^n$ be some fixed subspace of dimension at least d, and $\mathsf{Leak} : \{0,1\}^* \to \{0,1\}^\ell$ be an arbitrary function. Then, for a random $V \overset{\$}{\leftarrow} \mathbb{F}_q^{n\times u}$, $U \overset{\$}{\leftarrow} \mathbb{F}_q^{n\times u}$ and $A \overset{\$}{\leftarrow} \mathsf{Rk}_d(\mathbb{F}_q^{n\times n} \mid \mathsf{row} \in \mathcal{W})$ we have*

$$(\mathsf{Leak}(A), AS, V, AV) \overset{\text{stat}}{\approx} (\ \mathsf{Leak}(A), AS, V, U)$$

as long as $(d-s-u)\log(q) - \ell = \omega(\log(\lambda))$ and $q = \omega(\log(\lambda))$.

Proof. Sampling $A \overset{\$}{\leftarrow} \mathsf{Rk}_d(\mathbb{F}_q^{n\times n} \mid \mathsf{row} \in \mathcal{W})$ is equivalent to sampling $C \overset{\$}{\leftarrow} \mathsf{Rk}_d(\mathbb{F}_q^{n\times d})$, $R \overset{\$}{\leftarrow} \mathsf{Rk}_d(\mathbb{F}_q^{d\times n} \mid \mathsf{row} \in \mathcal{W})$ and setting $A = CR$. The corollary then follows by applying Lemma 6.2.1 to the matrix C while thinking of R as public. In particular, ℓ-bit leakage $\mathsf{Leak}(A)$ on the matrix $A = CR$ is equivalent to ℓ-bit leakage $\mathsf{Leak}'(C)$ on the matrix C. Furthermore $AS = CRS = CS'$ for some $S' \in \mathbb{F}_q^{n\times s}$ and $AV = CRV = CV'$ where V' is uniformly random over the choice of V since R is full rank. We only use the fact that q is super-polynomial to switch from $C \overset{\$}{\leftarrow} \mathsf{Rk}_d(\mathbb{F}_q^{n\times d})$ to $C \overset{\$}{\leftarrow} \mathbb{F}_q^{n\times d}$. So, by Lemma 6.2.1, we have $(\mathsf{Leak}'(C), AS', V', AV') \overset{\text{stat}}{\approx} (\mathsf{Leak}'(C), AS', V', U)$ which gives us our corollary. $\square$

Extended Rank Hiding in the Exponent. We define an *extended rank hiding assumption* which says that one cannot distinguish between matrices of different ranks *in the exponent*, even given some additional vectors in the kernel of the matrix.

Definition 6.2.5 (Extended Rank Hiding Assumption)**.** *The* k extended rank hiding assumption *for a group* $(\mathbb{G}, q, \mathbf{g}) \leftarrow \mathcal{G}(1^\lambda)$ *states that for any integer constants* i, j, n, m *satisfying* $k \le i < j \le \min\{n, m\}$ *and for* $t := m - j$*, we get the indistinguishability property:*

$$\Big(\mathsf{prms}, \mathbf{g}^X, \vec{v}_1, \dots, \vec{v}_t \;\Big|\; (\mathbb{G}, q, \mathbf{g}) \leftarrow \mathcal{G}(1^\lambda), X \xleftarrow{\$} \mathsf{Rk}_i(\mathbb{F}_q^{n \times m}), \{\vec{v}_\rho\}_{\rho=1}^t \xleftarrow{\$} \mathsf{ker}(X)\Big)$$
$$\overset{\text{comp}}{\approx}$$
$$\Big(\mathsf{prms}, \mathbf{g}^X, \vec{v}_1, \dots, \vec{v}_t \;\Big|\; (\mathbb{G}, q, \mathbf{g}) \leftarrow \mathcal{G}(1^\lambda), X \xleftarrow{\$} \mathsf{Rk}_j(\mathbb{F}_q^{n \times m}), \{\vec{v}_\rho\}_{\rho=1}^t \xleftarrow{\$} \mathsf{ker}(X)\Big).$$

We say that the assumption holds for a pairing $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, q, \mathbf{g}, \mathbf{h}) \leftarrow \mathcal{G}_{pair}(1^\lambda)$ *if it holds in* both *the left group* $(\mathbb{G}_1, q, \mathbf{g})$ *and the right group* $(\mathbb{G}_2, q, \mathbf{h})$*.*

Lemma 6.2.6. *The* k extended rank hiding *assumption is implied by the (regular)* k rank hiding *assumption which is in-turn implied by the* k-linear *assumption.*

Proof. The second part of the lemma (k-linear $\Rightarrow$ k rank hiding) is shown in [NS09] and hence we skip it. For the first part of the lemma we show a reduction that converts a rank hiding challenge into an extended rank hiding challenge. In particular, the reduction is given the challenge $(\mathsf{prms}, \mathbf{g}^{X'})$ where either (I) $X' \xleftarrow{\$} \mathsf{Rk}_i(\mathbb{F}_q^{n \times j})$ or (II) $X' \xleftarrow{\$} \mathsf{Rk}_j(\mathbb{F}_q^{n \times j})$ for some i, j, n satisfying $i < j \le n$. The reduction chooses a random matrix $R \xleftarrow{\$} \mathsf{Rk}_j(\mathbb{F}_q^{j \times m})$ and sets $\mathbf{g}^X = \mathbf{g}^{X'R}$ (which can be computed efficiently without knowing X'). It also chooses $\vec{v}_1, \dots, \vec{v}_t \xleftarrow{\$} \mathsf{ker}(R)$ and outputs the challenge $(\mathsf{prms}, \mathbf{g}^X, \vec{v}_1, \dots, \vec{v}_t)$.

Assume that the received challenge if of type (II). Then, by Lemma 4.2.2, we see that the distribution of $X = X'R$ is the same as a random sample from $\mathsf{Rk}_j(\mathbb{F}_q^{n \times m})$. Furthermore since $\mathsf{rowspan}(X) = \mathsf{rowspan}(R)$, the vectors $\vec{v}_1, \dots, \vec{v}_t$ are random over $\mathsf{ker}(R) = \mathsf{rowspan}(R)^\perp = \mathsf{rowspan}(X)^\perp$. Therefore the outputs of the reduction is distributed the same as an extended rank-hiding assumption challenge with rank j.

Assume that the received challenge is of type (I). Then, by Lemma 4.2.2, we can sample X' via $X' = X_1 X_2$ where $X_1 \xleftarrow{\$} \mathsf{Rk}_i(\mathbb{F}_q^{n \times i})$ and $X_2 \xleftarrow{\$} \mathsf{Rk}_i(\mathbb{F}_q^{i \times j})$. Applying Lemma 4.2.2 once more, we get $X_2 R$ is uniformly random over $\mathsf{Rk}_i(\mathbb{F}_q^{i \times m})$ and applying it once again we see that $X = X'R = X_1(X_2 R)$ is uniformly random over $\mathsf{Rk}_i(\mathbb{F}_q^{n \times m})$. Furthermore $\mathsf{rowspan}(X) = \mathsf{rowspan}(X_2 R) \subseteq \mathsf{rowspan}(R)$ and so $\mathsf{ker}(R) \subseteq \mathsf{ker}(X)$. Moreover, $\mathsf{ker}(R)$ is a random $t = (m - j)$ dimensional subspace of the $(m-i)$ dimensional space $\mathsf{ker}(X)$. Since sampling t random vectors from $\mathsf{ker}(X)$ is statistically close to first choosing a random t-dimensional subspace $\mathsf{ker}(R) \subseteq \mathsf{ker}(X)$ and then sampling t random vectors from that, we see the joint distribution on $\mathbf{g}^X, \vec{v}_1, \ldots, \vec{v}_t$ produced by the reduction is (statistically close to) the extended rank hiding assumption challenge with rank i. $\square$

6.3 Warm Up: Weakly Secure Constructions

In this section, we build up toward our main construction of CLR PKE by first presenting several simpler constructions that only achieve weakened notions of security with leakage. Our presentation is mainly used to build up intuition and hence we do not give formal definitions of these notion or complete proofs here. For simplicity, all of the schemes we show will be based the DDH/SXDH (equivalently $k = 1$ linear) assumption, but we note that they can all be easily generalized to the k linear assumption for arbitrary k.

6.3.1 The Bounded Leakage Model

We first present a simple encryption scheme that achieves leakage resilience in the *bounded leakage model.* That is, this scheme has the standard syntax of a PKE and there is *no* extra key-update algorithm. Nevertheless, semantic security is preserved even if an attacker can get up to ℓ bits of leakage in total on the scheme's (single) secret key prior to seing the challenge ciphertext. This security property was initially studied by [AGV09, NS09] and the following scheme is a simple modification of a scheme analyzed in [NS09] and first proposed by [BHHO08]. It is based on the DDH assumption and does *not* require pairings. We use "linear algebra in the exponent" notation to describe the scheme (see preliminaries).

Construction. Let m be an integer parameter. The scheme is defined as follows.

KeyGen$(1^\lambda) \to (pk, sk)$ **:** Sample $(\mathbb{G}, \mathbf{g}, q) \leftarrow \mathcal{G}(1^\lambda)$.

Choose $\vec{p} \xleftarrow{\$} \mathbb{F}_q^m$ and set $\mathsf{prms} = (\mathbb{G}, \mathbf{g}, q, \mathbf{g}^{\vec{p}})$ to be the *public parameters* of the system. These parameters can then be reused to create the public/secret keys of all future users. For convenience, we *implicitly* think of prms as a part of each public key pk and as an input to all of the other algorithms.

Choose $\vec{t} \xleftarrow{\$} \mathbb{F}_q^m$ and set $pk := \mathbf{g}^{\langle \vec{p}, \vec{t} \rangle}$, $sk := \vec{t}$.

Encrypt$_{pk}(\mathbf{m}) \to ct$ **:** To encrypt $\mathbf{m} \in \mathbb{G}$ under $pk = \mathbf{f}$,
choose $u \xleftarrow{\$} \mathbb{F}_q$ and output: $ct := (\mathbf{g}^{u\vec{p}}, \mathbf{f}^u \cdot \mathbf{m})$.

Decrypt$_{sk}(ct) \to \mathbf{m}$**:** To decrypt, parse the secret key $sk = \vec{t}$ and the ciphertext $ct = (\mathbf{g}^{\vec{c}}, \mathbf{g}')$. Output: $\mathbf{m} := \mathbf{g}' / \mathbf{g}^{\langle \vec{c}, \vec{t} \rangle}$.

Proof Sketch. Now we can show that the above scheme is secure in the *bounded leakage model* as long as the overall amount of leakage is $\ell = (m-2)\log(q) - \omega(\log(\lambda))$. Note that this is approximately a $(m-2)/m$ fraction of the total key size and hence one can asymptotically interpret the above as saying that the scheme remains secure even if almost the entire key can leak.

Let us sketch the proof, following the main ideas of [NS09]. We use several hybrids where we modify how the challenge ciphertext is chosen.

***Game 0*:** This is the original semantic security game in the bounded leakage model, where the challenge ciphertext $ct := (\mathbf{g}^{\vec{c}}, \mathbf{g}')$ encrypting a message $\mathbf{m}$ is computed by choosing $u \xleftarrow{\$} \mathbb{F}_q$ and setting

$$\mathbf{g}^{\vec{c}} := \mathbf{g}^{u\vec{p}} \quad , \quad \mathbf{g}' := \mathbf{f}^u \cdot \mathbf{m}.$$

***Game 1*:** In this game, the challenge ciphertext $ct := (\mathbf{g}^{\vec{c}}, \mathbf{g}')$ is computed using the secret key $\vec{t}$ by choosing $u \xleftarrow{\$} \mathbb{F}_q$, setting

$$\mathbf{g}^{\vec{c}} := \mathbf{g}^{u\vec{p}} \quad , \quad \mathbf{g}' := \mathbf{g}^{\langle \vec{c}, \vec{t} \rangle}\mathbf{m}.$$

This retains the exact same distributions as in Game 0, since

$$\mathbf{f}^u = \mathbf{g}^{u(\langle \vec{p}, \vec{t} \rangle)} = \mathbf{g}^{\langle u\vec{p}, \vec{t} \rangle} = \mathbf{g}^{\langle \vec{c}, \vec{t} \rangle}.$$

Therefore, this change is purely syntactical.

***Game 2*:** In this game, the challenge ciphertext $ct := (\mathbf{g}^{\vec{c}}, \mathbf{g}')$ is computed by choosing $\vec{c} \xleftarrow{\$} \mathbb{F}_q^m$ at random and computing $\mathbf{g}' := \mathbf{g}^{\langle \vec{c}, \vec{t} \rangle}\mathbf{m}$ as in *Game 1*. This is *computationally indistinguishable* by the DDH assumption (or the $k = 1$ rank hiding assumption, by thinking $\mathbf{g}^{\vec{p}}, \mathbf{g}^{\vec{c}}$ as two rows of a matrix which is either uniformly random or a random rank 1 matrix). Note that the indistinguishability between Games 1 and 2 holds even if the attacker sees the secret key $\vec{t}$ *in full*.

***Game 3*:** In this game, we ignore the message $\mathbf{m}$ and set the challenge ciphertext to $ct := (\mathbf{g}^{\vec{c}}, \mathbf{g}')$ where $\vec{c} \xleftarrow{\$} \mathbb{F}_q^m$ and $\mathbf{g}' \xleftarrow{\$} \mathbb{G}$ are both uniformly random.

We show that this is statistically indistinguishable from the previous case. This is because, given the public key pk all the leakage that the attacker sees prior to the challenge ciphertext, the value $\vec{t}$ has at least $(m-1)\log(q) - \ell = \log(q) + \omega(\log(\lambda))$ bits of min entropy. By thinking of $\langle \vec{c}, \vec{t} \rangle$ as a randomness extractor with $\vec{t}$ being the source and $\vec{c}$ being the seed and the output consisting of $\log(q)$ bits, we note that the value $\mathbf{g}^{\langle \vec{c}, \vec{t} \rangle}$ is statistically close to uniform, even conditioned on everything else that the attacker sees in *Game 2*. Here we crucially rely on the fact that the leakage occurs only prior to when the attacker sees the challenge ciphertext and therefore the seed $\vec{c}$. Therefore, Games 2 and 3 are statistically close.

Since *Game 3* is independent of the encrypted message, the attacker now has no advantage and therefore we showed semantic security holds.

6.3.2 Secret Leak-Free Updates ("Floppy Model")

Now that we have a scheme in the *bounded leakage model*, we would like to get a method for updating the secret key. Indeed, taking the scheme from the previous section and fixing $\mathbf{g}^{\vec{p}}$, $sk = \vec{t}$, $pk = \mathbf{g}^{\langle \vec{p}, \vec{t} \rangle} = \mathbf{g}^{\alpha}$, we see that there is an entire $m-1$ dimensional *affine space*

$$\mathcal{S} := \{t' \in \mathbb{F}_q^m \ : \ \langle \vec{t'}, \vec{p} \rangle = \alpha\} = \{\vec{t} + \vec{w} \ : \ \vec{w} \in (\vec{p})^{\perp}\}$$

of secret keys which can be used to perform decryption correctly. Unfortunately, given a single secret key $\vec{t}$, it is computationally hard to find another secret key $\vec{t'} \in \mathcal{S}$. Therefore, we cannot just update the secret keys without any extra knowledge.

As a warm-up, let's first consider a method for updating the secret keys using additional secret information which never leaks. That is, we consider a PKE scheme where the key generation, in addition to outputting pk, sk, outputs a secret update key uk. In addition to the encryption and decryption procedures, there is also a secretly keyed update process $sk' \xleftarrow{\$} \mathsf{Update}_{uk}(sk)$. The security property is the same as that of CLR PKE with Leak-Free Updates: we assume that the secret key sk can leak up to ℓ bits in each time period, but the randomness of the updates and the update key uk do not leak. We note that this security property may be motivated in practice, by thinking of a device storing the secret key sk as being used "in the field" for some amount of time, during which the secret key can leak up to ℓ bits, but then always being brought back to a "secure base" where the secret key sk is updated to $sk' \xleftarrow{\$} \mathsf{Update}_{uk}(sk)$ using an update key uk which is stored securely and externally (say, on a floppy disk). We will refer to this as the "floppy model". Eventually, we will want to get rid of the requirement that updates require an additional piece of leak-free secret information uk, and even that they must be performed securely without leaking their randomness.

Construction. The main idea of extending our construction from the previous section to the *floppy model* is for the key generation algorithm to also output some vectors $(\vec{w}_1, \ldots, \vec{w}_{m-2})$ from the space $(\vec{p})^{\perp}$ as part of the update key uk. With overwhelming probability, these vectors will span an $m-2$ dimensional subspace of the $m-1$ dimensional space $(\vec{p})^{\perp}$. These vectors are then used to update the secret key $\vec{t}$ to $\vec{t'} := \vec{t} + \vec{w}$ where $\vec{w}$ is chosen at random from $\mathsf{span}(\vec{w}_1, \ldots, \vec{w}_{m-1})$. Let us write this formally.

KeyGen$(1^\lambda) \to (pk, sk, uk)$: Sample $\mathsf{prms} = (\mathbb{G}, \mathbf{g}, q) \leftarrow \mathcal{G}(1^\lambda)$.

Choose $\vec{p}, \vec{t} \xleftarrow{\$} \mathbb{F}_q^m$ and let $\vec{w}_1, \ldots, \vec{w}_{m-2} \xleftarrow{\$} (\vec{p})^\perp$. Let $\vec{w} \xleftarrow{\$} \mathsf{span}(\vec{w}_1, \ldots, \vec{w}_{m-2})$ and set $\vec{s} := \vec{t} + \vec{w}$.

Set $pk := (\mathbf{g}^{\vec{p}}, \mathbf{f} = \mathbf{g}^{\langle \vec{p}, \vec{t} \rangle})$, $sk := \vec{s}, uk := (\vec{w}_1, \ldots, \vec{w}_{m-2})$.

Update$_{uk}(sk) \to sk'$: Parse $sk = \vec{s}$, $uk = (\vec{w}_1, \ldots, \vec{w}_{m-2})$.

Choose $\vec{w} \xleftarrow{\$} \mathsf{span}(\vec{w}_1, \ldots, \vec{w}_{m-2})$ at random. Output $sk' := \vec{s} + \vec{w}$.

Encrypt$_{pk}(\mathbf{m})$, Decrypt$_{sk}(ct)$: These are defined the same way as in the previous scheme.

Proof Sketch. We now show that the above scheme remains semantically secure in the floppy model, even if the attacker can leak up to $\ell = (m-3)\log(q) - \omega(\log(\lambda))$ bits of information on *each* of arbitrarily many secret keys in between updates. Note that this is approximately a $(m-3)/m$ fraction of the total key size and hence one can asymptotically interpret the above as saying that the scheme remains secure even if almost the entire key can leak in between updates.

***Game 0*:** This is the original semantic security game in the "floppy model", where the challenge ciphertext $ct := (\mathbf{g}^{\vec{c}}, \mathbf{g}')$ encrypting a message $\mathbf{m}$ is computed by choosing $u \xleftarrow{\$} \mathbb{F}_q$ and setting

$$\mathbf{g}^{\vec{c}} := \mathbf{g}^{u\vec{p}} \quad , \quad \mathbf{g}' := \mathbf{f}^u \cdot \mathbf{m}.$$

***Game 1*:** In this game, the challenge ciphertext $ct := (\mathbf{g}^{\vec{c}}, \mathbf{g}')$ is computed using the secret key $\vec{t}$ by choosing $u \xleftarrow{\$} \mathbb{F}_q$, setting

$$\mathbf{g}^{\vec{c}} := \mathbf{g}^{u\vec{p}} \quad , \quad \mathbf{g}' := \mathbf{g}^{\langle \vec{c}, \vec{t} \rangle}\mathbf{m}.$$

This retains the exact same distributions as in Game 0, since

$$\mathbf{f}^u = \mathbf{g}^{u(\langle \vec{p}, \vec{t} \rangle)} = \mathbf{g}^{\langle u\vec{p}, \vec{t} \rangle} = \mathbf{g}^{\langle \vec{c}, \vec{t} \rangle}.$$

Therefore, this change is purely syntactical.

***Game 2*:** In this game, during key generation, we choose $\vec{p}, \vec{t}$ as before, but now we also choose $\vec{c} \xleftarrow{\$} \mathbb{F}_q^m$ and set $uk := (\vec{w}_1, \ldots, \vec{w}_{m-2}) \xleftarrow{\$} (\vec{p}, \vec{c})^\perp$. We set pk, sk as before as well. Lastly, we set the challenge ciphertext to:

$$ct = (\mathbf{g}^{\vec{c}} \quad , \quad \mathbf{g}' := \mathbf{g}^{\langle \vec{c}, \vec{t} \rangle}\mathbf{m}).$$

Note that the challenge ciphertext as produced in *Game 2* has the wrong distribution since it is unlikely that $\vec{c} \in \mathsf{span}(\vec{p})$. However, it is still correctly decrypted to $\mathbf{m}$ by every version of the updated secret key, since the update vectors $\vec{w} \in \mathsf{span}(\vec{w}_1, \ldots, \vec{w}_{m-2})$ are always orthogonal to $\vec{c}$ and hence do not affect decryption.

Games 1 and 2 are computationally indistinguishable by the $k = 1$ extended rank-hiding assumption (which is implied by DDH). To see this, think of $\mathbf{g}^{\vec{p}}, \mathbf{g}^{\vec{c}}$ as the two rows of an $2 \times m$ matrix $\mathbf{g}^X$ and the values $\vec{w}_1, \ldots, \vec{w}_{m-2}$ as being chosen randomly in $\mathsf{ker}(X)$. Now, if $X \xleftarrow{\$} \mathsf{Rk}_1(\mathbb{F}_q^{2\times m})$ then $\vec{c} = u\vec{p}$ for some random $u \in \mathbb{F}_q$ and hence we get the distribution of *Game 1.* On the other hand, if $X \xleftarrow{\$} \mathsf{Rk}_2(\mathbb{F}_q^{2\times m})$ then $\vec{p}$ and $\vec{c}$ are (statistically close to) random and independent and hence we get (statistically close to) the distribution of *Game 2.*

***Game 3*:** In this game, we modify how the secret keys are chosen. In particular, we can always write each secret key as $\vec{s} = \vec{t} + \vec{w}$. We can think of *Game 2* as choosing each such key independently with $\vec{w} \xleftarrow{\$} \mathsf{span}(\vec{w}_1, \ldots, \vec{w}_{m-2})$ instead of updating the previous key. We define *Game 3* by choosing each key independently with $\vec{w} \xleftarrow{\$} (\vec{p})^\perp$ coming from a larger subspace than previously.

We show that Games 2, 3 are statistically indistinguishable using *subspace hiding* (Corollary 6.2.3). We do so by defining more hybrid games *Game 2.0* - *Game 2.l* where l is the total number of keys leaked upon by the attacker. In *Game 2.i* the first i secret keys are chosen as in *Game 3* and the rest as in *Game 2.* Therefore *Game* 2.0 is equivalent to *Game 2* and *Game* 2.l is equivalent to *Game 3.*

Assume there is an attacker that distinguishes between Games 2.i and 2.$(i+1)$ for some i. Let $\mathcal{W} = (\vec{p})^\perp \subseteq \mathbb{F}_q^m$ be a known space, and let $A = [\vec{w}_1^\top || \ldots || \vec{w}_{m-2}] \xleftarrow{\$} \mathcal{W}^{m-2}$. Let us write the $(i+1)$ st secret key as $\vec{s} = \vec{t} + \vec{w}$. Then we can write $\vec{w} = A\vec{v}$ for a random $\vec{v} \xleftarrow{\$} \mathbb{F}_q^{m-2}$ in *Game* 2.i and $\vec{w} \xleftarrow{\$} \mathcal{W}$ is uniform in *Game* 2.$(i+1)$. Furthermore, the matrix A is random over $\mathcal{W}^{m-2}$ and the attacker does not get any information about it in the first i rounds of leakage. Therefore, the attacker can distinguish the two distributions of $\vec{w}$ by leaking ℓ bits of it first, and only later seing (information related to) the matrix A. (Note: we can think of the vector $\vec{c}$ in the challenge ciphertext as being chosen later via $\vec{c} \xleftarrow{\$} (\vec{w}_1, \ldots, \vec{w}_{m-2})^\perp$, a randomized process depending on A.) This directly contradicts Corollary 6.2.3.

Notice that, in *Game* 2.i, the first i secret keys will no longer correctly decrypt the challenge ciphertext to the encrypted message $\mathbf{m}$ because the component $\vec{w}$ in the secret key is no longer orthogonal to the ciphertext vector $\vec{c}$ as required for correctness of decryption!

***Game 4*:** We now change the challenge ciphertext to be independent of the message $\mathbf{m}$. That is, we set it to a completely random value $(\mathbf{g}^{\vec{c}}, \mathbf{g}')$. The only difference from the previous game is that now $\mathbf{g}'$ is random rather than being $\mathbf{g}' := \mathbf{g}^{\langle \vec{c}, \vec{t} \rangle}\mathbf{m}$.

However, its easy to see that Games 3 and 4 are statistically indistinguishable since Game 3 does not reveal anything about $\vec{t}$ beyond the value of $\langle \vec{t}, \vec{p} \rangle$, and therefore the value $\langle \vec{c}, \vec{t} \rangle$ is random and independent.

6.3.3 Public Leak-Free Updates

Finally, we are ready to give a construction of CLR PKE with leak-free updates. The main improvement over the previous construction is that now, the update algorithm does not require any secret update key uk. However, we still assume that updates are performed securely and that the randomness of the updates cannot leak. This construction is a slight variant of [BKKV10] and will build intuition for our main construction which does not require leak-free updates.

In the previous construction, we had a secret update key consisting of the vectors $uk = (\vec{w}_1, \ldots, \vec{w}_{m-2})$. One way to get rid of a secret uk would be to just post these vectors publicly. Unfortunately, this will be insecure since, once such vectors are made public, each updated secret key can be uniquely and efficiently written as $\vec{t} + c_1\vec{w}_1 + \ldots + c_{m-2}\vec{w}_{m-2}$ and therefore the attacker can always compute $\vec{t}$ from each secret key and leak it one bit at a time. Once $\vec{t}$ is leaked in full, the attacker can decrypt arbitrary ciphertexts.

Instead, we will change the scheme so that the secret key contains $\vec{t}$ *in the exponent* rather than in the clear. Also, instead of posting the vectors $\vec{w}_i$ publicly in the clear, we will now post only a single such vector $\vec{w}$ in the exponent. We will show that, although the updates now come from a significantly smaller space, they are indistinguishable from performing secretly keyed updates from the larger space as before (but in the exponent). Unfortunately, by placing $\vec{t}$ and $\vec{w}$ in the exponent, we loose the ability to decrypt efficiently. To regain it, we will need to modify the previous scheme to rely on *pairings*.

Construction. We now give a construction of CLR PKE with leak-free updates, to build intuition toward our full construction.

KeyGen$(1^\lambda) \to (pk, sk)$: Sample $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathbf{g}, \mathbf{h}, q) \leftarrow \mathcal{G}_{pair}(1^\lambda)$.
Choose $\vec{p}, \vec{w} \in \mathbb{F}_q^m$ at random subject to $\langle \vec{p}, \vec{w} \rangle = 0$ and set

$$\mathsf{prms} = ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathbf{g}, \mathbf{h}, q), \mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}})$$

to be the *public parameters* of the system. These parameters can then be reused to create the public/secret keys of all future users. For convenience, we *implicitly* think of prms as a part of each public key pk and as an input to all of the other algorithms.

Choose $\vec{t} \xleftarrow{\$} \mathbb{F}_q^m$ and set $pk := e(\mathbf{g}, \mathbf{h})^{\langle \vec{p}, \vec{t} \rangle}$, $sk = \mathbf{h}^{\vec{t} + r\vec{w}}$ where $r \xleftarrow{\$} \mathbb{F}_q$.

Update$(sk) \to sk'$**:** Parse $sk = \mathbf{h}^{\vec{s}}$ (without knowing $\vec{s}$). Output $sk' := \mathbf{h}^{\vec{s}+r\vec{w}}$ where $r \xleftarrow{\$} \mathbb{F}_q$.

Encrypt$_{pk}(\mathbf{m}) \to ct$ **:** To encrypt $\mathbf{m} \in \mathbb{G}_T$ under $pk = \mathbf{f}$, choose $u \in \mathbb{F}_q$ and output: $ct := (\mathbf{g}^{u\vec{p}}, \mathbf{f}^u \cdot \mathbf{m})$.

Decrypt$_{sk}(ct) \to \mathbf{m}$**:** To decrypt, parse the secret key matrix $sk = \mathbf{h}^{\vec{s}}$. Parse the ciphertext $ct = (\mathbf{g}^{\vec{c}}, \mathbf{z})$. Output: $\mathbf{m} := \mathbf{z}/e(\ \mathbf{g}^{\vec{c}}\ ,\ \mathbf{h}^{\vec{s}^\top}\ )$.

Proof Sketch. We now show that the above scheme is a secure ℓ-CLR-PKE *with leak-free updates*, where $\ell = (m-4)\log(q) - \omega(\log(\lambda))$. That is, the attacker can leak up to ℓ bits on each secret key in between updates (which is almost the entire key!), but cannot leak on the randomness of the update process. Security holds under the SXDH assumption in bilinear groups, and we sketch the series of hybrids below.

***Game 0*:** This is the original security game.

***Game 0.5*:** In this game, during key generation, we choose $\vec{p}, \vec{t}$ as before, but now we also choose $\vec{w}_1, \ldots, \vec{w}_{m-2} \xleftarrow{\$} (\vec{p})^\perp$ and set $\vec{w} := \vec{w}_1$. As before, we use $\vec{p}, \vec{t}, \vec{w}$ to construct the public key and parameters. However, we now choose each secret key (including the initial one) uniformly and independently at random via $sk := \mathbf{h}^{\vec{t} + \sum_{i=1}^{m-2} r_i \vec{w}_i}$ where $r_i \xleftarrow{\$} \mathbb{F}_q$.

We rely on the *extended rank hiding assumption* (Definition 6.2.5), implied by SXDH, to show indistinguishability. In particular, assume we are given a challenge $\mathbf{h}^W \in \mathbb{G}_2^{(m-2)\times m}$ and $\vec{p} \xleftarrow{\$} \mathsf{ker}(W)$, where W is either a random rank 1 matrix, or a random rank $m-2$ matrix. Let us label the rows of $W \in \mathbb{F}_q^{m-2\times m}$ by $\vec{w}_1, \ldots, \vec{w}_{m-2}$. Then we can use these values to simulate the CLR-PKE security game by choosing $\vec{t} \xleftarrow{\$} \mathbb{F}_q^m$, using $\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}_1}, e(\mathbf{g}, \mathbf{h})^{\langle \vec{p}, \vec{t} \rangle}$ to define the public parameters/key. We choose each secret key via $sk := \mathbf{h}^{\vec{t} + \sum_{i=1}^{m-2} r_i \vec{w}_i}$ where $r_i \xleftarrow{\$} \mathbb{F}_q$ (note: we can do this knowing only $\mathbf{h}^W$ and without knowing the exponent vectors $\vec{w}_i$). If W is of rank 1, then this is equivalent to Game 0 since we can write each secret key as $sk := \mathbf{h}^{\vec{t} + r\vec{w}_1}$ for a random $r \xleftarrow{\$} \mathbb{F}_q$. On the other hand if W is of rank $m-2$, then this is equivalent to Game 0.5. Therefore a distinguisher between these games will break extended rank hiding.

Now, the scheme in *Game 0.5* is beginning to look a lot like our previous scheme in the "floppy model". Indeed the rest of the games and the security proof are almost identical as before, with the main difference being that now there is an extra vector $\mathbf{h}^{\vec{w}_1}$ posted publicly. As we will see this only has a small affect on the leakage parameter.

***Game 1*:** In this game, the challenge ciphertext $ct := (\mathbf{g}^{\vec{c}}, \mathbf{z})$ is computed using the secret key $\vec{t}$ by choosing $u \xleftarrow{\$} \mathbb{F}_q$, setting

$$\mathbf{g}^{\vec{c}} := \mathbf{g}^{u\vec{p}} \quad , \quad \mathbf{z} := e(\mathbf{g}, \mathbf{h})^{\langle \vec{c}, \vec{t} \rangle}\mathbf{m}.$$

This retains the exact same distributions as in Game 0, since

$$\mathbf{f}^u = e(\mathbf{g}, \mathbf{h})^{u(\langle \vec{p}, \vec{t} \rangle)} = e(\mathbf{g}, \mathbf{h})^{\langle u\vec{p}, \vec{t} \rangle} = e(\mathbf{g}, \mathbf{h})^{\langle \vec{c}, \vec{t} \rangle}.$$

Therefore, this change is purely syntactical.

***Game 2*:** In this game, during key generation, we choose $\vec{p}, \vec{t}$ as before, but now we also choose $\vec{c} \xleftarrow{\$} \mathbb{F}_q^m$ and $(\vec{w}_1, \ldots, \vec{w}_{m-2}) \xleftarrow{\$} (\vec{p}, \vec{c})^\perp$. We set pk as before as well, and use the vectors $\vec{w}_1, \ldots, \vec{w}_{m-2}$ to select the secret keys as in Game 1. Lastly, we set the challenge ciphertext to:

$$ct = (\mathbf{g}^{\vec{c}} \quad , \quad \mathbf{z} := e(\mathbf{g}, \mathbf{h})^{\langle \vec{c}, \vec{t} \rangle}\mathbf{m}).$$

Games 1 and 2 are computationally indistinguishable by the $k = 1$ extended rank-hiding assumption (which is implied by DDH). To see this, think of $\mathbf{g}^{\vec{p}}, \mathbf{g}^{\vec{c}}$ as the two rows of an $2 \times m$ matrix $\mathbf{g}^X$ and the values $\vec{w}_1, \ldots, \vec{w}_{m-2}$ as being chosen randomly in $\mathsf{ker}(X)$. Now, if $X \xleftarrow{\$} \mathsf{Rk}_1(\mathbb{F}_q^{2\times m})$ then $\vec{c} = u\vec{p}$ for some random $u \in \mathbb{F}_q$ and hence we get the distribution of *Game 1*. On the other hand, if $X \xleftarrow{\$} \mathsf{Rk}_2(\mathbb{F}_q^{2\times m})$ then $\vec{p}$ and $\vec{c}$ are (statistically close to) random and independent and hence we get (statistically close to) the distribution of *Game 2*.

***Game 3*:** In this game, we modify how the secret keys are chosen. In particular, we can always write each secret key $sk = \mathbf{h}^{\vec{s}} = \mathbf{h}^{\vec{t}+\vec{w}'}$. Instead of choosing $\vec{w}' \xleftarrow{\$} \mathsf{span}(\vec{w}_1, \ldots, \vec{w}_{m-2})$, in *Game 3* we now choose $\vec{w}' \xleftarrow{\$} (\vec{p})^\perp$ coming from a larger subspace than previously.

As before, we define hybrid games *Game 2.0* - *Game 2.l* where l is the total number of keys leaked upon by the attacker. In *Game 2.i* the first i secret keys are chosen as in *Game 3* and the rest as in *Game 2*.

Assume there is an attacker that distinguishes between Games 2.i and 2.$(i+1)$ for some i. Let $\mathcal{W} = (\vec{p})^\perp \subseteq \mathbb{F}_q^m$ be a known space, $\vec{w} = \vec{w}_1$ be a known vector, and let $A = [\vec{w}_2^\top || \ldots || \vec{w}_{m-2}] \xleftarrow{\$} \mathcal{W}^{m-3}$ be a random matrix. Let us write the $(i+1)$ st secret key as $\mathbf{h}^{\vec{s}} = \mathbf{h}^{\vec{t}+r\vec{w}_1+\vec{w}''}$. Then we can write $\vec{w}'' = A\vec{v}$ for a random $\vec{v} \xleftarrow{\$} \mathbb{F}_q^{m-2}$ in *Game* 2.i and $\vec{w}'' \xleftarrow{\$} \mathcal{W}$ is uniform in *Game* 2.$(i+1)$. Furthermore, the matrix A is random over $\mathcal{W}^{m-3}$ and the attacker does not get any information about it from the public parameters/key or in the first i rounds of leakage. Therefore, the attacker can distinguish the two distributions of $\vec{w}''$ by leaking ℓ bits of it first, and only later seing (information related to) the matrix A. (Note: we can think of the vector $\vec{c}$ in the challenge ciphertext as being chosen later via $\vec{c} \xleftarrow{\$} (\vec{w}_1, \ldots, \vec{w}_{m-2})^{\perp}$, a randomized process depending on A.) This directly contradicts Corollary 6.2.3.

Notice that, in *Game* 2.i, the first i secret keys will no longer correctly decrypt the challenge ciphertext to the encrypted message $\mathbf{m}$ because the component $\vec{w}$ in the secret key is no longer orthogonal to the ciphertext vector $\vec{c}$ as required for correctness of decryption!

***Game 4*:** We now change the challenge ciphertext to be independent of the message $\mathbf{m}$. That is, we set it to a completely random value $(\mathbf{g}^{\vec{c}}, \mathbf{z})$. The only difference from the previous game is that now $\mathbf{z}$ is random rather than being $\mathbf{z} := e(\mathbf{g}, \mathbf{h})^{\langle \vec{c}, \vec{t} \rangle}\mathbf{m}$.

However, its easy to see that Games 3 and 4 are statistically indistinguishable since Game 3 does not reveal anything about $\vec{t}$ beyond the value of $\langle \vec{t}, \vec{p} \rangle$, and therefore the value $\langle \vec{c}, \vec{t} \rangle$ is random and independent.

We note that the relation between public and secret keys of the above scheme defined by

$$\mathcal{R} = \{(pk, sk) \ : \ pk = (\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}}, \mathbf{f}) sk = \mathbf{h}^{\vec{s}} \text{ s.t.} e(\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{s}}) = \mathbf{f}\}$$

right away gives us a CLR-OWR with leak-free updates. Therefore, this can be seen as an alternative to our construction from Chapter 5. Moreover, as shown generically in Section 5.6, we can also get full security (with leakage of updates) where the leakage bound ℓ becomes logarithmic in the security parameter. In fact, the same result can easily be extended to the above CLR PKE scheme, showing that it also achieves full security with a logarithmic leakage bound ℓ. Our main goal now will be to construct a scheme with full security tolerating a significantly higher leakage threshold.

6.3.4 The Difficulty with Leaking on Updates

It is easy to see that the previous scheme is *not* secure if the attacker can leak even just $\lceil \log(q) \rceil$ bits during each update and 1 extra bit on each secret key. In particular, since the update randomness consists of a single field element $r \in \mathbb{F}_q$, if the attacker were to leak it in full during each update, then it could efficiently compute the original secret key sk from every future updated version by reversing the updates. In that case, the attacker could simply leak the original key in full one bit at a time by leaking an additional bit of information on each future key.

Indeed, the proof of security (going from *Game 0* to *Game 0.5*) relies on switching the distribution of secret keys in such a way that they are no longer consistent with *any* value of the update randomness. Our main construction will be designed to avoid this step. Instead of modifying the distribution of various secret keys individually, we will only modify the distribution of the original secret key and the randomness of the updates. However, at each point, there will be some choice of update randomness that produces each successive key from the previous one.

6.4 Construction with Full Security

The main difference between the previous scheme (with leak-free updates) and our full scheme will be the structure of the secret keys and the update process itself. In our new scheme, a secret key can be thought of as consisting of n independent secret keys from the previous scheme. An update will just take n linear combinations of the keys to create n fresh keys. Although this does not appear too different from the previous scheme, we will see that the security argument requires several new ideas.

Let m, n, d be integer parameters with $n \geq d$. The scheme is defined as follows.

KeyGen$(1^\lambda) \to (pk, sk)$ **:** Sample $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathbf{g}, \mathbf{h}, q) \leftarrow \mathcal{G}_{pair}(1^\lambda)$.

Choose $\vec{p}, \vec{w} \in \mathbb{F}_q^m$ at random subject to $\langle \vec{p}, \vec{w} \rangle = 0$ and set

$$\mathsf{prms} = ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathbf{g}, \mathbf{h}, q), \mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}})$$

to be the *public parameters* of the system. These parameters can then be reused to create the public/secret keys of all future users. For convenience, we *implicitly* think of prms as a part of each public key pk and as an input to all of the other algorithms.

Choose $\vec{t} \xleftarrow{\$} \mathbb{F}_q^m$ and set $pk := e(\ \mathbf{g}^{\vec{p}}\ ,\ \mathbf{h}^{\vec{t}^\top}\ ) = e(\mathbf{g}, \mathbf{h})^\alpha$ where $\alpha = \langle \vec{p}, \vec{t} \rangle$. Choose $\vec{r} = (r_1, \ldots, r_n) \xleftarrow{\$} \mathbb{F}_q^n$ and set $sk := \mathbf{h}^S$, where S is the $n \times m$ matrix:

$$S \ := \ \begin{bmatrix} r_1\vec{w} + \vec{t} \\ \cdots \\ r_n\vec{w} + \vec{t} \end{bmatrix} \ = \ \begin{bmatrix} \\ \vec{r}^\top \\ \\ \end{bmatrix} \begin{bmatrix} \quad \vec{w} \quad \end{bmatrix} + \begin{bmatrix} \\ \vec{1}^\top \\ \\ \end{bmatrix} \begin{bmatrix} \quad \vec{t} \quad \end{bmatrix}.$$

In other words, each row of S is chosen at random from the 1-dimensional affine subspace $\vec{t} + \mathsf{span}(\vec{w})$.
(Note that $\mathbf{h}^S$ can be computed from $\mathbf{h}^{\vec{w}}$, without knowing $\vec{w}$.)

$\mathsf{Encrypt}_{pk}(\mathbf{m}) \to ct$ **:** To encrypt $\mathbf{m} \in \mathbb{G}_T$ under $pk = \mathbf{f}$, choose $u \in \mathbb{F}_q$ and output: $ct := (\mathbf{g}^{u\vec{p}}, \mathbf{f}^u \cdot \mathbf{m})$.

$\mathsf{Decrypt}_{sk}(ct) \to \mathbf{m}$**:** To decrypt, parse the secret key matrix $sk = \mathbf{h}^S$, whose first wrote we denote $\mathbf{h}^{\vec{s}}$. Parse the ciphertext $ct = (ct^{(1)}, ct^{(2)})$ where $ct^{(1)} = \mathbf{g}^{\vec{c}}$, $ct^{(2)} = e(\mathbf{g}, \mathbf{h})^z$. Output: $\mathbf{m} := e(\mathbf{g}, \mathbf{h})^z / e(\ \mathbf{g}^{\vec{c}}\ ,\ \mathbf{h}^{\vec{s}^\top}\ )$.

$\mathsf{Update}(sk) \to sk'$ **:** Choose a random $n \times n$ matrix of rank d: $A' \xleftarrow{\$} \mathsf{Rk}_d(\mathbb{F}_q^{n\times n})$. Derive A by "rescaling" each row of A' so that its components sum up to 1. That is, set $(A)_{i,j} := (A')_{i,j} / (\sum_{k=1}^{n} (A')_{i,k})$. This ensures $A\vec{1}^\top = \vec{1}^\top$.
If the current secret key is $sk = \mathbf{h}^S$, output the updated key $sk' := \mathbf{h}^{AS}$.

Theorem 6.4.1. *For any integers $n \geq 10, m \geq 5, d := n - m + 3$, the above scheme is an ℓ-CLR PKE scheme under the SXDH assumption with leakage $\ell = \min(n - 3m + 6, (m-4)/3) \log(q) - \omega(\log(\lambda))$.*

Notice that the amount of tolerated leakage ℓ can be made an arbitrarily large polynomial in the security parameter by increasing n, m sufficiently. However, the ratio of leakage to key size is maximized at $m = 5, n = 10$ to the constant fraction 1/150.

Correctness. Let $(\mathsf{prms}, pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and let $ct = (ct^{(1)}, ct^{(2)}) \leftarrow \mathsf{Encrypt}_{pk}(\mathbf{m})$. Then we can write $sk = \mathbf{g}^S$, $ct^{(1)} = \mathbf{g}^{\vec{c}}, ct^{(2)} = e(\mathbf{g}, \mathbf{h})^z$ and there will be some values $W, \vec{t}$ satisfying:

$$S = W + \vec{1}^\top \vec{t}\ \ ,\ \ z = \vec{c} \cdot \vec{t}^\top + \mu \quad : \quad \mathsf{rowspan}(W) \perp \mathsf{span}(\vec{c}) \tag{6.1}$$

where μ is given by $\mathbf{m} = e(\mathbf{g}, \mathbf{h})^\mu$.

First, we show that for any sk and ct satisfying equation (6.1), we get correctness with: $\mathsf{Decrypt}_{sk}(ct) = \mathbf{m}$. This is because, if we label the first row of $sk = h^S$ by $\mathbf{h}^{\vec{s}}$, then we have $\vec{s} = \vec{w} + \vec{t}, z = \langle \vec{c}, \vec{t} \rangle + \mu$ for some vectors $\vec{w}$ with $\langle \vec{c}, \vec{w} \rangle = 0$. Therefore decryption correctly recovers:

$$e(\mathbf{g}, \mathbf{h})^z / e(\ \mathbf{g}^{\vec{c}}\ ,\ \mathbf{h}^{s^\top}\ ) = e(\mathbf{g}, \mathbf{h})^{\langle \vec{c}, \vec{t} \rangle + \mu} / e(\mathbf{g}, \mathbf{h})^{\langle \vec{c}, \vec{w} + \vec{t} \rangle} = e(\mathbf{g}, \mathbf{h})^\mu = \mathbf{m}$$

Next we show, that updates preserve the key/ciphertext structure of equation (6.1). Assume that we update the initial secret key with the matrices $A_1, A_2, \ldots, A_i$. Define $\bar{A} = A_i A_{i-1} \cdots A_1$. Since the update matrices are "rescaled" we know that $\bar{A}\vec{1}^\top = \vec{1}^\top$. Therefore we can write the updated values as $sk_i = \mathbf{g}^{\bar{A}S}$ where

$$(\bar{A}S) = (\bar{A}W) + \vec{1}^\top \vec{t} \qquad : \qquad \mathsf{rowspan}(\bar{A}W) \perp \mathsf{span}(\vec{c}).$$

So key updates preserve equation (6.1) and therefore correctness.

6.5 Proof Part I: Alternate Distributions

Our proof of security will follow by a hybrid argument. In the real security game, every secret key in every time period will correctly decrypt the challenge ciphertext. Our goal is to move to a game where the secret keys no longer decrypt the challenge ciphertext correctly, and so even if the attacker could leak them in full, it would not help break security. We do so by slowly modifying how the challenger chooses the initial key, ciphertext and the update matrices. We first introduce several alternate distributions for selecting keys and ciphertexts, some of which decrypt correctly and some don't. We also show how to select update matrices to modify the key type.

Alternate Key Generation. Assume that the vectors $\vec{p}, \vec{w}, \vec{t}$ are chosen honestly, defining the public parameters/key values $\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}}, pk = e(\mathbf{g}, \mathbf{h})^{\langle \vec{p}, \vec{t} \rangle}$ with $\langle \vec{p}, \vec{w} \rangle = 0$. Fix $\vec{w}_1 := \vec{w}$, and randomly complete a full basis $(\vec{w}_1, \ldots, \vec{w}_{m-1})$ of the space $(\vec{p})^{\perp}$. The secret key is always set to $\mathbf{h}^S$ for some $n \times m$ matrix S of the form

$$S = \begin{bmatrix} | & & | \\ \vec{r}_1^{\top} & \cdots & \vec{r}_i^{\top} \\ | & & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ & \cdots & \\ - & \vec{w}_i & - \end{bmatrix} + \begin{bmatrix} \vec{1}^{\top} \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix} \qquad (6.2)$$

where $\vec{r}_1, \ldots, \vec{r}_i \in \mathbb{F}_q^n$ are chosen randomly. Equivalently, each of the n rows of S is chosen randomly from the affine space: $\mathsf{span}(\vec{w}_1, \ldots, \vec{w}_i) + \vec{t}$. The honest key generation algorithm uses $i = 1$ and we call these ***honest keys***. In addition, we define ***mid keys*** which are chosen with $i = 2$ and ***high keys*** which are chosen with $i = m - 1$. Notice that honest/mid/high keys all correctly decrypt honestly generated ciphertexts since $\mathsf{span}(\vec{w}_1, \ldots, \vec{w}_{m-1}) \perp \mathsf{span}(\vec{p})$.

Ciphertext Distributions. The encryption of the message $\mathbf{m}$ is always set to $ct = (ct^{(1)}, ct^{(2)})$ where $ct^{(1)} = \mathbf{g}^{\vec{c}}$ and $ct^{(2)} = e(\mathbf{g}, \mathbf{h})^{\langle \vec{c}, \vec{t} \rangle}\mathbf{m}$. Notice that the second component $ct^{(2)}$ can always be efficiently and deterministically computed from $ct^{(1)}$, $\vec{t}$ and $\mathbf{m}$ without knowing the exponent vector $\vec{c}$. The different ciphertext distributions only differ in how $ct^{(1)} = \mathbf{g}^{\vec{c}}$ is chosen.

For the ***honest ciphertexts***, we set $\vec{c} = u\vec{p}$ for a uniformly random $u \xleftarrow{\$} \mathbb{F}_q$. (In this case, $ct^{(2)} = e(\mathbf{g}, \mathbf{h})^{u\langle \vec{p}, \vec{t} \rangle}\mathbf{m} = (pk)^u\mathbf{m}$ can be computed efficiently knowing pk, $ct^{(1)}$ and u and without knowing $\vec{t}$.) We define ***dishonest ciphertext*** by choosing $\vec{c}$ uniformly at random from the space $(\vec{w}_1)^{\perp}$. Lastly, we define ***correlated dishonest ciphertexts***, by choosing $\vec{c}$ uniformly at random from $(\vec{w}_1, \vec{w}_2)^{\perp}$. That is, the exponent vector $\vec{c}$ in the ciphertext is correlated with the basis vector $\vec{w}_2$ so as to be orthogonal. Notice that there is no difference between dishonest ciphertexts and correlated dishonest ciphertexts, *unless* the attacker gets to see a mid key; this is because only mid keys depend on the basis vector $\vec{w}_2$.

keys $\rightarrow$ $\downarrow$ ciphertexts	honest	mid	high
honest	Yes	Yes	Yes
correlated dishonest	Yes	Yes	No
dishonest	Yes	No	No

Figure 6.1: Interactions between keys and challenge ciphertext.

Note that honest ciphertexts are correctly decrypted by honest/mid/high keys, correlated dishonest ciphertext are (only) correctly decrypted by honest/mid keys and dishonest ciphertexts are (only) correctly decrypted by honest keys. This is summarized in Figure 6.1.

Programmed Updates. Honest *key updates* in period i are performed by choosing $A'_i \xleftarrow{\$} \mathsf{Rk}_d(\mathbb{F}_q^{n\times n})$ and rescaling its rows to get A_i. Let $\bar{A}_i = A_iA_{i-1}\cdots A_1$ be the product of all key-update matrices up to and including period i (as a corner case, define $\bar{A}_0$ to be the identity matrix). We say that the update A_i is *programmed to annihilate the vectors* $\vec{v}_1, \ldots, \vec{v}_j \in \mathbb{F}_q^n$ if we instead choose

$$A'_i \xleftarrow{\$} \mathsf{Rk}_d(\mathbb{F}_q^{n\times n} \mid \mathsf{row} \in \mathcal{V}) \quad \text{where} \quad \mathcal{V} = (\bar{A}_{i-1}\vec{v}_1^\top, \ldots, \bar{A}_{i-1}\vec{v}_j^\top)^\perp.$$

In other words, a programmed update A_i has the vectors $\{\bar{A}_{i-1}\vec{v}_\rho\}_{\rho=1}^{j}$ in its *kernel.*

By programming the update matrices, we can have updates which reduce the rank of the key (e.g. reduce high keys to mid keys and mid keys to low keys etc.). Let us go through an example. Assume the initial key is high with $sk_1 = \mathbf{g}^S$ for S given by equation (6.2), and the updates $A_1, \ldots, A_{i-1}$ are chosen honestly, A_i is programmed to annihilate the vectors $\vec{r}_3, \ldots, \vec{r}_{m-1}$ and A_{i+1} is programmed to annihilate $\vec{r}_2$. Then the corresponding secret keys $sk_1, \ldots, sk_i$ in periods 1 to i will be high keys, sk_{i+1} will be a mid key and sk_{i+2} will be an honest key. To see this, notice that the exponent of (e.g.) the key sk_{i+1} will follow equation (6.2) with $\vec{r}_j^\top$ replaced by $A_iA_{i-1}\cdots A_1\vec{r}_j^\top$. Since A_i is programmed to annihilate the vectors $\vec{r}_j$ for $j \geq 3$, these values will be replaced by $\vec{\mathbf{0}}$ in period $i+1$.

6.6 Proof Part II: (Re)Programming Updates.

Here we show two useful results about the scenario where we continually update a key and the attacker is continually leaking on the process. The first lemma (Lemma 6.6.1) shows that after continually updating a key of some rank (e.g. low, mid, high) honestly, we get a random key of the same rank at end. If we program updates to reduce the rank, we get a random key of the appropriate reduced rank at the end. This even holds if some of the updates are programmed to annihilate random vectors unrelated to the key. The second lemma (Lemma 6.6.2) says that an attacker who is leaking sufficiently few bits of information on the update process cannot tell whether some updates are programmed to annihilate random vectors unrelated to the key.

We consider a randomized process called the *Programming Experiment*, defined as follows. First, we choose a random matrix $R \xleftarrow{\$} \mathbb{F}_q^{n\times l}$ and label its columns $R = [\vec{r}_1^\top \mid \cdots \mid \vec{r}_l^\top]$. Then we choose a series of (some polynomial number) t update matrices $A_1, \ldots, A_t \in \mathsf{Rk}_d(\mathbb{F}_q^{n\times n})$ and iteratively set $R_{i+1} = A_i R_i$ where $R_1 = R$. The experiment is defined by an *update regime* $\mathcal{R}$ which specifies if/how the various updates are programmed. Each update can be programmed to annihilate some of the columns of R and some additional random vectors. That is, for each update i, the regime $\mathcal{R}$ specifies a (possibly empty) subset $J_i \subseteq \{1, \ldots, l\}$ of column-indices to be annihilated and a number $u_i \geq 0$ of random vectors to be annihilated, subject to $|J_i| + u_i \leq n - d$. The programming experiment then chooses update i so that A_i is programmed to annihilate the vectors $\{\vec{r}_j | j \in J_i\} \cup \{\vec{v}_{i,1}, \ldots, \vec{v}_{i,u_i}\}$ for fresh and random $\vec{v}_{i,j} \xleftarrow{\$} \mathbb{F}_q^n$. The update regime is fully specified by $\mathcal{R} = (J_1, u_1, \ldots, J_t, u_t)$.

Lemma 6.6.1 (Program Correctness)**.** *Let $\mathcal{R} = (J_1, u_1, \ldots, J_t, u_t)$ be an update regime that annihilates the columns with indices $\bigcup_{i=1}^t J_i = \{\rho+1, \ldots, l\}$ along with $u = \sum_{i=1}^t u_i$ additional random vectors. Assume $(d - l - u - 1)\log(q) = \omega(\log(\lambda))$. Then the distribution of the final matrix R_t is statistically close to*

$$R_t = \begin{bmatrix} | & & | & \ddots & \\ \vec{v}_1^\top & \cdots & \vec{v}_\rho^\top & & \mathbf{0} \\ | & & | & & \ddots \end{bmatrix}$$

for uniformly random and independent columns $\vec{v}_1, \ldots, \vec{v}_\rho \in \mathbb{F}_q^n$. [1]

Let *GameProgram*$(\mathcal{R})$ be a game between a challenger and an attacker, where the challenger runs the programming experiment with the regime $\mathcal{R}$. In each round $i = 1, 2, \ldots, t$, the challenger gives the attacker the matrix R_i *in full* and answers ℓ leakage queries on the update matrix A_i (the attacker can choose leakage queries adaptively depending on its view thus far). The output of the game is the view of the attacker at the end of the game.

Lemma 6.6.2 (Reprogramming)**.** *Let*

$$\mathcal{R} = (J_1, u_1, \ldots, J_t, u_t) \quad \textit{and} \quad \mathcal{R}' = (J_1, u'_1, \ldots, J_t, u'_t)$$

be two update regimes that agree on which columns of R are annihilated and when (i.e. the sets J_i) but not on how many additional random vectors are annihilated

[1]Note: in this lemma, we do *not* condition on seing any information about the initial matrix R or the update matrices A_i.

and when (i.e. the values u_i, u_i'). Let $u^* = \max\left(\sum_{i=1}^t u_i, \sum_{i=1}^t u_i'\right)$ be the maximum number of additional random vectors annihilated by either of the regimes. If $(d - l - u^* - 1)\log(q) - \ell = \omega(\log(\lambda))$ then $\mathrm{GameProgram}(\mathcal{R}) \overset{\text{stat}}{\approx} \mathrm{GameProgram}(\mathcal{R}')$.

Both of the above lemmas (Lemma 6.6.1, Lemma 6.6.2) follow as special cases from the following lemma.

Lemma 6.6.3. *Let $\mathcal{R} = (J_1, u_1, \ldots, J_t, u_t)$ be an update regime and let $u = \sum_{i=1}^t u_i$ be the total number of random vectors it annihilates. Let $\mathcal{R}^* = (J_1, 0, \ldots, J_t, 0)$ be an update regime that agrees with $\mathcal{R}$ on which columns of R are annihilated and when (i.e. the sets J_i) but does not annihilate any additional random vectors (i.e. $u_i^* = 0$). Let $D = A_t \cdots A_1$ be the product of all the update matrices at the end of the game and let $\vec{\mu}_1, \ldots, \vec{\mu}_\rho \overset{\$}{\leftarrow} \mathbb{F}_q^n$ be fresh random vectors. Then*

$$\left(\mathit{GameProgram}(\mathcal{R}), D\vec{\mu}_1^\top, \ldots, D\vec{\mu}_\rho^\top\right) \overset{\text{stat}}{\approx} \left(\mathit{GameProgram}(\mathcal{R}^*), \vec{\mu}_1^\top, \ldots, \vec{\mu}_\rho^\top\right)$$

as long as $(d - l - u - \rho - 1)\log(q) - \ell = \omega(\log(\lambda))$.

Proof. For $i \geq j \geq 0$, we define $D[i,j] := A_i A_{i-1} \cdots A_j$ and $D_i := D[i,1]$ to be products of update matrices A_i (as a a corner case, D_0, A_0 are defined to be the identity matrix as is $D[i,j]$ for $i < j$).

To prove the lemma, we slowly move from the distribution on the left to the one on the right. That is, we define a series of hybrid games *GameHybrid* j where *GameHybrid* 0 is the left-hand distribution and *GameHybrid* $t+1$ is the right-hand one. In *GameHybrid* j, the update matrices A_i for $i \leq j$ are chosen as specified by the regime $\mathcal{R}^*$. Moreover, the update matrices A_i for $i > j$ are chosen by placing u_i random vectors from the span of $D[i-1,j]$ into their kernel (instead of the span of D_{i-1}). More formally, for $i > j$ the update A_i in *GameHybrid* j is chosen by rescaling a randomly sampled $A_i' \overset{\$}{\leftarrow} \mathsf{Rk}_d(\mathbb{F}_q^{n\times n} \mid \mathcal{W})$ where

$$\mathcal{W} = \left(\ \{\ D_{i-1}\vec{r}_k \mid k \in J_i\ \}\ ,\ \{\ D[i-1,j]\vec{v}_{i,k} \mid 1 \leq k \leq u_i\}\ \right)^\perp$$

and $\vec{v}_{i,k} \in \mathbb{F}_q^n$ are uniformly random. Lastly, at the end of *GameHybrid* j, we append the additional vectors chosen as $D[t,j]\vec{\mu}_1, \ldots, D[t,j]\vec{\mu}_\rho$ for uniformly random $\{\vec{\mu}_k \overset{\$}{\leftarrow} \mathbb{F}_q^n\}_{k=1}^\rho$.

We show that for each $j \in \{0, \ldots, t\}$ we have *GameHybrid* $j \overset{\text{stat}}{\approx}$ *GameHybrid* $j+1$. We do this by relying on the *subspace hiding lemma* (Corollary 6.2.4) and showing a "reduction" which uses a distinguishing strategy for the above two hybrids to get a distinguishing strategy for the two distributions in the subspace hiding lemma. The reduction chooses the initial matrix $R_1 \overset{\$}{\leftarrow} \mathbb{F}_q^{n\times l}$.

For updates $i < j$, the reduction chooses the update matrices A_i according to the regime $\mathcal{R}^*$ and gives the attacker the corresponding leakage. It also gives the attacker the values $R_{i+1} = A_i R_i$ in full.

For the update A_j, the reduction chooses its *row space* $\mathcal{W}_j$ honestly as specified by the regime $\mathcal{R}^*$ so that A_j should be sampled by rescaling a random sample $A'_j \xleftarrow{\$} \mathsf{Rk}_d(\mathbb{F}_q^{n\times n} \mid \mathsf{row} \in \mathcal{W}_j)$. However, the reduction does not choose A_j itself, but instead uses a *subspace hiding challenge* (Corollary 6.2.4). That is, let Leak_j be the leakage function chosen by the attacker in round j (on matrix A_j) and let Leak'_j be a modified function which, given matrix A'_j first rescales its rows to get A_j and then outputs $\mathsf{Leak}_j(A_j)$. Define the matrix $S = [R_j|\vec{1}^\top]$. The reduction specifies $S, \mathcal{W}_j, \mathsf{Leak}'_j$ and gets a *subspace hiding* challenge of the form:

$$(\mathsf{Leak}'_j(A'_j), A'_jS, V')$$

where $A'_j \in \mathsf{Rk}_d(\mathbb{F}_q^{n\times n} \mid \mathsf{row} \in \mathcal{W}_j)$ and V' is either of the form A'_iU or just U for a random $n \times (u+\rho)$ matrix U. We call the former distribution *type I* and the latter *type II*. The reduction gives the first component of the challenge to the attacker as its leakage on A_j. It uses the value $\vec{\eta}^\top = A'_j\vec{1}^\top$ (from the second component of the challenge) to compute an $n \times n$ rescaling-matrix N whose diagonals are the entries of $\vec{\eta}$ so that $A_j = NA'_j$ is the rescaled version of A'_j. It sets $R_{j+1} := N(A'_jR_j)$ (where A'_jR_j is in the second component of the challenge) and gives R_{j+1} to the attacker. It also sets $V := NV' \in \mathbb{F}_q^{n\times(u+\rho)}$ and interprets the columns of V as the $u + \rho$ vectors

$$\{\vec{v}_{i,k} \mid 1 \le i \le t, 1 \le k \le u_i\} \quad , \quad \{\vec{\mu}_k\}_{k=1}^{\rho}$$

For each update $i > j$, the reduction chooses the updates A_i iteratively by rescaling a random $A'_i \xleftarrow{\$} \mathsf{Rk}_d(\mathbb{F}_q^{n\times n} \mid \mathsf{row} \in \mathcal{W}_i)$ where

$$\mathcal{W}_i = (\ \{ D_{i-1}\vec{r}_k \mid k \in J_i \} \quad , \quad \{ D[i-1, j+1]\vec{v}_{i,k} \mid 1 \le k \le u_i\} \ )^\perp$$

and the values $D_{i-1}\vec{r}_k$ are derived from R_{j+1} and $A_{j+1}, A_{j+2} \ldots$ (without knowing A_j). Similarly, The reduction also samples the final ρ vectors as $\{ D[t, j+1]\vec{\mu}_k\}_{k=1}^{\rho}$. It gives the attacker leakage on the updates A_i, the full values R_i, and finally the ρ vectors $\vec{\mu}_k$.

If the challenge is of *type I* and $V' = A'_iU$ for a uniform U then the reduction produces the distribution *GameHybrid j*. This is because, for any columns $\vec{v}$ of V we can write $D[i-1, j+1]\vec{v} = D[i-1, j+1]A_j\vec{u} = D[i-1, j]\vec{u}$ where $\vec{u}$ is uniformly random. So all of the spaces $\mathcal{W}_i$ and the final vectors $\vec{\mu}_k$ are distributed as in *GameHybrid j*. On the other hand, if the challenge is *type II*, then the reduction produces the distribution *GameHybrid* $j + 1$. This is because $V' = U$ is uniformly random and hence $V = NU$ is also uniformly random (since N is full-rank and U is independent of N).

Therefore the hybrids are indistinguishable from each other and hence the left and right hand distributions of the lemma are indistinguishable.

□

Proof of Lemma 6.6.2. We just apply Lemma 6.6.3 twice with $\rho = 0$ to get

$$\text{GameProgram}(\mathcal{R}) \overset{\text{stat}}{\approx} \text{GameProgram}(\mathcal{R}^*) \overset{\text{stat}}{\approx} \text{GameProgram}(\mathcal{R}').$$

Proof of Lemma 6.6.1. Let us write the initial matrix as $R = [R^{\text{left}}|R^{\text{right}}]$ where R^{left} consists of the first ρ columns. Then, the choice of the updates $A_1, \ldots, A_t$ in the programming experiment is completely independent of R^{left}. Therefore, we can think of playing the programming experiment with *only* R^{right} and choosing the ρ columns of R^{left} randomly afterwards. Therefore, applying Lemma 6.6.3 to only the $l' = l - \rho$ sub-matrix R^{right} (and setting $\ell = 0$) we get statistical indistinguishability between the distributions on the ρ columns given by $R^{\text{left}}_t = D_t R^{\text{left}} \overset{\text{stat}}{\approx} V$ where V is uniformly random over $\mathbb{F}_q^{n \times \rho}$. It's also clear that $R^{\text{right}}_t = D_t R^{\text{right}} = \mathbf{0}$, and so $R_t = [R^{\text{left}}_t | R^{\text{right}}_t] \overset{\text{stat}}{\approx} [V|\mathbf{0}]$, as we wanted to show.

6.7 Proof Part III. Series of Hybrids

6.7.1 Informal Overview

We first give an informal overview of the main proof strategy, which should prove helpful in understanding the formal proof. However, all of the notions we introduce here will be defined formally later and hence this section is not strictly necessary if one wishes to follow the full formal proof.

We use a series of hybrid arguments, where each step either takes advantage of the fact that the adversary is *computationally bounded* and cannot distinguish the rank of various matrices in the exponent, or of the fact that the adversary is *leakage bounded* and is only seing partial leakage on any key and ciphertext. When we use computational steps, we will even assume that the attacker gets *full* leakage and so we *cannot* modify any property of the game that could be efficiently tested given the keys and ciphertexts in full – in particular, we cannot modify whether any secret key correctly decrypts the challenge ciphertext in any time period. When we use leakage steps, we can even assume that the attacker is computationally unbounded and hence we cannot modify the distribution of any individual key/ciphertext/update – but we can modify various *correlations* between them, which may not be testable given just partial leakage on the individual values.[2]

The main strategy is to move from a game where all keys/updates and the challenge ciphertext are *honest* to a game where the keys no longer decrypt the

[2]This is a good way of viewing essentially all prior results in leakage resilient cryptography. Since we do not have computational assumptions that allow us to reason about leakage directly, we alternate between using computational steps that work even in the presence of unrestricted leakage and information theoretic steps that take advantage of the leakage being partial.

challenge ciphertext correctly. We can use *computational steps* to change the distribution of the initial key (honest/mid/high) or ciphertext (honest/dishonest) but *only* if they still decrypt correctly. For example, we can make the initial key a *mid key* and ciphertext a *correlated dishonest ciphertext* while maintaining computational indistinguishability. We then want to use a *leakage step* to argue that the attacker cannot notice if the correlation disappears and decryption stops working. For this, we would like to rely on orthogonality hiding Corollary 6.2.4. Unfortunately, since leakage on keys is continual, we cannot use this lemma directly. To get around this, we carefully program our updates to reduce the rank of future keys, so that the leakage on the vector $\vec{w}_2$ only occurs in a single mid key. We carefully arrange the hybrids so as to make this type of argument on each key/ciphertext pair, one at a time.

Hybrid Games. We start with the *Real Security Game*, which follows the definition of CLR PKE (Definition 6.1.1). That is, the challenger chooses the initial secret key sk_1 honestly, chooses the update matrices A_i honestly, and chooses the challenge ciphertext ct honestly as an encryption of the challenge message $\mathbf{m}_b$ where $b \xleftarrow{\$} \{0,1\}$ is chosen randomly.

Next we introduce a series of hybrid games, called *Game* $i = 0, 1, 2, \ldots$, which are all of the following type. For $i \geq 2$, the challenger chooses the initial key sk_1 as a high key, the first $i-2$ updates are honest (and hence the keys $sk_1, \ldots, sk_{i-1}$ are high keys), the update A_{i-1} is programmed to reduce the key to a mid key sk_i, and the update A_i is programmed to reduce the key to a honest key sk_{i+1}. The rest of the updates are honest and hence the keys $sk_{i+1}, sk_{i+2}, \ldots$ are honest. For the special case $i = 1$, the initial key sk_1 already starts out as a mid key and the first update A_1 reduces it to an honest key. For the special case $i = 0$, the initial key sk_1 is already an honest key. In all these games, the challenge ciphertext is chosen as an (uncorrelated) *dishonest* ciphertext encrypting the message $\mathbf{m}_b$.

We also define analogous games *GameCor* i where the initial secret key and the update matrices are chosen as in *Game* i, but the challenge ciphertext is chosen as a*correlated dishonest ciphertext* encrypting the message $\mathbf{m}_b$.

Sequence of Steps. Our first step is to move from *Real Game* to *Game* 0. The only difference in this step is that we change the challenge ciphertext from an *honest ciphertext* to a *dishonest ciphertext*. All secret keys in the game still correctly decrypt the challenge ciphertext, since they are all honest keys. This is a computational step and is indistinguishable even given full leakage.

Next, our goal is to keep moving from *Game* i to *Game* $i+1$. Unfortunately, we *cannot* just increment i in a single step since each such move changes sk_{i+1} from an honest key to a mid key and hence changes it from correctly decrypting the dishonest challenge ciphertext to not. Instead, we move from *Game* i to *GameCor*

$i+1$ in a single computational step. Even though the key sk_{i+1} changes from an honest key in *Game* i to a mid key in *GameCor* $i+1$, by making the challenge ciphertext correlated, it is still decrypted by the mid keys sk_{i+1} and all future honest keys, but none of the prior high keys. Therefore, these games cannot be distinguished even given full leakage.

Next, we move from *GameCor* $i+1$ to *Game* $i+1$. The only difference is the correlation between the basis vector $\vec{w}_2$ in the mid secret key sk_{i+1} and the challenge-ciphertext vector $\vec{c}$. In *GameCor* $i+1$ they are orthogonal, but in *Game* $i+1$ they are random. However, we now use a leakage step, relying on *subspace hiding* Corollary 6.2.3, to show that these two games are indistinguishable. In particular, the vector $\vec{w}_2$ is either uniformly random or a random vector orthogonal to $\vec{c}$, but the attacker can only get partial leakage on it prior to seing any information about $\vec{c}$.

Assume the attacker makes at most q_{sk} update queries. Then eventually, we move to *Game* q_{sk} where all of the keys in the game are high keys (i.e. the initial key is a high key and all the updates are honest). Therefore, none of the keys that the attacker leaks on in this game can correctly decrypt the challenge ciphertext. We can now use a computational step to argue that, even if the attacker got the keys in full, the challenge ciphertext hides the plaintext $\mathbf{m}_b$ and hence the challenge bit b.

Under the Rug. The above discussion is slightly oversimplified. The main issue is that the computational transitions, e.g. from *Game* i to *GameCor* $i+1$, are not computationally indistinguishable the way we defined the games. This is because in *Game* i the update matrix A_{i+1} is unlikely to annihilate any vectors (i.e. its kernel is unlikely to contain non-zero vectors from the span of the previous updates) while in *GameCor* $i+1$ it is programmed to annihilate vectors so as to reduce the dimension of the key. This can be efficiently tested given full leakage of the update matrices. Therefore, in the full proof, we define the games *Game* and *GameCor* slightly differently with some updates programmed to annihilate additional uniformly random vectors. With this modification, we can prove computational indistinguishability. We also need extra information theoretic steps to argue that the attacker cannot tell if updates are programmed to annihilate some random vectors, given limited leakage.

In Section 6.7.2, we define the main hybrid games that are used in the proof. Then, in Section 6.7.3, we proceed to prove the indistinguishability of the various hybrid games.

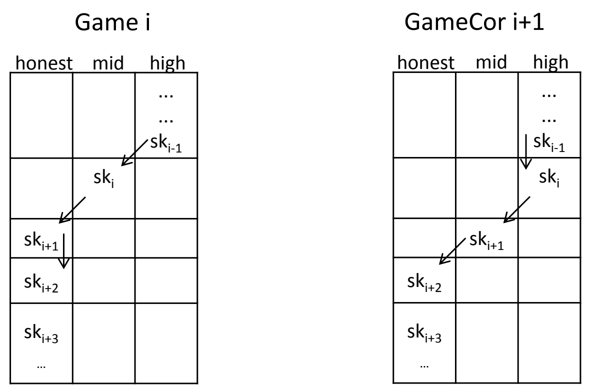


Figure 6.2: The main hybrid games. Arrows correspond to programmed updates.

6.7.2 Hybrid Game Definitions

We first define several hybrid games. The main part of the proof is to show computational/statistical indistinguishability between these games. The output of each game consists of the view of the attacker $\mathcal{A}$ as well as a bit b chosen by the challenger representing its choice of which message $\mathbf{m}_0, \mathbf{m}_1$ to encrypt. We assume that the attacker $\mathcal{A}$ makes a maximum of q_{sk} update queries on the secret-key during the course of the game.

Real Game : This is the original "ℓ-CLR-PKE" security game between the adversary and the challenger (see Definition 6.1.1). The initial secret key sk_1 as well as all of the updates and the challenge ciphertext are *honest*. We define the output of the game to consist of the view of the attacker $\mathcal{A}$ and the bit b chosen by the challenger. Therefore, given the output of the game, we can efficiently tell if "$\mathcal{A}$ wins".

Game' 0 : In this game, the challenger chooses the initial ciphertext incorrectly as a *dishonest* ciphertext of the message $\mathbf{m}_b$ instead of encrypting it honestly. The initial secret key sk_1 is chosen honestly and all key updates are chosen honestly as before.

Game i : We define Game i for $i = 0, \dots, q_{sk} + 1$. In all future game definitions, we will include two additional "dummy keys" sk_{-1}, sk_0 and key-update matrices A_{-1}, A_0 chosen by the challenger (but not observed or leaked on by the attacker). That is, the challenger (in its head) always initially chooses sk_{-1}, then updates it to sk_0 using an update matrix A_{-1}, then updates sk_0 to sk_1 using an update matrix A_0 and so on. However, the values $sk_{-1}, A_{-1}, sk_0, A_0$ are then ignored, and the first key and update matrix that the attacker can ask leakage queries on are sk_1 and A_1 respectively. In every Game i, the initial key sk_{-1} is chosen as a *high key* with the exponent matrix:

$$S = \begin{bmatrix} | & \cdots & | \\ \vec{r}_1^{\top} & \cdots & \vec{r}_{m-1}^{\top} \\ | & \cdots & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ & \cdots & \\ - & \vec{w}_{m-1} & - \end{bmatrix} + \begin{bmatrix} \vec{1}^{\top} \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

The first key update matrices $A_{-1}, \dots, A_{i-2}$ are chosen honestly. The update A_{i-1} is *programmed to annihilate* the $m - 3$ vectors $(\vec{r}_3, \dots, \vec{r}_{m-1})$ reducing the key to a *mid*. The update A_i is programmed to annihilate $\vec{r}_2$ along with $m - 4$ random vectors, reducing the key to an *honest* key. The update A_{i+1} is programmed to annihilate a single random vector. All other key updates are chosen honestly. Note that, in Game i, the keys $sk_{-1}, \dots, sk_{i-1}$ are high keys, sk_i is a mid key, and $sk_{i+1}, \dots$ are honest keys.[3] The challenge ciphertext is a *dishonest ciphertext*. Notice that the secret keys $sk_{-1}, \dots, sk_i$ do *not* decrypt the challenge ciphertexts correctly, but all future keys $sk_{i+1}, \dots, sk_{q_{sk}+1}$ do.

GameCor $i + 1$: We define GameCor $i + 1$ for $i = 0, \dots, q_{sk} + 1$. This game is similar to *Game $i + 1$* with the main difference being tat now the challenge ciphertext is chosen as an *correlated dishonest ciphertext*. However, the sequence of updates also differs in which updates annihilate additional random vectors. In particular, the initial key sk_{-1} is chosen as a high key, just like in *Game $i+1$*. The first key update matrices $A_{-1}, \dots, A_{i-2}$ are chosen honestly. The update A_{i-1} is *programmed to annihilate* the $m - 3$ uniformly random and independent vectors. The update A_i is programmed to annihilate the secret key vectors $\vec{r}_{m-1}, \dots, \vec{r}_3$,

[3]In particular, in *Games 0* the initial observed (non-dummy) key sk_1 is an honest key, in *Game* 1 it is a mid key, and in all future games it is a high key. For the proof, it becomes easier to just pretend that there are some "dummy" mid and high keys sk_{-1}, sk_0 even in Games 0,1 so as not to have to define special corner cases for these games.

reducing the key to an *mid* key. The update A_{i+1} is programmed to annihilate $\vec{r}_2$, reducing the key to an honest key. All other key updates are chosen honestly.

GameFinal : This game is defined the same way as *Game* $q_{sk}+1$, except that instead of using the message $\mathbf{m}_b$ in the dishonest ciphertext, we just use message $1_{\mathbb{G}_T}$. More specifically, in *GameFinal*, the secret key is a random *high key* and all of the relevant key updates that the attacker leaks on are honest. The ciphertext is a dishonest ciphertext encryting the message $1_{\mathbb{G}_T}$ and therefore the view of the attacker in GameFinal is independent of the challenger's bit b.

6.7.3 Hybrid Indistinguishability Arguments

Lemma 6.7.1. *Real* $\overset{\text{comp}}{\approx}$ *Game' 0.*

Proof. We show a reduction from the $(k=1)$-*extended rank hiding assumption* (Definition 6.2.5, Lemma 6.2.6). The reduction is given a challenge $\mathbf{g}^P$, $\vec{w}$ where $P \xleftarrow{\$} \mathsf{Rk}_x(\mathbb{F}_q^{2\times m})$ is either of rank $x=1$ or $x=2$ and $\vec{w} \xleftarrow{\$} \mathsf{ker}(P)$. Let us denote the two rows of P by $\vec{p}, \vec{c}$ respectively. The reduction puts the values $\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}}$ in prms and chooses its own random $\vec{t}$ to set up the initial (honest) public/secret key pk, sk. To create the encryption of $\mathbf{m}_b$, the reduction sets $ct = (\mathbf{g}^{\vec{c}}, e(\mathbf{g},\mathbf{h})^{\langle\vec{c},\vec{t}\rangle}\mathbf{m}_b)$. The reduction chooses all of the key/ciphertext update matrices honestly and answers all update/leakage queries honestly.

If the challenge has $x=1$ then $\vec{c}_1 = u'\vec{p}$ for some scalar u' and hence the ciphertext is a correctly distributed *honest* encryption of $\mathbf{m}_b$, so the reduction produces the distribution of the *Real Game.* If $x=2$ then $\vec{c}_1$ is a random and independent vector in the space $(\vec{w})^{\perp}$ and hence the ciphertext is a correctly distributed *dishonest* encryption of $\mathbf{m}_b$, so the reduction produces the distribution of *Game' 0.* Therefore the two are computationally indistinguishable. $\square$

Lemma 6.7.2. *If* $\ell \le (n-3m+6)\log(q) - \omega(\log(\lambda))$ *then* Game' *0* $\overset{\text{stat}}{\approx}$ Game *0.*

Proof. The only difference between *Game' 0* and *Game 0* is the joint distribution on the initial secret key sk_1 and the update matrix A_1. Conditioned on sk_1, A_1, all other values are sampled the same way in the two games. In *Game' 0*, the key sk_1 is a randomly chosen *honest key*, and A_1 is an *honest update.* In *Game 0* the distribution on sk_1, A_1 is slightly more complicated. First we choose a random *high key* $sk_{-1} = \mathbf{h}^{S-1}$ with exponent

$$S_{-1} = \begin{bmatrix} | & \cdots & | \\ \vec{r}_1^{\top} & \cdots & \vec{r}_{m-1}^{\top} \\ | & \cdots & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ & \cdots & \\ - & \vec{w}_{m-1} & - \end{bmatrix} + \begin{bmatrix} \vec{1}^{\top} \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

Then we choose the updates: A_{-1} programmed to annihilate $\vec{r}_3, \ldots, \vec{r}_{m-1}$ yielding a mid-key sk_0, A_0 is programmed to annihilate $\vec{r}_2$ along with $m-4$ random vectors yielding an honest key sk_1, and A_1 is programmed to annihilate a single random vector.

We claim this joint distribution on A_1 and sk_1 in *Game 0* is statistically indistinguishable from that of *Game' 0*. To see this, we first apply the *reprogramming lemma* (Lemma 6.6.2 with $u^* = m-3, d = n-m+3, l = m-1$) to switch from A_1 being programmed to annihilate a single vector to just being honest. Next we just use *program correctness lemma* (Lemma 6.6.1; with $l = m-1$, $\rho = 1$, $u = m-4$, $d = n-m+3$) to argue that the distribution of the key sk_1 in *Game 0* is statistically close to choosing a fresh honest key. In particular, the lemma tells us that $A_0 A_{-1}[\vec{r}_1^\top | \cdots | \vec{r}_{m-1}^\top] \overset{\text{stat}}{\approx} [\vec{r}^\top | \mathbf{0} \cdots]$ for a uniformly random $\vec{r} \in \mathbb{F}_q^n$. So the exponent of the key sk_1 in *Game 0* is

$$A_0 A_{-1} S_{-1} = A_0 A_{-1} \vec{r}_1^\top \vec{w}_1 + \vec{1}^\top \vec{t} \quad \overset{\text{stat}}{\approx} \quad \vec{r}^\top \vec{w} + \vec{1}^\top \vec{t}$$

and hence statistically indistinguishable from that of a random honest key as in *Game' 0*.

□

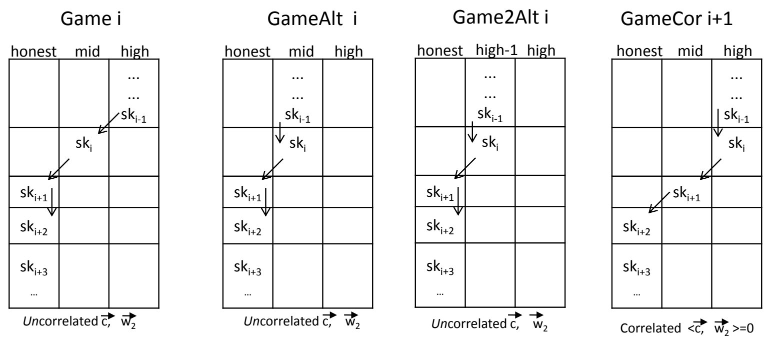


Figure 6.3: Additional hybrids for Lemma 6.7.3

Lemma 6.7.3. *For $i \in \{0, \ldots, q_{sk}\}$: Game $i \overset{\text{comp}}{\approx}$ GameCor $i+1$.*

Proof. For the proof of the lemma, we introduce two additional intermediate games (see Figure 6.3). Firstly, we define *GameAlt i*. In this game, the initial secret key sk_{-1} is a random *mid key* with exponent matrix

$$S \overset{\text{def}}{=} \begin{bmatrix} | & | \\ \vec{r}_1^\top & \vec{r}_2^\top \\ | & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ - & \vec{w}_2 & - \end{bmatrix} + \begin{bmatrix} \vec{1}^\top \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

The key update matrices $A_1, \ldots, A_{i-2}$ are chosen honestly, the update A_{i-1} is programmed to annihilate $m-3$ random vectors, the update A_i is programmed to annihilate $\vec{r}_2$ along with $m-4$ random vectors, and the update matrix A_{i+1} is programmed to annihilate a single random vector. The challenge ciphertext is chosen as in *Game i*.

Claim 6.7.4. *Game i* $\overset{\text{comp}}{\approx}$ *GameAlt i*

Proof. We show a reduction from the ($k=1$)-*extended rank hiding assumption* (Definition 6.2.5, Lemma 6.2.6). The challenger gets a challenge $\mathbf{h}^{W'}$ and $\vec{p}$, where $W' \xleftarrow{\$} \mathsf{Rk}_x(\mathbb{F}_q^{(m-2)\times m})$ is either of rank $x = 1$ or of rank $x = m-2$ and $\vec{p} \in \mathsf{ker}(W')$. Let us label the rows of W' by $\vec{w}_2, \ldots, \vec{w}_{m-1}$. Choose $\vec{w}_1 \xleftarrow{\$} (\vec{p})^\perp$ and $\vec{t} \xleftarrow{\$} \mathbb{F}_q^m$. Set $\mathsf{prms} = (\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}_1})$, $pk = e(\mathbf{g}, \mathbf{h})^{\vec{t}\cdot\vec{p}}$ and $sk_{-1} = \mathbf{h}^S$ where

$$S = \begin{bmatrix} | & & | \\ \vec{r}_1^\top & \cdots & \vec{r}_{m-1}^\top \\ | & & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ & \cdots & \\ - & \vec{w}_{m-1} & - \end{bmatrix} + \begin{bmatrix} \vec{1}^\top \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

for uniformly random $\vec{r}_1, \ldots, \vec{r}_{m-1}$ (note: the challenger can do this efficiently given $\mathbf{h}^{W'}$ without knowing W'). Create a dishonest ciphertext by choosing a random vector $\vec{c} \xleftarrow{\$} (\vec{w}_1)^\perp$. Run the rest of *Game i* correctly as a challenger, where the key-update A_{i-1} is programmed to annihilate $\vec{r}_3, \ldots, \vec{r}_{m-1}$, the key-update A_i is programmed to annihilate $\vec{r}_2$ and $m-4$ random vectors and the key update A_{i+1} update is programmed to annihilate a single random vector.

If W' is of rank $x = m-2$, then it is easy to see that the above distribution is that of *Game i*.

If W' is of rank $x = 1$ then we claim that the above is distribution is that of *GameAlt i*. Firstly, we can write $\vec{w}_3 = \mu_3\vec{w}_2, \ldots, \vec{w}_{m-1} = \mu_{m-1}\vec{w}_2$ for some scalars $\mu_3, \ldots, \mu_{m-1}$ in $\mathbb{F}_q$. Letting $\vec{\mu} = (1, \mu_3, \ldots, \mu_{m-1}) \in \mathbb{F}_q^{m-2}$, we can write

$$S = \begin{bmatrix} | & | \\ \vec{r}_1^\top & \vec{r}_2{}'^\top \\ | & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ - & \vec{w}_2 & - \end{bmatrix} + \begin{bmatrix} \vec{1}^\top \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

for

$$\begin{bmatrix} \vec{r}_2\,'^\top \end{bmatrix} = \begin{bmatrix} | & & | \\ \vec{r}_2^\top & \cdots & \vec{r}_{m-1}^\top \\ | & & | \end{bmatrix} \begin{bmatrix} \vec{\mu}^\top \end{bmatrix}$$

So sk_{-1} is a random *mid* key as in *GameAlt i*. Moreover, the vectors $\vec{r}_3, \ldots, \vec{r}_{m-1}$ are random and independent of $S, \vec{w}_1, \vec{w}_2, \vec{r}_1, \vec{r}_2\,'$. Therefore, the update matrix A_{i-1} is programmed to annihilate $m-3$ random and independent vectors $\vec{r}_3, \ldots, \vec{r}_{m-1}$ as in *GameAlt i*. Finally, the update A_i is programmed to annihilates $\vec{r}_2$ (along with $m-4$ random vectors), which is equivalent to being programmed to annihilate $\vec{r}_2'$ since $\mathsf{span}(D_{i-1}\vec{r}_2\,') = \mathsf{span}(D_{i-1}\vec{r}_2)$ where D_{i-1} is the product of all update matrices prior to A_i. So we see that the, when W' is of rank 1 then the choice of sk_{i-1} and all the update matrices is distributed correctly as in *GameAlt i*. Hence an attacker that distinguishes *Game i* and *GameAlt i* breaks rank-hiding. □

We now introduce a second intermediate game called *Game2Alt i* where the initial secret key sk_{-1} has an exponent of the form

$$S_{-1} \stackrel{\text{def}}{=} \begin{bmatrix} | & \cdots & | \\ \vec{r}_1^\top & \cdots & \vec{r}_{m-2}^\top \\ | & \cdots & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ & \cdots & \\ - & \vec{w}_{m-2} & - \end{bmatrix} + \begin{bmatrix} \vec{1}^\top \end{bmatrix} \begin{bmatrix} \vec{t} \end{bmatrix}$$

so that the rows are chosen randomly from the $m-2$ dimensional affine space $\vec{t} + \mathsf{span}(\vec{w}_1, \ldots, \vec{w}_{m-2})$. The key update matrices $A_1, \ldots, A_{i-2}$ are chosen honestly, the update A_{i-1} is programmed to annihilate $m-3$ random vectors, the update A_i is programmed to annihilate the vectors $\vec{r}_2, \ldots, \vec{r}_{m-2}$, and the update matrix A_{i+1} is programmed to annihilate a single random vector. The challenge ciphertext is chosen as in *Game i*.

We now show (in 2 steps) that *Game2Alt i* is computationally indistinguishable from *GameAlt i* and from *GameCor i*+1, which completes the proof of Lemma 6.7.3

Claim 6.7.5. *GameAlt i* $\stackrel{\text{comp}}{\approx}$ *Game2Alt i.*

Proof. We show a reduction from the $(k=1)$-*extended rank hiding assumption* (Definition 6.2.5, Lemma 6.2.6). The challenger gets a challenge $\mathbf{h}^{W'}$ and $\vec{p}$, where $W' \stackrel{\$}{\leftarrow} \mathsf{Rk}_x(\mathbb{F}_q^{(m-3)\times m})$ is either rank $x = 1$ or rank $x = (m-3)$ and $\vec{p} \in \mathsf{ker}(W')$. Let us label the rows of W' by $\vec{w}_2, \ldots, \vec{w}_{m-2}$. Choose $\vec{w}_1 \stackrel{\$}{\leftarrow} (\vec{p})^\perp$ and $\vec{t} \stackrel{\$}{\leftarrow} \mathbb{F}_q^m$. Set $\mathsf{prms} = (\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}_1})$, $pk = e(\mathbf{g}, \mathbf{h})^{\vec{t}\cdot\vec{p}}$ and $sk_{-1} = \mathbf{h}^S$ where

$$S = \begin{bmatrix} | & & | \\ \vec{r}_1^\top & \cdots & \vec{r}_{m-2}^\top \\ | & & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ & \cdots & \\ - & \vec{w}_{m-2} & - \end{bmatrix} + \begin{bmatrix} \vec{1}^\top \end{bmatrix} \begin{bmatrix} \vec{t} \end{bmatrix}$$

for uniformly random $\vec{r}_1, \ldots, \vec{r}_{m-2}$ (note: the challenger can do this efficiently given $\mathbf{h}^{W'}$ without knowing W'). Create a dishonest ciphertext by choosing a random vector $\vec{c} \xleftarrow{\$} (\vec{w}_1)^{\perp}$. The key-update A_{i-1} is programmed to annihilate $m-3$ random vectors, the key-update A_i is programmed to annihilate $\vec{r}_2, \ldots, \vec{r}_{m-2}$ and the key update A_{i+1} update is programmed to annihilate a single random vector.

If W' is of rank $x = m-3$, then it is easy to see that the above distribution is that of *Game2Alt i*.

If W' is of rank $x = 1$ then we claim that the above distribution is that of *GameAlt i*. This follows since we can write

$$S = \begin{bmatrix} | & | \\ \vec{r}_1^{\top} & \vec{r}_2\,'^{\top} \\ | & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ - & \vec{w}_2 & - \end{bmatrix} + \begin{bmatrix} \vec{1}^{\top} \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

for

$$\begin{bmatrix} \vec{r}_2\,'^{\top} \end{bmatrix} = \begin{bmatrix} | & & | \\ \vec{r}_2^{\top} & \cdots & \vec{r}_{m-2}^{\top} \\ | & & | \end{bmatrix} \begin{bmatrix} \vec{\mu}^{\top} \end{bmatrix}$$

and some vector $\vec{\mu}^{\top} \in \mathbb{F}_q^{m-3}$. So sk_{-1} is a random *mid key* as in *GameAlt i*. Moreover, an update A_i which is programmed to annihilate $\vec{r}_2, \ldots, \vec{r}_{m-2}$ is equivalent to being programmed to annihilate $\vec{r}_2\,'$ along with $m-4$ random vectors since $\mathsf{span}(\vec{r}_2, \ldots, \vec{r}_{m-2}) = \mathsf{span}(\vec{r}_2\,', \vec{r}_3, \ldots, \vec{r}_{m-2})$ where $\vec{r}_3, \ldots, \vec{r}_{m-2}$ are random an independent of sk_{-1} or any of the previous updates. Therefore sk_{-1} and all of the update matrices are distributed as in *GameAlt i*. □

Claim 6.7.6. *Game2Alt* $i \overset{\text{comp}}{\approx}$ *GameCor* $i+1$.

Proof. We show a reduction from the $(k=1)$-*extended rank hiding assumption* (Definition 6.2.5, Lemma 6.2.6). The reduction gets a challenge $\mathbf{h}^{W'}$ and $\vec{p}, \vec{c}$, where $W' \xleftarrow{\$} \mathsf{Rk}_x(\mathbb{F}_q^{2\times m})$ is either rank $x=1$ or rank $x=2$ and $\vec{p}, \vec{c} \in \mathsf{ker}(W')$. Let us label the rows of W' by $\vec{w}_1, \vec{w}_2$. The reduction chooses $\vec{w}_3, \ldots, \vec{w}_{m-1} \xleftarrow{\$} (\vec{p})^{\perp}$, $\vec{t} \xleftarrow{\$} \mathbb{F}_q^m$ and sets $\mathsf{prms} = (\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}_1})$, $pk = e(\mathbf{g}, \mathbf{h})^{\vec{t}\cdot\vec{p}}$ and $sk_{-1} = \mathbf{h}^{S}$ where

$$S = \begin{bmatrix} | & & | \\ \vec{r}_1^{\top} & \cdots & \vec{r}_{m-1}^{\top} \\ | & & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ & \cdots & \\ - & \vec{w}_{m-1} & - \end{bmatrix} + \begin{bmatrix} \vec{1}^{\top} \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

for uniformly random $\vec{r}_1, \ldots, \vec{r}_{m-1}$. (Note: the reduction can do this efficiently given $\mathbf{h}^{W'}$ without knowing W'). Lastly, the reduction creates a dishonest ciphertext using the vector $\vec{c}$ that comes from the challenge. The reduction does everything else as in *GameCor' i* $+1$ where the key-update A_{i-1} is programmed to annihilate $m-3$ random vectors, the key-update A_i is programmed to annihilate $\vec{r}_3, \ldots, \vec{r}_{m-1}$ and the key update A_{i+1} update is programmed to annihilate $\vec{r}_2$.

If W' is of rank $x = 2$, then it is easy to see that the above distribution is that of *GameCor'* $i+1$. Notice that the key and ciphertext basis are random *correlated* basis with $\vec{c} \in (\vec{w}_1, \vec{w}_2)^\perp$.

If W' is of rank $x = 1$ then we claim that the above distribution is that of *Gam2Alt i*. This follows since the exponent of sk_{-1} can be written as

$$S = \begin{bmatrix} | & | & & | \\ \vec{r}_1{}'^\top & \vec{r}_3^\top & \cdots & \vec{r}_{m-1}^\top \\ | & | & & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ - & \vec{w}_3 & - \\ & \cdots & \\ - & \vec{w}_{m-1} & - \end{bmatrix} + \begin{bmatrix} \vec{1}^\top \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

where $\vec{r}_1{}'^\top = \mu_1 \vec{r}_1^\top + \mu_2 \vec{r}_2^\top$ for some scalars μ_1, μ_2. Note that $\vec{r}_1{}'$ is therefore uniformly random and independent of $\vec{r}_2$. By relabeling, this has the same distribution as the initial key in *Game2Alt i*. Moreover, the update A_i is programmed to annihilate the $m-3$ vectors $\vec{r}_3, \ldots, \vec{r}_{m-1}$ in the initial secret key, while the update A_{i+1} annihilates the vector $\vec{r}_2$ which is random and independent of the initial secret key. Therefore the initial secret key and all of the secret key updates are distributed as in *Game2Alt i*. □

Putting Claim 6.7.4, Claim 6.7.5 and Claim 6.7.6 together, we complete the proof of the lemma. □

Lemma 6.7.7. *If* $\ell \le \min((m-3)/3, n-3m+6)\log(q) - \omega(\log(\lambda))$ *then*

$$\text{GameCor } i+1 \overset{\text{stat}}{\approx} \text{Game } i+1$$

Proof. There are two differences between *GameCor* $i+1$ and *Game* $i+1$. The first difference lies in wether the ciphertext is dishonest or correlated dishonest. The second difference lies in the update regime and which updates annihilate additional random vectors (although in both games, the update regimes agree on which updates reduce the rank of the key from high to mid to honest). We show statistical indistinguishability of each of these changes. We define *GameInter* $i+1$ which is just like *GameCor* $i+1$, with the same distribution of the secret key and the update matrices, but where the ciphertext is dishonest (not correlated).

Claim 6.7.8. *If* $\ell \le (m-4)\log(q)/3 - \omega(\log(\lambda))$, *then*

$$\textit{GameCor } i+1 \overset{\text{stat}}{\approx} \textit{GameInter } i+1$$

Proof. This follows by *subspace hiding* (Corollary 6.2.3). Let us fix the vectors $\vec{p}, \vec{w} = \vec{w}_1, \vec{t}$, and let us fix the space $\mathcal{W} = (\vec{p})^\perp$. Let us also fix vectors $\vec{w}_3, \dots, \vec{w}_{m-1}$ in the space $\mathcal{W}$ *but not* the vector $\vec{w}_2$. These values are sufficient to define the public key *pk* of the scheme and the initial *high* secret key in Game $i+1$ and GameCor $i+1$. That is, each row of the initial secret key is chosen as $\mathbf{h}^{\vec{t}+\vec{w}'}$ where $\vec{w}' \stackrel{\$}{\leftarrow} \mathcal{W}$ is chosen randomly. Moreover, the first $i-1$ updates are performed honestly or they annihilate some random vectors (in both games) and hence we can simulate all these values without fixing $\vec{w}_2$.

Let $M \stackrel{\$}{\leftarrow} \mathcal{W}^{m-3}$ be random matrix and let $\vec{w}_2$ be some vector which is either chosen as (I) $\vec{w}_2 := M\vec{v}^\top$ for $\vec{v} \stackrel{\$}{\leftarrow} \mathbb{F}_q^{m-3}$ or (II) $\vec{w}_2 \stackrel{\$}{\leftarrow} \mathcal{W}$. We will use $\vec{w}_2$ to simulate leakage on the updates A_i, A_{i+1} and the secret keys sk_{i+1}. In particular, once we fix $\vec{w}_2$, we also fix the vectors $\vec{r}_1, \dots, \vec{r}_{m-1}$ so that the initial high key can be written as $\mathbf{h}^S$ with

$$S = \begin{bmatrix} | & \cdots & | \\ \vec{r}_1^\top & \cdots & \vec{r}_{m-1}^\top \\ | & \cdots & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ & \cdots & \\ - & \vec{w}_{m-1} & - \end{bmatrix} + \begin{bmatrix} \vec{1}^\top \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

Then we choose update A_i to annihilate $\vec{r}_3, \dots, \vec{r}_{m-1}$, and A_{i+1} to annihilate $\vec{r}_2$. Note that the secret key sk_{i+2} and the rest of the updates are then completely independent of $\vec{w}_2$ (we can choose it ahead of time). Therefore, in total, the attacker gets leakage on $\vec{w}_2$ in only 3 rounds of leakage and therefore gets at most 3ℓ bits of information about $\vec{w}_2$. We can now simulate the rest of the game by choosing updates honestly. To give the dishonest challenge ciphertext, we choose a vector $\vec{c}$ at random orthogonal to the columns of M and $\vec{w}_1$. Note that (information about) the matrix M is only used *after* leakage occurs on the vector $\vec{w}_2$.

If the challenge if of type (I) then this gives us the distribution of *GameCor* $i+1$ with $\vec{w}_2 \stackrel{\$}{\leftarrow} (\vec{c}, \vec{p})^\perp$ and otherwise we get the distribution of *Game* $i+1$ with $\vec{w}_2 \stackrel{\$}{\leftarrow} (\vec{p})^\perp$. Therefore, these two games are statistically indistinguishable by Corollary 6.2.3.

□

We now turn to changing the regime of ciphertext updates from *GameInter* $i+1$ to *Game* $i+1$. But for this we just use the *reprogramming lemma* (Lemma 6.6.2) on the ciphertext updates. In particular, we rely on the fact that the regime of ciphertext updates between *GameInter* and *Game* only differs in which updates annihilate additional random vectors (but *not* in how they annihilate the ciphertext exponent vectors $\vec{u}_i$). Therefore, using the reprogramming lemma, we see that the two games are indistinguishable as long as $\ell \leq (n - 3m + 6)\log(q) - \omega(\log(\lambda))$.
□

Lemma 6.7.9. Game $q_{sk}+1 \overset{\text{stat}}{\approx}$ GameFinal.

Proof. Notice that in *Game* $q_{sk}+1$, the initial secret key sk_{-1} is a high key and all of the update matrices that the attacker leaks on are honest. Therefore, the attacker does not learn any information about the vector $\vec{t}$ created during key generation, beyond just the value $\langle \vec{p}, \vec{t} \rangle$ in the public key (this is true even if the attacker got full leakage on all secret keys). Now, the dishonest challenge ciphertext if of the form $\mathbf{g}^{\vec{c}}, e(\mathbf{g},\mathbf{h})^{\langle \vec{c}, \vec{t} \rangle}\mathbf{m}_b$, where $\vec{c}$ is uniformly random and independent of $\vec{t}$. Therefore, given everything the attacker sees in the game prior to the challenge ciphertext, the value $\langle \vec{c}, \vec{t} \rangle$ is uniformly random, and hence the message $\mathbf{m}_b$ and the bit b are perfectly hidden by the ciphertext. $\square$

6.7.4 Putting It All Together

Using Lemmata 6.7.1 - 6.7.9, we get the indistinguishability: *Real* $\overset{\text{comp}}{\approx}$ *GameFinal.* Recall that the output of each game includes the view of the attacker $\mathcal{A}$ at the end of the experiment along with the challenger's selection bit b. Since the attacker's guess $\tilde{b}$ at the end of the game can be efficiently computed from the view of the attacker, the predicate ($\mathcal{A}$ wins) $\Leftrightarrow$ ($\tilde{b} \overset{?}{=} b$) can be efficiently computed from the output of each game. In *GameFinal* the view of the attacker is independent of the random bit b and hence we have $\Pr[\mathcal{A} \text{ wins }] = \frac{1}{2}$. Therefore, in the *Real* game, we must have $|\Pr[\mathcal{A} \text{ wins }] - \frac{1}{2}| \leq negl(\lambda)$ since the two games are indistinguishable. This concludes the proof of Theorem 6.4.1.

6.8 Generalization to k-Linear

In this section, we provide a generalized scheme which we can prove secure under the k-linear assumption for arbitrary choices of k. When $k = 1$, the scheme description and proof coincide exactly with the original.

6.8.1 Scheme Description

Let k, n, m, d be integer parameters of the system with $n \geq d$.

KeyGen(1^λ) $\rightarrow (pk, sk)$: Sample $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathbf{g}, \mathbf{h}, q) \leftarrow \mathcal{G}(1^\lambda)$ to be the description of a bilinear group of primer order q, with an efficient pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, and generators $\mathbf{g} \in \mathbb{G}_1, \mathbf{h} \in \mathbb{G}_2$.

Choose matrices $P, W \in \mathbb{F}_q^{k \times m}$ at random subject to $\mathsf{rowspan}(P) \perp \mathsf{rowspan}(W)$ and set

$$\mathsf{prms} = ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathbf{g}, \mathbf{h}, q), \mathbf{g}^P, \mathbf{h}^W)$$

to be the *public parameters* of the system.[4]

Choose $\vec{t} \xleftarrow{\$} \mathbb{F}_q^m$ and set $pk := e(\ \mathbf{g}^P\ ,\ \mathbf{h}^{\vec{t}^\top}\ ) = e(\mathbf{g},\mathbf{h})^{\vec{\alpha}}$ where $\vec{\alpha}^\top = P\vec{t}^\top$. Choose $R \xleftarrow{\$} \mathbb{F}_q^{n\times k}$ and set $sk := \mathbf{h}^S$, where S is the $n \times m$ matrix given by

$$S \ := \ \begin{bmatrix} \\ R \\ \\ \end{bmatrix} \begin{bmatrix} \quad W \quad \end{bmatrix} + \begin{bmatrix} \\ \vec{1}^\top \\ \\ \end{bmatrix} \begin{bmatrix} \quad \vec{t} \quad \end{bmatrix}.$$

In other words, each row of S is chosen at random from the affine subspace $\vec{t} + \mathsf{rowspan}(W)$. (Note that $\mathbf{h}^S$ can be computed from the components $\mathbf{h}^W$, $\vec{t}$, R without knowing W.)

Encrypt$_{pk}(\mathbf{m}) \to ct$ **:** To encrypt $\mathbf{m} \in \mathbb{G}_T$ under $pk = e(\mathbf{g},\mathbf{h})^{\vec{\alpha}}$, choose $\vec{u} \in \mathbb{F}_q^k$ and output: $ct = (\mathbf{g}^{\vec{u}P}, e(\mathbf{g},\mathbf{h})^{\vec{u}\cdot\vec{\alpha}^\top}\ \mathbf{m}\ )$.

Decrypt$_{sk}(ct) \to \mathbf{m}$**:** To decrypt, we only need to look at the first row of the secret key. Given the first row $\mathbf{h}^{\vec{s}}$ of the secret key $sk = \mathbf{h}^S$, and the ciphertext component $ct^{(1)} = \mathbf{g}^{\vec{c}}, ct^{(2)} = e(\mathbf{g},\mathbf{h})^z$, the decryption algorithm outputs: $\mathbf{m} = e(\mathbf{g},\mathbf{h})^z / e(\ \mathbf{g}^{\vec{c}}\ ,\ \mathbf{h}^{\vec{s}^\top}\ )$.

Update$(sk) \to sk'$ **:** Choose a random matrix $A' \xleftarrow{\$} \mathsf{Rk}_d(\mathbb{F}_q^{n\times n})$. Derive A by "rescaling" each row of A' so that its components sum up to 1. That is, set $(A)_{i,j} := (A')_{i,j} / (\sum_{l=1}^n (A')_{i,l})$.

If the current secret key is $sk = \mathbf{h}^S$, output the updated key $sk' := \mathbf{h}^{AS}$.

Theorem 6.8.1. *For any integers $m \geq 4k, n \geq 3m - 7k + 1$ and $d := n - m + 3k$ the scheme* (KeyGen, Encrypt, Decrypt,Update) *is an ℓ-CLR-PKE scheme under the k-linear assumption for any*

$$\ell = \min\left((m - 3k - 1)/3, n - 3m + 7k - 1\right)\log(q) - \omega(\log(\lambda)).$$

The proof of the theorem closely follows that of the SXDH scheme and we only sketch the needed modifications.

6.8.2 The Generalized Proof (Sketch)

We now give an overview of the modifications to the previous proof necessary to generalize it to the k-linear assumption. The overall structure of the proof and the hybrid games is exactly the same, except that we modify how low/mid/high keys and honest/dishonest ciphertexts are defined.

[4] We can interpret the above as choosing W at random and choosing each row of P at random from $\mathsf{ker}(W)$.

Alternate Key and Ciphertext Distributions. Assume the matrices P, W and the vector $\vec{t}$ are fixed defining $\mathbf{g}^P, \mathbf{h}^W$ and $pk = e(\mathbf{g}, \mathbf{h})^{P\vec{t}^\top}$. Let us label the rows of W by $\vec{w}_1, \dots, \vec{w}_k$ and let $(\vec{w}_1, \dots, \vec{w}_{(m-k)})$ be a basis of $\mathsf{ker}(P) = \mathsf{rowspan}(P)^\perp$.

We define the various key distributions on $sk = \mathbf{h}^S$ the same way as before with

$$S = \begin{bmatrix} | & & | \\ \vec{r}_1^\top & \cdots & \vec{r}_i^\top \\ | & & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ & \cdots & \\ - & \vec{w}_i & - \end{bmatrix} + \begin{bmatrix} \\ \vec{1}^\top \\ \\ \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix} \tag{6.3}$$

but now we have $i = k$ for ***honest***, $i = 2k$ for ***mid*** and $i = (m-k)$ for ***high*** keys.

Similarly, *ciphertexts* are always of the form $ct = (\mathbf{g}^{\vec{c}}, e(\mathbf{g}, \mathbf{h})^{\langle \vec{c}, \vec{t} \rangle}\mathbf{m})$. For ***honest ciphertexts*** we have $\vec{c} = \vec{u}P$ and for ***dishonest ciphertexts*** we have $\vec{c} \xleftarrow{\$} (\vec{w}_1, \dots, \vec{w}_k)^\perp = \mathsf{ker}(W)$. Lastly, for ***correlated dishonest ciphertexts*** we have $\vec{c} \xleftarrow{\$} (\vec{w}_1, \dots, \vec{w}_{2k})^\perp$. The interactions between keys and ciphertexts still follows the outline given in Figure 6.1.

The Hybrids. We also define the various games *Game i*, *GameCor i* analogously as before with appropriate modifications. For example, in *Game i*, the key updates $\dots, A_{i-2}$ are honest, the update A_i (high to mid) is now programmed to annihilate the $m - 3k$ vectors $\vec{r}_{2k+1}, \dots, \vec{r}_{m-k}$, the update A_{i+1} (mid to low) is programmed to annihilate the k vectors $\vec{r}_{k+1}, \dots, \vec{r}_{2k}$ along with $m - 4k$ random vectors, and the update A_{i+1} is now programmed to annihilate k random vectors. The other game definitions are all analogous.

All of the computational steps are performed analogously, but now under the *k-extended rank hiding assumption* which follows from k-linear. The information theoretic step in Lemma 6.7.2, is also analogous and the new bound becomes: $\ell \le n - 3m + 7k - 1$. [5]

The only part that requires a little more work is the analogue of Claim 6.7.8, where we go from a correlated dishonest ciphertext to just plain dishonest. Again, let us think of the matrices P and W as fixed, defining the columns of W to be $\vec{w}_1, \dots, \vec{w}_k$. Let us also fix the vectors $\vec{w}_{2k+1}, \dots, \vec{w}_{m-k}$ as fixed (but not the mid vectors $\vec{w}_{k+1}, \dots, \vec{w}_{2k}$.

We can think of switching from *GameCor* $i+1$ to *Game* $i+1$ as switching whether the key vectors $\vec{w}_{k+1}, \dots, \vec{w}_{2k}$ come from $\mathcal{W} := \mathsf{ker}(P)$ in the latter case or

[5]To demystify this slightly, we need to set $d = n - m + 3k$ since we need to be able to go from a high key of dimension $m - k$ to a mid key of dimension $2k$ in one step, therefor annihilating $m - 3k$ vectors with a single update matrix. To use the "program correctness" and "reprogramming" lemmas, we need $d - l - u - 1 = \omega(\log(\lambda))$ where $d = n - m + 3k$ is the rank of the update matrices, $l = m - k$ is the rank of the high keys, and $u = m - 3k$ is the number of random vectors that are annihilated in our update regimes.

a smaller subspace $\mathcal{W}' := \mathcal{W} \cap (\vec{c})^{\perp}$ in the former case. Again, we can think of the subspace $\mathcal{W}'$ as being chosen by first selecting an $m \times (m - 2k - 1)$ matrix $M \xleftarrow{\$} \mathcal{W}^{m-2k-1}$, defining $\mathcal{W}' = \mathsf{colspan}(M)$ and then selecting $\vec{c} \xleftarrow{\$} \mathsf{colspan}([M||W])^{\perp}$. We can then think of the vectors $\vec{w}_{k+1}, \ldots, \vec{w}_{2k}$ as being chosen at random form either $\mathsf{colspan}(M)$ or from all of $\mathcal{W}$. By applying the subspace hiding lemma (Corollary 6.2.3), we get that this is indistinguishable as long as the leakage on $\vec{w}_{k+1}, \ldots, \vec{w}_{2k}$ is bounded by $(m - 3k - 1)\log(q) - \omega(\log(\lambda))$. Since the leakage on these vectors only occurs in at most 3 time periods, it is bounded by 3ℓ, giving us the requitement $\ell = ((m - 3k - 1)/3)\log(q) - \omega(\log(\lambda))$.

Chapter 7

CLR Secret Sharing and Storage

In this chapter, we deviate somewhat from previous chapters in that we now consider *two* independently leaky devices instead of a single leaky device. Each of these devices stores a share of some shared message, and the shares are periodically updated by the devices in an individualized and asynchronous manner. The attacker can continually observe partial leakage on the individual state of each device (including its share and update randomness) while ensuring that the secret remains securely hidden. As mentioned earlier, we can also think of this as providing a method for storing a value secretly on a single leaky device in a restricted model of leakage, where two sub-components of the device are leaking individually (i.e. the attacker can leak arbitrary information from each component individually, but not a global function of the entire state of the system).

7.1 Definitions

7.1.1 Continual-Leakage-Resilient Sharing (CLRS)

We now formally define the notion of a *continual-leakage-resilient sharing (CLRS)* scheme between two devices. The scheme has the following syntax:

ShareGen$(1^\lambda, \mathbf{m}) \to (\mathsf{sh}_1, \mathsf{sh}_2)$ **:** The share generation algorithm takes as input the security parameter λ and a secret message $\mathbf{m}$. It outputs two *shares*, sh_1 and sh_2 respectively.

Update$_b(\mathsf{sh}_b) \to \mathsf{sh}'_b$ **:** The *randomized* update algorithm takes in the index b and the current version of the share sh_b and outputs an updated version sh'_b. We use the notation $\mathsf{Update}^i_b(\mathsf{sh}_b)$ to denote the operation of updating the share sh_b successively i times in a row.

Reconstruct$(\mathsf{sh}_1, \mathsf{sh}_2) \to \mathbf{m}$ **:** The reconstruction algorithm takes in some version of secret shares $\mathsf{sh}_1, \mathsf{sh}_2$ and it outputs the secret message $\mathbf{m}$.

Correctness. We say that the scheme is *correct* if for any shares $(\mathsf{sh}_1, \mathsf{sh}_2) \leftarrow \mathsf{ShareGen}(1^\lambda, \mathbf{m})$ and any sequence of $i \geq 0, j \geq 0$ updates resulting in $\mathsf{sh}_1' \leftarrow \mathsf{Update}_1^i(\mathsf{sh}_1)$, $\mathsf{sh}_2' \leftarrow \mathsf{Update}_2^j(\mathsf{sh}_2)$, we get $\mathsf{Reconstruct}(\mathsf{sh}_1', \mathsf{sh}_2') = \mathbf{m}$. Note that i and j are arbitrary, and are not required to be equal.

Security. We define ℓ-CLR security as an interactive game between an attacker $\mathcal{A}$ and a challenger. The attacker chooses two messages: $\mathbf{m}_0, \mathbf{m}_1 \in \{0,1\}^*$ with $|\mathbf{m}_0| = |\mathbf{m}_1|$. The challenger chooses a bit $b \leftarrow \{0,1\}$ at random, runs $(\mathsf{sh}_1, \mathsf{sh}_2) \leftarrow \mathsf{ShareGen}(1^\lambda, \mathbf{m}_b)$. The challenger also chooses randomness $rand_1, rand_2$ for the next update of the shares 1,2 respectively and sets $\mathsf{state}_1 := (\mathsf{sh}_1, rand_1), \mathsf{state}_2 := (\mathsf{sh}_2, rand_2)$. It also initializes the counters $L_1 := 0, L_2 := 0$. The attacker $\mathcal{A}$ can adaptively make any number of the following types of queries to the challenger in any order of its choosing:

Leakage Queries: The attacker specifies an index $i \in \{1,2\}$ and a query to the leakage oracle $\mathcal{O}_{\mathsf{state}_i}$. If $L_i < \ell$ it gets the corresponding response and the challenger sets $L_i := L_i + 1$.

Update Queries: The attacker specifies an index $i \in \{1,2\}$. The challenger parses $\mathsf{state}_i = (\mathsf{sh}_i, rand_i)$ and computes updates $\mathsf{sh}_i' := \mathsf{Update}_i(\mathsf{sh}_i; rand_i)$ using randomness $rand_i$. It samples fresh randomness $rand_i'$ and sets $\mathsf{state}_i := (\mathsf{sh}_i', rand_i')$, $L_i := 0$.

At any point in the game, the attacker $\mathcal{A}$ can output a guess $\tilde{b} \in \{0,1\}$. We say that $\mathcal{A}$ *wins* if its guess matches the choice of the challenger $\tilde{b} = b$. We say that an ℓ-CLRS scheme is *secure* if for any PPT attacker $\mathcal{A}$ running in the above game, we have $|\Pr[\mathcal{A} \text{ wins }] - \frac{1}{2}| \leq negl(\lambda)$.

Remarks on the Definition. The inclusion of the update randomness $rand_i$ in the state of the device models leakage during the update process itself when this randomness is used. In fact, the definition just assumes (without loss of generality) that the attacker always leaks only during the update processes when the most information is available on the device. Note that we do not need to explicitly include the next share $\mathsf{sh}_i' = \mathsf{Update}_i(\mathsf{sh}_i; rand_i)$ in the state since it is already efficiently computable from sh_i and $rand_i$.

The given definition also already implies semantic security even if one of the shares is revealed *fully* at the end of the game (but no leakage on the other share is allowed afterwards). To see this, assume that at some point in the game, there is a distinguishing strategy D that uses a fully revealed share sh_i to break security. Then we could also just leak the single predicate $D(\mathsf{sh}_i)$ to break security.

7.1.2 CLRS-Friendly Encryption

We consider an approach of instantiating CLRS via a *public key encryption* scheme $(\mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ having the usual syntax. Given any such encryption scheme, we can define a sharing scheme where the two shares are the secret key $\mathsf{sh}_1 = sk$ and the ciphertext $\mathsf{sh}_2 = ct$ respectively. Formally, we define:

$\mathsf{ShareGen}(1^\lambda; \mathbf{m})$: Sample $(pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$, $ct \leftarrow \mathsf{Encrypt}_{pk}(\mathbf{m})$. Output $\mathsf{sh}_1 := sk, \mathsf{sh}_2 := ct$.

$\mathsf{Reconstruct}(\mathsf{sh}_1, \mathsf{sh}_2)$: Parse $\mathsf{sh}_1 = sk, \mathsf{sh}_2 = ct$. Output $\mathbf{m} = \mathsf{Decrypt}_{sk}(ct)$.

We say that an encryption scheme is *updatable* if it comes with two additional (non-standard) procedures $sk' \leftarrow \mathsf{SKUpdate}(sk)$, $ct' \leftarrow \mathsf{CTUpdate}(ct)$ for updating the secret keys and ciphertexts respectively. These procedures can naturally be used to define updates for the corresponding sharing via $\mathsf{Update}_1(\mathsf{sh}_1) := \mathsf{SKUpdate}(sk)$, $\mathsf{Update}_2(\mathsf{sh}_2) := \mathsf{CTUpdate}(ct)$, where $\mathsf{sh}_1 = sk, \mathsf{sh}_2 = ct$. The above gives us a natural syntactical transformation from an updatable encryption scheme to a corresponding CLRS scheme. We say that an updatable encryption scheme is an *ℓ-CLRS-Friendly Encryption* if:

- The corresponding CLRS scheme satisfies *correctness*.
- The corresponding CLRS scheme satisfies a *strengthening* of *ℓ-CLRS security* where the attacker is first given the public key pk and then adaptively chooses the messages $\mathbf{m}_1, \mathbf{m}_2$.
  (*i.e. The challenger first chooses* $(pk, \mathsf{sh}_1 = sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ *and gives pk to the attacker, who chooses* $\mathbf{m}_1, \mathbf{m}_2$. *The challenger then generates* $\mathsf{sh}_2 \leftarrow \mathsf{Encrypt}_{pk}(\mathbf{m}_b)$ *and the game continues as before.*)

Remarks. We note that the additional functionality provided by CLRS-friendly encryption on top of a plain CLRS may be useful even in the context of sharing a secret between leaky devices. For example, we can imagine a system where one (continually leaky) *master device* stores a secret key share and we publish the corresponding public key. Then other devices can enter the system in an ad-hoc manner by just encrypting their data individually under the public key to establish a shared value with the master device (i.e. no communication is necessary to establish the sharing). The same secret-key share on the master device can be *reused* to share many different messages with many different devices.

7.2 Construction

We now describe our construction of CLRS going through CLRS-Friendly Encryption. In fact, the encryption scheme is going to be remarkably similar to our construction of CLR PKE from the previous chapter, with an identical key generation procedure and key update procedure. The main difference is that we now need our ciphertexts to also be updatable. To accomplish this, a ciphertext in our new scheme will consist of n independently generated ciphertexts of the same message under the original scheme. For ciphertext updates, we notice that our original scheme is already *homomorphic* – to update the n component ciphertexts we will create n fresh component ciphertexts, each of which is a random linear combination of the previous ones using coefficients that sum up to 1. Therefore, each new component ciphertext encrypts the same message as the previous ones. We describe the scheme formally below. As before, we first describe the simplest scheme secure under the SXDH assumption ($k = 1$ linear) and give a formal proof of security. Later we generalize this to the k-linear assumption in bilinear groups, for an arbitrary integer k, and sketch the proof.

Let m, n, d be integer parameters with $n \geq d$. The scheme is defined as follows.

KeyGen$(1^\lambda) \to (pk, sk)$ **:** Sample $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathbf{g}, \mathbf{h}, q) \leftarrow \mathcal{G}(1^\lambda)$ to be the description of a bilinear group of primer order q, with an efficient pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and generators $\mathbf{g} \in \mathbb{G}_1, \mathbf{h} \in \mathbb{G}_2$.

Choose $\vec{p}, \vec{w} \in \mathbb{F}_q^m$ at random subject to $\langle \vec{p}, \vec{w} \rangle = 0$ and set

$$\mathsf{prms} = ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathbf{g}, \mathbf{h}, q), \mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}})$$

to be the *public parameters* of the system. These parameters can then be re-used to create the public/secret keys of all future users. For convenience, we *implicitly* think of prms as a part of each public key pk and as an input to all of the other algorithms.

Choose $\vec{t} \xleftarrow{\$} \mathbb{F}_q^m$ and set $pk := e(\ \mathbf{g}^{\vec{p}}\ ,\ \mathbf{h}^{\vec{t}^\top}\ ) = e(\mathbf{g}, \mathbf{h})^\alpha$ where $\alpha = \langle \vec{p}, \vec{t} \rangle$. Choose $\vec{r} = (r_1, \ldots, r_n) \xleftarrow{\$} \mathbb{F}_q^n$ and set $sk := \mathbf{h}^S$, where S is the $n \times m$ matrix given by

$$S \ := \ \begin{bmatrix} r_1\vec{w} + \vec{t} \\ \cdots \\ r_n\vec{w} + \vec{t} \end{bmatrix} \ = \ \begin{bmatrix} \\ \vec{r}^\top \\ \\ \end{bmatrix} \begin{bmatrix} \quad \vec{w} \quad \end{bmatrix} + \begin{bmatrix} \\ \vec{1}^\top \\ \\ \end{bmatrix} \begin{bmatrix} \quad \vec{t} \quad \end{bmatrix}.$$

In other words, each row of S is chosen at random from the 1-dimensional affine subspace $\vec{t} + \mathsf{span}(\vec{w})$. (Note that $\mathbf{h}^S$ can be computed from the components $\mathbf{h}^{\vec{w}}$, $\vec{t}$, $\vec{r}$ without knowing $\vec{w}$.)

Encrypt$_{pk}(\mathbf{m}) \to ct$ **:** To encrypt $\mathbf{m} \in \mathbb{G}_T$ under $pk = \mathbf{f} = e(\mathbf{g}, \mathbf{h})^\alpha$, choose $\vec{u} = (u_1, \ldots, u_n) \xleftarrow{\$} \mathbb{F}_q^n$ and output $ct = (ct^{(1)}, ct^{(2)})$ where:

$$ct^{(1)} = \begin{bmatrix} \mathbf{g}^{u_1\vec{p}} \\ \dots \\ \mathbf{g}^{u_n\vec{p}} \end{bmatrix} \quad , \quad ct^{(2)} = \begin{bmatrix} \mathbf{f}^{u_1} \cdot \mathbf{m} \\ \dots \\ \mathbf{f}^{u_n} \cdot \mathbf{m} \end{bmatrix}$$

Each row is an independent encryption of the (same) message $\mathbf{m}$ using the basic *encryption* scheme from the previous chapter. Equivalently, we can write the ciphertext as $ct^{(1)} = \mathbf{g}^C$, $ct^{(2)} = e(\mathbf{g}, \mathbf{h})^{\vec{z}^\top}$ for:

$$\begin{aligned} C &= \begin{bmatrix} \\ \vec{u}^\top \\ \\ \end{bmatrix} \begin{bmatrix} & \vec{p} & \end{bmatrix} \\ \vec{z}^\top &= \begin{bmatrix} \\ \vec{u}^\top \\ \\ \end{bmatrix} \alpha + \begin{bmatrix} \\ \vec{1}^\top \\ \\ \end{bmatrix} \mu \; = \; \begin{bmatrix} \\ & C & \\ \\ \end{bmatrix} \begin{bmatrix} \\ t^\top \\ \\ \end{bmatrix} + \begin{bmatrix} \\ \vec{1}^\top \\ \\ \end{bmatrix} \mu \end{aligned}$$

where μ is given by $\mathbf{m} = e(\mathbf{g}, \mathbf{h})^\mu$ and $\alpha = \langle \vec{p}, \vec{t} \rangle$.

Decrypt$_{sk}(ct) \rightarrow$ **m:** To decrypt, we only need to look at the first rows of the secret key and the ciphertext matrices. Given the first row $\mathbf{h}^{\vec{s}}$ of the secret key $sk = \mathbf{h}^S$, the first row $\mathbf{g}^{\vec{c}}$ of the ciphertext component $ct^{(1)} = \mathbf{g}^C$, and the first scalar component $e(\mathbf{g}, \mathbf{h})^z$ of $ct^{(2)} = e(\mathbf{g}, \mathbf{h})^{\vec{z}^\top}$, the decryption algorithm outputs: $\mathbf{m} = e(\mathbf{g}, \mathbf{h})^z / e(\ \mathbf{g}^{\vec{c}}\ ,\ \mathbf{h}^{\vec{s}^\top}\ )$.

SKUpdate$(sk) \rightarrow sk'$ **:** Choose a random matrix $A' \xleftarrow{\$} \mathsf{Rk}_d(\mathbb{F}_q^{n \times n})$. Derive A by "rescaling" each row of A' so that its components sum up to 1. That is, set $(A)_{i,j} := (A')_{i,j} / (\sum_{k=1}^n (A')_{i,k})$, so that $A\vec{1}^\top = \vec{1}^\top$.

If the current secret key is $sk = \mathbf{h}^S$, output the updated key $sk' := \mathbf{h}^{AS}$.

CTUpdate$(ct) \rightarrow ct'$ **:** Choose a random matrix $B' \xleftarrow{\$} \mathsf{Rk}_d(\mathbb{F}_q^{n \times n})$. Derive B by "rescaling" each row of B' so that its components sum up to 1. That is, set $(B)_{i,j} := (B')_{i,j} / (\sum_{k=1}^n (B')_{i,k})$, so $B\vec{1}^\top = \vec{1}^\top$.

If the current ciphertext is $ct = (\mathbf{g}^C, e(\mathbf{g}, \mathbf{h})^{\vec{z}^\top})$, output the updated ciphertext $ct' := (\mathbf{g}^{BC}, e(\mathbf{g}, \mathbf{h})^{B\vec{z}^\top})$.

Theorem 7.2.1. *For any integers $m \geq 6, n \geq 3m - 6, d := n - m + 3$ the above scheme is an ℓ-CLRS-friendly encryption scheme under the SXDH assumption for $\ell = \min(m/6 - 1, n - 3m + 6)\log(q) - \omega(\log(\lambda))$.*

In the above theorem, the absolute leakage (ℓ) scales linearly as $\min(m, n-3m)$ or $\log(q)$ grow. The *ratio* of leakage to share size is $\ell/(nm\log(q))$, and is maximized at $m = 7, n = 16$ to roughly $1/672$.

Corollary 7.2.2. *For any polynomial $\ell = \ell(\lambda)$, there exist ℓ-CLRS schemes under the SXDH assumption. Furthermore, ℓ is a* constant fraction *of the share size.*

Correctness. Let $(\mathsf{prms}, pk, sk) \leftarrow \mathsf{KeyGen}(1^\lambda)$ and let $ct = (ct^{(1)}, ct^{(2)}) \leftarrow \mathsf{Encrypt}_{pk}(\mathbf{m})$. Then we can write $sk = \mathbf{g}^S$, $ct^{(1)} = \mathbf{h}^C, ct^{(2)} = e(\mathbf{g},\mathbf{h})^{\vec{z}}$ for some values $S, C, \vec{z}$ and $W, \vec{t}$ satisfying:

$$S = W + \vec{1}^\top \vec{t} \ , \quad \vec{z}^\top = C\vec{t}^\top + \vec{1}^\top \mu \quad : \quad \mathsf{rowspan}(W) \perp \mathsf{rowspan}(C) \tag{7.1}$$

with μ given by $\mathbf{m} = e(\mathbf{g},\mathbf{h})^\mu$.

First, we show that for any sk and ct satisfying equation (7.1), we get $\mathsf{Decrypt}_{sk}(ct) = \mathbf{m}$. This is because decryption looks at the first row of $sk, ct^{(1)}, ct^{(2)}$ respectively, which are of the form $\mathbf{h}^{\vec{s}}, \mathbf{g}^{\vec{c}}, e(\mathbf{g},\mathbf{h})^z$ where $\vec{s} = \vec{w} + \vec{t}, z = \langle \vec{c}, \vec{t} \rangle + \mu$ for some vectors $\vec{w}, \vec{c}, \vec{t}$ with $\langle \vec{c}, \vec{w} \rangle = 0$. Therefore decryption correctly recovers:

$$e(\mathbf{g},\mathbf{h})^z / e(\ \mathbf{g}^{\vec{c}}\ ,\ \mathbf{h}^{s^\top}\ ) = e(\mathbf{g},\mathbf{h})^{\langle \vec{c}, \vec{t} \rangle + \mu} / e(\mathbf{g},\mathbf{h})^{\langle \vec{c}, \vec{w} + \vec{t} \rangle} = e(\mathbf{g},\mathbf{h})^\mu = \mathbf{m}$$

Next we show, that updates preserve the key/ciphertext structure of equation (7.1). Assume that we update the secret key with the matrices $A_1, A_2, \ldots, A_i$ and the ciphertext with the matrices $B_1, B_2, \ldots, B_j$. Define $\bar{A} = A_i A_{i-1} \cdots A_1$, $\bar{B} = B_j B_{j-1} \cdots B_1$. Since the update matrices are "rescaled" we know that $\bar{A}\vec{1}^\top = \bar{B}\vec{1}^\top = \vec{1}^\top$. Therefore we can write the updated values as $sk_i = \mathbf{g}^{\bar{A}S}$, $ct_j^{(1)} = \mathbf{h}^{\bar{B}C}, ct_j^{(2)} = e(\mathbf{g},\mathbf{h})^{\bar{B}\vec{z}^\top}$ satisfying:

$$(\bar{A}S) = (\bar{A}W) + \vec{1}^\top \vec{t} \ , \ (\bar{B}\vec{z}^\top) = (\bar{B}C)\vec{t}^\top + \vec{1}^\top \mu \ : \ \mathsf{rowspan}(\bar{A}W) \perp \mathsf{rowspan}(\bar{B}C).$$

So equation (7.1) is satisfied by the updated keys and ciphertexts and we get $\mathsf{Decrypt}_{sk_i}(ct_j) = \mathbf{m}$.

7.3 Proof of Security

Our proof of security will follow by a hybrid argument and use many of the components of the proof from the previous chapter. First, we prove a basic information theoretic lemma which will be one of the key ingredients of our analysis.

7.3.1 Hiding Orthogonality

The following lemma can be interpreted as saying that if two random vectors X, Y leak individually, and the total amount of leakage Z is sufficiently small, an attacker cannot distinguish whether X, Y are random orthogonal vectors or uniformly random vectors. A slightly different but related lemma with a related proof strategy appears in [DDV10].

Lemma 7.3.1 (Orthogonality Hiding)**.** *Let $X, Y, Z = \mathsf{Leak}(X, Y)$ be correlated r.v. where X, Y have their support in $\mathbb{F}_q^m$ and are independent conditioned on Z. Let E be the event that $\langle X, Y\rangle = 0$ and let $(X', Y', Z' = \mathsf{Leak}(X', Y')) := (X, Y, Z \mid E)$ be their joint distribution conditioned on the event E. Then $Z \overset{\text{stat}}{\approx} Z'$ as long as $\widetilde{\mathbf{H}}_\infty(X \mid Z) + \widetilde{\mathbf{H}}_\infty(Y \mid Z) - (m + 3)\log(q) = \omega(\log(\lambda))$.*

Proof. We rely on the fact that the inner product is a good *two source extractor* (Lemma 4.1.4). In particular, let ϵ' be the bound from Lemma 4.1.4 such that $(Z, \langle X, Y\rangle) \overset{\text{stat}}{\approx}_{\epsilon'} (Z, U)$ where U is uniform over $\mathbb{F}_q$. Assume there is a statistical test D such that $\Pr[D(Z') = 1] - \Pr[D(Z) = 1] = \epsilon$. Then we claim that there is a statistical test D' that distinguishes $(Z, \langle X, Y\rangle)$ from (Z, U) with advantage $\epsilon/q - \epsilon'$. This implies that $\epsilon/q - \epsilon' \le \epsilon' \Rightarrow \epsilon \le 2q\epsilon'$ and, using the bound on ϵ' from Lemma 4.1.4, our lemma follows. Therefore we are left to describe and analyze the statistical test D'.

The test $D'(\cdot, \cdot)$ just outputs 0 if the second component is non-zero and otherwise outputs the evaluation of D on the first component. This gives

$$\begin{aligned}
\Pr[D'(Z, \langle X, Y\rangle) = 1] &= \Pr[D'(Z, \langle X, Y\rangle) = 1 \mid \langle X, Y\rangle = 0]\Pr[\langle X, Y\rangle = 0] \\
&\ge \Pr[D(Z') = 1](1/q - \epsilon') \\
\Pr[D'(Z, U) = 1] &= \Pr[D'(Z, U) = 1 \mid U = 0]\Pr[U = 0] \\
&= \Pr[D(Z) = 1]/q
\end{aligned}$$

and so D' has the claimed advantage $\epsilon/q - \epsilon'$. □

7.3.2 Alternate Distributions for Keys, Ciphertexts and Updates

Assume that the vectors $\vec{p}, \vec{w}, \vec{t}$ are fixed, defining the public values $\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}}, pk = e(\mathbf{g}, \mathbf{h})^{\langle \vec{p}, \vec{t}\rangle}$. Fix $\vec{w}_1 := \vec{w}$, and let $(\vec{w}_1, \ldots, \vec{w}_{m-1})$ be some *basis* of $(\vec{p})^\perp$ and $(\vec{c}_1, \ldots, \vec{c}_{m-1})$ be some *basis* of $(\vec{w})^\perp$. We define various distributions of keys and ciphertexts relative to these bases.

Key Distributions. Firstly, we define the same additional key distributions as those used in the previous chapter to analyze CLR PKE. That is, the secret key is always set to $\mathbf{h}^S$ for some $n \times m$ matrix S of the form

$$S = \begin{bmatrix} | & & | \\ \vec{r}_1^\top & \cdots & \vec{r}_i^\top \\ | & & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ & \cdots & \\ - & \vec{w}_i & - \end{bmatrix} + \begin{bmatrix} \\ \vec{1}^\top \\ \\ \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix} \tag{7.2}$$

where $\vec{r}_1, \ldots, \vec{r}_i \in \mathbb{F}_q^n$ are chosen randomly. Equivalently, each of the n rows of S is chosen randomly from the affine space: $\mathsf{span}(\vec{w}_1, \ldots, \vec{w}_i) + \vec{t}$. The honest key

keys → ↓ ciphertexts	honest	mid	high
honest	Yes	Yes	Yes
low	Yes	If Correlated or Super-Correlated	No
mid	Yes	If Super-Correlated	No
high	Yes	No	No

Figure 7.1: Do alternate keys correctly decrypt alternate ciphertexts?

generation algorithm uses $i = 1$ and we call these ***honest keys***. In addition, we define ***mid keys*** which are chosen with $i = 2$ and ***high keys*** which are chosen with $i = m - 1$. Notice that honest/mid/high keys all correctly decrypt honestly generated ciphertexts since $\mathsf{span}(\vec{w}_1, \ldots, \vec{w}_{m-1}) \perp \mathsf{span}(\vec{p})$.

Ciphertext Distributions. The encryption of the message $\mathbf{m} = e(\mathbf{g}, \mathbf{h})^{\mu} \in \mathbb{G}_T$ is always set to $ct = (ct^{(1)}, ct^{(2)})$ where $ct^{(1)} = \mathbf{g}^C$ and $ct^{(2)} = e(\mathbf{g}, \mathbf{h})^{\vec{z}^\top}$ with $\vec{z}^\top = C\vec{t}^\top + \vec{1}^\top \mu$. The second component $ct^{(2)}$ can always be efficiently and deterministically computed from $\mathbf{g}^C$, given $\vec{t}$ and $\mathbf{m}$, without knowing the exponents C, μ. The different ciphertext distributions only differ in how $ct^{(1)} = \mathbf{g}^C$ is chosen.

For the ***honest ciphertexts***, we set $C = \vec{u}^\top \vec{p}$ for a uniformly random $\vec{u} \in \mathbb{F}_q^n$. That is, every row of C is chosen at random from the space $\mathsf{span}(\vec{p})$. In addition to the honest way of choosing C, we define three *additional* distributions on C given by:

$$C = \begin{bmatrix} | & & | \\ \vec{u}_1^\top & \cdots & \vec{u}_j^\top \\ | & & | \end{bmatrix} \begin{bmatrix} - & \vec{c}_1 & - \\ & \cdots & \\ - & \vec{c}_j & - \end{bmatrix} \tag{7.3}$$

where $\vec{u}_1, \ldots, \vec{u}_j \in \mathbb{F}_q^n$ are chosen randomly. Equivalently the rows of the C are chosen randomly from the subspace: $\mathsf{span}(\vec{c}_1, \ldots, \vec{c}_j)$. When $j = 1$, we call these ***low ciphertexts***, when $j = 2$ we call these ***mid ciphertexts*** and when $j = (m - 1)$, we call these ***high ciphertexts***. Notice that honest/low/mid/high ciphertexts are all correctly decrypted by *honest secret keys* since $\mathsf{span}(\vec{w}) \perp \mathsf{span}(\vec{c}_1, \ldots, \vec{c}_{m-1})$.

Bases Correlations. By default, we choose the bassis $(\vec{w}_1, \ldots, \vec{w}_{m-1})$ of the space $(\vec{p})^\perp$ and the basis $(\vec{c}_1, \ldots, \vec{c}_{m-1})$ of the space $(\vec{w})^\perp$ uniformly at random and independently subject to fixing $\vec{w}_1 := \vec{w}$. This is statistically close to choosing $\vec{w}_2, \ldots, \vec{w}_{m-1} \overset{\$}{\leftarrow} (\vec{p})^\perp$ and $\vec{c}_1, \ldots, \vec{c}_{m-1} \overset{\$}{\leftarrow} (\vec{w})^\perp$ (Lemma 4.2.1, part II). We call this choice of bases ***uncorrelated***. We will also consider two alternate distributions. We say that the bases are ***correlated*** if we instead choose $\vec{c}_1 \overset{\$}{\leftarrow} (\vec{w}_1, \vec{w}_2)^\perp$ and all other vectors as before. We say that the bases are ***super-correlated*** if we instead choose $\vec{c}_1, \vec{c}_2 \overset{\$}{\leftarrow} (\vec{w}_1, \vec{w}_2)^\perp$ and all other vectors as before. If the key and ciphertext bases are correlated then mid keys correctly decrypt low ciphertexts and if they are super-correlated then mid keys correctly decrypt low and mid ciphertexts. The table in Figure 7.1 summarizes which types of secret keys can correctly decrypt which types of ciphertexts.

Programmed Updates. We will rely on the same terminology and definitions of programming updates as we did in the case of CLR PKE. Please review these definitions from the previous chapter (in particular, see Section 6.5 and Section 6.6).

7.3.3 Overview of Hybrid Games

We now present a series of hybrid game definitions and transitions between them. The goal, as in the proof of CLR PKE security, is to move from the original security game, in which every key/ciphertext pair ever created decrypts to the correct shared message, to a game where none of the pairs decrypt correctly.

Hybrid Games. A helpful pictorial representation of the main hybrid games appears in Figure 7.2. Our hybrid games, called *Game* (i, j) are all of the following type. For $i \geq 2$, the challenger chooses the initial key sk_1 as a high key, the first $i - 2$ updates are honest (and hence the keys $sk_1, \ldots, sk_{i-1}$ are high keys), the update A_{i-1} is programmed to reduce the key to a mid key sk_i, and the update A_i is programmed to reduce the key to a honest key sk_{i+1}. The rest of the updates are honest and hence the keys $sk_{i+1}, sk_{i+2}, \ldots$ are honest. For the special case $i = 1$, the initial key sk_1 already starts out as a mid key and the first update A_1 reduces it to an honest key. For the special case $i = 0$, the initial key sk_1 is already an honest key. This description is mirrored by the ciphertexts. When $j \geq 2$, the initial ciphertext ct_1 is a high ciphertext, the first $j - 2$ ciphertext updates are honest (and hence the ciphertexts $ct_1, \ldots, ct_{j-1}$ are high), the update B_{j-1} is programmed to reduce the ciphertext to a mid ct_j, and the update B_j is programmed to reduce the ciphertext to a *low* ciphertext ct_{j+1}. The rest of the updates are honest and hence the other ciphertexts stay *low*. For the special case $j = 1$, the initial ciphertext ct_1 is already mid and the first update B_1 reduces it to low. For the special case $j = 0$, the initial ciphertext ct_1 is already low.

We write *Game* i as short for *Game* $(i, j = 0)$. In *Game* (i, j) the ciphertext and key bases are *un*correlated. We also define analogous games: *GameCor* (i, j) where the bases are *correlated* and *GameSuperCor* (i, j) where the bases are *super-correlated.*

Sequence of Steps. A helpful table describing the sequence of hybrid arguments appears in Figure 7.3. Our first step is to move from *Real Game* to *Game* 0 (i.e. $i = 0$, $j = 0$). The only difference in this step is that we change the initial ciphertext ct_1 from an *honest ciphertext* to a *low ciphertext.* This is a computational step and all secret keys still correctly decrypt all ciphertexts in the game.

Next, our goal is to keep moving from *Game* i to *Game* $i+1$. We call this the *outer loop* where we increment i. Unfortunately, we *cannot* just increment i in a single step since each such move changes sk_{i+1} from an honest key to a mid key and hence changes it from decrypting all of the low ciphertexts in the game to decrypting none of them. A single computational or leakage step cannot suffice.

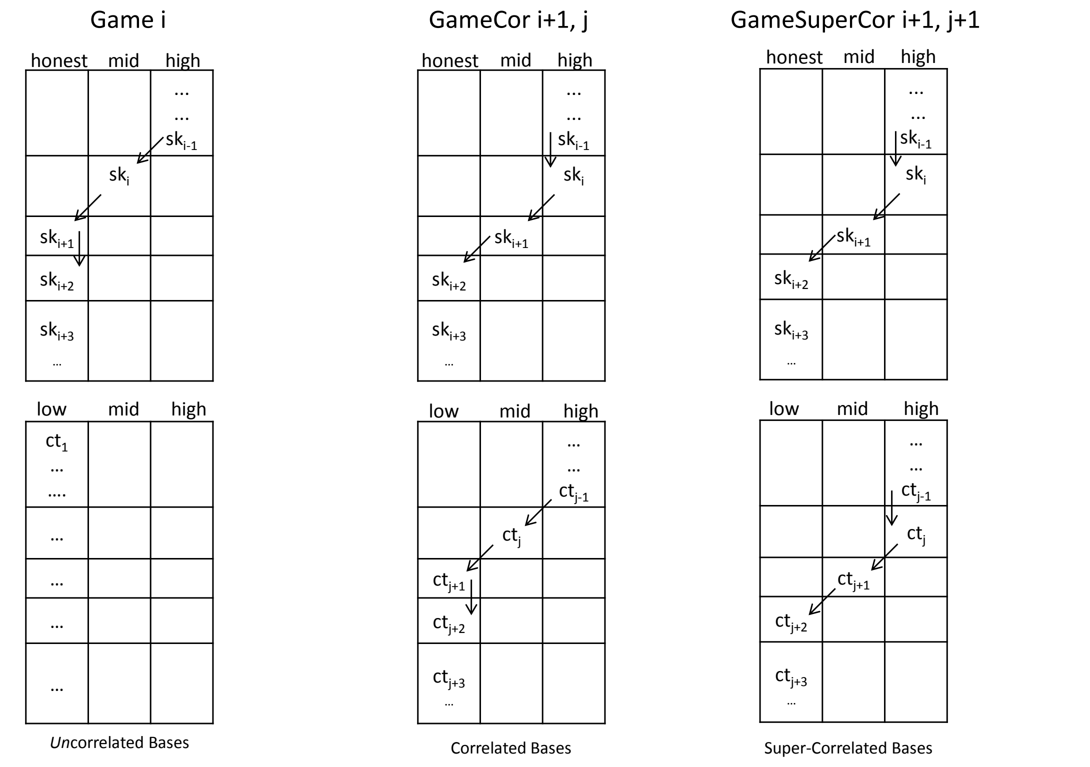
Instead, we *can* move from *Game* i to *GameCor* $i+1$ in a single computational step. Even though the key sk_{i+1} changes from an honest key in *Game* i to a mid key in *GameCor* $i+1$, by making the bases correlated we ensure that it still correctly decrypts all of the low ciphertexts in the game. Therefore, these games cannot be distinguished even given full leakage.

(*Now, we might be tempted to make an information theoretic step that moves us from* GameCor $i+1$ *to* Game $i+1$, *by arguing that a leakage-bounded attacker cannot tell if the key/ciphertext bases are correlated. Indeed, this would correspond to the proof of security for CLR PKE rom the previous chapter. As before, we can still argue that the leakage on the secret-key basis vector* $\vec{w}_2$ *is bounded overall, as this vector only occurs in the single mid key* sk_{i+1}. *Unfortunately, the leakage on the ciphertext basis vector* $\vec{c}_1$ *is* not *bounded overall as it occurs in every single ciphertext, and so a computationally unbounded attacker can learn* $\mathsf{span}(\vec{c}_1)$ *in full and test if* $\vec{w}_2 \perp \vec{c}_1$. *Therefore, we will require more intermediate steps to make this tanssition.)*

To move from *GameCor* $i+1$ to *Game* $i+1$, we first introduce an *inner loop* in which we slowly increment j. Starting with $j=0$, we move from *GameCor* $(i+1, j)$ to *GameSuperCor* $(i+1, j+1)$. This is a single computational step. Even though we change ct_{j+1} from a low ciphertext to a mid ciphertext, it is still correctly decrypted (only) by the mid and low keys in periods $i+1$ and later, since the bases are super-correlated. Therefore, these games cannot be distinguished even given full leakage. Finally, we use an information theoretic step to move from *GameSuperCor* $(i+1, j+1)$ to *GameCor* $(i+1, j+1)$. Here we are actually changing whether a single key sk_{i+1} correctly decrypts a single ciphertext ct_{j+1} (it does in *GameSuperCor* but not in *GameCor*). We use the fact that the adversary is leakage-bounded to argue that it cannot notice whether the bases are *correlated* or *super-correlated*. In particular, because the bases vectors $\vec{w}_2$ and $\vec{c}_2$ only occur in the single mid key sk_{i+1} and the single mid ciphertext ct_{j+1} respectively, the leakage on these vectors is bounded overall. We argue that such partial leakage *hides* whether $\vec{w}_2 \perp \vec{c}_2$, which determines if the bases are correlated or super-correlated.

Assume that the attacker makes at most q_{ct} update queries on the ciphertext and at most q_{sk} update queries on the secret key. By repeatedly increasing j in the inner loop, we move from *GameCor* $(i+1, 0)$ to *GameCor* $(i+1, q_{ct}+1)$ where *all* of the ciphertexts that the attacker can leak on are *high* ciphertexts. Therefore the mid key sk_{i+1} does not decrypt any of them correctly (but all future honest keys still do). We now apply another computational step to move from *GameCor* $(i+1, q_{ct}+1)$ to *Game* $i+1$ and therefore (finally) incrementing i in the outer loop. This step preserves all interactions between keys and ciphertexts and therefore these games cannot be distinguished even given full leakage. Lastly, by repeatedly increasing i in the outer loop, we can move from *Game* 0 to *Game* $q_{sk}+1$ where all of the secret keys that the attacker can leak on are *high keys* and all of the ciphertexts are *low ciphertexts*. Therefore, in *Game* $q_{sk}+1$ *none* of the keys correctly decrypts *any* of the ciphertexts. Hence we can argue that even an attacker that has full leakage in *Game* $q_{sk}+1$ cannot learn any information about the shared/encrypted message $\mathbf{m}$.

Under the Rug. The above discussion is slightly oversimplified. The main issue is that the computational transitions, e.g. from *Game* i to *GameCor* $i+1$, are not computationally indistinguishable the way we defined the games. This is because in *Game* i the update matrix A_{i+1} is unlikely to annihilate any vectors (i.e. its kernel is unlikely to contain non-zero vectors from the span of the previous updates) while in *GameCor* $i+1$ it is programmed to annihilate vectors so as to reduce the dimension of the key. This can be efficiently tested given full leakage of the update matrices. Therefore, in the full proof, we define the games *Game*, *GameCor* and *GameSuperCor* slightly differently with some updates programmed to annihilate additional uniformly random vectors. With this modification, we can prove computational indistinguishability. We also need extra information theoretic steps to argue that the attacker cannot tell if updates are programmed to annihilate some random vectors, given limited leakage.



The arrows ↓ indicate *programmed updates.*

Figure 7.2: An Overview of the Main Hybrid Games

Real $\overset{\text{comp}}{\approx}$ Game 0.

For $i \in \{0, \ldots, q_{sk}\}$:

Game i $\overset{\text{comp}}{\approx}$ GameCor $(i+1, 0)$.

For $j \in \{0, \ldots, q_{ct}\}$: GameCor $(i+1, j)$ $\overset{\text{comp}}{\approx}$ GameSuperCor $(i+1, j+1)$

$\overset{\text{stat}}{\approx}$ GameCor $(i+1, j+1)$.

GameCor $(i+1, q_{ct}+1)$ $\overset{\text{comp}}{\approx}$ Game $i+1$.

Game $q_{sk}+1$ $\overset{\text{comp}}{\approx}$ GameFinal.

Figure 7.3: Sequence of Hybrid Arguments Showing $Real \overset{\text{comp}}{\approx} GameFinal$.

7.3.4 Formal Hybrid Game Definitions

We first define several hybrid games. The main part of the proof is to show computational/statistical indistinguishability between these games. The output of each game consists of the view of the attacker $\mathcal{A}$ as well as a bit b chosen by the challenger representing its choice of which message $\mathbf{m}_0, \mathbf{m}_1$ to encrypt. We assume that the attacker $\mathcal{A}$ makes a maximum of q_{sk} update queries on the secret-key share, and at most q_{ct} update queries on the ciphertext share during the course of the game. We describe the games in detail below, and also give a helpful pictorial representation in Figure 7.2.

Real Game : This is the original "ℓ-CLRS-Friendly Encryption" security game between the adversary and the challenger (see Section 7.1.2). The output of the game consists of the view of the attacker $\mathcal{A}$ and the bit b chosen by the challenger.

Game' 0 : In this game, the challenger chooses the initial ciphertext incorrectly as a *low* ciphertext of the message $\mathbf{m}_b$ instead of encrypting it honestly (the key/ciphertext bases are chosen as random *un*correlated bases). The initial secret key is chosen honestly and all ciphertext/key updates are chosen honestly as before.

Game i : We define Game i for $i = 0, \ldots, q_{sk}+1$. In all future game definitions, we will include two additional "dummy keys" sk_{-1}, sk_0 and key-update matrices A_{-1}, A_0 chosen by the challenger (but not observed or leaked on by the attacker). That is, the challenger (in its head) always initially chooses sk_{-1}, then updates it to sk_0 using an update matrix A_{-1}, then updates sk_0 to sk_1 using an update matrix A_0 and so on. However, the values $sk_{-1}, A_{-1}, sk_0, A_0$ are then ignored, and the first key and update matrix that the attacker can ask leakage queries on are sk_1 and A_1 respectively. In every Game i, the initial key sk_{-1} is chosen as a *high key* with the exponent matrix:

$$S = \begin{bmatrix} | & \cdots & | \\ \vec{r}_1^{\top} & \cdots & \vec{r}_{m-1}^{\top} \\ | & \cdots & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ & \cdots & \\ - & \vec{w}_{m-1} & - \end{bmatrix} + \begin{bmatrix} \\ \vec{1}^{\top} \\ \\ \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

The first key update matrices $A_{-1}, \ldots, A_{i-2}$ are chosen honestly. The update A_{i-1} is *programmed to annihilate* the $m-3$ vectors $(\vec{r}_3, \ldots, \vec{r}_{m-1})$ reducing the key to a *mid*. The update A_i is programmed to annihilate $\vec{r}_2$ along with $m-4$ random vectors, reducing the key to an *honest* key. The update A_{i+1} is programmed to annihilate a single random vector. All other key updates are chosen honestly. Note that, in Game i, the keys $sk_{-1}, \ldots, sk_{i-1}$ are high keys, sk_i is a mid key, and

$sk_{i+1}, \ldots$ are honest keys.[1] The initial ciphertext ct_1 is chosen as in Game' 0 (a low ciphertext) and all ciphertext updates are performed honestly. The ciphertext and key bases are random and *un*correlated. Notice that the secret keys $sk_{-1}, \ldots, sk_i$ do *not* decrypt the low ciphertexts correctly, but all future keys $sk_{i+1}, \ldots, sk_{q_{sk}+1}$ do.

GameCor' $(i+1, 0)$ **:** We define GameCor' $(i+1, 0)$ for $i = 0, \ldots, q_{sk}+1$. In this game, the initial secret key sk_{-1} is a *high* key and the initial ciphertext ct_1 is a *low* ciphertext as in *Game i*. However, the ciphertext and key bases are now *correlated.* Also, the regime of key updates is modified from *Game i*. The first updates $A_{-1}, \ldots, A_{i-2}$ are chosen honestly. The update A_{i-1} is programmed to annihilate $m-3$ random vectors, the update A_i is programmed to annihilate the vectors $(\vec{r}_3, \ldots, \vec{r}_{m-1})$ reducing the key to a *mid*, the update A_{i+1} is programmed to annihilate $\vec{r}_2$ along with $m-4$ random vectors reducing the key to *honest.* All future key updates are then chosen honestly. All ciphertext updates are also chosen honestly (as in Game i). Note that, in GameCor' $(i+1, 0)$, the keys $sk_{-1}, \ldots, sk_i$ are high keys and do not decrypt the low ciphertexts correctly, sk_{i+1} is a mid key but does decrypt the low ciphertexts since the bases are correlated, and $sk_{i+2}, \ldots$ are honest keys which always decrypt correctly.

GameCor $(i+1, j)$ **:** We define GameCor' $(i+1, j)$ for $i = 0, \ldots, q_{sk}+1$ and $j = 0, \ldots, q_{ct}+1$. As in GameCor' $(i+1, 0)$, the key and ciphertext bases are *correlated.* Also, the initial secret key sk_{-1} and the regime of key updates is chosen the same way as in GameCor' $(i+1, 0)$. In all future game definitions, we will include two additional "dummy ciphertexts" ct_{-1}, ct_0 and ciphertext-update matrices B_{-1}, B_0 chosen by the challenger (but not observed or leaked on by the attacker). That is, the challenger initially chooses ct_{-1}, then updates it to ct_0 using an update matrix B_{-1}, then updates that to ct_1 using an update matrix B_0 and so on. However, the values $ct_{-1}, B_{-1}, ct_0, B_0$ are then ignored, and the first ciphertext and ciphertext-update matrix that the attacker can leak on are ct_1 and B_1 respectively. The initial ciphertext ct_{-1} is chosen as a *high ciphertext* using the exponent matrix

$$C = \begin{bmatrix} | & & | \\ \vec{u}_1^\top & \cdots & \vec{u}_{m-1}^\top \\ | & & | \end{bmatrix} \begin{bmatrix} - & \vec{c}_1 & - \\ & \cdots & \\ - & \vec{c}_{m-1} & - \end{bmatrix}$$

[1]In particular, in *Games 0* the initial observed (non-dummy) key sk_1 is an honest key, in *Game* 1 it is a mid key, and in all future games it is a high key. For the proof, it becomes easier to just pretend that there are some "dummy" mid and high keys sk_{-1}, sk_0 even in Games 0,1 so as not to have to define special corner cases for these games.

The first ciphertext updates $B_{-1}, \ldots, B_{j-2}$ are chosen honestly, the update B_{j-1} is programmed to annihilate the $m-3$ vectors $(\vec{u}_3, \ldots, \vec{u}_{m-1})$ reducing the ciphertext to a*mid*, the update B_j is programmed to annihilate $\vec{u}_2$ along with $m-4$ other random vectors reducing the ciphertext to a *low*, the update matrix B_{j+1} is programmed to annihilate a single random vector. All future ciphertext updates are chosen honestly. Note that, in GameCor $(i+1, j)$, the initial ciphertexts $ct_{-1}, \ldots, ct_{j-1}$ are high, ct_j is mid, and $ct_{j+1}, \ldots$ are low.[2]

GameSuperCor $(i+1, j+1)$ **:** We define GameSuperCor $(i+1, j+1)$ for $i = 0, \ldots, q_{sk}+1$ and $j = 0, \ldots, q_{ct}+1$. In this game, the initial secret-key sk_{-1} is a *high key* and the initial ciphertext ct_{-1} is a high ciphertext as in GameCor $(i+1, j)$. However, the key and ciphertext bases are now *super-correlated*. Also, the regime of ciphertext updates is modified from GameCor $(i+1, j)$. The first updates $B_{-1}, \ldots, B_{j-2}$ are chosen honestly, the update B_{j-1} is programmed to annihilate $m-3$ uniformly random vectors, the update matrix B_j is programmed to annihilate the vectors $(\vec{u}_3, \ldots, \vec{u}_{m-1})$ reducing the ciphertext to a *mid*, the update B_{j+1} is programmed to annihilate $\vec{u}_2$ reducing the ciphertext to a *low*. All future ciphertext updates are then chosen honestly. The regime of key updates is the same way as in GameCor $(i+1, j)$. Note that the ciphertexts $ct_{-1}, \ldots, ct_j$ are high, ct_{j+1} is mid, and $ct_{j+2}, \ldots$ are low.

GameFinal : This game is defined the same way as *Game* $q_{sk}+1$, except that instead of using the message $\mathbf{m}_b$ in the (low) ciphertext, we just use message $1_{\mathbb{G}_T}$. More specifically, in *GameFinal*, the secret key is a random *high key* and all of the key updates are honest. The ciphertext is a random *low ciphertext* of the message $1_{\mathbb{G}_T}$ and all of the ciphertext updates are honest as well. In particular, the view of the attacker in GameFinal is independent of the challenger's bit b.

7.3.5 Hybrid Indistinguishability Arguments

In Figure 7.3, we show the sequence of hybrid arguments that is used to derive the indistinguishability:
Real $\overset{\text{comp}}{\approx}$ *GameFinal*. In this section, we prove each of the necessary sub-steps in separate lemmas.

Lemma 7.3.2. *Real* $\overset{\text{comp}}{\approx}$ *Game' 0.*

[2]In particular, in *GamesCor* $i+1, 0$ the initial observed (non-dummy) ciphertext ct_1 is low, in *GameCor* $i+1, 1$ it is mid, and in all future games it is high. For the proof, it becomes easier to just pretend that there are some "dummy" mid and high ciphertexts ct_{-1}, ct_0 even when $j = 0, 1$ so as not to have to define special corner cases for these games.

Proof. We show a reduction from the $(k = 1)$-*extended rank hiding assumption* (Definition 6.2.5, Lemma 6.2.6). The reduction is given a challenge $\mathbf{g}^P$, $\vec{w}$ where $P \xleftarrow{\$} \mathsf{Rk}_x(\mathbb{F}_q^{2\times m})$ is either of rank $x = 1$ or $x = 2$ and $\vec{w} \xleftarrow{\$} \mathsf{ker}(P)$. Let us denote the two rows of P by $\vec{p}, \vec{c}_1$ respectively. The reduction puts the values $\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}}$ in prms and chooses its own random $\vec{t}$ to set up the initial public/secret key pk, sk. To create the encryption of $\mathbf{m}_b$, the reduction sets $ct = (\mathbf{g}^C, e(\mathbf{g}, \mathbf{h})^{\vec{z}})$ where it uses the challenge $\mathbf{g}^{\vec{c}_1}$ to compute $\mathbf{g}^C$ for $C = \vec{u}^\top \vec{c}_1$ where $\vec{u} \xleftarrow{\$} \mathbb{F}_q^n$ is random. The matching second component $e(\mathbf{g}, \mathbf{h})^{\vec{z}}$ can then be computed efficiently as a deterministic function of $\mathbf{g}^C, \vec{t}$ and the message $\mathbf{m}_b$. The reduction chooses all of the key/ciphertext update matrices honestly and answers all update/leakage queries honestly.

If the challenge has $x = 1$ then $\vec{c}_1 = u'\vec{p}$ for some scalar u' and hence the ciphertext is a correctly distributed *honest* encryption of $\mathbf{m}_b$, so the reduction produces the distribution of the *Real Game.* If $x = 2$ then $\vec{c}_1$ is a random and independent vector in the space $(\vec{w})^\perp$ and hence the ciphertext is a correctly distributed *low* encryption of $\mathbf{m}_b$, so the reduction produces the distribution of *Game' 0.* Therefore the two are computationally indistinguishable. □

Lemma 7.3.3. *If* $\ell \leq (n - 3m + 6)\log(q) - \omega(\log(\lambda))$ *then* Game' *0* $\overset{\text{stat}}{\approx}$ Game *0.*

Proof. The only difference between *Game' 0* and *Game 0* is the joint distribution on the initial secret key sk_1 and the update matrix A_1. Conditioned on sk_1, A_1, all other values are sampled the same way in the two games. In *Game' 0*, the key sk_1 is a randomly chosen *honest key*, and A_1 is an *honest update.* In *Game 0* the distribution on sk_1, A_1 is slightly more complicated. First we choose a random *high key* $sk_{-1} = \mathbf{h}^{S-1}$ with exponent

$$S_{-1} = \begin{bmatrix} | & \cdots & | \\ \vec{r}_1^\top & \cdots & \vec{r}_{m-1}^\top \\ | & \cdots & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ & \cdots & \\ - & \vec{w}_{m-1} & - \end{bmatrix} + \begin{bmatrix} \vec{1}^\top \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

Then we choose the updates: A_{-1} programmed to annihilate $\vec{r}_3, \ldots, \vec{r}_{m-1}$ yielding a mid-key sk_0, A_0 is programmed to annihilate $\vec{r}_2$ along with $m-4$ random vectors yielding an honest key sk_1, and A_1 is programmed to annihilate a single random vector.

We claim this joint distribution on A_1 and sk_1 in *Game 0* is statistically indistinguishable from that of *Game' 0.* To see this, we first apply the *reprogramming lemma* (Lemma 6.6.2 with $u^* = m - 3, d = n - m + 3, l = m - 1$) to switch from A_1 being programmed to annihilate a single vector to just being honest. Next we just use *program correctness lemma* (Lemma 6.6.1; with $l = m - 1$, $\rho = 1$, $u = m - 4$, $d = n - m + 3$) to argue that the distribution of the key sk_1 in *Game 0* is statistically close to choosing a fresh honest key. In particular, the lemma tells us that $A_0 A_{-1}[\vec{r}_1^\top | \cdots | \vec{r}_{m-1}^\top] \overset{\text{stat}}{\approx} [\vec{r}^\top | \mathbf{0} \cdots]$ for a uniformly random $\vec{r} \in \mathbb{F}_q^n$. So the exponent of the key sk_1 in *Game 0* is

$$A_0 A_{-1} S_{-1} = A_0 A_{-1} \vec{r}_1^\top \vec{w}_1 + \vec{1}^\top \vec{t} \quad \overset{\text{stat}}{\approx} \quad \vec{r}^\top \vec{w} + \vec{1}^\top \vec{t}$$

and hence statistically indistinguishable from that of a random honest key as in *Game' 0.*

□

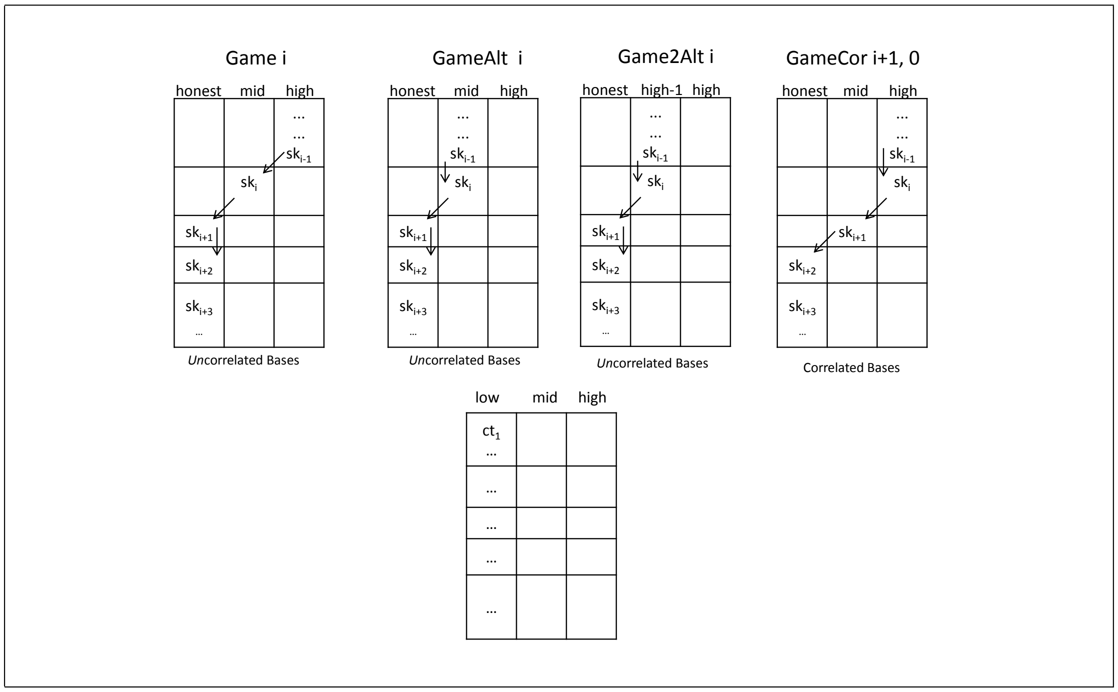


Figure 7.4: Intermediate Hybrids for Lemma 7.3.4.

Lemma 7.3.4. *For $i \in \{0, \ldots, q_{sk}\}$: Game i $\overset{\text{comp}}{\approx}$ GameCor' $i+1$.*

Proof. For the proof of the lemma, we introduce two additional intermediate games (see Figure 7.4). Firstly, we define *GameAlt i*. In this game, the initial secret key sk_{-1} is a random *mid key* with exponent matrix

$$S \overset{\text{def}}{=} \begin{bmatrix} | & | \\ \vec{r}_1^\top & \vec{r}_2^\top \\ | & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ - & \vec{w}_2 & - \end{bmatrix} + \begin{bmatrix} \\ \vec{1}^\top \\ \\ \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

The key update matrices $A_1, \ldots, A_{i-2}$ are chosen honestly, the update A_{i-1} is programmed to annihilate $m-3$ random vectors, the update A_i is programmed to annihilate $\vec{r}_2$ along with $m-4$ random vectors, and the update matrix A_{i+1} is programmed to annihilate a single random vector. The initial ciphertext and ciphertext update matrices are chosen as in *Game i*.

Claim 7.3.5. *Game i* $\overset{\text{comp}}{\approx}$ *GameAlt i*

Proof. We show a reduction from the $(k = 1)$-*extended rank hiding assumption* (Definition 6.2.5, Lemma 6.2.6). The challenger gets a challenge $\mathbf{h}^{W'}$ and $\vec{p}$, where $W' \xleftarrow{\$} \mathsf{Rk}_x(\mathbb{F}_q^{(m-2)\times m})$ is either of rank $x = 1$ or of rank $x = m - 2$ and $\vec{p} \in \mathsf{ker}(W')$. Let us label the rows of W' by $\vec{w}_2, \ldots, \vec{w}_{m-1}$. Choose $\vec{w}_1 \xleftarrow{\$} (\vec{p})^\perp$ and $\vec{t} \xleftarrow{\$} \mathbb{F}_q^m$. Set $\mathsf{prms} = (\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}_1})$, $pk = e(\mathbf{g}, \mathbf{h})^{\vec{t}\cdot\vec{p}}$ and $sk_{-1} = \mathbf{h}^S$ where

$$S = \begin{bmatrix} | & & | \\ \vec{r}_1^{\top} & \cdots & \vec{r}_{m-1}^{\top} \\ | & & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ & \cdots & \\ - & \vec{w}_{m-1} & - \end{bmatrix} + \begin{bmatrix} \vec{1}^{\top} \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

for uniformly random $\vec{r}_1, \ldots, \vec{r}_{m-1}$ (note: the challenger can do this efficiently given $\mathbf{h}^{W'}$ without knowing W'). Create a "low" ciphertext by choosing a random vector $\vec{c} \xleftarrow{\$} (\vec{w}_1)^\perp$. Run the rest of *Game i* correctly as a challenger, where the key-update A_{i-1} is programmed to annihilate $\vec{r}_3, \ldots, \vec{r}_{m-1}$, the key-update A_i is programmed to annihilate $\vec{r}_2$ and $m - 4$ random vectors and the key update A_{i+1} update is programmed to annihilate a single random vector.

If W' is of rank $x = m - 2$, then it is easy to see that the above distribution is that of *Game i*.

If W' is of rank $x = 1$ then we claim that the above is distribution is that of *GameAlt i*. Firstly, we can write $\vec{w}_3 = \mu_3\vec{w}_2, \ldots, \vec{w}_{m-1} = \mu_{m-1}\vec{w}_2$ for some scalars $\mu_3, \ldots, \mu_{m-1}$ in $\mathbb{F}_q$. Letting $\vec{\mu} = (1, \mu_3, \ldots, \mu_{m-1}) \in \mathbb{F}_q^{m-2}$, we can write

$$S = \begin{bmatrix} | & | \\ \vec{r}_1^{\top} & \vec{r}_2\,'^{\top} \\ | & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ - & \vec{w}_2 & - \end{bmatrix} + \begin{bmatrix} \vec{1}^{\top} \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

for

$$\begin{bmatrix} \vec{r}_2\,'^{\top} \end{bmatrix} = \begin{bmatrix} | & & | \\ \vec{r}_2^{\top} & \cdots & \vec{r}_{m-1}^{\top} \\ | & & | \end{bmatrix} \begin{bmatrix} \vec{\mu}^{\top} \end{bmatrix}$$

So sk_{-1} is a random *mid* key as in *GameAlt i*. Moreover, the vectors $\vec{r}_3, \ldots, \vec{r}_{m-1}$ are random and independent of $S, \vec{w}_1, \vec{w}_2, \vec{r}_1, \vec{r}_2\,'$. Therefore, the update matrix A_{i-1} is programmed to annihilate $m - 3$ random and independent vectors $\vec{r}_3, \ldots, \vec{r}_{m-1}$ as in *GameAlt i*. Finally, the update A_i is programmed to annihilates $\vec{r}_2$ (along with $m - 4$ random vectors), which is equivalent to being programmed to annihilate $\vec{r}_2'$ since $\mathsf{span}(D_{i-1}\vec{r}_2\,') = \mathsf{span}(D_{i-1}\vec{r}_2)$ where D_{i-1} is the product of all update matrices prior to A_i. So we see that the, when W' is of rank 1 then the choice of sk_{i-1} and all the update matrices is distributed correctly as in *GameAlt i*. Hence an attacker that distinguishes *Game i* and *GameAlt i* breaks rank-hiding. □

We now introduce a second intermediate game called *Game2Alt i* where the initial secret key sk_{-1} has an exponent of the form

$$S_{-1} \stackrel{\text{def}}{=} \begin{bmatrix} | & \cdots & | \\ \vec{r}_1^{\top} & \cdots & \vec{r}_{m-2}^{\top} \\ | & \cdots & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ & \cdots & \\ - & \vec{w}_{m-2} & - \end{bmatrix} + \begin{bmatrix} \\ \vec{1}^{\top} \\ \\ \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

so that the rows are chosen randomly from the $m-2$ dimensional affine space $\vec{t} + \mathsf{span}(\vec{w}_1, \ldots, \vec{w}_{m-2})$. The key update matrices $A_1, \ldots, A_{i-2}$ are chosen honestly, the update A_{i-1} is programmed to annihilate $m-3$ random vectors, the update A_i is programmed to annihilate the vectors $\vec{r}_2, \ldots, \vec{r}_{m-2}$, and the update matrix A_{i+1} is programmed to annihilate a single random vector. The initial ciphertext and ciphertext update matrices are chosen as in *Game i*.

We now show (in 2 steps) that *Game2Alt i* is computationally indistinguishable from *GameAlt i* and from *GameCor* $(i+1, 0)$, which completes the proof of Lemma 7.3.4

Claim 7.3.6. *GameAlt i* $\overset{\text{comp}}{\approx}$ *Game2Alt i.*

Proof. We show a reduction from the $(k=1)$-*extended rank hiding assumption* (Definition 6.2.5, Lemma 6.2.6). The challenger gets a challenge $\mathbf{h}^{W'}$ and $\vec{p}$, where $W' \xleftarrow{\$} \mathsf{Rk}_x(\mathbb{F}_q^{(m-3)\times m})$ is either rank $x = 1$ or rank $x = (m-3)$ and $\vec{p} \in \mathsf{ker}(W')$. Let us label the rows of W' by $\vec{w}_2, \ldots, \vec{w}_{m-2}$. Choose $\vec{w}_1 \xleftarrow{\$} (\vec{p})^{\perp}$ and $\vec{t} \xleftarrow{\$} \mathbb{F}_q^m$. Set $\mathsf{prms} = (\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}_1})$, $pk = e(\mathbf{g}, \mathbf{h})^{\vec{t}\cdot\vec{p}}$ and $sk_{-1} = \mathbf{h}^S$ where

$$S = \begin{bmatrix} | & & | \\ \vec{r}_1^{\top} & \cdots & \vec{r}_{m-2}^{\top} \\ | & & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ & \cdots & \\ - & \vec{w}_{m-2} & - \end{bmatrix} + \begin{bmatrix} \\ \vec{1}^{\top} \\ \\ \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

for uniformly random $\vec{r}_1, \ldots, \vec{r}_{m-2}$ (note: the challenger can do this efficiently given $\mathbf{h}^{W'}$ without knowing W'). Create a "low" ciphertext by choosing a random vector $\vec{c} \xleftarrow{\$} (\vec{w}_1)^{\perp}$. The key-update A_{i-1} is programmed to annihilate $m-3$ random vectors, the key-update A_i is programmed to annihilate $\vec{r}_2, \ldots, \vec{r}_{m-2}$ and the key update A_{i+1} update is programmed to annihilate a single random vector.

If W' is of rank $x = m-3$, then it is easy to see that the above distribution is that of *Game2Alt i*.

If W' is of rank $x = 1$ then we claim that the above distribution is that of *GameAlt i*. This follows since we can write

$$S = \begin{bmatrix} | & | \\ \vec{r}_1^{\top} & \vec{r}_2\,'^{\top} \\ | & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ - & \vec{w}_2 & - \end{bmatrix} + \begin{bmatrix} \\ \vec{1}^{\top} \\ \\ \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

for

$$\begin{bmatrix} \vec{r}_2{}'^{\top} \end{bmatrix} = \begin{bmatrix} | & & | \\ \vec{r}_2^{\top} & \cdots & \vec{r}_{m-2}^{\top} \\ | & & | \end{bmatrix} \begin{bmatrix} \vec{\mu}^{\top} \end{bmatrix}$$

and some vector $\vec{\mu}^{\top} \in \mathbb{F}_q^{m-3}$. So sk_{-1} is a random *mid key* as in *GameAlt i*. Moreover, an update A_i which is programmed to annihilate $\vec{r}_2, \ldots, \vec{r}_{m-2}$ is equivalent to being programmed to annihilate $\vec{r}_2{}'$ along with $m-4$ random vectors since $\mathsf{span}(\vec{r}_2, \ldots, \vec{r}_{m-2}) = \mathsf{span}(\vec{r}_2{}', \vec{r}_3, \ldots, \vec{r}_{m-2})$ where $\vec{r}_3, \ldots, \vec{r}_{m-2}$ are random an independent of sk_{-1} or any of the previous updates. Therefore sk_{-1} and all of the update matrices are distributed as in *GameAlt i*. □

Claim 7.3.7. *Game2Alt i* $\overset{\text{comp}}{\approx}$ *GameCor' i* $+1$.

Proof. We show a reduction from the $(k=1)$-*extended rank hiding assumption* (Definition 6.2.5, Lemma 6.2.6). The reduction gets a challenge $\mathbf{h}^{W'}$ and $\vec{p}, \vec{c}$, where $W' \xleftarrow{\$} \mathsf{Rk}_x(\mathbb{F}_q^{2\times m})$ is either rank $x=1$ or rank $x=2$ and $\vec{p}, \vec{c} \in \mathsf{ker}(W')$. Let us label the rows of W' by $\vec{w}_1, \vec{w}_2$. The reduction chooses $\vec{w}_3, \ldots, \vec{w}_{m-1} \xleftarrow{\$} (\vec{p})^{\perp}$, $\vec{t} \xleftarrow{\$} \mathbb{F}_q^m$ and sets $\mathsf{prms} = (\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}_1})$, $pk = e(\mathbf{g}, \mathbf{h})^{\vec{t}\cdot\vec{p}}$ and $sk_{-1} = \mathbf{h}^S$ where

$$S = \begin{bmatrix} | & & | \\ \vec{r}_1^{\top} & \cdots & \vec{r}_{m-1}^{\top} \\ | & & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ & \cdots & \\ - & \vec{w}_{m-1} & - \end{bmatrix} + \begin{bmatrix} \vec{1}^{\top} \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

for uniformly random $\vec{r}_1, \ldots, \vec{r}_{m-1}$. (Note: the reduction can do this efficiently given $\mathbf{h}^{W'}$ without knowing W'). Lastly, the reduction creates a "low" ciphertext using the vector $\vec{c}$ that comes from the challenge. The reduction does everything else as in *GameCor' i*$+1$ where the key-update A_{i-1} is programmed to annihilate $m-3$ random vectors, the key-update A_i is programmed to annihilate $\vec{r}_3, \ldots, \vec{r}_{m-1}$ and the key update A_{i+1} update is programmed to annihilate $\vec{r}_2$.

If W' is of rank $x=2$, then it is easy to see that the above distribution is that of *GameCor' i* $+1$. Notice that the key and ciphertext basis are random *correlated* basis with $\vec{c} \in (\vec{w}_1, \vec{w}_2)^{\perp}$.

If W' is of rank $x=1$ then we claim that the above distribution is that of *Gam2Alt i*. This follows since the exponent of sk_{-1} can be written as

$$S = \begin{bmatrix} | & | & & | \\ \vec{r}_1{}'^{\top} & \vec{r}_3^{\top} & \cdots & \vec{r}_{m-1}^{\top} \\ | & | & & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ - & \vec{w}_3 & - \\ & \cdots & \\ - & \vec{w}_{m-1} & - \end{bmatrix} + \begin{bmatrix} \vec{1}^{\top} \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix}$$

where $\vec{r}_1\,'^{\top} = \mu_1\vec{r}_1^{\top} + \mu_2\vec{r}_2^{\top}$ for some scalars μ_1, μ_2. Note that $\vec{r}_1\,'$ is therefore uniformly random and independent of $\vec{r}_2$. By relabeling, this has the same distribution as the initial key in *Game2Alt i*. Moreover, the update A_i is programmed to annihilate the $m-3$ vectors $\vec{r}_3, \ldots, \vec{r}_{m-1}$ in the initial secret key, while the update A_{i+1} annihilates the vector $\vec{r}_2$ which is random and independent of the initial secret key. Therefore the initial secret key and all of the secret key updates are distributed as in *Game2Alt i*. □

Putting Claim 7.3.5, Claim 7.3.6 and Claim 7.3.7 together, we complete the proof of the lemma. □

Lemma 7.3.8. *If* $\ell \leq (n-3m+6)\log(q) - \omega(\log(\lambda))$ *then* GameCor' $i+1 \overset{\text{stat}}{\approx}$ GameCor $(i+1, 0)$.

Proof. The only difference between the two games is the joint distribution on the initial ciphertext ct_1 and ciphertext-update matrix B_1. In *GameCor' i + 1* the ciphertext ct_1 is chosen as a low ciphertext and the update B_1 is honest. In *GameCor* $(i+1, 0)$, we first choose a high ciphertext ct_{-1} with the exponent

$$C_{-1} \overset{\text{def}}{=} \begin{bmatrix} | & \cdots & | \\ \vec{u}_1^{\top} & \cdots & \vec{u}_{m-1}^{\top} \\ | & \cdots & | \end{bmatrix} \begin{bmatrix} - & \vec{c}_1 & - \\ & \cdots & \\ - & \vec{c}_{m-1} & - \end{bmatrix}$$

We then choose updates: B_{-1} programmed to annihilate $(\vec{u}_3, \ldots, \vec{u}_{m-1})$ yielding a mid ciphertext ct_0, B_0 programmed to annihilate $\vec{u}_2$ and $m-4$ random vectors yielding a low ciphertext ct_1, and the update B_1 programmed to annihilate a single random vector. We claim that ct_1, B_1 are distributed the same way in the two games. The proof exactly follows that of Lemma 7.3.3. □

Lemma 7.3.9. *GameCor* $(i+1, j) \overset{\text{comp}}{\approx}$ *GameSuperCor* $(i+1, j+1)$.

Proof. The proof of this lemma is analogous to the proof of Lemma 7.3.4. We introduce two additional intermediate games (see Figure 7.5) called *GameCorAlt* $(i+1, j)$ and *GameCor2Alt* $(i+1, j)$. In all these games the key basis $(\vec{w}_1, \ldots, \vec{w}_{m-1}) \in (\vec{p})^{\perp}$ and the ciphertext basis $(\vec{c}_1, \ldots, \vec{c}_{m-1}) \in (\vec{w}_1)^{\perp}$ are *correlated* with $\vec{w}_2 \in (\vec{c}_1)^{\perp}$.

Firstly, we define *GameCorAlt* $(i+1, j)$ where the initial ciphertext ct_{-1} is a random *mid ciphertext* using exponent matrix

$$C \overset{\text{def}}{=} \begin{bmatrix} | & | \\ \vec{u}_1^{\top} & \vec{u}_2^{\top} \\ | & | \end{bmatrix} \begin{bmatrix} - & \vec{c}_1 & - \\ - & \vec{c}_2 & - \end{bmatrix}$$

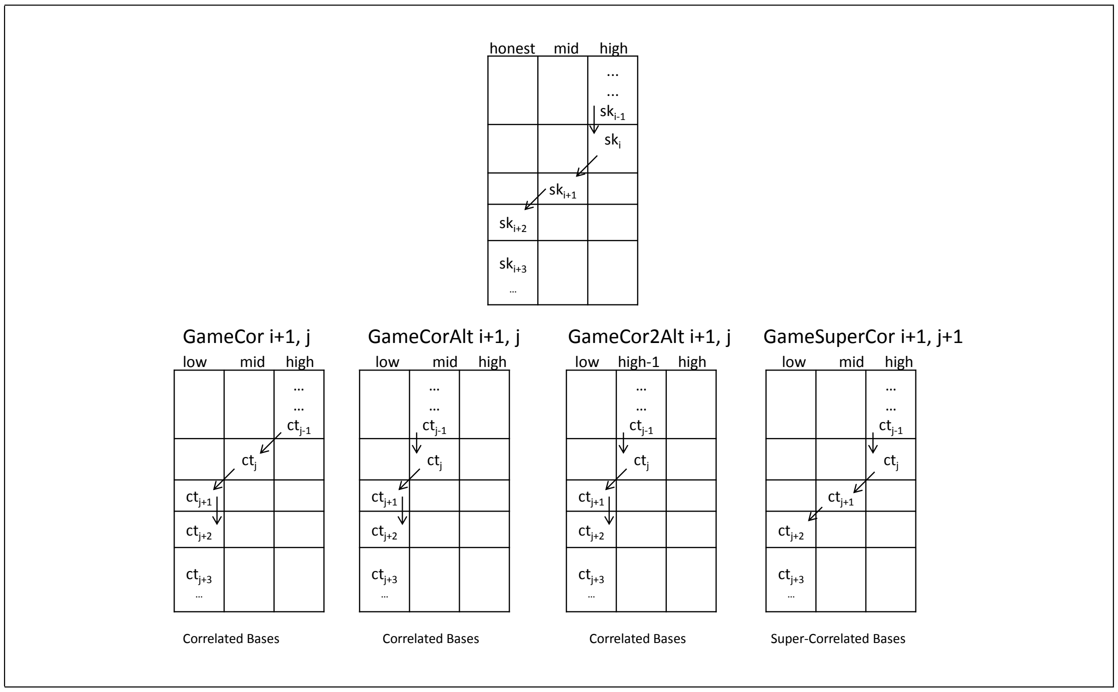


Figure 7.5: Intermediate Hybrids for Lemma 7.3.9.

for uniformly random $\vec{u}_1, \vec{u}_2$. The initial ciphertext-update matrices $\ldots, B_{j-2}$ are chosen honestly, the update B_{j-1} is programmed to annihilate $m-3$ random vectors, the update B_j is programmed to annihilate $\vec{u}_2$ along with $m-4$ random vectors, and the update matrix B_{j+1} is programmed to annihilate a single random vector. The initial secret key and all key update matrices are chosen as in *GameCor* $(i+1, j)$.

Claim 7.3.10. *GameCor* $(i+1, j) \overset{\text{comp}}{\approx}$ *GameCorAlt* $(i+1, j)$.

Proof. We show a reduction from the $(k = 1)$-*extended rank hiding assumption* (Definition 6.2.5, Lemma 6.2.6). The challenge gets a challenge $\mathbf{g}^{C'}$ and $\vec{w}_1$, where $C' \overset{\$}{\leftarrow} \mathsf{Rk}_x(\mathbb{F}_q^{(m-2)\times m})$ is either of rank $x = 1$ or $x = m-2$ and $\vec{w}_1 \in \mathsf{ker}(C')$. Let us label the rows of C' by $\vec{c}_2, \ldots, \vec{c}_{m-1}$. Choose $\vec{c}_1, \vec{p} \overset{\$}{\leftarrow} (\vec{w}_1)^\perp$ and $\vec{t} \overset{\$}{\leftarrow} \mathbb{F}_q^m$. Set $\mathsf{prms} = (\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}_1})$, $pk = e(\mathbf{g}, \mathbf{h})^{\vec{t}\cdot\vec{p}}$. The initial ciphertext $ct_{-1}$is chosen so that the exponent matrix in the first component $\mathbf{g}^C$ is

$$C = \begin{bmatrix} | & & | \\ \vec{u}_1^\top & \cdots & \vec{u}_{m-1}^\top \\ | & & | \end{bmatrix} \begin{bmatrix} - & \vec{c}_1 & - \\ & \cdots & \\ - & \vec{c}_{m-1} & - \end{bmatrix}$$

for uniformly random $\vec{u}_1, \ldots, \vec{u}_{m-1}$ (note: the challenger can do this efficiently given $\mathbf{g}^{C'}$ without knowing C'). The ciphertext-update matrices are chosen as in *GameCor* $(i+1, j)$ where B_{j-1} is programmed to annihilate $\vec{u}_3, \ldots, \vec{u}_{m-1}$, B_j is programmed to annihilate $\vec{u}_2$ and $m-4$ random vectors and B_{j+1} is programmed to annihilate a single random vector.

The initial secret key is chosen by first sampling $\vec{w}_2, \ldots, \vec{w}_{m-1}$ so that, along with $\vec{w}_1$, they form a random *correlated* basis of $(\vec{p})^\perp$, having $\vec{w}_2 \in (\vec{c}_1)^\perp$. This basis is then used to sample the initial *high* secret key sk_{-1}. The secret key updates are chosen as in *GameCor* $(i+1, j)$.

If C' is of rank $x = m-2$, then the above distribution is just that of *GameCor* $(i+1, j)$. On the other hand, if C' is of rank $x = 1$, we argue that the above distribution is that of *GameCorAlt* $(i+1, j)$. The argument mirrors that of Claim 7.3.5.

$\square$

We now introduce a second intermediate game called *GameCor2Alt* $(i+1, j)$ where the initial ciphertext ct_{-1} is chosen so that the exponent in the first component $\mathbf{g}^C$ is

$$C = \begin{bmatrix} | & \cdots & | \\ \vec{u}_1^\top & \cdots & \vec{u}_{m-2}^\top \\ | & \cdots & | \end{bmatrix} \begin{bmatrix} - & \vec{c}_1 & - \\ & \cdots & \\ - & \vec{c}_{m-2} & - \end{bmatrix}$$

for uniformly random $\vec{u}_1, \ldots, \vec{u}_{m-2}$. The initial ciphertext-update matrices $\ldots, B_{j-2}$ are chosen honestly, the update B_{j-1} is programmed to annihilate $m-3$ random vectors, the update B_j is programmed to annihilate the $m-3$ vectors $\vec{u}_2, \ldots, \vec{u}_{m-2}$, and the update matrix B_{j+1} is programmed to annihilate a single random vector. The initial secret key and all key update matrices are chosen as in *GameCor* $(i+1, j)$.

We now show (in 2 steps) that *GameCor2Alt* $(i+1, j)$ is computationally indistinguishable from *GameAlt* $(i+1, j)$ and from *GameSuperCor* $(i+1, j+1)$, which completes the proof of Lemma 7.3.9.

Claim 7.3.11. *GameCorAlt* $(i+1, j) \overset{\text{comp}}{\approx}$ *GameCor2Alt* $(i+1, j)$.

Proof. We show a reduction from the $(k=1)$-*extended rank hiding assumption* (Definition 6.2.5, Lemma 6.2.6). The challenger gets a challenge $\mathbf{g}^{C'}$ and $\vec{w}_1$, where $C' \overset{\$}{\leftarrow} \mathsf{Rk}_x(\mathbb{F}_q^{(m-3 \times m)})$ is either rank $x = 1$ or rank $x = m-3$ and $\vec{w}_1 \in \mathsf{ker}(C')$. Let us label the rows of C' by $\vec{c}_2, \ldots, \vec{c}_{m-2}$. Choose $\vec{c}_1, \vec{p} \overset{\$}{\leftarrow} (\vec{w}_1)^\perp$

and $\vec{t} \xleftarrow{\$} \mathbb{F}_q^m$. Set $\mathsf{prms} = (\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}_1})$, $pk = e(\mathbf{g}, \mathbf{h})^{\vec{t} \cdot \vec{p}}$. The initial ciphertext ct_{-1} is chosen so that the exponent in the first component $\mathbf{g}^C$ is

$$C = \begin{bmatrix} | & & | \\ \vec{u}_1^\top & \cdots & \vec{u}_{m-2}^\top \\ | & & | \end{bmatrix} \begin{bmatrix} - & \vec{c}_1 & - \\ & \cdots & \\ - & \vec{c}_{m-2} & - \end{bmatrix}$$

for random $\vec{u}_1, \ldots, \vec{u}_{m-2} \in \mathbb{F}_q^m$ (note: the challenger can do this efficiently given $\mathbf{g}^{C'}$ without knowing C'). The ciphertext-update B_{j-1} is programmed to annihilate $m-3$ random vectors, B_j is programmed to annihilate $\vec{u}_2, \ldots, \vec{u}_{m-2}$, and B_{j+1} is programmed to annihilate a single random vector.

The initial secret key is chosen by first sampling $\vec{w}_2, \ldots, \vec{w}_{m-1}$ so that, along with $\vec{w}_1$, they form a random *correlated* basis of $(\vec{p})^\perp$, having $\vec{w}_2 \in (\vec{c}_1)^\perp$. This basis is then used to sample the initial *high* secret key sk_{-1}. The secret key updates are chosen as in *GameCor* $(i+1, j)$.

If C' is of rank $x = m-3$, then the above distribution is just that of *GameCor2Alt* $(i+1, j)$. On the other hand, if C' is of rank $x = 1$, we argue that the above distribution is that of *GameCorAlt* $(i+1, j)$. The argument mirrors that of Claim 7.3.6. □

Claim 7.3.12. *GameCor2Alt* $(i+1, j) \overset{\text{comp}}{\approx}$ *GameSuperCor* $(i+1, j+1)$.

Proof. We show a reduction from the $(k=1)$-*extended rank hiding assumption* (Definition 6.2.5, Lemma 6.2.6). The challenger gets a challenge $\mathbf{g}^{C'}$ and $\vec{w}_1, \vec{w}_2$, where $C' \xleftarrow{\$} \mathsf{Rk}_x(\mathbb{F}_q^{2 \times m})$ is either rank $x = 1$ or rank $x = 2$ and $\vec{w}_1, \vec{w}_2 \in \mathsf{ker}(C')$. Let us label the rows of C' by $\vec{c}_1, \vec{c}_2$. Choose $\vec{p}, \vec{c}_3, \ldots, \vec{c}_{m-1} \xleftarrow{\$} (\vec{w}_1)^\perp$ and $\vec{t} \xleftarrow{\$} \mathbb{F}_q^m$. Set $\mathsf{prms} = (\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}_1})$, $pk = e(\mathbf{g}, \mathbf{h})^{\vec{t} \cdot \vec{p}}$. The initial ciphertext ct_{-1} is chosen so that the exponent in the first component $\mathbf{g}^C$ is

$$C = \begin{bmatrix} | & & | \\ \vec{u}_1^\top & \cdots & \vec{u}_{m-1}^\top \\ | & & | \end{bmatrix} \begin{bmatrix} - & \vec{c}_1 & - \\ & \cdots & \\ - & \vec{w}_{c-1} & - \end{bmatrix}$$

for random $\vec{u}_1, \ldots, \vec{u}_{m-2} \in \mathbb{F}_q^m$ (note: the challenger can do this efficiently given $\mathbf{g}^{C'}$ without knowing C'). The ciphertext-update B_{j-1} is programmed to annihilate $m-3$ random vectors, B_j is programmed to annihilate $\vec{u}_3, \ldots, \vec{u}_{m-1}$, and B_{j+1} is programmed to annihilate $\vec{u}_2$.

The initial secret key is chosen by sampling $\vec{w}_3, \ldots, \vec{w}_{m-1}$ so that, along with $\vec{w}_1, \vec{w}_2$, they form a basis of $(\vec{p})^\perp$. This basis is then used to sample the initial *high* secret key sk_{-1}. The secret key updates are chosen as in *GameCor* $(i+1, j)$.

If C' is of rank $x = 2$, then the above distribution is just that of *GameSuperCor* $(i+1, j+1)$. Notice that, in this case, the key and ciphertext bases are super-correlated with $\vec{c}_1, \vec{c}_2 \in (\vec{w}_1, \vec{w}_2)^\perp$. On the other hand, if C' is of rank $x = 1$, we argue that the above distribution is that of *GameCor2Alt* $(i+1, j)$. The argument mirrors that of Claim 7.3.7. □

Putting Claim 7.3.10, Claim 7.3.11 and Claim 7.3.12 together, we complete the proof of the lemma.

□

Lemma 7.3.13. *If* $\ell \leq \min(m/6 - 1, n - 3m + 6)\log(q) - \omega(\log(\lambda))$ *then*

$$\text{GameSuperCor}(i+1, j+1) \overset{\text{stat}}{\approx} \text{GameCor}(i+1, j+1).$$

Proof. There are two differences between *GameSuperCor* and *GameCor*. The first difference lies in wether the ciphertext and key bases are correlated or super-correlated. The second difference lies in the regime of ciphertext updates. We define an intermediate game, *GameCorInter* $(i+1, j+1)$, in which the ciphertext bases are (only) *correlated*, but the regime of ciphertext updates follows that of *GameSuperCor* $(i+1, j+1)$.

Claim 7.3.14. *If* $\ell \leq (m/6 - 1)\log(q) - \omega(\log(\lambda))$, *then*

$$\textit{GameSuperCor}(i+1, j+1) \overset{\text{stat}}{\approx} \textit{GameCorInter}(i+1, j+1).$$

Proof. This follows by the *orthogonality hiding lemma* (Lemma 7.3.1). Notice that, once we fix $\vec{p}, \vec{w}_1$ and $\vec{c}_1$ in *GamrCor'*, we can think of $\vec{w}_2 \overset{\$}{\leftarrow} (\vec{c}_1, \vec{p})^\perp$ and $\vec{c}_2 \overset{\$}{\leftarrow} (\vec{w}_1)^\perp$ as two independent sources. Moreover, we claim that the only leaked-upon values in the two games above that are related to $\vec{w}_2$ are the key updates A_i, A_{i+1} and the secret key sk_{i+1}. Therefore only *three* time periods contain relevant information about $\vec{w}_2$ This is because the initial *high* key sk_{-1} has rows from the full space $(\vec{p})^\perp$ (and hence does not depend on $\vec{w}_2$) and the updates $A_{-1}, \ldots, A_{i-1}$ do not depend on the key at all. (On the other hand the secret key sk_{i+1} has the span of $\vec{w}_1, \vec{w}_2$ in the exponent, and the updates A_i, A_{i+1} annihilate vectors that are correlated to sk_{i+1} and hence also to $\vec{w}_2$). Lastly the keys $sk_{i+2}, \ldots$ are random *low* keys and the update $A_{i+2}, \ldots$ are chosen honestly and hence have no information about $\vec{w}_2$. Therefore, the leakage on $\vec{w}_2$ is bounded by 3ℓ. Similarly, the only leaked-upon values in the two games above that are related to $\vec{c}_2$ are the ciphertext updates B_j, B_{j+1} and the ciphertext ct_{j+1}. Hence the leakage on $\vec{c}_2$ is bounded by 3ℓ as well. Lastly, since the secret key and ciphertext leak independently, if we condition on the leakage

Z observed by the attacker, the distributions of $\vec{c}_2$ and $\vec{w}_2$ are independent. We get $\widetilde{\mathbf{H}}_\infty(\vec{w}_2 \mid Z) \geq (m-2)\log(q) - 3\ell, \widetilde{\mathbf{H}}_\infty(\vec{c}_2 \mid Z) \geq (m-1)\log(q) - 3\ell$. Since *GameSuperCor* is equivalent to *GameCorInter* is we condition on the event $\langle \vec{w}_2, \vec{c}_2 \rangle = 0$, we can apply the orthogonality hiding lemma (Lemma 7.3.1) to bound the statistical distance between these games. In particular, the games *GameSuperCor* and *GameCorInter* are statistically indistinguishable as long as $\ell \leq (m/6 - 1)\log(q) - \omega(\log(\lambda))$. □

We now turn to changing the regime of ciphertext updates from *GameCorInter* to *GameCor*. But for this we just use the *reprogramming lemma* (Lemma 6.6.2) on the ciphertext updates. In particular, we rely on the fact that the regime of ciphertext updates between *GameCorInter* and *GameCor* only differs in which updates annihilate additional random vectors (but *not* in how they annihilate the ciphertext exponent vectors $\vec{u}_i$). Therefore, using the reprogramming lemma, we see that the two games are indistinguishable as long as $\ell \leq (n - 3m + 6)\log(q) - \omega(\log(\lambda))$.

□

Lemma 7.3.15. *If* $\ell \leq (n - 3m + 6)\log(q) - \omega(\log(\lambda))$ *then* GameCor $(i+1, q_{ct}+1) \overset{\text{comp}}{\approx}$ Game $i+1$.

Proof. There are three differences between the above games: (I) in *GameCor* $(i + 1, q_{ct} + 1)$ the ciphertext and key bases are *correlated* ($\langle \vec{c}_2, \vec{w}_1 \rangle = 0$) whereas in *Game* $i+1$ they are *un*correlated, (II) the regime of programmed *key updates* differs between the two games in when and how many random vectors the updates are programmed to annihilate, (III) in *GameCor* $(i+1, q_{ct}+1)$ the initial ciphertext is *high* while in *Game* $i+1$ it is low (all ciphertext updates are honest in both games).

Firstly, we can ignore (I) since both games are completely independent of the choice of the basis vector $\vec{c}_2$ as there are no mid ciphertexts in either game. In particular, the distribution of the initial low/high ciphertext in the two games does not depend at all on whether the bases are correlated or not.

Secondly, we use the reprogramming lemma (Lemma 6.6.2) to change the regime of key updates from that of *GameCor* $(i+1, q_{ct}+1)$ to that of *Game* $i+1$. The total number of random vectors that are programmed to be annihilated in either game is $u = m-3$. Therefore, the change is statistically indistinguishable as long as $\ell \leq (n - 3m + 6)\log(q) - \omega(\log(\lambda))$.

Lastly, we provide a simple reduction from the ($k = 1$)-*extended rank hiding assumption* (Definition 6.2.5, Lemma 6.2.6) to distinguishing whether the initial ciphertext is low or high. The reduction gets a challenge $\mathbf{g}^C$ and $\vec{w}$ where $C \xleftarrow{\$} \mathsf{Rk}_x(\mathbb{F}_q^{n\times m})$ is a random matrix of either rank $x = m-1$ or rank $x = 1$, and $\vec{w} \in \ker(C)$. The reduction samples a random $\vec{t} \xleftarrow{\$} \mathbb{F}_q^n$, $\vec{p} \xleftarrow{\$} (\vec{w})^\perp$ and

$\vec{w}_2, \ldots, \vec{w}_{m-1} \xleftarrow{\$} (\vec{p})^\perp$. It uses these to create the public parameters $\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}}$, public key $pk = e(\mathbf{g}, \mathbf{h})^{\langle \vec{p}, \vec{t} \rangle}$ and the initial high secret key $\mathbf{h}^S$. For the initial ciphertext, it just uses $ct^{(1)} = \mathbf{g}^C$ and creates the matching second component $ct^{(2)}$ efficiently using $ct^{(1)}, \mathbf{m}_b, \vec{t}$. It runs the rest of the game by choosing the key updates as specified in *Game* $i+1$ and all the ciphertext updates honestly. It's easy to see that if the challenge $\mathbf{g}^C$ is of rank $x = m-1$ then this the same as *GameCor* $(i+1, q_{ct}+1)$ with the modified key updates as above, while if the challenge is of rank $x = 1$ then this is just *Game* $i+1$.

□

Lemma 7.3.16. Game $q_{sk}+1 \overset{\text{comp}}{\approx}$ GameFinal.

Proof. We define several hybrid distributions for choosing the initial ciphertext ct. Recall that we can think of ct as consisting of n rows where each row is a ciphertext under the "simple" (un-updatable) encryption scheme. We will consider the following distributions on the ciphertext-rows:

1. Low Encryptions of $\mathbf{m}$: The ciphertext-row is of the form $(\mathbf{g}^{\vec{c}}, e(\mathbf{g}, \mathbf{h})^{\langle \vec{c}, \vec{t} \rangle} \mathbf{m})$ where $\vec{c} = u\vec{c}_1$ for a random $u \xleftarrow{\$} \mathbb{F}_q$ and the ciphertext-basis vector $\vec{c}_1$.
2. Mid Encryptions of $\mathbf{m}$: The ciphertext-row is of the form $(\mathbf{g}^{\vec{c}}, e(\mathbf{g}, \mathbf{h})^{\langle \vec{c}, \vec{t} \rangle} \mathbf{m})$ where $\vec{c} = u_1\vec{c}_1 + u_2\vec{c}_2$ for random $u_1, u_2 \xleftarrow{\$} \mathbb{F}_q$ and the ciphertext-basis vectors $\vec{c}_1, \vec{c}_2$.

In *Game* $q_{sk}+1$, the initial ciphertext ct has all of its n rows chosen as random *low encryptions of* $\mathbf{m}_b$. In *GameFinal*, the initial ciphertext ct has all of its n rows chosen as random *low encryptions of* $1_{\mathbb{G}_T}$.

We define hybrid games *GameHyb* $i = 0, \ldots, n$, where the first i rows of the initial ciphertext ct are chosen as random *low encryptions of* $1_{\mathbb{G}_T}$ and the rest are random *low encryptions of* $\mathbf{m}_b$. Note that *GameHyb* 0 is really just *Game* $q_{sk}+1$ and *GameHyb* n is really just *GameFinal*.

We also define the hybrid game *GameHybMid* $i = 0, \ldots, n-1$ where the first i rows of the initial ciphertext ct are random low encryptions of $1_{\mathbb{G}_T}$, the row $i+1$ is a random *mid* encryption of $\mathbf{m}_b$, and the rest of the rows are random low encryptions of $\mathbf{m}_b$.

In all these games, the ciphertext updates are all honest, the secret key is a high key and the key updates are all honest (as in *Game* $q_{sk}+1$).

Claim 7.3.17. *For* $i = 0, \ldots, n-1$*:* *GameHyb* $i \overset{\text{comp}}{\approx}$ *GameHybMid* i.

Proof. We show a reduction from the $(k=1)$-*extended rank hiding assumption* (Definition 6.2.5, Lemma 6.2.6) to distinguishing whether the $(i+1)$ row is a low or mid encryption of $\mathbf{m}_b$. The reduction gets a challenge $\mathbf{g}^{C'}$ and $\vec{w}$ where

$C' \xleftarrow{\$} \mathsf{Rk}_x(\mathbb{F}_q^{2\times m})$ is a random matrix of either rank $x = 2$ or rank $x = 1$, and $\vec{w} \in \mathsf{ker}(C)$. Let us label the rows of C' by $\vec{c}_1, \vec{c}_2$ The reduction samples a random $\vec{t} \xleftarrow{\$} \mathbb{F}_q^n$, $\vec{p} \xleftarrow{\$} (\vec{w})^\perp$ and $\vec{w}_2, \dots, \vec{w}_{m-1} \xleftarrow{\$} (\vec{p})^\perp$. It uses these to create the public parameters $\mathbf{g}^{\vec{p}}, \mathbf{h}^{\vec{w}}$, public key $e(\mathbf{g}, \mathbf{h})^{\langle \vec{p}, \vec{t}\rangle}$ and the initial high secret key $\mathbf{h}^S$.

For the initial ciphertext, it sets $ct^{(1)} = \mathbf{g}^C$ where $C = \vec{u}^T\vec{c}_1 + \vec{e}^\top\vec{c}_2$ for uniformly random $\vec{u} \in \mathbb{F}_q^n$ and the vector $\vec{e} \in \mathbb{F}_q^n$ being it $i+1$ standard basis vector whose $i+1$ coordinate is 1 and all others are 0. In other words each row j of the matrix C can be written as $u_j\vec{c}_1 + e_j\vec{c}_2$ where u_j are random, $e_{i+1} = 1$ and $e_j = 0$ for $j \neq i+1$. The reduction creates the matching second component $ct^{(2)}$ efficiently using $ct^{(1)}, \mathbf{m}_b, \vec{t}$ as either an encryption of $1_{\mathbb{G}_T}$ (for rows $j \leq i$) or the message $\mathbf{m}_b$ (for rows $j > i$). It runs the rest of the game by choosing the key updates the ciphertext updates honestly. It's easy to see that if the challenge is of rank $x = 1$ then this the same distribution as *GameHyb i* while if the challenge is of rank $x = 2$ then this is the same distribution as *GameHybMid i*. □

Claim 7.3.18. *For $i = 0, \dots, n-1$: GameHybMid i $\overset{\text{stat}}{\approx}$ GameHyb $i+1$.*

Proof. In *GameHybMid i*, the $(i+1)$ row of the initial ciphertext ct is of the form

$$(\mathbf{g}^{u_1\vec{c}_1 + u_2\vec{c}_2}, e(\mathbf{g}, \mathbf{h})^{u_1\beta + u_2\gamma}\mathbf{m}_b)$$

where $\beta = \langle \vec{c}_1, \vec{t}\rangle, \gamma = \langle \vec{c}_2, \vec{t}\rangle$. We claim that γ information theoretically "blinds" the message m. That is, we claim that given everything else in the game other than $e(\mathbf{g}, \mathbf{h})^{u_1\beta+u_2\gamma}\mathbf{m}_b$, the value γ looks uniformly random. This is because the only information that's available in the entire game about $\vec{t}$ is the value $\alpha = \langle \vec{t}, \vec{p}\rangle$ given in the public key and the value β which is revealed by the other rows of the ciphertext ct. Since the secret key is a *high key*, the rows of the secret key are chosen uniformly at random from the space

$$\mathsf{span}(\vec{w}_1, \dots, \vec{w}_{m-1}) + \vec{t} \quad = (\vec{p})^\perp + \vec{t} \quad = \{\vec{s} \in \mathbb{F}_q^m \mid \langle \vec{s}, \vec{p}\rangle = \alpha\}$$

which does not depend on $\vec{t}$ beyond its dependence on α. If $m \geq 3$ then the value $\vec{c}_2$ is linearly independent of $\vec{p}, \vec{c}_1$ (w.o.p.) and hence the value γ is a random and independent of α, β and everything else observed in the game. Therefore, the message $\mathbf{m}_b$ contained in the $(i+1)$st row of the ciphertext is statistically hidden in *GameHybMid i*. So we may as well replace the $(i+1)$ row from containing $\mathbf{m}_b$ to just containing $1_{\mathbb{G}_T}$.

We now repeat the same argument as in the proof of Claim 7.3.17 to change the $(i+1)$ row from being a *mid encryption of* $1_{\mathbb{G}_T}$ to being a *low encryption of* $1_{\mathbb{G}_T}$, which gets us to *GameHyb* $i+1$. □

Combining these hybrids, we get the statement of the lemma. □

7.3.6 Putting It All Together

Using Lemmata 7.3.2 - 7.3.16 in the sequence of hybrids given by Figure 7.3, we get the indistinguishability:
Real $\overset{\text{comp}}{\approx}$ *GameFinal*. Recall that the output of each game includes the view of the attacker $\mathcal{A}$ at the end of the experiment along with the challenger's selection bit b. Since the attacker's guess $\tilde{b}$ at the end of the game can be efficiently computed from the view of the attacker, the predicate ($\mathcal{A}$ wins) $\Leftrightarrow$ ($\tilde{b} \overset{?}{=} b$) can be efficiently computed from the output of each game. In *GameFinal* the view of the attacker is independent of the random bit b and hence we have $\Pr[\mathcal{A} \text{ wins }] = \frac{1}{2}$. Therefore, in the *Real* game, we must have $|\Pr[\mathcal{A} \text{ wins }] - \frac{1}{2}| \le negl(\lambda)$ since the two games are indistinguishable. This concludes the proof of Theorem 7.2.1.

7.4 Generalization to k-Linear

In this section, we provide a generalized scheme which we can prove secure under the k-linear assumption for arbitrary choices of k. When $k = 1$, the scheme description and proof coincide exactly with the original.

7.4.1 Scheme Description

Let k, n, m, d be integer parameters of the system with $n \ge d$.

KeyGen$(1^\lambda) \to (pk, sk)$: Sample $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathbf{g}, \mathbf{h}, q) \leftarrow \mathcal{G}(1^\lambda)$ to be the description of a bilinear group of prime order q, with an efficient pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, and generators $\mathbf{g} \in \mathbb{G}_1, \mathbf{h} \in \mathbb{G}_2$.

Choose matrices $P, W \in \mathbb{F}_q^{k \times m}$ at random subject to $\mathsf{rowspan}(P) \perp \mathsf{rowspan}(W)$ and set

$$\mathsf{prms} = ((\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, \mathbf{g}, \mathbf{h}, q), \mathbf{g}^P, \mathbf{h}^W)$$

to be the *public parameters* of the system.[3]

Choose $\vec{t} \xleftarrow{\$} \mathbb{F}_q^m$ and set $pk := e(\ \mathbf{g}^P\ ,\ \mathbf{h}^{\vec{t}^\top}\ ) = e(\mathbf{g}, \mathbf{h})^{\vec{\alpha}}$ where $\vec{\alpha}^\top = P\vec{t}^\top$. Choose $R \xleftarrow{\$} \mathbb{F}_q^{n \times k}$ and set $sk := \mathbf{h}^S$, where S is the $n \times m$ matrix given by

$$S \ := \ \begin{bmatrix} \\ R \\ \\ \end{bmatrix} \begin{bmatrix} \ W \ \end{bmatrix} + \begin{bmatrix} \\ \vec{1}^\top \\ \\ \end{bmatrix} \begin{bmatrix} \ \vec{t} \ \end{bmatrix}.$$

In other words, each row of S is chosen at random from the affine subspace $\vec{t} + \mathsf{rowspan}(W)$.

[3]We can interpret the above as choosing W at random and choosing each row of P at random from $\mathsf{ker}(W)$.

(Note that $\mathbf{h}^S$ can be computed from the components $\mathbf{h}^W$, $\vec{t}$, R without knowing W.)

Encrypt$_{pk}(\mathbf{m}) \rightarrow ct$ **:** To encrypt $\mathbf{m} \in \mathbb{G}_T$ under $pk = e(\mathbf{g}, \mathbf{h})^{\vec{\alpha}}$, choose $U \xleftarrow{\$} \mathbb{F}_q^{n \times k}$ and label its rows $\vec{u}_1, \ldots, \vec{u}_n$. Output $ct = (ct^{(1)}, ct^{(2)})$ where:

$$ct^{(1)} = \begin{bmatrix} \mathbf{g}^{\vec{u}_1 P} \\ \ldots \\ \mathbf{g}^{\vec{u}_n P} \end{bmatrix} \quad , \quad ct^{(2)} = \begin{bmatrix} e(\mathbf{g}, \mathbf{h})^{\vec{u}_1 \cdot \vec{\alpha}^\top} \cdot \mathbf{m} \\ \ldots \\ e(\mathbf{g}, \mathbf{h})^{\vec{u}_n \cdot \vec{\alpha}^\top} \cdot \mathbf{m} \end{bmatrix}$$

Each row is an independent encryption of the (same) message $\mathbf{m}$ using the *simple encryption* process. Equivalently, we can write the ciphertext as $ct^{(1)} = \mathbf{g}^C$, $ct^{(2)} = e(\mathbf{g}, \mathbf{h})^{\vec{z}^\top}$ for:

$$\begin{aligned} C &= \begin{bmatrix} \\ U \\ \\ \end{bmatrix} \begin{bmatrix} & P & \end{bmatrix} \\ \vec{z}^\top &= \begin{bmatrix} \\ U \\ \\ \end{bmatrix} \vec{\alpha}^\top + \begin{bmatrix} \\ \vec{1}^\top \\ \\ \end{bmatrix} \mu = \begin{bmatrix} \\ & C & \\ \\ \end{bmatrix} \begin{bmatrix} \\ t^\top \\ \\ \end{bmatrix} + \begin{bmatrix} \\ \vec{1}^\top \\ \\ \end{bmatrix} \mu. \end{aligned}$$

where μ is given by $\mathbf{m} = e(\mathbf{g}, \mathbf{h})^\mu$ and $\vec{\alpha}^\top = P\vec{t}^\top$.

Decrypt$_{sk}(ct) \rightarrow \mathbf{m}$**:** To decrypt, we only need to look at the first rows of the secret key and the ciphertext matrices. Given the first row $\mathbf{h}^{\vec{s}}$ of the secret key $sk = \mathbf{h}^S$, the first row $\mathbf{g}^{\vec{c}}$ of the ciphertext component $ct^{(1)} = \mathbf{g}^C$, and the first scalar component $e(\mathbf{g}, \mathbf{h})^z$ of $ct^{(2)} = e(\mathbf{g}, \mathbf{h})^{\vec{z}^\top}$, the decryption algorithm outputs: $\mathbf{m} = e(\mathbf{g}, \mathbf{h})^z / e(\ \mathbf{g}^{\vec{c}}\ ,\ \mathbf{h}^{\vec{s}^\top}\ )$.

SKUpdate$(sk) \rightarrow sk'$ **:** Choose a random matrix $A' \xleftarrow{\$} \mathsf{Rk}_d(\mathbb{F}_q^{n \times n})$. Derive A by "rescaling" each row of A' so that its components sum up to 1. That is, set $(A)_{i,j} := (A')_{i,j} / (\sum_{l=1}^n (A')_{i,l})$.

If the current secret key is $sk = \mathbf{h}^S$, output the updated key $sk' := \mathbf{h}^{AS}$.

CTUpdate$(ct) \rightarrow ct'$ **:** Choose a random matrix $B' \xleftarrow{\$} \mathsf{Rk}_d(\mathbb{F}_q^{n \times n})$. Derive B by "rescaling" each row of B' so that its components sum up to 1. That is, set $(B)_{i,j} := (B')_{i,j} / (\sum_{l=1}^n (B')_{i,l})$.

If the current ciphertext is $ct = (\mathbf{g}^C, e(\mathbf{g}, \mathbf{h})^{\vec{z}})$, output the updated ciphertext $ct' := (\mathbf{g}^{BC}, e(\mathbf{g}, \mathbf{h})^{B\vec{z}})$.

Theorem 7.4.1. *For any integers $m \geq 7k, n \geq 3m - 7k + 1$ and $d := n - m + 3k$ the scheme* (KeyGen, $\mathsf{Encrypt}$, $\mathsf{Decrypt}$, $\mathsf{SKUpdate}$, $\mathsf{CTUpdate}$) *is an ℓ-CLRS-Friendly Encryption scheme under the k-linear assumption for any* $\ell \leq \min\left((m - 5k - 1)/6, n - 3m + 7k - 1\right)\log(q) - \omega(\log(\lambda))$.

The argument for correctness is the same as for the original scheme in Section 7.2 and we sketch the modifications needed to make the proof of security go through below.

7.4.2 The Generalized Proof (Sketch)

We now give an overview of the modifications to the previous proof necessary to generalize it to the k-linear assumption. The overall structure of the proof and the hybrid games is exactly the same, except that we modify how low/mid/high keys and ciphertexts are defined.

Alternate Key and Ciphertext Distributions. Assume the matrices P, W and the vector $\vec{t}$ are fixed defining $\mathbf{g}^P, \mathbf{h}^W$ and $pk = e(\mathbf{g}, \mathbf{h})^{P\vec{t}^\top}$. Let us label the rows of W by $\vec{w}_1, \ldots, \vec{w}_k$ and let $(\vec{w}_1, \ldots, \vec{w}_{(m-k)})$ be a basis of $\mathsf{ker}(P) = \mathsf{rowspan}(P)^\perp$ and let $(\vec{c}_1, \ldots, \vec{c}_{(m-k)})$ be a basis of $\mathsf{ker}(W) = \mathsf{rowspan}(W)^\perp$. We define the various key distributions on $sk = \mathbf{h}^S$ the same way as before with

$$S = \begin{bmatrix} | & & | \\ \vec{r}_1^\top & \cdots & \vec{r}_i^\top \\ | & & | \end{bmatrix} \begin{bmatrix} - & \vec{w}_1 & - \\ & \cdots & \\ - & \vec{w}_i & - \end{bmatrix} + \begin{bmatrix} \\ \vec{1}^\top \\ \\ \end{bmatrix} \begin{bmatrix} & \vec{t} & \end{bmatrix} \tag{7.4}$$

but now we have $i = k$ for ***honest***, $i = 2k$ for ***mid*** and $i = (m - k)$ for ***high*** keys. Similarly, we define the low/mid/high ciphertext distributions the same as before with $ct^{(1)} = \mathbf{g}^C$ where

$$C = \begin{bmatrix} | & & | \\ \vec{u}_1^\top & \cdots & \vec{u}_j^\top \\ | & & | \end{bmatrix} \begin{bmatrix} - & \vec{c}_1 & - \\ & \cdots & \\ - & \vec{c}_j & - \end{bmatrix} \tag{7.5}$$

but now we have $j = k$ for ***low***, $j = 2k$ for ***mid*** and $j = (m - k)$ for ***high*** ciphertexts.

By default, we choose the ciphertext and keys basses ***uncorrelated*** with $\vec{w}_{k+1}, \ldots, \vec{w}_{m-k} \xleftarrow{\$} \mathsf{ker}(P)$ and $\vec{c}_1, \ldots, \vec{c}_{m-k} \xleftarrow{\$} \mathsf{ker}(W)$. We say that the bases are ***correlated*** if we instead choose $\vec{c}_1, \ldots, \vec{c}_k \xleftarrow{\$} (\vec{w}_1, \ldots \vec{w}_{2k})^\perp$ and all other vectors as before. We say that the bases are ***super-correlated*** if we instead choose $\vec{c}_1, \ldots, \vec{c}_{2k} \xleftarrow{\$} (\vec{w}_1, \ldots \vec{w}_{2k})^\perp$ and all other vectors as before. The table in Section 7.3.2, Figure 7.1 still accurately summarizes interactions between keys and ciphertexts.

The Hybrids. We also define the various games *Game i*, *GameCor (i,j)*, *Game-SuperCor (i,j)* analogously as before with appropriate modifications. For example, in *Game i*, the key updates $\ldots, A_{i-2}$ are honest, the update A_i (high to mid) is now programmed to annihilate the $m-3k$ vectors $\vec{r}_{2k+1}, \ldots, \vec{r}_{m-k}$, the update A_{i+1} (mid to low) is programmed to annihilate the k vectors $\vec{r}_{k+1}, \ldots, \vec{r}_{2k}$ along with $m-4k$ random vectors, and the update A_{i+1} is now programmed to annihilate k random vectors. The other game definitions are all analogous.

All of the computational steps are performed analogously, but now under the *k-extended rank hiding assumption* which follows from k-linear. The information theoretic steps in Lemma 7.3.3, Lemma 7.3.8, are also analogous and the new bound becomes: $\ell \leq n - 3m + 7k - 1$.

The only aspect that's non-trivial becomes the information theoretic argument where we change from *super-correlated* bases to just *correlated* bases (the analogue of Claim 7.3.14). We want to switch the condition

$$(\vec{w}_{k+1}, \ldots, \vec{w}_{2k}) \perp (\vec{c}_{k+1}, \ldots, \vec{c}_{2k})$$

from being true to being false. We do this by using an information theoretic argument on each pair $\vec{w}_i, \vec{c}_j$ separately (in any order). Whenever we do so, we think of all other basis components as fixed. That is $\vec{w}_i$ *always* comes from the space orthogonal to $\mathsf{rowspan}(P||\vec{c}_1|| \ldots, \vec{c}_{j-1}||\vec{c}_{j+1}|| \ldots ||\vec{c}_{2k})$ and $\vec{c}_j$ *always* comes from the space orthogonal to $\mathsf{rowspan}(W||\vec{w}_{k+1}|| \ldots ||\vec{w}_{i-1}||\vec{w}_{i+1}|| \ldots ||\vec{w}_{2k})$. We are only changing the condition $\vec{w}_i \perp \vec{c}_j$ by applying arguing that leakage on these vectors is bounded by 3ℓ since they only "occur" in *three* time periods, and then applying Lemma 7.3.1. We then get indistinguishability assuming the parameters $\ell \leq ((m-5k-1)/6)\log(q) - \omega(\log(\lambda))$.

7.5 Information-Theoretic Impossibility

We notice that, unlike public-key encryption and digital signatures, traditional secret sharing schemes (including 2-out-2 secret sharing) can achieve *information-theoretic* security. Namely, the secret message remains perfectly hidden even against a *computationally unbounded* attacker. Unfortunately, we show that the same cannot be true for CLRS schemes, which must withstand *continuous* leakage of shares. Namely, by specifying *arbitrary* (as opposed to efficient) leakage predicates, the attacker can reconstruct the hidden secret with probability arbitrarily close to 1. In fact, the result holds even when the following additional restrictions are placed on the attacker:

- The leakage bound $\ell = 1$. Namely, at most one bit can leak in between successive share updates.

- Each leakage predicate can only depend on the current share value, but not on the randomness for the next update. Namely, we can assume *leak-free updates*.
- If s_1 is the bit size of the first share and s_2 is the bit size of the second share, the attacker will use only s_1 **Leakage** queries on the first share and only s_2 **Leakage** queries on the second share.[4]
- The sequence of **Leakage** and **Update** queries is specified *non-adaptively*.[5]
- The attacker can even break *one-wayness* of the CLRS scheme, and not just semantic security. Namely, the shared message $\mathbf{m}$ can be chosen at random (as opposed to being either $\mathbf{m}_0$ or $\mathbf{m}_1$ chosen by the attacker) and will be recovered in full with probability $1 - \epsilon$ (for any $\epsilon > 0$).

As it turns out, all these properties will easily follow from the following more general attack, which we call *Continuous Leakage of Consistent Value* (CLCV) attack. The attack, parameterized by an arbitrary key refreshing procedure Update, shows how to leak a value "Update-consistent" with an s-bit initial secret, using at most s non-adaptive (but computationally unbounded) **Leakage** queries. After specifying this attack below, the attack on the CLRS scheme will simply perform a separate CLCV attack on each share, and then run the honest reconstruction algorithm to recover the secret.

CLCV Attack. Assume $\mathsf{Update} : \{0,1\}^s \to \{0,1\}^s$ is an arbitrary randomized procedure. Given such a procedure, we say that a value x' is *consistent* with x if either $x = x'$ or there exists some $i \geq 1$ and a sequence of randomness strings $r_1, \ldots, r_i$ such that $x' = \mathsf{Update}(\ldots \mathsf{Update}(x; r_1) \ldots; r_i)$. We let $C(x)$ denote the set of all strings x' consistent with x.

We define the following *CLRV game* between a (computationally unbounded) attacker $\mathcal{A}$ and a challenger $\mathcal{C}$. The challenger $\mathcal{C}$ gets an input $x \in \{0,1\}^s$ and sets $x_0 = x$, $i = 0$. The attacker $\mathcal{A}$ can adaptively make any number of the following two queries:

Update Queries: $\mathcal{C}$ picks a random string r_{i+1}, sets $x_{i+1} = \mathsf{Update}(x_i; r_{i+1})$, and increments i.

Leakage Queries: $\mathcal{A}$ specifies a predicate $\mathsf{Leak} : \{0,1\}^s \to \{0,1\}$ and gets back the value $\mathsf{Leak}(x_i)$ from $\mathcal{C}$. (Notice, the leakage predicate does not take the

[4]This is essentially optimal, since otherwise both shares still have some entropy left after leakage, and the results of [DDV10] give an information-theoretic CLRS scheme in this setting.

[5]Alternatively, one can use s_1 non-adaptive queries on the first share, and a single *adaptive* query on the second share.

update randomness as its input.) $\mathcal{C}$ then automatically makes an **Update** query described above, updating x_i to x_{i+1} and incrementing i.[6]

At the end of the game the attacker outputs a value x' and wins if x' is consistent with x: $x' \in C(x)$.

Lemma 7.5.1. *For any randomized procedure* $\mathsf{Update} : \{0,1\}^s \to \{0,1\}^s$ *and any* $\epsilon > 0$ *there exist an (inefficient) attacker* $\mathcal{A}^*$ *such that, for all* $x \in \{0,1\}^s$, $\mathcal{A}^*$ *wins the CLCV game against* $\mathcal{C}(x)$ *with probability* $1-\epsilon$. *Moreover,* $\mathcal{A}^*$ *is non-adaptive and makes only* s ***Leakage*** *queries.*

We prove the lemma below, but, as an immediate corollary, we get the attack on the CLRS mentioned at the beginning of the section. Namely, we set the failure parameter to $\epsilon/2$ and run the CLCV attacker from the Lemma above on both shares sh_1 and sh_2. Then, with probability $1-\epsilon$ we get correct share values sh'_1 and sh'_2 consistent with sh_1 and sh_2. By perfect correctness of CLRS, running the reconstruction procedure on sh'_1 and sh'_2 will return the correct message $\mathbf{m}$.

In fact, we notice that the CLCV attack essentially rules out information-theoretic security for any cryptographic primitive in the continuous leakage model enjoying perfect correctness. As we mentioned, however, this is primarily interesting for cryptographic primitives which permit information-theoretic solutions (without leakage) in the first place, such as secret sharing, one-time pad, one-time MACs, etc.

Proof of Lemma 7.5.1. We start with some notation before we describe our CLCV attacker $\mathcal{A}^*$. Given a permutation $\pi : \{0,1\}^s \to \{0,1\}^s$ and a non-empty set $X \subseteq \{0,1\}^s$, we let $\mathsf{smallest}_\pi(X)$ denote the (unique) string $x \in X$ having the lexicographically smallest value $\pi(x)$ among $\{\pi(x') \mid x' \in X\}$. Notice, if π is a random permutation, then $\mathsf{smallest}_\pi(X)$ is simply a random element of X. More generally, for any sequence $X_i \supseteq X_{i+1} \supseteq \ldots \supseteq X_{i+s-1}$ of s non-empty "shrinking" sets, if π is a random permutation, the probability that *all* s values $\mathsf{smallest}_\pi(X_{i+j})$ are the same only depends on the ratio between the sizes of the smallest and the largest sets:

$$\Pr_\pi\left[\mathsf{smallest}_\pi(X_i) = \mathsf{smallest}_\pi(X_{i+1}) = \ldots = \mathsf{smallest}_\pi(X_{i+s-1})\right] = \frac{|X_{i+s-1}|}{|X_i|} \quad (7.6)$$

Indeed, since the sets are contained in each other, all the values $\mathsf{smallest}_\pi(X_{i+j})$ are equal if and only if $\mathsf{smallest}_\pi(X_i) \in X_{i+s-1}$. However, since π is a random permutation, the latter happens with probability precisely $|X_{i+s-1}|/|X_i|$.

For $i \geq 0$, let $X_i = C(x_i)$ be the (non-empty) set of values consistent with the i-th secret x_i. The non-adaptive strategy of $\mathcal{A}^*$ is the following (recall, ϵ is the maximum allowed failure probability):

[6]This corresponds to the leakage bound $\ell = 1$, meaning that at most 1 bit can leak in between the updates.

- Pick a random permutation $\pi : \{0,1\}^s \to \{0,1\}^s$, and a random integer $t \in \{1, \dots, N \stackrel{\text{def}}{=} \frac{4s}{\epsilon^2 \log e}\}$.
- Perform the sequence of $i \stackrel{\text{def}}{=} (t-1)s$ **Update** queries, so that the current secret is x_i.
- Perform s **Leakage** queries, where query $j \in [s]$ will leak the j-th bit of $\mathsf{smallest}_\pi(X_{i+j-1})$. (Formally, the j-th leakage predicate $\mathsf{Leak}_j(y)$ returns the j-th bit of $\mathsf{smallest}_\pi(C(y))$.)
- Let $x' = b_1, \dots, b_s$ be the concatenation of s answers to the **Leakage** queries above. Output x' as a candidate secret consistent with $x = x_0$.

It remains to argue that the probability that x' is consistent with x is at least $1-\epsilon$. For that, let $y_j \stackrel{\text{def}}{=} \mathsf{smallest}_\pi(X_{i+j-1})$ be the value whose j-th bit we leak in the j-th **Leakage** query. Let us call these s values $y_1, \dots, y_s$ *critical.* Notice, each critical value is consistent with x, by definition. Thus, it suffices to argue that the probability all the critical values are *the same* is at least $1-\epsilon$. Indeed, in this case the leaked value x' is actually equal to all the critical values, and, hence, consistent with x. This also gives the intuition behind our construction of $\mathcal{A}^*$. Essentially, $\mathcal{A}^*$ choose a "random" index i hoping that the set of consistent values $X_i = C(x_i)$ will not shrink too much during the next s updates. If this is so (which we prove below happens with high probability), $\mathcal{A}^*$ wants to choose a "consistent representative" from the slowly shrinking sets $X_i, X_{i+1}, \dots, X_{i+s-1}$. Since this "shrinkage" might be adversarial, $\mathcal{A}^*$ randomly permutes the elements of the "slowly shrinking" sets, and hopes that the smallest element of these permuted sets does not "disappear" as the sets slowly shrink. Luckily, Equation (7.6) tells us precisely this, as long as the smallest set X_{i+s-1} is almost as big as the original set X_i. The formal details are given below.

Let us call a sequence of s consecutive updates an *epoch*, and let us examine the consistent sets $X_0, X_s, X_{2s}, \dots$ at the end of each epoch. We say that epoch $t \geq 1$ is "bad" if $|X_{ts}| < (1 - \frac{\epsilon}{2}) \cdot |X_{(t-1)s}|$; namely, the set of consistent values shrunk by more than a factor $(1-\frac{\epsilon}{2})$. Notice, since the set of consistent values has size at most 2^s and at least 1, we know that the maximal number n of bad epochs must satisfy the relation $2^s(1-\frac{\epsilon}{2})^n \geq 1$. Solving for n, we get the number of bad epochs $n \leq 2s/(\epsilon \log e)$.

Recall now that $\mathcal{A}^*$ choose a random epoch t in the range $\{1, \dots, N \stackrel{\text{def}}{=} \frac{4s}{\epsilon^2 \log e}\}$. Since there are at most $n \leq 2s/(\epsilon \log e)$ bad epochs, we get that the probability $\mathcal{A}^*$ chose a bad epoch is at most $n/N \leq \epsilon/2$. Otherwise, if $\mathcal{A}^*$ chose a "good" epoch, we know that $|X_{ts}| \geq (1-\frac{\epsilon}{2}) \cdot |X_{(t-1)s}|$. Since π was random and epoch t is good, Equation (7.6) then tells us that s critical values $y_j = \mathsf{smallest}_\pi(X_{i+j-1})$ are *all the same* with probability $1-\epsilon/2$, meaning they are not the same with probability at most $\epsilon/2$. Hence, we see that $\mathcal{A}^*$ fails either it chose a bad epoch t (probability at most $\epsilon/2$), or the epoch was good but the critical values were inconsistent (again, probability at most $\epsilon/2$). Summing these, we get that $\mathcal{A}^*$ fails with probability at most ϵ, completing the proof. $\square$

Chapter 8

Conclusions

In this thesis, we proposed a new model to capture the leakage available to an attacker through side-channel attacks against an implementation of a cryptographic primitive. We then showed how to construct many of the most useful cryptographic primitives (i.e. one-way relations, signatures, public-key encryption and secret sharing or secure storage) which remain provably secure even in the presence of such formally modeled leakage.

Many interesting and important questions remain unanswered. Perhaps most importantly, the proposed model of leakage may still not be the “right one”. There is certainly a need for better empirical analysis of whether the model is sufficient to capture all realistic examples of side-channel attacks. Nevertheless, we believe that having a good understanding of our current model, with its simplicity and elegance, will prove useful in any future work on more fine-tuned models. It also remains an important open problem to come up with more constructions in our model, realizing more advanced primitives (identity based encryption, fully-homomorphic encryption ...), getting better efficiency and leakage rates, and under more assumptions. Lastly, given our result showing how to securely store a secret on a device consisting of multiple components that leak individually, it remains an interesting open problem to also construct schemes for securely performing (arbitrary?) computations on such secrets, so that nothing but the outputs of the computation are revealed even to an attacker that gets side-channel leakage on the state of each component during the computations.

Bibliography

[AARR02] Dakshi Agrawal, Bruce Archambeault, Josyula R. Rao, and Pankaj Rohatgi. The EM side-channel(s). In Burton S. Kaliski Jr., Çetin Kaya Koç, and Christof Paar, editors, *CHES*, volume 2523 of *Lecture Notes in Computer Science*, pages 29–45. Springer, August 13-15 2002.

[ACM87] *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, New York City, 25–27 May 1987.

[ADN$^+$10] Joël Alwen, Yevgeniy Dodis, Moni Naor, Gil Segev, Shabsi Walfish, and Daniel Wichs. Public-key encryption in the bounded-retrieval model. In Gilbert [Gil10], pages 113–134.

[ADR02] Yonatan Aumann, Yan Zong Ding, and Michael O. Rabin. Everlasting security in the bounded storage model. *IEEE Transactions on Information Theory*, 48(6):1668–1680, 2002.

[ADW09a] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In Halevi [Hal09], pages 36–54.

[ADW09b] Joël Alwen, Yevgeniy Dodis, and Daniel Wichs. Survey: Leakage resilience and the bounded retrieval model. In *ICITS*, pages 1–18, 2009.

[AGH10] Adi Akavia, Shafi Goldwasser, and Carmit Hazay. Distributed public key schemes secure against continual leakage. Unpublished Manuscript, 2010.

[AGV09] Adi Akavia, Shafi Goldwasser, and Vinod Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In Omer Reingold, editor, *Sixth Theory of Cryptography Conference — TCC 2007*, volume 5444 of *Lecture Notes in Computer Science*. Springer-Verlag, 2009.

[Ajt10] Miklós Ajtai. Oblivious rams without cryptogrpahic assumptions. In Leonard J. Schulman, editor, *STOC*, pages 181–190. ACM, 2010.

[Ajt11] Miklós Ajtai. Secure computation with information leaking to an adversary. In Lance Fortnow and Salil P. Vadhan, editors, *STOC*, pages 715–724. ACM, 2011.

[And97] Ross Anderson. Invited lecture. In *Fourth Annual Conference on Computer and Communications Security.* ACM, 1997. Summary appears in [And01].

[And01] Ross Anderson. Two remarks on public key cryptology. `http://www.cl.cam.ac.uk/users/rja14/`, 2001.

[AR99] Yonatan Aumann and Michael O. Rabin. Information theoretically secure communication in the limited storage space model. In *CRYPTO*, pages 65–79, 1999.

[AR00] Michel Abdalla and Leonid Reyzin. A new forward-secure digital signature scheme. In Tatsuaki Okamoto, editor, *Advances in Cryptology—ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 116–129, Kyoto, Japan, 3–7 December 2000. Springer-Verlag. Full version available from the Cryptology ePrint Archive, record 2000/002, `http://eprint.iacr.org/`.

[BBCM95] Charles H. Bennett, Gilles Brassard, Claude Crépeau, and Ueli M. Maurer. Generalized privacy amplification. *IEEE Transactions on Information Theory*, 41(6):1915–1923, 1995.

[BBR88] Charles H. Bennett, Gilles Brassard, and Jean-Marc Robert. Privacy amplification by public discussion. *SIAM Journal on Computing*, 17(2):210–229, 1988.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In *CRYPTO*, pages 41–55, 2004.

[BCC$^+$09] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In *CRYPTO*, pages 108–125, 2009.

[BDL97] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of checking cryptographic protocols for faults. In Walter Fumy, editor, *Advances in Cryptology—EUROCRYPT 97*, volume 1233 of *LNCS*, pages 37–51. Springer-Verlag, 1997.

[BE03] Hagai Bar-El. Known attacks against smartcards, 2003. last accessed: August 26, 2009. `http://www.hbarel.com/publications/Known_Attacks_Against_Smartcards.pdf`.

[BG10] Zvika Brakerski and Shafi Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In Rabin [Rab10], pages 1–20.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, pages 1–10, Chicago, Illinois, 2–4 May 1988.

[BHHO08] Dan Boneh, Shai Halevi, Michael Hamburg, and Rafail Ostrovsky. Circular-secure encryption from decision diffie-hellman. In *CRYPTO*, pages 108–125, 2008.

[BKKV10] Zvika Brakerski, Jonathan Katz, Yael Kalai, and Vinod Vaikuntanathan. Overcomeing the hole in the bucket: Public-key cryptography against resilient to continual memory leakage. In *FOCS* [IEE10], pages 501–510.

[BM99] Mihir Bellare and Sara Miner. A forward-secure digital signature scheme. In Wiener [Wie99], pages 431–448. Revised version is available from `http://www.cs.ucsd.edu/~mihir/`.

[Bon03] Dan Boneh, editor. *Advances in Cryptology—CRYPTO 2003*, volume 2729 of *LNCS*. Springer-Verlag, 2003.

[Boy99] Victor Boyko. On the security properties of the OAEP as an all-or-nothing transform. In Wiener [Wie99], pages 503–518.

[BS97] Eli Biham and Adi Shamir. Differential fault analysis of secret key cryptosystems. In Kaliski [Kal97], pages 513–525.

[BSW11] Elette Boyle, Gil Segev, and Daniel Wichs. Fully leakage-resilient signatures. In Kenneth G. Paterson, editor, *EUROCRYPT*, volume 6632 of *Lecture Notes in Computer Science*, pages 89–108. Springer, 2011.

[CCS09] Jan Camenisch, Nishanth Chandran, and Victor Shoup. A public key encryption scheme secure against key dependent chosen plaintext and adaptive chosen ciphertext attacks. In Joux [Jou09b], pages 351–368.

[CDD$^+$07] David Cash, Yan Zong Ding, Yevgeniy Dodis, Wenke Lee, Richard J. Lipton, and Shabsi Walfish. Intrusion-resilient key exchange in the bounded retrieval model. In *TCC*, pages 479–498, 2007.

[CDH$^+$00] Ran Canetti, Yevgeniy Dodis, Shai Halevi, Eyal Kushilevitz, and Amit Sahai. Exposure-resilient functions and all-or-nothing transforms. In Bart Preneel, editor, *Advances in Cryptology—EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 453–469. Springer-Verlag, 2000.

[CDRW10] Sherman S. M. Chow, Yevgeniy Dodis, Yannis Rouselakis, and Brent Waters. Practical leakage-resilient identity-based encryption from simple assumptions. In Ehab Al-Shaer, Angelos D. Keromytis, and Vitaly Shmatikov, editors, *ACM Conference on Computer and Communications Security*, pages 152–161. ACM, 2010.

[CG88] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM Journal on Computing*, 17(2):230–261, 1988.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, pages 255–271, 2003.

[CK01] Ran Canetti and Hugo Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT*, pages 453–474, 2001.

[CKM10] Seung Geol Choi, Aggelos Kiayias, and Tal Malkin. Bitr: Built-in tamper resilience. Cryptology ePrint Archive, Report 2010/503, 2010. `http://eprint.iacr.org/`.

[CLW06] Giovanni Di Crescenzo, Richard J. Lipton, and Shabsi Walfish. Perfectly secure password protocols in the bounded retrieval model. In *TCC*, pages 225–244, 2006.

[CS98] Ronald Cramer and Victor Shoup. A practical public key cryptosystem provably secure against chosen ciphertext attack. In Hugo Krawczyk, editor, *Advances in Cryptology—CRYPTO '98*, volume 1462 of *LNCS*. Springer-Verlag, 23–27 August 1998.

[CS02] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In Knudsen [Knu02], pages 45–64.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *STOC*, pages 542–552, 1991.

[DDV10] Francesco Davì, Stefan Dziembowski, and Daniele Venturi. Leakage-resilient storage. In Juan A. Garay and Roberto De Prisco, editors, *SCN*, volume 6280 of *Lecture Notes in Computer Science*, pages 121–137. Springer, 2010.

[DF89] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In G. Brassard, editor, *Advances in Cryptology—CRYPTO '89*, volume 435 of *LNCS*, pages 307–315. Springer-Verlag, 1990, 20–24 August 1989.

[DF11] Stefan Dziembowski and Sebastian Faust. Leakage-resilient cryptography from the inner-product extractor, 2011. Manuscript in submission.

[DFK$^+$03] Yevgeniy Dodis, Matt Franklin, Jonathan Katz, Atsuko Miyaji, and Moti Yung. Intrusion-resilient public-key encryption. In Marc Joye, editor, *Progress in Cryptology — CT-RSA 2003*, LNCS. Springer-Verlag, April 13-17 2003.

[DFK$^+$04] Yevgeniy Dodis, Matthew K. Franklin, Jonathan Katz, Atsuko Miyaji, and Moti Yung. A generic construction for intrusion-resilient public-key encryption. In Tatsuaki Okamoto, editor, *Progress in Cryptology — CT-RSA 2004*, volume 2964 of *LNCS*, pages 81–98. Springer-Verlag, February 23-27 2004.

[DGK$^+$10] Yevgeniy Dodis, Shafi Goldwasser, Yael Tauman Kalai, Chris Peikert, and Vinod Vaikuntanathan. Public-key encryption secure against auxiliary input. In Daniele Micciancio, editor, *Theory of Cryptography Conference*, volume 5978 of *LNCS*, pages 361–381. Springer, 2010.

[DHLW10a] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS* [IEE10], pages 511–520.

[DHLW10b] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 613–631. Springer, 2010.

[DKL09] Yevgeniy Dodis, Yael Tauman Kalai, and Shachar Lovett. On cryptography with auxiliary input. In Michael Mitzenmacher, editor, *STOC*, pages 621–630. ACM, 2009.

[DKRS06] Yevgeniy Dodis, Jonathan Katz, Leonid Reyzin, and Adam Smith. Robust fuzzy extractors and authenticated key agreement from close secrets. In Cynthia Dwork, editor, *Advances in Cryptology—CRYPTO 2006*, volume 4117 of *LNCS*, pages 232–250. Springer-Verlag, 20–24 August 2006.

[DKXY02] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Key-insulated public key cryptosystems. In Knudsen [Knu02].

[DKXY03] Yevgeniy Dodis, Jonathan Katz, Shouhuai Xu, and Moti Yung. Strong key-insulated signature schemes. In *PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *LNCS*. Springer-Verlag, 2003.

[DLWW11] Yevgeniy Dodis, Allison Lewko, Brent Waters, and Daniel Wichs, 2011.

[DMN11] Ivan Damgård, Sigurd Meldgaard, and Jesper Buus Nielsen. Perfectly secure oblivious ram without random oracles. In Ishai [Ish11], pages 144–163.

[Dod00] Yevgeniy Dodis. *Exposure-Resilient Cryptography.* PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2000.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38(1):97–139, 2008.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *49th Symposium on Foundations of Computer Science*, pages 293–302, Philadelphia, PA, USA, October 25–28 2008. IEEE Computer Society.

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In Andrew Chi-Chih Yao, editor, *ICS*, pages 434–452. Tsinghua University Press, 2010.

[DSS01] Yevgeniy Dodis, Amit Sahai, and Adam Smith. On perfect and adaptive security in exposure-resilient cryptography. In Birgit Pfitzmann, editor, *Advances in Cryptology—EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 301–324. Springer-Verlag, 2001.

[DvOW92] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. Authentication and authenticated key exchanges. *Des. Codes Cryptography*, 2(2):107–125, 1992.

[DW09] Yevgeniy Dodis and Daniel Wichs. Non-malleable extractors and symmetric key cryptography from weak secrets. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 601–610, Bethesda, MD, USA, 2009. ACM.

[Dzi06] Stefan Dziembowski. Intrusion-resilience via the bounded-storage model. In *TCC*, pages 207–224, 2006.

[ECR] ECRYPT. Side channel cryptanalysis lounge. last accessed: May 1, 2011. `http://www.emsec.rub.de/research/projects/sclounge/`.

[FGMY97] Yair Frankel, Peter Gemmell, Philip D. MacKenzie, and Moti Yung. Optimal resilience proactive public-key cryptosystems. In *FOCS*, pages 384–393, 1997.

[FKPR10] Sebastian Faust, Eike Kiltz, Krzysztof Pietrzak, and Guy N. Rothblum. Leakage-resilient signatures. In *TCC*, pages 343–360, 2010.

[FPV11] Sebastian Faust, Krzysztof Pietrzak, and Daniele Venturi. Tamper-proof circuits: How to trade leakage for tamper-resilience. ICALP, 2011.

[FRR$^+$10] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In Gilbert [Gil10], pages 135–156.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology—CRYPTO '86*, volume 263 of *LNCS*, pages 186–194. Springer-Verlag, 1987, 11–15 August 1986.

[Gem97] P. Gemmel. RSA CryptoBytesm, 2(3) 7 - 12, 1997.

[Gil10] Henri Gilbert, editor. *Advances in Cryptology - EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*. Springer, 2010.

[GKR08] Shafi Goldwasser, Yael Tauman Kalai, and Guy N. Rothblum. One-time programs. In David Wagner, editor, *Advances in Cryptology - CRYPTO 2008*, volume 5157 of *LNCS*, pages 39–56. Springer-Verlag, 2008.

[GLM$^+$03] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic Tamper-Proof (ATP) security: Theoretical foundations for security against hardware tampering. In Naor [Nao04], pages 258–277.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or a completeness theorem for protocols with honest majority. In ACM [ACM87], pages 218–229.

[GO96] Oded Goldreich and Rafail Ostrovsky. Software protection and simulation on oblivious rams. *J. ACM*, 43(3):431–473, 1996.

[Gol87] Oded Goldreich. Towards a theory of software protection and simulation by oblivious rams. In ACM [ACM87], pages 182–194.

[GR04] Steven D. Galbraith and Victor Rotger. Easy decision-diffie-hellman groups. *LMS Journal of Computation and Mathematics*, 7:2004, 2004.

[GR10] Shafi Goldwasser and Guy N. Rothblum. Securing computation against continuous leakage. In Rabin [Rab10], pages 59–79.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology - EUROCRYPT 2008*, volume 4965 of *LNCS*, pages 415–432. Springer-Verlag, 2008.

[Hal09] Shai Halevi, editor. *Advances in Cryptology - CRYPTO 2009*, volume 5677 of *LNCS*. Springer-Verlag, 2009.

[HJKY95] Amir Herzberg, Stanislaw Jarecki, Hugo Krawczyk, and Moti Yung. Proactive secret sharing or: How to cope with perpetual leakage. In Don Coppersmith, editor, *CRYPTO*, volume 963 of *LNCS*, pages 339–352. Springer-Verlag, 27–31 August 1995.

[HK07] Dennis Hofheinz and Eike Kiltz. Secure hybrid encryption from weakened key encapsulation. In *CRYPTO*, pages 553–571, 2007.

[HL11] Shai Halevi and Huijia Lin. After-the-fact leakage in public-key encryption. In Ishai [Ish11], pages 107–124.

[HSH$^+$08] J. Alex Halderman, Seth D. Schoen, Nadia Heninger, William Clarkson, William Paul, Joseph A. Calandrino, Ariel J. Feldman, Jacob Appelbaum, and Edward W. Felten. Lest we remember: Cold boot attacks on encryption keys. In Paul C. van Oorschot, editor, *USENIX Security Symposium*, pages 45–60. USENIX Association, 2008.

[IEE10] IEEE. *51th Symposium on Foundations of Computer Science*, Las Vegas, NV, USA, October 23–26 2010.

[IPSW06] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits II: Keeping secrets in tamperable circuits. In Serge Vaudenay, editor, *Advances in Cryptology—EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 308–327. Springer-Verlag, 2006.

[IR02] Gene Itkis and Leonid Reyzin. SIBIR: Signer-base intrusion-resilient signatures. In Yung [Yun02]. Available from `http://eprint.iacr.org/2002/054/`.

[Ish11] Yuval Ishai, editor. *Theory of Cryptography - 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings*, volume 6597 of *Lecture Notes in Computer Science*. Springer, 2011.

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Boneh [Bon03].

[JGS11] Abhishek Jain, Sanjam Garg, and Amit Sahai. Leakage-resilient zero knowledge. In *CRYPTO*, 2011.

[Jou09a] Antoine Joux, editor. *Advances in Cryptology - EUROCRYPT 2009*, volume 5479 of *LNCS*. Springer-Verlag, 2009.

[Jou09b] Antoine Joux, editor. *Advances in Cryptology - EUROCRYPT 2009, 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings*, volume 5479 of *Lecture Notes in Computer Science*. Springer, 2009.

[JV10] Ali Juma and Yevgeniy Vahlis. Protecting cryptographic keys against continual leakage. In Rabin [Rab10], pages 41–58.

[Kal97] Burton S. Kaliski, Jr., editor. *Advances in Cryptology—CRYPTO '97*, volume 1294 of *LNCS*. Springer-Verlag, 1997.

[KJJ99] Paul Kocher, Joshua Jaffe, and Benjamin Jun. Differential power analysis. In Wiener [Wie99], pages 388–397.

[Knu02] Lars Knudsen, editor. *Advances in Cryptology—EUROCRYPT 2002*, volume 2332 of *LNCS*. Springer-Verlag, 28 April–2 May 2002.

[Koc96] Paul Kocher. Timing attacks on implementations of Diffie-Hellman, RSA, DSS, and other systems. In Neal Koblitz, editor, *Advances in Cryptology—CRYPTO '96*, volume 1109 of *LNCS*, pages 104–113. Springer-Verlag, 18–22 August 1996.

[KP10] Eike Kiltz and Krzysztof Pietrzak. Leakage resilient elgamal encryption. In Masayuki Abe, editor, *ASIACRYPT*, volume 6477 of *Lecture Notes in Computer Science*, pages 595–612. Springer, 2010.

[KR09] Bhavana Kanukurthi and Leonid Reyzin. Key agreement from close secrets over unsecured channels. In Joux [Jou09a], pages 206–223.

[KV09] Jonathan Katz and Vinod Vaikuntanathan. Signature schemes with bounded leakage resilience. In Mitsuru Matsui, editor, *Advances in Cryptology—ASIACRYPT 2009*, LNCS. Springer-Verlag, 2009. To Appear.

[LLTT05] Chia-Jung Lee, Chi-Jen Lu, Shi-Chun Tsai, and Wen-Guey Tzeng. Extracting randomness from multiple independent sources. *IEEE Transactions on Information Theory*, 51(6):2224–2227, 2005.

[LLW11] Allison B. Lewko, Mark Lewko, and Brent Waters. How to leak on key updates. In *STOC*, 2011. To Appear.

[Lu02] Chi-Jen Lu. Hyper-encryption against space-bounded adversaries from on-line strong extractors. In Yung [Yun02].

[Mau92] Ueli Maurer. Conditionally-perfect secrecy and a provably-secure randomized cipher. *Journal of Cryptology*, 5(1):53–66, 1992.

[MMM02] Tal Malkin, Daniele Micciancio, and Sara K. Miner. Efficient generic forward-secure signatures with an unbounded number of time periods. In Lars R. Knudsen, editor, *EUROCRYPT*, volume 2332 of *Lecture Notes in Computer Science*, pages 400–417. Springer, 2002.

[MR04] Silvio Micali and Leonid Reyzin. Physically observable cryptography (extended abstract). In Naor [Nao04], pages 278–296.

[MTVY11] Tal Malkin, Isamu Teranishi, Yevgeniy Vahlis, and Moti Yung. Signatures resilient to continual leakage on memory and computation. In Ishai [Ish11], pages 89–106.

[MW97] Ueli Maurer and Stefan Wolf. Privacy amplification secure against active adversaries. In Kaliski [Kal97], pages 307–321.

[Nao04] Moni Naor, editor. *First Theory of Cryptography Conference — TCC 2004*, volume 2951 of *LNCS*. Springer-Verlag, February 19–21 2004.

[NS09] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. In Halevi [Hal09], pages 18–35.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–53, 1996.

[Pie09] Krzysztof Pietrzak. A leakage-resilient mode of operation. In Joux [Jou09a], pages 462–482.

[PSP$^+$08] Christophe Petit, Franois-Xavier Standaert, Olivier Pereira, Tal G. Malkin, and Moti Yung. A Block Cipher based Pseudo Random Number Generator Secure Against Side-Channel Key Recovery. In M. Abe and V. Gligor, editors, *ASIACCS '08: Proceedings of the 2008 ACM symposium on Information, computer and communications security*, pages 56–65. ACM, 3 2008.

[QK02] Jean-Jaques Quisquater and François Koene. Side channel attacks: State of the art, October 2002. `http://www.ipa.go.jp/security/enc/CRYPTREC/fy15/doc/1047_Side_Channel_report.pdf`, last accessed: August 26, 2009.

[QS01] Jean-Jacques Quisquater and David Samyde. Electromagnetic analysis (ema): Measures and counter-measures for smart cards. In Isabelle Attali and Thomas P. Jensen, editors, *E-smart*, volume 2140 of *LNCS*, pages 200–210. Springer-Verlag, September 19-21 2001.

[Rab10] Tal Rabin, editor. *Advances in Cryptology - CRYPTO 2010, 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings*, volume 6223 of *Lecture Notes in Computer Science*. Springer, 2010.

[Rel] Reliable Computing Laboratory, Boston University. Side channel attacks database. `http://www.sidechannelattacks.com`, last accessed: August 26, 2009.

[Riv97] Ronald L. Rivest. All-or-nothing encryption and the package transform. In Eli Biham, editor, *FSE*, volume 1267 of *Lecture Notes in Computer Science*, pages 210–218. Springer, 1997.

[Rom90] John Rompel. One-way functions are necessary and sufficient for secure signatures. In *Proceedings of the Twenty Second Annual ACM Symposium on Theory of Computing*, pages 387–394, Baltimore, Maryland, 14–16 May 1990.

[RW03] Renato Renner and Stefan Wolf. Unconditional authenticity and privacy from an arbitrarily weak secret. In Boneh [Bon03], pages 78–95.

[Sco02] Mike Scott. Authenticated id-based key exchange and remote log-in with simple token and pin number. Cryptology ePrint Archive, Report 2002/164, 2002. `http://eprint.iacr.org/`.

[SDFY94] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *STOC*, pages 522–533, 1994.

[Sha79] Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[Sha07] Hovav Shacham. A cramer-shoup encryption scheme from the linear assumption and from progressively weaker linear variants, 2007. Cryptology ePrint Archive, Report 2007/074.

[SMY09] François-Xavier Standaert, Tal Malkin, and Moti Yung. A unified framework for the analysis of side-channel key recovery attacks. In Joux [Jou09b], pages 443–461.

[Vad04] S. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *J. Cryptology*, 17(1), 2004.

[Ver04] Eric R. Verheul. Evidence that xtr is more secure than supersingular elliptic curve cryptosystems. *J. Cryptology*, 17(4):277–296, 2004.

[Wat] Brent Waters. Personal Communication. April 2010.

[Wie99] Michael Wiener, editor. *Advances in Cryptology—CRYPTO '99*, volume 1666 of *LNCS*. Springer-Verlag, 15–19 August 1999.

[Yao82] Andrew Chi-Chih Yao. Protocols for secure computations. In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, Chicago, Illinois, 3–5 November 1982. IEEE.

[YSPY10] Yu Yu, Franois-Xavier Standaert, Olivier Pereira, and Moti Yung. Practical leakage-resilient pseudorandom generators. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 141–151. ACM, 10 2010.

[Yun02] Moti Yung, editor. *Advances in Cryptology—CRYPTO 2002*, volume 2442 of *LNCS*. Springer-Verlag, 18–22 August 2002.