The security of the one-more discrete-logarithm assumption and blind Schnorr signatures

Antoine Plouviez

PSL
UNIVERSITÉ PARIS

THÈSE DE DOCTORAT
DE L'UNIVERSITÉ PSL

Préparée à l'École Normale Supérieure

The security of the One-More Discrete-Logarithm assumption and Blind Schnorr Signatures

Soutenue par
Antoine PLOUVIEZ
Le 29 Octobre 2021

École doctorale n°386
Sciences Mathématiques de Paris Centre

Spécialité
Informatique

Composition du jury :

Georg Fuchsbauer TU Wien	*Directeur de thèse*
David Pointcheval CNRS, École Normale Supérieure	*Directeur de thèse*
Damien Vergnaud Sorbonne Université	*Président du jury*
Marc Fischlin Darmstadt University of Technology	*Rapporteur*
Yannick Seurin ANSSI	*Examinateur*
Esha Ghosh Microsoft, MSR	*Examinatrice*

ENS | PSL

The security of the One-More Discrete-Logarithm assumption and Blind Schnorr Signatures

Antoine PLOUVIEZ

29 Octobre 2021

Abstract

In cryptography, many protocols are based on security assumptions such as the Discrete Logarithm (DL) assumption and the Computationnal Diffie-Hellman (CDH) assumption. The security of many protocols rely on the hardness of such assumptions, and the variety of protocols brought some more analogous assumptions such as the one-more assumptions.

The one more-discrete logarithm assumption (OMDL) is central to the security analysis of identification protocols, multi-signature schemes such as the recent MuSig2 multi-signatures and blind signatures, most notably the Blind Schnorr Signatures which we analyse in this work.

Despite OMDL wide use, surprisingly, those one-more assumptions are lacking any rigorous analysis; there is not even a proof that it holds in the generic group model (GGM). (We show that a claimed proof is flawed.) OMDL is also assumed for many impossibility results that show that certain security reductions cannot exist.

We give rigorous proofs in the GGM of OMDL and a related assumption, the one-more computational Diffie-Hellman assumption. We do so by deviating from prior GGM proofs replacing the use of the Schwartz-Zippel Lemma by a new argument.

In this work we also analyse the Schnorr blind signing protocol which allows blind issuing of Schnorr signatures, one of the most widely used signatures. Despite its practical relevance, its security analysis is also unsatisfactory. The only known security proof is rather informal and in the combination of the GGM and the random oracle model (ROM) assuming that the "ROS problem" is hard.

We analyze the security of these schemes in the algebraic group model (AGM), an idealized model closer to the standard model than the GGM. We first prove tight security of Schnorr signatures from the DL assumption in the AGM+ROM. We then give a rigorous proof for blind Schnorr signatures in the AGM+ROM assuming hardness of the OMDL problem and ROS.

As ROS can be solved in sub-exponential time using Wagner's algorithm, we propose a simple modification of the signing protocol, which leaves the signatures unchanged. It is therefore compatible with systems that already use Schnorr signatures, such as blockchain protocols. We show that the security of our modified scheme relies on the hardness of a problem related to ROS that appears much harder.

Finally, as the situation is similar for (Schnorr-)signed ElGamal encryption (a simple CCA2-secure variant of ElGamal), we give tight reductions, again in the AGM+ROM, of the CCA2 security of signed ElGamal encryption to DDH and signed hashed ElGamal key encapsulation to DL.

Acknowledgments

Je souhaite dans un premier temps remercier Georg et David sans qui cette thèse n'aurait pas existé. J'ai pu découvrir la côte est des Etats-Unis très tôt dans mon parcours, et cela grâce à la confiance que mes superviseurs m'ont accordées.

Je souhaite aussi remercier les membres du jury de thèse d'avoir bien voulu prendre le temps de s'intéresser aux travaux que j'ai réunis dans ce document.

Je remercie aussi Esha Ghosh et Melissa Chase qui m'ont supervisé pour le projet avec MSR (Microsoft) et qui m'ont accompagné dans ma découverte de Seattle.

Enfin, même si la fin de thèse fut chaotique d'un point de vue interaction au laboratoire à cause du Covid, je souhaite chaleureusement remercier toute l'équipe du laboratoire qui m'a vraiment aidé à tenir pendant ces années pas toujours faciles.

Merci donc à Melissa, Chloé et Romain pour ce voyage magnifique en Californie après Crypto, et dont je me souviendrai toute ma vie. Merci à Louiza, Aurélien, Anca, Michele, Jeremy les anciens qui ont su m'accueillir et me motiver, que j'ai vu partir les uns après les autres après avoir soutenu leurs thèse ! Merci Balthazar, qui a continué à travailler avec moi même après sa soutenance !! Merci à Huy, qui a commencé sa thèse en même temps que moi, ça fait plaisir ! Merci à Azam et son humour :-p Je me souviendrai longtemps des parties de jeu de plateau qu'on a organisé dans le labo avant le covid ! C'était chouette.

Merci aux permanents Brice, Céline, Michel qui ont bien voulu m'attendre finir mon repas à la cantine pendant les repas en commun à 11h45 (parce que je suis long à manger). Et enfin merci aux nouveaux venus, Lenaick, Leonard, Théo et les très nouveaux que je n'ai malheureusement pas eu le temps de rencontrer à causes des conditions covid : Paola, Michael, Hugo.

Enfin, bien sûr il me faut sortir du laboratoire pour remercier mon entourage qui m'a soutenu sans faille pendant ces trois années. Je remercie Janine qui a pu supporter tout le déroulé de mes humeurs vacillantes et en déduire que "jamais je ferai une thèse moi, si c'est pour être dans ces états là". Je remercie mes parents, Philippe et Nathalie qui m'ont beaucoup encouragé, et même accueilli pendant le covid pour télétravailler depuis chez eux à la campagne. Mes soeurs Margaux et Marion, merci, merci.

Bon et puis y a tous les anciens potes de l'ENS, mais là la liste est trop longue. Les collocs, Clément et François; Les improvisateurs, Glen, Aaron, Quentin, Charlie, Nobody, Léo, Aura, Alice, Anne Gaelle; Les matheux Vianney, Timothée, Alexandre, Shmuel, Nicolas et Noémie. Il y a aussi My-An et Léo, et enfin les dessinateurs Thomas, Bob, Camille.... Et en plus j'en oublie...!

Contents

Chapter 1

Intoduction

1.1 Context

Generalities

Signing is an essential procedure in human societies. From the seal on an historical manuscript to your own artistic calligraphy on your ID card, it seems that the signature procedure will always be part of every human life. Producing a signature is not a big deal for anyone though, since a prerequisite for an efficient signature is to be easy to build. In fact signatures could be even thought as a gift from nature that makes every human (and every living being) different. There exist some signatures that we carry with us without even thinking of then: we could think about our appearance (although appearance impersonation becomes more and more easy with fake videos), our DNA or our fingerprints. Those are used by the police as unintentional signatures left by the criminals which identify them.

As a child, we first encounter signatures when our teachers tell us to make our parent sign our grade book. We quickly learn two things: 1- signature falsification can make us impersonate our elders and 2- although they seem easy to produce for their author, it needs some training and effort to get the ability of copying a signature, but also a risk to be caught.

That's why a signature is so important: it should prove that you and only you could have seen / read / approved / written / sent a message and it thus should be impossible to forge (unforgeable). But progress in technology makes it more and more easy for a signature to be forged. The hand-written signatures can now easily be copied and pasted onto a numerical document, and even signed documents can be modified on an imagery software afterwards. Also it's unclear how to sign an email or an online transaction using a handwritten signature.

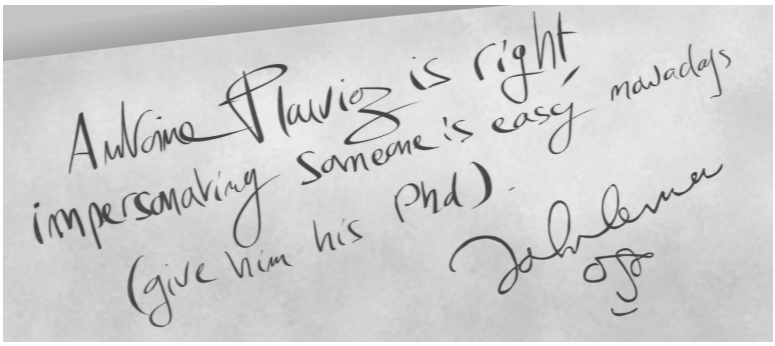

Figure 1.1: Message of John Lennon saying that the author of this Phd is right.

The digital development has obliged us to search for stronger signatures but also signatures protocols that can be used by a computer. Fortunately, computer science also came up with a lot of interesting solutions to solve this. Researchers designed signatures protocols based on mathematical problems, in which the signer who signs the message often knows a secret number x, and communicates a public number X which is linked to the secret but doesn't give information on it. Every message sent by the signer will then be "sealed" using the secret x, and everyone will be able to use X to verify that the seal is correct, thus that the message comes from the signer.

The more the technology was used, the more specific protocols were required for different uses. For example signatures protocols were invented in which a user who cannot sign (doesn't know the secret x) gets a signature from an authority (who knows x) who we call the signer. You could think of the user as the child who want a signature on his grade book from his parents who are the signers. Blind signature is a protocol in which the user actually wants a signature from the signer,

but doesn't want the signer to be able to recognize the signed document afterwards. The signer is blindfolded while signing the document provided by the user.

In this work we were interested in *Schnorr signatures*; which is a protocol of signature based on the Schnorr identification scheme. It's one of the simplest and oldest signature scheme designs based on prime-order groups. This protocol is widely deployed in a lot of numerical architectures and will soon be implemented in the Bitcoin. More precisely, we studied the security (unforgeability) of a blind signature version of this protocol which is actually harder to study, but also appeared to have some security weaknesses (which we will discuss in the work). Thus we propose a new blind signature version of this protocol which we call the Clause Blind Schnorr signature scheme.

In cryptography, the study of the security of protocols relies on mathematical reasoning and on assumptions. The idea is that some mathematical problems are hard to solve, and thus we can use them to hide a secret. If a signer has a secret which is protected behind a mathematical problem which cannot be solved efficiently, then this secret is considered to be safe and the protocol is secure. For example, say the signer knows 5 and 7, multiplies them together and shows you 35, it may be easy for you to deduce the decomposition $35 = 5 \times 7$. But now if the signer shows you 16637 and asks you to guess what is the decomposition of it, maybe you will eventually find that $131 \times 127 = 16637$, but we can be sure it will take you much more time to guess. In fact the factorisation problem of finding the two primes p, q such as $N = pq$, from only seeing N is a problem called the factorisation problem and is considered to be hard. This means the larger the primes p ans q are, the harder the problem is to solve. This problem is at the foundation of the RSA protocols.

In our work, we encountered a few mathematical problems, some of which are derivatives of the discrete logarithm problem. The security of blind Schnorr signature security (unforgeability) relies on two problems which are the one-more discrete logarithm (OMDL) and the ROS problem. The ROS problem was proved to be the main weakness of the blind Schnorr protocol, but we found that OMDL actually had not been formally analysed either in the literature.

We found that OMDL was a widely used mathematical assumption and that many results relied on it in the research literature, but since it has never been proved to be hard even in some idealized models, for example the generic group model, we decided to write a proof for it. It turned out to be more complex than expected.

Now let's get to the serious work.

Models

Generic Group Model: The Generic Group Model (GGM) is an idealized model for the security analysis of hardness assumptions such as DL, CDH or OMDL. It was introduced by Shoup [Nec94, Sho97] and it formalizes the idea that group elements don't give any information about the structure of the group. That's what elliptic curves aim on doing: generate group elements which are so hard to understand that they do not reveal information about the group. In fact in this model, an algorithm having access to some group elements as input can only do two actions which are: compute the composition of two group elements and check equality between two group elements.

In the generic group model, an adversary playing in a security game (the adversary is an algorithm which tries to break a security assumption) is given some group elements $(X_1, \ldots, X_n) \in \mathbb{G}^n$ as input. To formalize that those group elements do not leak more information that they should, we build an injective handle function $\Xi : \mathbb{G} \to E$ that associates each element of the group to an element of a set E that does not have a group structure. For example E could be $\{0,1\}^{\log_2(p)}$ and Ξ associate each X_i to a random string of E. Or E could be equal to $\mathbb{Z}_p$ and $\Xi(X_i) = i$ (and if exists j such as $i > j$ and $X_j = X_i$ then $\Xi(X_i) = j$).

In the Generic Group Model, the adversary will have as input only the handles $(\Xi(X_1), \Xi(X_2), \ldots, \Xi(X_n))$. Since those are not associated to a group structure, the adversary won't be able to compute a sum

of group elements alone, so we give to the adversary an access to a group computation oracle, we call GCmp which takes as input a pair $(\Xi(X_i), \Xi(X_j))$ and outputs $\Xi(X_i + X_j)$ to the adversary.

For an hardness assumption based on a secret $\overrightarrow{x} = (x_1, x_2, \ldots, x_n)$ that the adversary must guess, the Generic Group Model often allows to simulate the security game for the adversary without actually defining the secret $\overrightarrow{x}$. Say for example that the adversary gets $(\Xi(G), \Xi(X_1), \ldots, \Xi(X_n))$ as input with the secret $\overrightarrow{x}$ such that $X_i = x_i G$. The challenger simulating the hardness game can just build polynomial variables $(\mathsf{X}_1, \ldots, \mathsf{X}_n) \in \mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$ that represent each group element challenge $X_1, \ldots, X_n$. The challenger can give handles to the adversary on polynomials of $\mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$ instead of handles on group elements. Let's call $\Xi' : \mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n] \to E'$ the new injective handle function.

The adversary get as input $(\Xi'(1), \Xi'(\mathsf{X}_1), \ldots, \Xi'(\mathsf{X}_n))$ and on call to oracle $\text{GCmp}(\Xi'(P), \Xi'(Q))$ with $P, Q \in \mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$ the adversary get $\Xi'(P + Q)$. More generally, for a polynomial P the adversary gets access to $\Xi'(P)$ but the adversary expects $\Xi(P(\overrightarrow{x})G)$. In the end, the adversary should guess the secret $\overrightarrow{x}$, so it outputs $\overrightarrow{x}^*$ and since the challenger has simulated the whole game to the adversary with only polynomials (so without defining the secret $\overrightarrow{x}$), it can now pick the secret $\overrightarrow{x}$ at random. Since the secret $\overrightarrow{x}$ is picked by the challenger after the adversary gives its output $\overrightarrow{x}^*$ we get that the adversary has a very low chance to win the game with $\overrightarrow{x}^* = \overrightarrow{x}$.

Still, after the challenger picks a random secret $\overrightarrow{x}$, he needs to check that the simulation would have been the same if it was done with group elements based on this secret. Indeed, if for some polynomials $P \neq Q$ computed by the adversary, we have $P(\overrightarrow{x}) = Q(\overrightarrow{x})$, then we have $\Xi'(P) \neq \Xi'(Q)$ but $\Xi(P(\overrightarrow{x})G) = \Xi(Q(\overrightarrow{x})G)$ is what the adversary should have seen in the real game ! So the simulation is incorrect.

Usually, we bound the probability of failure by using the Schwarz-Zippel Lemma (described in the preliminaries section). But we will see that this Lemma does not apply for the OMDL assumption.

Algebraic Group Model: Notice that in the previous example, since the adversary can only build simple additions, it is not able to compute polynomials of degree more than 1. So every polynomial computed will have at most degree 1. In fact, all the group elements computed by the adversary can be decomposed using the input of the adversary: If the adversary get the handle for the polynomial $P = \alpha_0 + \alpha_1 \mathsf{X}_1 + \ldots + \alpha_n \mathsf{X}_n$, it means that from the group element point of view, it computes the group element $X = \alpha_0 G + \alpha_1 X_1 + \ldots + \alpha_n X_n$. So for every element that the adversary computes, the challenger knows which decomposition the adversary used with from input it got.

The *Algebraic Group Model* (AGM) was designed to be able to use this property [FKL18]. This model lies between the GGM and the standard model. The AGM considers that the adversary in the security game is *algebraic*, which means that every time the adversary outputs a group elements or gives a group element as input to an oracle, it should also provide its decomposition with respect to the input it got.

For example, if an adversary gets as input the group elements $(X_1, \ldots, X_n)$ and outputs Y, it must also output the vector $\overrightarrow{\alpha}$ such that $Y = \alpha_0 G + \alpha_1 X_1 + \ldots + \alpha_n X_n$. Security results in the AGM are proved via reductions to computationally hard problems like in the standard model.

Random Oracle Model: In the *Random Oracle Model* (ROM) we consider that there exists an oracle that gives a truly uniformly random output depending on the input. This algorithm should be deterministic, meaning that it outputs always the same value for the same input. This oracle is used as a blackbox by the challenger during the game. In practice it is an idealized model for the hash functions, meaning that every hash function implemented in this model will be represented by a random oracle in the ROM.

1.2 One More-Discrete Logarithm

Provable security is the prevailing paradigm in present-day cryptography. To analyze the security of a cryptographic scheme, one first defines a formal model for what it means to break the scheme and then gives a rigorous proof that this is infeasible assuming that some computational problem is hard.

Classical hardness assumption like RSA and the discrete logarithm assumption in various groups have received much scrutiny over the years, but there are now myriads of less studied assumptions. This has attracted criticism [KM07, KM10], as the value of a security proof is unclear, when it is by reduction from an (often newly introduced) assumption that is not well understood. A sanity check that is considered a minimum requirement for assumptions in cyclic groups is a proof in the GGM, which guarantees that there are no efficient solvers that work for any group.

In this work we give the first proof that the *one-more discrete logarithm assumption*, a widely used hardness assumption, holds in the GGM. While prior proofs in the GGM have followed a common blueprint, the nature of OMDL differs from that of other assumptions and its proof requires a different approach, which we propose in this paper. We then extend our proof so that it also covers the *one-more Diffie-Hellman assumption*.

OMDL. The one-more discrete logarithm problem, introduced by Bellare et al. [BNPS03], is an extension of the discrete logarithm (DL) problem. Instead of being given one group element X of which the adversary must compute the discrete logarithm w.r.t. some basis G, for OMDL the adversary can ask for as many challenges X_i as it likes. Moreover, it has access to an oracle that returns the discrete logarithm of any group element submitted by the adversary. The adversary's goal is to compute the DL of all challenges X_i, of which there must be *one more* than the number of DL oracle calls it made.

Applications of OMDL

Security of blind signatures. Blind signature schemes [Cha82] let a user obtain a signature from a signer without the latter learning the message it signed. Their security is formalized by *one-more unforgeability*, which requires that after q signing interactions with the signer, the user should not be able to compute signatures on more than q messages.

The signatures in the *blind Schnorr signature scheme* [CP93] are standard Schnorr signatures [Sch91], which, in the form of EdDSA [BDL+12] are increasingly used in practice and considered for standardization by NIST [NIS19]. They are now used in OpenSSL, OpenSSH, GnuPG and considered to be supported by Bitcoin [Wui18].

In this work, we give a security analysis for blind Schnorr signatures one-more unforgeabilily which relies on OMDL.

Multi-signatures. *Multi-signature schemes* [IN83] allow a group of signers, each having individual verification and signing keys, to sign a message on behalf of all of them via a single signature. In recent work, Nick et al. [NRS20] present a (concurrently secure) two-round multi-signature scheme called *MuSig2* (a variant of the *MuSig* scheme [MPSW19]), which they prove secure under the OMDL assumption. The resulting signatures are ordinary Schnorr signatures (under an *aggregated* verification key, which is of the same form as a key for Schnorr); they are thus fully compatible with blockchain systems already using Schnorr and will help ease scalability issues, as a single aggregate signature can replace a set of individual signatures to be stored on the blockchain.

Earlier, Bellare and Neven [BN06] instantiated another signature primitive called *transitive signatures* [MR02] assuming OMDL.

IDENTIFICATION SCHEMES. Bellare and Palacio [BP02] assume OMDL to prove that the Schnorr identification protocol is secure against active and concurrent attacks, and Gennaro et al. [GLSY04] use it for a batched version of the scheme. Bellare and Shoup [BS07] prove that the Schnorr identification scheme verifies *special soundness* under concurrent attack from OMDL. Bellare et al. [BNN04] assume OMDL to prove their ID-based identification protocol secure against impersonation under concurrent attacks.

NEGATIVE RESULTS. OMDL has also been assumed in proofs of impossibility results. Paillier and Vergnaud [PV05] prove that unforgeability of Schnorr signatures cannot be proven under the discrete logarithm assumption. In particular, they show that there is no algebraic reduction to DL in the standard model if OMDL holds. There are further results about the N-OMDL assumptions (OMDL limited to N challenges) which show that these assumptions are not equivalent to each other. It is done by considering algebraic white-box reduction [BMV08] or standard black-box reduction [Bro07]. These results convinced us furthermore to directly consider the strongest version of OMDL where the adversary chooses adaptively the number of challenges (called $*$-OM-DL in [Bro07]). Seurin [Seu12] shows that, assuming OMDL, the security bound for Schnorr signatures by Pointcheval and Stern [PS96b] using the forking lemma is optimal in the ROM under the DL assumption. More precisely, the paper shows that if the OMDL assumption holds, then any algebraic reduction of Schnorr signatures must lose the same factor as a proof via the forking lemma. Fischlin and Fleischhacker [FF13] generalize this impossibility result to a large class of reductions, they call the single-instance reductions, again assuming OMDL.

Finally, Drijvers et al. [DEF+19] show under the OMDL assumption that many multi-signature schemes, namely *CoSi* [STV+16], *MuSig* [MPSW19], *BCJ* [BCJ08] and *MWLD* [MWLD10], cannot be proven secure from DL or OMDL. Many other works prove negative results about the security of Schnorr signatures as long as OMDL holds. [GBL08, FF13, FJS14, FH21].

The Generic Security of OMDL

Despite its wide use, surprisingly, OMDL is lacking of rigorous analysis. So far, OMDL has only been compared to the other DL assumptions [KM08]. OMDL assumption is trivially easier to break than DL. Using the index calculus algorithm, the OMDL assumption seems to be strictly easier to break than DL in jacobian groups of genus 3 or more [KM08]. The only analysis in the GGM is a relatively recent proof sketch by Coretti, Dodis, and Guo [CDG18, eprint version], which we show is flawed.[1] (The authors confirmed this.)

Their analysis follows the blueprint of earlier GGM proofs, which goes back to Shoup's [Sho97] proof of the hardness of DL in the GGM. However, as we explain below, the adversary can make their simulation of the GGM OMDL game fail with overwhelming probability. The particularity of OMDL compared to other assumptions, which lend themselves more easily to a GGM proof, is that via its DL oracle, the adversary can obtain information about the secret values chosen by the experiment.

Bauer et al. [BFL20] have recently given further evidence that the analysis of the generic security of OMDL must differ from that of other assumptions. They show that in the algebraic group model, a large class of assumptions, captured by an extension of the *uber assumption* framework [BBG05, Boy08], is implied by the hardness a parametrized discrete-logarithm problem: in q-DLog the adversary is given $(xG, x^2G, \ldots, x^qG)$ and must find x. While in the AGM q-DLog implies assumptions as diverse as the strong Diffie-Hellman [BB08], the Gap Diffie-Hellman [OP01], and the

[1] The authors study the security of assumptions (including OMDL) and schemes in an extension of the generic group model that models preprocessing attacks. They give a proof sketch for the security of OMDL with preprocessing. While we show that their sketch is flawed (see p. 7), their preprocessing techniques can be adapted to our proof. Thus their result for OMDL in the preprocessing GGM still holds, except for a change of bounds.

LRSW assumption [LRSW99], this is not the case for OMDL. Using the meta-reduction technique, Bauer et al. [BFL20] show that it is impossible to prove OMDL from q-DLog, for any q, in the AGM.

Challenges in the GGM proof of OMDL. We proceed like Shoup [Sho97] in that we replace all challenges x_i in the game by corresponding polynomials $\mathsf{X}_i \in \mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$. It seems tempting to then deduce, like for DL, that the probability that $P(x_1, \ldots, x_n) = Q(x_1, \ldots, x_n)$ for any $P \neq Q$ generated during the game is at most $\frac{1}{p}$ by Schwartz-Zippel. (This is what Coretti et al. [CDG18] do in their proof sketch.) This however ignores the fact that, via the discrete logarithm oracle $\mathrm{DLog}(\cdot)$, the adversary can obtain (a lot of) information on the challenges x_i and thereby easily cause such collisions. In more detail, such a straightforward proof has the following issues:

First, in the game simulated via polynomials, the adversary's oracle $\mathrm{DLog}(\cdot)$ must be simulated carefully. For example, suppose the adversary asks for the discrete logarithm of the first challenge by making the query $\mathrm{DLog}(\Xi(\mathsf{X}_1))$. Since the challenge x_1 is not defined yet, the challenger samples it randomly and gives it to the adversary. However, if the adversary later asks for $\Xi(\mathsf{X}_1 + 1)$ (via its group-operation oracle) and queries DLog on it, it expects the answer $x_1 + 1$, and not a random value. (In [CDG18], the DLog oracle always returns random values; the adversary can thus trivially decide that it is not playing the OMDL game in the GGM.)

Second, there is a more subtle issue. Again suppose that the adversary queried $\mathrm{DLog}(\Xi(\mathsf{X}_1))$ and was given the answer x_1. Let $P := \mathsf{X}_1$. Using the group-operation oracle, the adversary can compute (an encoding of) the constant polynomial $Q := x_1$, that is, it can obtain $\Xi(Q)$. Since $P(x_1) = Q(x_1) = x_1$, this means that the adversary can construct two polynomials P and Q such that $P(x_1, \ldots, x_n) = Q(x_1, \ldots, x_n)$ and $P \neq Q$. These situation can not occur in GGM proofs of other assumptions, because as long as there is no simulation failure, the adversary's polynomials are *independent* of $\vec{x}$. This is the reason that one can apply Schwartz-Zippel (SZ) in the end.

In summary, this use of SZ is not possible for OMDL (although [CDG18] uses it) because the adversary can obtain information on the challenge $(x_1, \ldots, x_n)$ even when there is no simulation failure, namely from its oracle DLog.

All these issues persist when using Maurer's model [Mau05], which is an abstraction of Shoup's GGM model in which all (logarithms of) group elements remain in a "black box". The adversary can ask for creating new entries in the box that are the sum of existing entries, or for values of its choice. For OMDL one would have to extend the model and allow the adversary to obtain values from the box to implement a DLog oracle. In proofs in this model [Mau05], the adversary wins if it creates a collision between values in the black box (which is what lets Maurer assume *non-adaptive* adversaries). However, an OMDL adversary can easily create collisions (e.g., get x_1 from the DLog oracle, then insert the constant x_1 into the black box).

Our GGM proof of OMDL In our proof of the hardness of OMDL in the GGM we follow the overall strategy of simulating the game using polynomials, but we take into account the issues just described. That is, the challenger monitors what the adversary has learned about the challenge and defines the simulation considering this knowledge, so the adversary cannot trivially distinguish the real game from the simulation. Of course, their might still be simulation failures due to "bad luck", which corresponds to the event that previous proofs bound via Schwartz-Zippel. As our simulation is quite different, we propose a new lemma that precisely corresponds to the situation in OMDL. That is, it bounds the probability that our (more complex) simulation fails.

Our strategy is similar to how Yun [Yun15] analyzed the generic security of the *multiple discrete logarithm* assumption, where the adversary must solve multiple DL challenges (but is not given a DLog oracle, which is what causes all the complication of the OMDL proof). Like Yun, we formalize

the knowledge about the challenge that adversary accumulates by affine hyperplanes in $\mathbb{Z}_p^n$. Due to the DLog oracle, this formalization is more complex for OMDL.

One might wonder if it was possible to still rely on the Schwartz-Zippel lemma (SZ) for proving OMDL, which would be the obvious approach. We have already argued that applying it once and at the end of the game, as in previous proofs, is not possible. But can SZ be applied before the end of the game?

A first idea could be to apply SZ at each call to the DLog oracle, but it does not work. Consider a call $\text{DLog}(\Xi(\mathsf{X}_1 + \mathsf{X}_2))$ answered with a uniform $v \leftarrow \mathbb{Z}_p$. One could now replace the variable X_1 by the expression $\mathsf{X}_2 - v$ in all polynomials P generated so far and use SZ to bound the probability that this creates a collision. One problem is that, P being a multivariate polynomial, SZ does not directly imply a bound on $\Pr[P(\mathsf{X}_2 - v, \mathsf{X}_2, \ldots, \mathsf{X}_n) = 0]$. Indeed, $P(\mathsf{X}_2 - v, \mathsf{X}_2, \ldots, \mathsf{X}_n)$ is the evaluation of the polynomial $\hat{P}(\mathsf{X}_1) := P(\mathsf{X}_1, \mathsf{X}_2, \ldots, \mathsf{X}_n)$ for $\mathsf{X}_1 = \mathsf{X}_2 - v$, so we need to bound $\Pr[\hat{P}(\mathsf{X}_2 - v) = 0]$ for a polynomial $\hat{P}$ with coefficients in the ring $\mathbb{Z}_p[\mathsf{X}_2, \ldots, \mathsf{X}_n]$. But SZ is defined for polynomials over fields. More generally, when the query $\text{DLog}(\Xi(P(\mathsf{X}_1, \ldots, \mathsf{X}_n))$ involves a more complex polynomial than $P = \mathsf{X}_1 + X_2$ then the substitution of one variable by a linear expression of the other is even more cumbersome to describe notationally. In our proof, these problems are avoided by replacing SZ with a lemma appropriate for OMDL.

Another idea would be to apply SZ each time a new encoding is computed. Indeed, assuming no collisions have occurred so far, then one could use SZ to bound the probability that the new encoding introduces a collision, and then proceed by induction.

But the resulting proof would require one game hop for every newly computed encoding: The j-th hybrid of this game would be the one in which the j first encodings are chosen all different independently of the real value of the challenge. The challenge $\overrightarrow{x}$ is picked by the game just before the $(j+1)$-th encoding, corresponding to the polynomial P_{j+1}, is defined. Using SZ, we can show that the probability that $P_{j+1}(\overrightarrow{x}) = P_i(\overrightarrow{x})$ for all $i \leq j$ is negligible.

But in fact, we need to be more cautious. When the adversary queries $\text{DLog}(\Xi(\mathsf{X}_1))$ and obtains x_1, to prevent the attack where the adversary generates the constant polynomial $P_{j+1} = x_1$, we need to adapt all polynomials so far defined to reflect the information revealed by the oracle Dlog. In this example, this is easy to formalize: update every polynomial by evaluating X_1 on x_1 and replace $P_k(x_1, \mathsf{X}_2, \ldots, \mathsf{X}_n)$ by some $P'_k(\mathsf{X}_2, \ldots, \mathsf{X}_n)$; the updated challenge $\overrightarrow{x}$ would be of size $n-1$. To generalize this, we would have to apply an affine transformation to all the variables of the polynomials at each call to $\text{DLog}()$. After as many game hops as there are queries by the adversary, we would arrive at a game in which all the encodings are random and the challenge is defined after the adversary gives its output.

We believe that both approaches just sketched lead to more complicated proofs than the one we give. In our proof, in the first game hop we abort if our simulation fails and we bound this probability by our new lemma. The remaining 3 game hops are purely syntactical, in that they do not change the adversary's winning probability.

One-More CDH

Another "one-more" assumption is the *one-more computational Diffie-Hellman* assumption [BNN04], also known as 1-*MDHP* [KM08, KM10], which is very similar to the *chosen-target CDH* assumption [Bol03]. In this problem, the adversary receives q pairs of group elements (X, Y_i), which all have the same first component $X = xG$, and its task is to compute xY_i for all i, for which it is provided an oracle $\text{CDH}_1()$ that computes xY for any Y of the adversary's choice. As for OMDL, the number of queries must be less than the number of challenges.

It turns out that this assumption can be proved to hold in the generic group model using standard techniques. Following the original GGM proof of DL [Sho97], we modify the simulation for the adversary from encoding logarithms to encoding polynomials in $\mathbb{Z}_p[\mathsf{X}, \mathsf{Y}_1, \ldots, \mathsf{Y}_n]$. The

challenges that the adversary receives are the monomials $\mathsf{X}, \mathsf{Y}_1, \ldots, \mathsf{Y}_n$, and when the adversary queries its oracle $\mathrm{CDH}_1()$ on an encoding that corresponds to a polynomial P, it receives an encoding corresponding to $\mathsf{X}P$, i.e., its polynomial multiplied by the indeterminate X. To win this "ideal" game, the adversary must construct encodings that correspond to $(\mathsf{XY}_1, \ldots, \mathsf{XY}_n)$. Making q calls to its $\mathrm{CDH}_1()$ oracle and using its group-operation oracle, it can only construct (encodings of) polynomials from $\mathsf{Span}(1, \mathsf{X}, \mathsf{Y}_1, \ldots, \mathsf{Y}_n, \mathsf{X}P_1, \ldots, \mathsf{X}P_q)$.

Ignoring polynomials of degree less than 2, the adversary wins the game if $\mathsf{Span}(\mathsf{XY}_1, \ldots, \mathsf{XY}_n) \subseteq \mathsf{Span}(\mathsf{X}P_1, \ldots, \mathsf{X}P_q)$. But it also has to satisfy the second winning condition, namely to solve more challenges than the number of its $\mathrm{CDH}_1()$ oracle queries; that is $q < n$. Using a dimension argument, we deduce that the above condition cannot be satisfied, and thus the adversary cannot win this game.

This "ideal" game is indistinguishable from the original one if the adversary does not create two distinct polynomials that agree on $x, y_1, \ldots, y_n$ (the discrete logarithms of $X, Y_1, \ldots, Y_n$). Because the degree of all polynomials is upper-bounded by $q + 1$, we can use the Schwartz-Zippel Lemma (as, e.g., done in [Boy08]) to upper-bound the statistical distance between the two games by $\mathcal{O}\left(\frac{(q+1)(m+q)^2}{p}\right)$, where m is the number of group operations made by the adversary. We can therefore derive the generic security of this assumption. (Another way of obtaining this result is by casting the assumption as an uber-assumption in the algebraic group model and applying [BFL20, Theorem 4.1].)

The situation is very different for a variant of the above problem, in which the first component of the challenge pairs is not fixed. That is, the adversary can request challenges, which are random pairs (X_i, Y_i) and is provided an oracle CDH(), which on input any pair $(X = xG, Y)$ returns the CDH solution of X and Y, that is xY. The adversary must compute the CDH solutions of the challenge pairs while making fewer queries to CDH(). In this paper we will refer to this assumption as OMCDH.

For this problem the standard proof methodology in the GGM fails. On a very high level, the reason is the following. Providing the adversary with an oracle $\mathrm{CDH}_1()$, as in the one-more Diffie-Hellman assumption with one component fixed (or a DLog oracle in OMDL) lets the adversary only construct polynomials of degree at most $q + 1$. In contrast, the CDH() oracle in OMCDH leads to a multiplication of the degrees, which enables the adversary to "explode" the degrees and make arguments à la Schwartz-Zippel impossible, since they rely on low-degree polynomials.

There is however, a neat way around this problem, namely to prove the following, *stronger* assumption: as in OMCDH, the adversary still has to compute CDH solutions, but now it is given a discrete-logarithm oracle. This hybrid assumption implies both OMDL (for which the goal is harder) and OMCDH (in which the oracle is less powerful) and we prove it in the GGM by extending our proof of OMDL.

1.3 Blind Schnorr Signature

Schnorr Signatures. The Schnorr signature scheme [Sch90, Sch91] is one of the oldest and simplest signature schemes based on prime-order groups. Its adoption was hindered for years by a patent which expired in February 2008, but it is by now widely deployed: EdDSA [BDL+12], a specific instantiation based on twisted Edward curves, is used for example in OpenSSL, OpenSSH, GnuPG and more. Schnorr signatures are also expected to be implemented in Bitcoin [Wui18], enabling multi-signatures supporting public key aggregation, which will result in considerable scalability and privacy enhancements [BDN18, MPSW19].

The security of the Schnorr signature scheme has been analyzed in the random oracle model (ROM) [BR93], an idealized model which replaces cryptographic hash functions by truly random functions. Pointcheval and Stern [PS96b, PS00] proved Schnorr signatures secure in the ROM under

the discrete logarithm assumption (DL). The proof, based on the so-called Forking Lemma, proceeds by rewinding the adversary, which results in a loose reduction (the success probability of the DL solver is a factor q_{h} smaller than that of the adversary, where q_{h} is the number of the adversary's random oracle queries). Using the "meta reduction" technique, a series of works showed that this security loss is unavoidable when the used reductions are either algebraic [PV05, GBL08, Seu12] or generic [FJS19]. Although the security of Schnorr signatures is well understood (in the ROM), the same cannot be said for two related schemes, namely blind Schnorr signatures and Schnorr-signed ElGamal encryption.

Blind Schnorr Signatures. A blind signature scheme allows a user to obtain a signature from a signer on a message m in such a way that (i) the signer is unable to recognize the signature later (*blindness*, which in particular implies that m remains hidden from the signer) and (ii) the user can compute one single signature per interaction with the signer (*one-more unforgeability*). Blind signature schemes were introduced by Chaum [Cha82] and are a fundamental building block for applications that guarantee user anonymity, e.g. e-cash [Cha82, CFN90, OO92, CHL05, FPV09], e-voting [FOO93], direct anonymous attestation [BCC04], and anonymous credential [Bra94, CL01, BCC+09, BL13a, Fuc11].

Constructions of blind signature schemes range from very practical schemes based on specific assumptions and usually provably secure in the random oracle model [PS96a, PS00, Abe01, Bol03, FHS15, HKL19] to theoretical schemes provably secure in the standard model from generic assumptions [GRS+11, BFPV13, GG14].

The blind Schnorr signature scheme derives quite naturally from the Schnorr signature scheme [CP93]. It is one of the most efficient blind signature schemes and increasingly used in practice. Anticipating the implementation of Schnorr signatures in Bitcoin, developers are already actively exploring the use of blind Schnorr signatures for *blind* coin swaps, trustless tumbler services, and more [Nic19].

While the hardness of computing discrete logarithms in the underlying group $\mathbb{G}$ is obviously necessary for the scheme to be unforgeable, Schnorr [Sch01] showed that another problem that he named ROS, which only depends on the order p of the group $\mathbb{G}$, must also be hard for the scheme to be secure. Informally, the ROS_ℓ problem, parameterized by an integer ℓ, asks to find $\ell+1$ vectors $\vec{\rho}_i = (\rho_{i,j})_{j\in[\ell]}$ such that the system of $\ell+1$ linear equations in unknowns $c_1,\ldots,c_\ell$ over $\mathbb{Z}_p$

$$\textstyle\sum_{j=1}^{\ell} \rho_{i,j} c_j = \mathsf{H}_{\mathrm{ros}}(\vec{\rho}_i)\ , \quad i \in [\ell+1]$$

has a solution, where $\mathsf{H}_{\mathrm{ros}}\colon (\mathbb{Z}_p)^\ell \to \mathbb{Z}_p$ is a random oracle. Schnorr showed that an attacker able to solve the ROS_ℓ problem can produce $\ell+1$ valid signatures while interacting (concurrently) only ℓ times with the signer. Slightly later, Wagner [Wag02] showed that the ROS_ℓ problem can be reduced to the $(\ell+1)$-sum problem, which can solved with time and space complexity $O((\ell+1)2^{\lambda/(1+\lfloor \lg(\ell+1)\rfloor)})$, where λ is the bit size of p. For example, for $\lambda = 256$, this attack yields 16 valid signatures after $\ell = 15$ interactions with the signer in time and space close to 2^{55}. For $\ell+1 = 2^{\sqrt{\lambda}}$, the attack has sub-exponential time and space complexity $O(2^{2\sqrt{\lambda}})$, although the number of signing sessions becomes arguably impractical. Asymptotically, this attack can be thwarted by increasing the group order, but this would make the scheme quite inefficient. Benhamouda et al. [BLOR20] recently presented a polynomial-time solver for ROS. This leads to forgeries of blind Schnorr signatures when the attacker is allowed to run *concurrent* executions of the signing protocol.

From a provable-security point of view, a number of results [FS10, Pas11, BL13b] indicate that blind Schnorr signatures cannot be proven one-more unforgeable under standard assumptions, not even in the ROM. The only positive result by Schnorr and Jakobsson [SJ99] and Schnorr [Sch01]

states that blind Schnorr signatures are secure in the combination of the generic group model and the ROM assuming hardness of the ROS problem.

The recent analysis by Hauck, Kiltz, and Loss [HKL19] of blind signatures derived from linear identification schemes does not apply to Schnorr. The reason is that the underlying linear function family $F\colon \mathbb{Z}_p \to \mathbb{G}, x \mapsto xG$ lacks the property of having a pseudo torsion-free element from the kernel (see [HKL19, Definition 3.1]). In particular, F is one-to-one, whereas Hauck et al. reduce blind signature unforgeability to collision resistance of the underlying function family.

Our Results on Blind Schnorr Signatures. Our starting point is the observation that in the combination[2] AGM+ROM Schnorr signatures have a *tight* security proof under the DL assumption. This is because we can give a reduction which works *straight-line*, i.e., unlike the forking-lemma-based reduction [PS96b, PS00], which must rewind the adversary, it runs the adversary only once.[3] Motivated by this, we then turn to blind Schnorr signatures, whose security in the ROM remains elusive, and study their security in the AGM+ROM.

Our first contribution is a rigorous analysis of the security of blind Schnorr signatures in the AGM+ROM. Concretely, we show that any algebraic adversary successfully producing $\ell+1$ forgeries after at most ℓ interactions with the signer must either solve the one-more discrete logarithm (OMDL) problem or the ROS_ℓ problem. Although this is not overly surprising in view of the previous results in the GGM [SJ99, Sch01], this gives a more satisfying characterization of the security of this protocol. Moreover, all previous proofs [SJ99, Sch01] were rather informal; in particular, the reduction solving ROS was not explicitly described. In contrast, we provide precise definitions (in particular for the ROS problem, whose exact specification is central for a security proof) and work out the details of the reductions to both OMDL and ROS, which yields the first rigorous proof.

Nevertheless, the serious threat by Wagner's attack for standard-size group orders remains. In order to remedy this situation, we propose a simple modification of the scheme which only alters the signing protocol (key generation and signature verification remain the same) and thwarts (in a well-defined way) any attempt at breaking the scheme by solving the ROS problem. The idea is that the signer and the user engage in two parallel signing sessions, of which the signer only finishes one (chosen at random) in the last round. Running this tweak takes thus around twice the time of the original protocol. We show that an algebraic adversary successfully mounting an $(\ell+1)$-forgery attack against this scheme must either solve the OMDL problem or a *modified* ROS problem, which appears much harder than the standard ROS problem for large values of ℓ, which is precisely when the standard ROS problem becomes tractable.

Our results are especially relevant to applications that impose the signature scheme and for which one then has to design a blind signing protocol. This is the case for blockchain-based systems where modifying the signature scheme used for authorizing transactions is a heavy process that can take years (if possible at all). We see a major motivation for studying blind Schnorr signatures in its real-world relevance for protocols that use Schnorr signatures or will in the near future, such as Bitcoin. For these applications, Wagner's attack represents a significant risk, which can be thwarted by using our modified signing protocol.

Chosen-Ciphertext-Secure ElGamal Encryption. Recall the ElGamal public-key encryption (PKE) scheme [ElG85]: given a cyclic group $(\mathbb{G}, +)$ of prime order p and a generator G, a secret/public key pair is of the form $(y, yG) \in \mathbb{Z}_p \times \mathbb{G}$. To encrypt a message $M \in \mathbb{G}$, one draws $x \xleftarrow{\$} \mathbb{Z}_p$, computes $X := xG$, and outputs ciphertext $(X, M + xY)$. This scheme is IND-CPA-secure

[2] This combination of idealized models was already considered when the AGM was first defined [FKL18].

[3] A similar result [ABM15] shows that Schnorr signatures, when viewed as non-interactive proofs of knowledge of the discrete logarithm of the public key, are simulation-sound extractable, via a straight-line extractor. Our proof is much simpler and gives a concrete security statement.

under the decisional Diffie-Hellman (DDH) assumption [TY98], that is, no adversary can distinguish encryptions of two messages. Since the scheme is homomorphic, it cannot achieve IND-CCA2 security, where the adversary can query decryptions of any ciphertext (except of the one it must distinguish). However, ElGamal has been shown to be IND-CCA1-secure (where no decryption queries can be made after receiving the challenge ciphertext) in the AGM under a "q-type" variant of DDH [FKL18].[4]

A natural way to make ElGamal encryption IND-CCA2-secure is to add a proof of knowledge of the randomness x used to encrypt. Intuitively, this makes the scheme *plaintext-aware* [BR95], which informally means that for any adversary producing a valid ciphertext, there exists an extractor that can retrieve the corresponding plaintext. The reduction of IND-CCA2 security can then use the extractor to answer the adversary's decryption queries. (For ElGamal, the extractor would extract the randomness x used to produce $(X = xG, C = M + xY)$ from the proof of knowledge and return the plaintext $M = C - xY$.) Since the randomness x together with the first part X of the ciphertext form a Schnorr key pair, a natural idea is to use a Schnorr signature [Jak98, TY98], resulting in what is usually called (Schnorr-)signed ElGamal encryption. This scheme has a number of attractive properties: ciphertext validity can be checked without knowledge of the decryption key, and one can work homomorphically with the "core" ElGamal ciphertext (a property sometimes called "submission-security" [Wik08]), which is very useful in e-voting.

Since Schnorr signatures are extractable in the ROM, one would expect that signed ElGamal can be proved IND-CCA2 under, say, the DDH assumption (in the ROM). However, turning this intuition into a formal proof has remained elusive. The main obstacle is that Schnorr signatures are not *straight-line* extractable in the ROM [BNW17]. As explained by Shoup and Gennaro [SG02], the adversary could order its random-oracle and decryption queries in a way that makes the reduction take exponential time to simulate the decryption oracle.

Schnorr and Jakobsson [SJ00] showed IND-CCA2 security in the GGM+ROM, while Tsiounis and Yung [TY98] gave a proof under a non-standard "knowledge assumption", which amounts to assuming that Schnorr signatures are straight-line extractable. On the other hand, impossibility results tend to indicate that IND-CCA2 security cannot be proved in the ROM [ST13, BFW16].

Our Results on Signed ElGamal Encryption. Our second line of contributions is twofold. First, we prove (via a tight reduction) that in the AGM+ROM, Schnorr-signed ElGamal encryption is IND-CCA2-secure under the DDH assumption. While intuitively this should follow naturally from the straight-line extractability of Schnorr proofs of knowledge for algebraic adversaries, the formal proof is technically quite delicate: since messages are group elements, the "basis" of group-element inputs in terms of which the adversary provides representations contains not only the three group elements of the challenge ciphertext but also grows as the adversary queries the decryption oracle.[5]

We finally consider the "hashed" variant of ElGamal (also known as DHIES) [ABR01], in which a key is derived as $k = \mathsf{H}(xY)$. In the ROM, the corresponding key-encapsulation mechanism (KEM) is IND-CCA2-secure under the strong Diffie-Hellman assumption (which states that CDH is hard even when given a DDH oracle) [CS03]. We propose to combine the two approaches: concretely, we consider the hashed ElGamal KEM together with a Schnorr signature proving knowledge of the randomness used for encapsulating the key and give a *tight* reduction of the IND-CCA2 security of this scheme to the DL problem in the AGM+ROM.

[4] [FKL18] showed IND-CCA1 security for the corresponding key-encapsulation mechanism, which returns a key $K = xY$ and an encapsulation of the key $X = xG$. The ElGamal PKE scheme is obtained by combining it with the one-time-secure data-encapsulation mechanism $M \mapsto M + K$. Generic results on hybrid schemes [HHK10] imply IND-CCA1 security of the PKE.

[5] Bernhard et al. [BFW16] hastily concluded that, in the AGM+ROM, IND-CCA2-security of signed ElGamal followed from straight-line extractability of Schnorr signatures showed in [ABM15]. Our detailed proof shows that this was a bit optimistic.

1.4 Other work

In this document, I do not discuss a major part of my work during my Phd which concerns the collaboration I had with Melissa Chase and Esha Ghosh from MSR.

This work aims to extend the principle of anonymous credentials in a more complicated setting than usual. Anonymous Credentials is a protocol in which a **user** get some credential (for example the id card) from a signing authority (for example the State) and wants to prove to a **verifier** that he is allowed to get access to some resource (for example, you can show your id card to prove you're a citizen of the country so that you can vote.) A trivial way of proving that you can get an access on the resource is to show your credential to the verifier, but this is not anonymous at all (for example your id card shows more information than only the fact that you're allowed to vote. It reveals your age, your name, etc...).

To make the process anonymous we usually use a zero knowledge proofs of knowledge, which proves that the user owns a valid credential. By doing this, we can show to a verifier that some users are allowed get access to the resources without revealing anything more. To enforce the scheme, we can also make the signing authority provide credential by using some blind signatures procedure, so that they cannot collude with the verifier to deduce the user's identity.

In credential transparency, we consider the situation in which the users give their credential to an authority called the **Cloud**. Since only the Cloud is able to manipulate the users' credentials in this setting, it is also the Cloud which shows the credential to the verifier so that a user can get an access a resource.

Thus we now need to consider what the Cloud could be mischievous. For example, a Cloud which has no access to a resource could use one user's credential to authenticate anonymously to a verifier. Thus we want the users to be aware of how their credentials are used by the Cloud. We want to make the Cloud accountable for all the use of credential that it does.

In this work, we designed a scheme we called credential transparency inspired from some already existing constructions such as certificate transparency, Coniks or SeemLess. The aim is to allow the auditing of the Cloud's behaviour so that we don't need to trust it. The main idea behind those is that the authority will have to maintain a log of all the actions it did. In credential transparency, the Cloud will have to update the log for every time it uses a credential to authenticate to a verifier. The logs are accumulators (for example merkle trees) that provide an efficient way of verifying that an element is included inside (proof of inclusion) and provide privacy. Every update of such a log by the cloud is released together with a snapshot and proof of update provided so that external parties called **auditors** can verify that the update was computed correctly. Since those snapshot are public, everyone who wants to check the inclusion of some information in the log can check that the snapshot they refer to is the same for everyone: the cloud cannot cheat.

With Melissa and Esha, we designed the credential transparency protocol and proved its security. Our scheme uses a lot of cryptographic primitives, such as Strong Accumulator (SA), append-only zero knowledge set (aZKS), simulatable Verifiable Pseudorandom Functions (sVRF), simulatable Commitment, and Zero Knowledge (ZK) scheme.

Chapter 2

Preliminaries

In this chapter, we introduce the notation and basic assumptions and primitives employed throughout this manuscript. We start by recalling some standard mathematical and computational notions, then we briefly introduce provable security. We also recall some well-known number-theoretic assumptions.

2.1 General Notation

INTEGERS, SETS, MODULUS: In this document, $\mathbb{R}$ is the set of real numbers, $\mathbb{Z}$ is the set of integers and $\mathbb{N}$ the set of positive integers. If $a, b \in \mathbb{Z}$ with $a < b$, we denote the closed integer interval from a to b by $[a, b]$.

If $p \in \mathbb{N}$ we call $\mathbb{Z}_p$ the ring of integers modulus p. We represent $\mathbb{Z}_p$ by the elements of $[0, p-1]$. Elements of $\mathbb{Z}_p$ can also be seen as the remainders of the integers in $\mathbb{Z}$ after Euclidian division by p. If a and b are equal modulo p, meaning that there exist $q, q' \in \mathbb{Z}$ such that $a - qp = b - q'p \in [0, p-1]$, we write $a \equiv_p b$ or $a \equiv b \pmod{p}$. If p is a prime then $\mathbb{Z}_p$ has the structure of a field.

If S is a set and $n \in \mathbb{N}$, $n > O$, a tuple of elements of S is noted $(s_1, \ldots, s_n)$. The set of tuples is denoted S^n. We use the vector notation $\overrightarrow{s}$ to represent a list of elements of S, meaning that there exists a unique $n \in \mathbb{N}$, such as $\overrightarrow{s} \in S^n$. We call n the size of $\overrightarrow{s}$ and we denote $|\overrightarrow{s}| = n$. We also use the notation $\overrightarrow{s} = (s_i)_{i=1}^n$. The empty list is denoted $()$. If S is a finite set of elements we call $|S|$ the cardinal of (number of elements in) S.

Thus $|\mathbb{Z}_p| = p$. Throughout this work, p will always part of the public parameters.

CYCLIC GROUPS: Every group of prime order (cardinal) p is cyclic and thus is homomorphic to $\mathbb{Z}_p$. In this work we call $\mathbb{G}$ such a group with neutral element $0_{\mathbb{G}} \in \mathbb{G}$ (or 0 when there is no possible confusion). We use additive notation, meaning that if $X_1, X_2 \in \mathbb{G}$ we have $X_1 + X_2 \in \mathbb{G}$ and $-X_1 \in \mathbb{G}$. if $n \in \mathbb{N}$ and $x \in \mathbb{G}$ we define $nX \in \mathbb{G}$ as the sum of n times the group element X. Since $\mathbb{G}$ has order p, if r is the remainder of n in the Euclidian division by p, we get $nX = rX$, thus we can define rX with $r \in \mathbb{Z}_p$. A *generator* G of $\mathbb{G}$ is a group element such that for any group element $X \in \mathbb{G}$, there exists a unique $x \in \mathbb{Z}_p$ such that $X = xG$. Since the order of $\mathbb{G}$ is a prime p, every element of $\mathbb{G}$ which is not $0_{\mathbb{G}}$ is a generator.

Thus we define the *discrete logarithm* of an element $X \in \mathbb{G}$ with respect to the generator G by the unique $x \in \mathbb{Z}_p$ such that $X = xG$. We write $\log_G(X) = x$ or $\log(X) = x$ when there is no ambiguity on the generator G. We denote by $(p, \mathbb{G}, G)$ the tuple representing the order p of the group $\mathbb{G}$ and its generator G.

POLYNOMIALS: If A is a ring, a polynomial $P \in A[\mathsf{X}]$ is represented as $P(\mathsf{X}) = \sum_{i=0}^n a_i \mathsf{X}^i$, with $(a_i)_{i=0}^n \in A^{n+1}$. Since $A[\mathsf{X}]$ is also a ring we can define $A[\mathsf{X}_1, \mathsf{X}_2] = A[\mathsf{X}_1][\mathsf{X}_2]$ and by induction

$A[\mathsf{X}_1, \ldots, \mathsf{X}_n]$. Elements $P \in A[\mathsf{X}_1, \ldots, \mathsf{X}_n]$ are called multivariates polynomials. Every multivariate polynomial can be written as a unique linear combination of the monomials $\mathsf{X}_1^{i_1}\mathsf{X}_2^{i_2}\ldots\mathsf{X}_n^{i_n}$ which have by definition degree $i_1+i_2+\ldots+i_n$. The degree of P written $\deg(P)$ is equal to the maximum degree of the monomials that compose it. For multivariate polynomials $P \in A[\mathsf{X}_1, \ldots, \mathsf{X}_n]$ we write $\overrightarrow{\mathsf{X}} \coloneqq (\mathsf{X}_1, \ldots, \mathsf{X}_n)$ and $P(\overrightarrow{x}) \coloneqq P(x_1, \ldots, x_n)$ for $\overrightarrow{x} \in A^n$. In the whole document the ring A will always be $\mathbb{Z}_p$.

Vector spaces: $\mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$ has the structure of a $\mathbb{Z}_p$-vector space. In this document we consider vector subspaces of $\mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$: if $L = (P_1, \ldots, P_q)$ is a list of polynomials then $\mathsf{Span}(L) \coloneqq \{\sum_{i\in[1,q]} \alpha_i P_i \mid \overrightarrow{\alpha} \in \mathbb{Z}_p^q\}$ is the smallest vector space containing the elements of L. If $L = \emptyset$ then $\mathsf{Span}(L) = \{0\}$. If F is a vector subspace of $\mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$, we denote by $\mathsf{dim}(F)$ the dimension of F.

If E is a vector space, an *affine subspace* of E is a couple $\mathcal{A} = (x, F)$ with $x \in E$ and F a vectorial subspace of E. We denote by $\mathsf{dim}(\mathcal{A}) = \mathsf{dim}(F)$ the dimension of the affine space $\mathcal{A}$. An affine hyperplane is an affine subspace of dimension $\mathsf{dim}(E) - 1$.

For the Euclidian space $(\mathbb{Z}_p^n, \langle\cdot,\cdot\rangle)$ we write the scalar product of two elements $\overrightarrow{x}, \overrightarrow{y} \in \mathbb{Z}_p^n$, as $\langle\overrightarrow{x}, \overrightarrow{y}\rangle = \sum_{i\in[1,n]} x_i y_i$. In this work, polynomials are typically of degree 1, so we can write $P = \rho_0 + \sum_{i=1}^n \rho_i \mathsf{X}_i$ as a scalar product: $P(\overrightarrow{\mathsf{X}}) = \rho_0 + \langle\overrightarrow{P}, \overrightarrow{\mathsf{X}}\rangle$, where we define $\overrightarrow{P} \coloneqq (\rho_i)_{i\in[1,n]}$, that is the vector of *non-constant* coefficients of P.

Probabilities: For a random variable X and a possible outcome x, we write $\Pr[X = x]$ to denote the probability of the event $X = x$. Given a non-empty finite set S, we let $x \xleftarrow{\$} S$ denote the operation of sampling an element x from S uniformly at random, meaning that if X is a random variable sampled uniformly from S and $s \in S$, we have $\Pr[X = s] = \frac{1}{|S|}$.

The Schwarz-Zippel Lemma We introduce here the Schwarz Zippel Lemma, which we mention a lot in our work. [DL77]:

Lemma 2.1. *Let $P \in \mathbb{Z}_p[X_1, \ldots, X_m]$ be a non-zero polynomial of total degree d. Let $r_1, \ldots, r_m$ be selected at random independently and uniformly from $\mathbb{Z}_p^*$. Then*

$$\Pr\left[P(r_1, \ldots, r_m) \equiv_p 0\right] \leq \frac{d}{p-1}.$$

Function Bounds: Given a function $f : \mathbb{N} \to \mathbb{R}$, the set $O(f)$ describes all the functions that f dominates, meaning they can be upper-bounded by f. Namely, $g \in O(f)$ means that exists $M \in \mathbb{R}$ such that for all $\lambda \in \mathbb{N}$ we have $|g(\lambda)| \leq M|f(\lambda)|$. We say that g is *polynomially bounded* if and only if $g \in O(f)$, with f a polynomial function.

A function $\mu\colon \mathbb{N} \to [0, 1]$ is *negligible* (denoted $\mu = \mathsf{negl}$) if for all $c \in \mathbb{N}$ there exists $\lambda_c \in \mathbb{N}$ such that $\mu(\lambda) \leq \lambda^{-c}$ for all $\lambda \geq \lambda_c$. A function ν is *overwhelming* if $1 - \nu = \mathsf{negl}$.

Algorithms: Algorithms are Turing Machines. If $\mathcal{A}$ is an algorithm, the length function $\mathcal{A}.rl(\lambda)$ of $\mathcal{A}$ is a polynomially bounded function from $\mathbb{N}$ to $\mathbb{N}$ in λ defining the length of the randomness for a probabilistic interactive Turing Machine. In this work all the algorithms are probabilistic unless stated otherwise meaning that an algorithm $\mathcal{A}(x_1, \ldots, x_n; r)$ is run on input $(x_1, \ldots, x_n)$ with random coins $r \in \{0,1\}^{\mathcal{A}.rl(\lambda)}$. By $y \leftarrow \mathcal{A}(x_1, \ldots, x_n)$ we denote the operation of running algorithm $\mathcal{A}$ on input $(x_1, \ldots, x_n)$ and uniformly random coin and letting y denote the output. In the algorithm instantiation, we denote by $y \coloneqq x$ the attribution of the value x to the variable y. If $\mathcal{A}$ has oracle access to some algorithm Oracle, we write $y \leftarrow \mathcal{A}^{\text{Oracle}}(x_1, \ldots, x_n)$.

2.2 Security notions

SECURITY GAMES A *security game* $\text{GAME}_{par}(\lambda)$ indexed by a set of parameters *par* consists of a main procedure and a collection of oracle procedures. The main procedure, on input the security parameter λ, initializes variables and generates input on which an adversary $\mathcal{A}$ is run. The adversary interacts with the game by calling oracles provided by the game and returns some output, based on which the game computes its own output bit b, which we write $b \leftarrow \text{GAME}^{\mathcal{A}}_{par}(\lambda)$. We identify **false** with 0 and **true** with 1. Games are either computational or decisional. If the game is computationnal, the *advantage* of $\mathcal{A}$ in $\text{GAME}_{par}(\lambda)$ is defined as

$$\mathbf{Adv}^{GAME}_{par,\mathcal{A}} := \Pr[1 \leftarrow \text{GAME}^{\mathcal{A}}_{par}(\lambda)]$$

If the game is decisionnal, the advantage of of $\mathcal{A}$ in $\text{GAME}_{par}(\lambda)$ is defined as

$$\mathbf{Adv}^{GAME}_{par,\mathcal{A}} := 2 \cdot \Pr[1 \leftarrow \text{GAME}^{\mathcal{A}}_{par}(\lambda)] - 1$$

where the probability is taken over the random coins of the game and the adversary. We say that GAME_{par} is *hard* if for any probabilistic polynomial-time (p.p.t.) adversary if $\mathcal{A}$, $\mathbf{Adv}^{GAME}_{par,\mathcal{A}} = \mathsf{negl}(\lambda)$. All games considered in this paper are computational unless stated otherwise (we only consider decisional games in Section 7.1 and Section 7.3 and 6.4.)

ALGEBRAIC SECURITY An *algebraic security game* (with respect to GrGen) is a game $\text{GAME}_{\mathsf{GrGen}}$ that (among other things) runs $\Gamma \leftarrow \mathsf{GrGen}(1^\lambda)$ and runs the adversary on input $\Gamma = (p, \mathbb{G}, G)$. An algorithm $\mathcal{A}_{\text{alg}}$ executed in an algebraic game $\text{GAME}_{\mathsf{GrGen}}$ is *algebraic* if for all group elements Z that it outputs, it also provides a representation of Z relative to all previously received group elements: if $\mathcal{A}_{\text{alg}}$ has so far received $\overrightarrow{X} = (X_0, \ldots, X_n) \in \mathbb{G}^{n+1}$ (where by convention we let $X_0 = G$), then $\mathcal{A}_{\text{alg}}$ must output Z together with $\overrightarrow{z} = (z_0, \ldots, z_n) \in (\mathbb{Z}_p)^{n+1}$ such that $Z = \sum_{i=0}^{n} z_i X_i$. We let $Z_{[\overrightarrow{z}]}$ denote such an augmented output. When writing $\overrightarrow{z}$ explicitly, we simply write $Z_{[z_0,\ldots,z_n]}$ (rather than $Z_{[(z_0,\ldots,z_n)]}$) to lighten the notation.

ALGEBRAIC ALGORITHMS IN THE RANDOM ORACLE MODEL. We assume the existence of a p.p.t. algorithm GrGen which, on input the security parameter 1^λ in unary, outputs a group description $\Gamma = (p, \mathbb{G}, G)$ where p is of bit-length λ. The original paper [FKL18] considered the algebraic group model augmented by a random oracle and proved tight security of BLS signatures [BLS04] in the AGM+ROM model. The random oracle in that work is of type $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}$, and as the outputs are group elements, the adversary's group element representations could depend on them.

In this work we analyze Schnorr-type cryptosystems, for which the RO is typically of type $\mathsf{H}\colon \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_p$. Thus, an algebraic adversary querying H on some input (Z, m) must also provide a representation $\overrightarrow{z}$ for the group-element input Z. In a game that implements the random oracle by lazy sampling, to ease readability, we will define an auxiliary oracle $\widetilde{\mathsf{H}}$, which is used by the game itself (and thus does not take representations of group elements as input) and implements the same function as H.

THE ONE-MORE DISCRETE LOGARITHM PROBLEM. The discrete logarithm (DL) problem consists in finding the discrete logarithm of a group element. From a group description $(p, \mathbb{G}, G)$ we resolve the DL problem if for $X \in \mathbb{G}$, we find $\log_G(X)$ in a polynomial time. The security game representing the DL problem is represented in figure 2.1.

For a group description $(p, \mathbb{G}, G)$ with input $X, Y \in G$ with secret $x, y \in \mathbb{Z}_p$ such as $X = xG$ and $Y = yG$, the Computational Diffie Hellman (CDH) problem consist in finding the group element Z such as $Z = xyG$. The security game representing the CDH problem is represented in Figure 2.2.

Game $\mathrm{DL}^{\mathcal{A}}_{\mathsf{GrGen}}(\lambda)$	Game $\mathrm{OMDL}^{\mathcal{A}}_{\mathsf{GrGen}}(\lambda)$	Oracle $\textsc{Chal}()$	Oracle $\mathrm{DLog}(X)$
$(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$	$(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$	$x \xleftarrow{\$} \mathbb{Z}_p\,;\ X := xG$	$q := q + 1$
$x \xleftarrow{\$} \mathbb{Z}_p\,;\ X := xG$	$\overrightarrow{x} := (\,)\,;\ q := 0$	$\overrightarrow{x} := \overrightarrow{x} \,\|\, (x)$	$x := \log_G(X)$
$y \leftarrow \mathcal{A}(p, \mathbb{G}, G, X)$	$\overrightarrow{y} \leftarrow \mathcal{A}^{\textsc{Chal},\mathrm{DLog}}(p, \mathbb{G}, G)$	**return** X	**return** x
return $(y = x)$	**return** $\left(\overrightarrow{y} = \overrightarrow{x} \ \wedge\ q < \lvert\overrightarrow{x}\rvert\right)$		

Figure 2.1: The DL and OMDL problems.

Game $\mathrm{CDH}^{\mathcal{A}}_{\mathsf{GrGen}}(\lambda)$	Game $\mathrm{OMCDH}^{\mathcal{A}}_{\mathsf{GrGen}}(\lambda)$	Oracle $\textsc{Chal}()$	Oracle $\mathrm{CDH}(X, Y)$
$(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$	$(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$	$x \xleftarrow{\$} \mathbb{Z}_p\,;\ X := xG$	$q := q + 1$
$x, y \xleftarrow{\$} \mathbb{Z}_p$	$\overrightarrow{Z} := (\,)\,;\ q := 0$	$y \xleftarrow{\$} \mathbb{Z}_p\,;\ Y := yG$	$x := \log_G(X)$
$X := xG\,;\ Y := yG$	$\overrightarrow{V} \leftarrow \mathcal{A}^{\textsc{Chal},\mathrm{CDH}}(p, \mathbb{G}, G)$	$\overrightarrow{Z} := \overrightarrow{Z} \,\|\, (xyG)$	$y := \log_G(Y)$
$V \leftarrow \mathcal{A}(p, \mathbb{G}, G, X, Y)$	**return** $\left(\overrightarrow{Z} = \overrightarrow{V} \wedge q < \lvert\overrightarrow{Z}\rvert\right)$	**return** X, Y	**return** (xyG)
return $(V = xyG)$			

Figure 2.2: The CDH and OMCDH problems

The one more-discrete logarithm (OMDL) problem is an extension of the DL problem and consists in finding the discrete logarithm of n group elements by making strictly less than n calls to an oracle solving the discrete logarithm problem. The security game representing the OMDL problem is represented in figure 2.1. It was introduced in [BNPS03] and used for example to prove the security of the Schnorr identification protocol against active and concurrent attacks [BP02].

Equivalently, the one more-computational Diffie Hellmann (OMCDH) is an extension of the CDH problem and is represented in figure Figure 2.2

Chapter 3

One More-Discrete Logarithm security in the Generic Group Model

In this chapter, we give a security proof for the OMDL assumption in the GGM. Despite the intuition, we need more work than the usual GGM proofs to show that OMDL is secure in the GGM. First, we define a lemma on which our proof will be based. Then we give an intuition for the proof, and finally we build the security proof.

After that, we give an insight on how the same proof idea can cover the OMCDH assumption security and further assumptions. Those are proved in the appendices.

3.1 A Technical Lemma for OMDL in the GGM

While a standard argument in GGM proofs uses the Schwartz-Zippel lemma, this argument cannot be made for OMDL since in this game the adversary obtains information on the challenge $\overrightarrow{x}$ *not* only when the simulation fails. We therefore cannot argue that $\overrightarrow{x}$ looks uniformly random to the adversary, which is a precondition for applying Schwartz-Zippel.

We therefore use a different lemma, which bounds the probability that for a given polynomial P, we have $P(\overrightarrow{x}) = 0$ when $\overrightarrow{x}$ is chosen uniformly from a set $\mathcal{C}$. This set $\mathcal{C} \subseteq \mathbb{Z}_p^n$ represents the knowledge the adversary has on the challenge $\overrightarrow{x}$.

The Schwartz-Zippel lemma applies when $\mathcal{C} = S^n$ with S a subset of $\mathbb{Z}_p$, whereas our lemma is for the case that P has degree 1 and $\mathcal{C}$ is defined by an intersection of affine hyperplanes $\mathcal{Q}_j$ from which we remove other affine hyperplanes $\mathcal{D}_i$, that is $\mathcal{C} \coloneqq (\bigcap_{j\in[1,q]} \mathcal{Q}_j) \setminus (\bigcup_{i\in[1,m]} \mathcal{D}_i)$.

We start with some notations. Consider m polynomials $D_i \in \mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$ of degree 1, and $q+1$ polynomials $Q_j \in \mathbb{Z}_p[X_1, \ldots, \mathsf{X}_n]$ also of degree 1. We can write them as

$$D_i(\overrightarrow{\mathsf{X}}) = D_{i,0} + \sum_{k=1}^{n} D_{i,k}\mathsf{X}_k = D_{i,0} + \langle \overrightarrow{D_i}, \overrightarrow{\mathsf{X}} \rangle \tag{3.1}$$

with $\overrightarrow{D_i} \coloneqq (D_{i,k})_{1\leq k\leq n}$. We define the sets of roots of these polynomials, which are affine hyperplanes of $\mathbb{Z}_p^n$:

$$\begin{aligned} \forall i \in [1,m]: \ \mathcal{D}_i &= \{\overrightarrow{x} \in \mathbb{Z}_p^n \,|\, D_i(\overrightarrow{x}) = 0\} \\ \forall j \in [1,q+1]: \ \mathcal{Q}_j &= \{\overrightarrow{x} \in \mathbb{Z}_p^n \,|\, Q_j(\overrightarrow{x}) = 0\} \ . \end{aligned} \tag{3.2}$$

From (3.1), we see that the vector $\overrightarrow{D_i}$ of non-constant coefficients defines the direction of the hyperplane $\mathcal{D}_i$. It contains the coefficients of the polynomial $D_i - D_i(0) = \sum_{k=1}^{n} D_{i,k}\mathsf{X}_k$.

We define the set

$$\mathcal{C} \coloneqq \Big(\bigcap_{j\in[1,q]} \mathcal{Q}_j \Big) \setminus \Big(\bigcup_{i\in[1,m]} \mathcal{D}_i \Big) , \tag{3.3}$$

that is, the set of points at which all Q_i's vanish but none of the D_i's do. The following lemma will be the heart of our proofs of one-more assumptions in the GGM.

Lemma 3.1. *Let $D_1, \ldots, D_m, Q_1, \ldots, Q_{q+1} \in \mathbb{Z}_p[\mathsf{X}_1, \ldots, \mathsf{X}_n]$ be of degree 1; let $\mathcal{C}$ be as defined in (3.2) and (3.3). Assume $\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$ and $\overrightarrow{Q}_{q+1}$ is linearly independent of $(\overrightarrow{Q}_j)_{j\in[1,q]}$. If $\overrightarrow{x}$ is picked uniformly at random from $\mathcal{C}$ then*

$$\frac{p-m}{p^2} \leq \Pr\left[Q_{q+1}(\overrightarrow{x}) = 0\right] \leq \frac{1}{p-m} .$$

Proof

Since $\overrightarrow{x}$ is picked uniformly in $\mathcal{C}$, we have $\Pr[\overrightarrow{x} \in \mathcal{Q}_{q+1}] = \dfrac{|\mathcal{Q}_{q+1} \cap \mathcal{C}|}{|\mathcal{C}|}$.

We first bound $|\mathcal{C}|$. We define the affine space $\mathcal{Q} \coloneqq \bigcap_{j\in[1,q]} \mathcal{Q}_j$ and let $d \coloneqq \mathsf{dim}(\mathcal{Q})$ denote its dimension. Thus, $\mathcal{Q}$ contains p^d elements. We rewrite $\mathcal{C}$:

$$\mathcal{C} = \mathcal{Q} \setminus \Big(\bigcup_{i\in[1,m]} (\mathcal{D}_i \cap \mathcal{Q}) \Big) .$$

Now for a fixed $i \in [1,m]$ we bound the size of $\mathcal{D}_i \cap \mathcal{Q}$. Since the polynomial D_i has degree one by definition, $\mathcal{D}_i$ is an hyperplane. There are three cases: either $\mathcal{Q} \subseteq \mathcal{D}_i$, which means $\mathcal{C} = \emptyset$. This contradicts the premise of the lemma, namely $\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$. Since $\mathcal{D}_i$ is an hyperplane, the remaining cases are $\mathcal{Q} \cap \mathcal{D}_i = \emptyset$ and $\mathcal{Q} \cap \mathcal{D}_i$ has dimension $\mathsf{dim}(\mathcal{Q}) - 1 = d - 1$. In both cases $\mathcal{D}_i \cap \mathcal{Q}$ contains at most p^{d-1} elements.

When we remove the sets $(\mathcal{D}_i)_{i\in[1,m]}$ from $\mathcal{Q}$, we remove at most mp^{d-1} elements, which yields

$$p^d - mp^{d-1} \leq |\mathcal{C}| \leq p^d . \tag{3.4}$$

We now use the same method to bound $|\mathcal{C} \cap \mathcal{Q}_{q+1}|$. We define $\mathcal{Q}' = \mathcal{Q}_{q+1} \cap \mathcal{Q}$. Since $\overrightarrow{Q}_{q+1}$ is linearly independent of $(\overrightarrow{Q}_j)_{j\in[1,m]}$, we get $\mathsf{dim}(\mathcal{Q}') = d - 1$.

For a fixed $i \in [1,m]$, since by assumption $\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$, we can proceed as with $\mathcal{Q}$ above: either $\mathcal{Q}' \cap \mathcal{D}_i = \emptyset$ or $\mathcal{Q}' \cap \mathcal{D}_i$ has dimension $d - 2$, which yields

$$p^{d-1} - mp^{d-2} \leq |\mathcal{Q}_{q+1} \cap \mathcal{C}| \leq p^{d-1} . \tag{3.5}$$

Combining equations (3.4) and (3.5) we obtain the following, which concludes the proof:

$$\frac{p^{d-1} - mp^{d-2}}{p^d} \leq \frac{|\mathcal{Q}_{q+1} \cap \mathcal{C}|}{|\mathcal{C}|} \leq \frac{p^{d-1}}{p^d - mp^{d-1}} .$$

3.2 Proof Overview

THE GENERIC GAME. We prove a lower bound on the computational complexity of the OMDL game in generic groups in the sense of Shoup [Sho97]. We follow the notation developed by Boneh and Boyen [BB08] for this proof.

In the generic group model, elements of $\mathbb{G}$ are encoded as arbitrary unique strings, so that no property other than equality can be directly tested by the adversary. The adversary performs operations on group elements by interacting with an oracle called GCMP.

To represent and simulate the working of the oracles, we model the opaque encoding of the elements of $\mathbb{G}$ using an injective function $\Xi\colon \mathbb{Z}_p \to \{0,1\}^{\lceil \log_2(p) \rceil}$ where p is the group order.

Game $\mathrm{OMDLGGM}^{\mathcal{A}}_{\mathsf{GrGen}}(\lambda)$

$\overrightarrow{x} := ()\,;\ a_0 := 1$
$j := 0\,;\ q := 0\,;\ n := 0$
$\overrightarrow{y} \leftarrow \mathcal{A}^{\textsc{Chal},\textsc{DLog},\textsc{GCmp}}(\textsc{Enc}())$
return $\left(\overrightarrow{y} = \overrightarrow{x}\ \wedge\ q < n\right)$

Oracle Chal()

$n := n + 1$
$x_n \xleftarrow{\$} \mathbb{Z}_p$
$j := j + 1$
$a_j := x_n$
return Enc()

Oracle DLog(ξ)

if $\xi \notin \{\xi_i\}_{i\in[0,j]}$
  then return $\perp$
$q := q + 1$
$i := \min\{k \in [0,j] \mid \xi = \xi_k\}$
return a_i

Enc()

if $\exists\, i \in [0, j-1] : a_j = a_i$
  then $\xi_j := \xi_i$
else
  $\xi_j \xleftarrow{\$} \{0,1\}^{\log(p)} \setminus \{\xi_i\}_{i\in[0,j-1]}$
return ξ_j

Oracle GCmp(ξ, ξ', b)

if $\xi \notin \{\xi_i\}_{i\in[0,j]}$ **or** $\xi' \notin \{\xi_i\}_{i\in[0,j]}$
  then return $\perp$
$i := \min\{k \in [0,j] \,|\, \xi = \xi_k\}$
$i' := \min\{k \in [0,j] \,|\, \xi' = \xi_k\}$
$j := j + 1\,;\ a_j := a_i + (-1)^b a_{i'}$
return Enc()

Figure 3.1: The OMDL game in the GGM

Internally, the simulator represents the elements of $\mathbb{G}$ by their discrete logarithms relative to a fixed generator G. This is captured by Ξ, which maps any integer a to the string $\xi := \Xi(a)$ representing $a \cdot G$. In the game we will use an encoding procedure Enc to implement Ξ.

We specify the game OMDL in the GGM in Figure 3.1. In contrast to Figure 2.1 there are no more group elements. The game instead maintains discrete logarithms $a \in \mathbb{Z}_p$ and gives the adversary their encodings $\Xi(a)$, which are computed by the procedure Enc. The challenger uses variable j to represent the number of created group elements, which is incremented before each call to Enc. The procedure Enc then encodes the latest scalar a_j. If a_j has already been assigned a string ξ, then Enc() outputs ξ, else it outputs a random string different from all previous ones. For this, the game maintains a list $(a_i, \xi_i)_{0\le i\le j}$ of logarithms and their corresponding encodings.

OMDLGGM initializes $j = 0$ and $a_0 = 1$, and runs the adversary on input $\xi_0 \leftarrow \textsc{Enc}()$ (ξ_0 is thus the encoding of the group generator). The oracle Chal increments a counter of challenges n, samples a new value x_n and returns its encoding by calling Enc(). Since it creates a new element, it first increments j and defines the $a_j := x_n$. The oracle DLog is called with a string ξ and returns $\perp$ if the string is not in the list of assigned strings $\{\xi_i\}_{i\in[0,j]}$. Else, it picks an index i (concretely: the smallest such index) such that $\xi_i = \xi$ and returns a_i, which is the Ξ-preimage of ξ (and thus the logarithm of the group element encoded by ξ).

The adversary also has access to the oracle GCmp for group operations, which takes as input two strings ξ and ξ' and a bit b, which indicates whether the adversary wants to compute the addition or the subtraction of the group elements. The oracle gets the (smallest) indexes i and i' such that $\xi = \xi_i$ and $\xi' = \xi_{i'}$. The oracle increments j, sets $a_j := a_i + (-1)^b a_{i'}$ and returns Enc(), which computes the encoding of a_j.

Proof overview. The aim of our proof is to simulate the game without ever computing scalars a_i by replacing them by polynomials P_i and show that with overwhelming probability this does not affect the game. Game_0 (defined by ignoring all the boxes, except the dashed ones, in Figure 3.2) is the same game as OMDLGGM, except for two syntactical changes, which will be useful in the proof. The main modification is that we now make n calls to the oracle DLog after $\mathcal{A}$ outputs its

answer $\overrightarrow{y}$: for $i \in [1, n]$ we set $x_i \coloneqq \text{DLOG}(\xi_{j_i})$, where indexes j_i are defined in the oracle CHAL so that $a_{j_i} = x_i$; thus $\text{DLOG}(\xi_{j_i})$ always outputs $a_{j_i} = x_i$, meaning this does not affect the game. Second, as calls to DLOG increase q, we put the condition "**if** $q < n$ **then return** 0" before those calls.

INTRODUCING POLYNOMIALS. Game_1, defined in Figure 3.2 by only ignoring the gray boxes, introduces the polynomials P_i. The polynomial $P_0 = 1$ represents $a_0 = 1$. In the n-th call to CHAL, the game defines a new polynomial $P_j = \mathsf{X}_n$, which represents the value x_n. We thus have

$$P_i(\overrightarrow{x}) = a_i \ , \tag{3.6}$$

and in this sense the polynomial P_i represents the scalar a_i (and thus implicitly the group element $a_i G$). The group operation oracle maintains this invariant; when computing $a_j := a_i + (-1)^b a_{i'}$, it also sets $P_j := P_i + (-1)^b P_{i'}$.

Note that there are many ways to represent a group element aG by a polynomial. E.g., the first challenge $x_1 G$ is represented by both the polynomial X_1 and the constant polynomial x_1. Intuitively, since x_1 is a challenge, it is unknown to $\mathcal{A}$, and as long as $\mathcal{A}$ does not query $\text{DLOG}(\xi)$, with $\xi \coloneqq \Xi(x_1)$, it does not know that the polynomials X_1 and x_1 represent the same group element. Game_1 introduces a list L that represents this knowledge of $\mathcal{A}$. E.g., when $\mathcal{A}$ calls $\text{DLOG}(\Xi(x_1))$, the game will append the polynomial $\mathsf{X}_1 - x_1$ to the list L. More generally, on call $\text{DLOG}(\xi_i)$ the game appends $P_i - P_i(\overrightarrow{x})$ to L, which represents the fact that $\mathcal{A}$ knows that the polynomial $P_i - P_i(\overrightarrow{x})$ represents the scalar 0 and the group element $0_{\mathbb{G}}$. The list L will be used to ensure consistency when we replace scalars by polynomials in the game.

Recall that our goal is to have the challenger only deal with polynomials when simulating the game for $\mathcal{A}$. As this can be done without actually defining the challenge $\overrightarrow{x}$, the challenger could then select $\overrightarrow{x}$ *after* $\mathcal{A}$ gave its output, making it impossible for $\mathcal{A}$ to predict the right answer.

This is done in the final game Game_4, defined in Figure 3.4, where the challenger is in the same position as $\mathcal{A}$: it does not know that x_1 is the answer to the challenge represented by the polynomial X_1 until $\text{DLOG}(\xi)$ is called with $\xi \coloneqq \Xi(x_1)$. In fact, x_1 is not even defined before this call, and, more generally, $\overrightarrow{x}$ does not exist until the proper DLOG queries are made.

To get to Game_4, we define two intermediate games. We will modify procedure ENC so that it later deals with polynomials only (instead of their evaluations, as $\overrightarrow{x}$ will not exist). Because of this, it will be unknown whether $P_j(\overrightarrow{x}) = P_i(\overrightarrow{x})$ for some $i \in [0, j-1]$ – *unless* $P_j - P_i \in \mathsf{Span}(L)$, since both the challenger and the adversary are aware that all polynomials in L evaluate to 0 at $\overrightarrow{x}$.

However, it can be the case that, when $\overrightarrow{x}$ is defined later, $P_j(\overrightarrow{x}) = P_i(\overrightarrow{x})$. That is, in the original game, we would have had $a_j = a_i$, but in the final game, ENC is not aware of this. This is precisely when the simulation fails, and we abort the game. We will then bound the probability of this event, for which we will use Lemma 3.1.

In "typical" GGM proofs an abort happens when $P_j(\overrightarrow{x}) = P_i(\overrightarrow{x})$ and $P_j \neq P_i$. For OMDL, because the adversary might have information on the $\overrightarrow{x}$ (and the challenger is aware of this), we allow that there are $P_j \neq P_i$ for which the current knowledge on $\overrightarrow{x}$ lets us deduce $P_j(\overrightarrow{x}) = P_i(\overrightarrow{x})$. With the formalism introduced above this corresponds exactly to the situation that $P_i - P_j \notin \mathsf{Span}(L)$. We introduce this abort condition in the procedure ENC in Game_1 (Figure 3.2). Because in the "ideal" game Game_4 (Figure 3.4), there are no more values a_i, we will express the abort condition differently (namely in oracle DLOG) and argue that the two conditions are equivalent.

ELIMINATING USES OF SCALARS. Using the abort condition in Game_1, we can replace some uses of the scalars a_i by their representations as polynomials P_i. This is what we do in Game_2, (Figure 3.2, including all boxes *except* the dashed box), which eliminates all occurrences of a_i's. In ENC, since the game aborts when $P_j(\overrightarrow{x}) = P_i(\overrightarrow{x})$ and $P_j - P_i \notin \mathsf{Span}(L)$, and because when $P_j - P_i \in \mathsf{Span}(L)$,

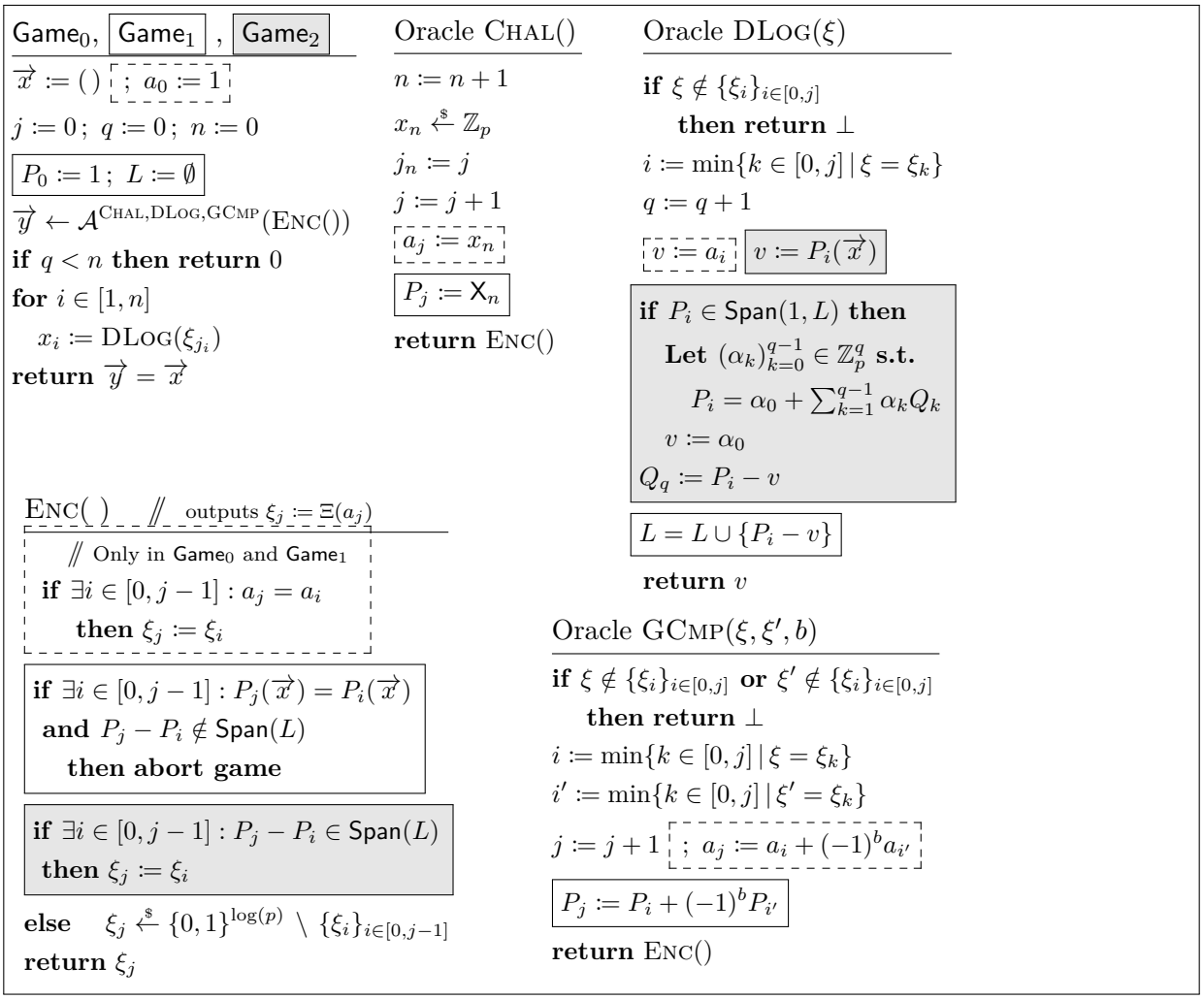

Figure 3.2: Game_0 (which only includes the dashed boxes) is the GGM version of OMDL. Game_1 (including all but the gray boxes) introduces the polynomials that represent the information that $\mathcal{A}$ obtains, and aborts when Game_0 cannot be simulated with polynomials. In Game_2 (including all but the dashed boxes) we eliminate the use of scalars (except for the abort condition) in oracles ENC and DLOG.

it implies that $P_j(\overrightarrow{x}) = P_i(\overrightarrow{x})$, we can replace the event $P_j(\overrightarrow{x}) = P_i(\overrightarrow{x})$ by $P_j - P_i \in \mathsf{Span}(L)$. Intuitively, we can now think of ENC() as encoding the polynomial P_j instead of the scalar a_j.

We next modify the oracle DLOG. The first change is that instead of returning a_i the oracle uses $P_i(\overrightarrow{x})$, which is equivalent by (3.6). The second change is that on input ξ, oracle DLOG checks if $\mathcal{A}$ already knows the answer to its query, in which case it computes the answer without using $\overrightarrow{x}$. E.g., assume $\mathcal{A}$ has only made one query CHAL(), and thus $q = 0$ and $L = \emptyset$: if $\mathcal{A}$ now queries DLOG(ξ) with $\xi := \Xi(x_1)$, the oracle first checks if $P_i = \mathsf{X}_1 \in \mathsf{Span}(1, L)$, (with i the current value of number of group elements seen by the adversary), which is not the case, and so it computes $v := P_i(\overrightarrow{x}) = x_1$. It then adds the polynomial $Q_1 := \mathsf{X}_1 - x_1$ to L and returns x_1. If for example $\mathcal{A}$ makes another call DLOG(ξ') with $\xi' := \Xi(2x_1 + 2)$, then it knows that the answer should be $2x_1 + 2$. And indeed, the oracle DLOG checks if $2\mathsf{X}_1 + 2 \in \mathsf{Span}(1, L)$, and since this is the case, it gets the decomposition

$$2\mathsf{X}_1 + 2 = (2x_1 + 2) + 2Q_1 = \alpha_0 + \alpha_1 Q_1$$

with $\alpha_0 = 2x_1 + 2$ and $\alpha_1 = 2$. The oracle uses this decomposition to compute its answer $v := \alpha_0 = 2x_1 + 2$.

More generally, on input ξ_i, the oracle DLOG checks if $P_i \in \mathsf{Span}(1, L)$. If so, it computes the answer using the decomposition of P_i in $\mathsf{Span}(1, L)$; else it uses $\overrightarrow{x}$ and outputs $a_i = P_i(\overrightarrow{x})$.

We have now arrived at a situation close to the "ideal" game, where the challenger only uses polynomials. The only uses of scalars are the abort condition in ENC (since it compares $P_j(\overrightarrow{x})$ and $P_i(\overrightarrow{x})$) and in DLOG, when computing the logarithm of an element that is not already known to $\mathcal{A}$. Towards our goal of simulating the game without defining $\overrightarrow{x}$, we modify those two parts next.

CHANGING THE ABORT CONDITION. The aim of Game_3 is precisely to modify the abort condition so that it does not use $\overrightarrow{x}$ anymore. Figure 3.3 recalls Game_2 and defines Game_3 by not including the dashed and the gray box. In Game_3 the challenger does not abort in the procedure ENC. This means that if $P_j - P_i \notin \mathsf{Span}(L)$ for some i, the challenger creates a string $\xi_j \neq \xi_i$ even when $P_j(\overrightarrow{x}) = P_i(\overrightarrow{x})$. This means that the simulation of the game is not correct anymore; but we will catch these inconsistencies in the oracle DLOG.

For concreteness consider the following example: let $\overrightarrow{x} = (x_1)$ and suppose $\mathcal{A}$ built the polynomials $P_{i_1} = x_1$ using the oracle GCMP and $P_{i_2} = \mathsf{X}_1$ using the oracle CHAL; suppose also that $\mathcal{A}$ has not queried DLOG yet, thus $L = \emptyset$. If $i_1 < i_2$ then Game_2 aborts on the call ENC() which encodes P_{i_2}, since $P_{i_1}(\overrightarrow{x}) = P_{i_2}(\overrightarrow{x})$ and $P_{i_2} - P_{i_1} \notin \mathsf{Span}(L)$. In contrast, in Game_3 the challenger defines $\xi_{i_1} \neq \xi_{i_2}$, which is inconsistent. But the abort will now happen during a call to the oracle DLOG.

Suppose $\mathcal{A}$ queries DLOG(ξ_{i_3}), with $\xi_{i_3} = \Xi(2\mathsf{X}_1 + 2)$. Game_3 now adds the polynomial $Q_1 = 2\mathsf{X}_1 + 2 - (2x_1 + 2) = 2(\mathsf{X}_1 - x_1)$ to L and checks for an inconsistency of this answer with all the polynomials that $\mathcal{A}$ computed. Since it finds that $P_{i_1} - P_{i_2} = x_1 - \mathsf{X}_1 \in \mathsf{Span}(L)$ but $\xi_{i_1} \neq \xi_{i_2}$, the game aborts. But Game_3 should also abort even if $\mathcal{A}$ does not query the oracle DLOG. This was precisely the reason for adding the final calls of the game to the oracle DLOG in Game_0. Since $P_{j_i} = \mathsf{X}_i$ and the challenger calls $x_i \leftarrow$ DLOG(ξ_{j_i}) for $i \in [1, n]$ at the end, the challenger makes the query DLOG(ξ_{j_1}), which adds $\mathsf{X}_1 - x_1$ to L, after which we have $P_{i_1} - P_{i_2} \in \mathsf{Span}(L)$ and therefore an abort.

More generally, in Game_3 the oracle DLOG aborts if there exists $(i_1, i_2) \in [0, j]^2$ such that $P_{i_1} - P_{i_2} \in \mathsf{Span}(L)$ and $\xi_{i_1} \neq \xi_{i_2}$. In the proof of Theorem 3.2 we show that this abort condition is equivalent to the abort condition in Game_2.

ELIMINATING ALL USES OF $\overrightarrow{x}$. In Game_3 the only remaining part that uses $\overrightarrow{x}$ is the operation $v := P_i(\overrightarrow{x})$ in oracle DLOG. Our final game hop will replace this by an equivalent operation. In Game_4, also presented in Figure 3.3, the challenger samples v uniformly from $\mathbb{Z}_p$ instead of evaluating P_i on the challenge. In the proof of Theorem 3.2, we will show that since the distribution of $P_i(\overrightarrow{x})$ is uniform for a fixed P_i, this change does not affect the game.

This is the only difference between Game_4 and Game_3, but since this modification removes all the uses of $\overrightarrow{x}$ for the challenger, we rewrite Game_4 explicitly in Figure 3.4, where we define $\overrightarrow{x}$ only after $\mathcal{A}$ outputs $\overrightarrow{y}$. Game_4 is thus a game which is easily seen to be hard to win for $\mathcal{A}$. The reason for this is that $\mathcal{A}$ cannot make enough queries to DLOG to constrain the construction of $\overrightarrow{x}$ at the end of the game and therefore cannot predict the challenge $\overrightarrow{x}$. We now make the intuition given above formal in the following theorem.

3.3 Formal Proof

Theorem 3.2. *Let $\mathcal{A}$ be an adversary that solves OMDL in a generic group of prime order p, making at most m oracle queries. Then*

$$\mathbf{Adv}^{\text{OMDLGGM}}_{\mathcal{A}} \leq \frac{1}{p} + \frac{m^2}{p - m^2} \ .$$

Game_2, $\boxed{\mathsf{Game}_3}$, $\boxed{\mathsf{Game}_4}$

$\overrightarrow{x} \coloneqq ()$
$j \coloneqq 0\,;\ q \coloneqq 0\,;\ n \coloneqq 0$
$P_0 \coloneqq 1\,;\ L \coloneqq \emptyset$
$\overrightarrow{y} \leftarrow \mathcal{A}^{\text{CHAL,DLOG,GCMP}}(\text{ENC}())$
if $q \geq n$ **then return** 0
for $i \in [1, n]$
  $x_i \coloneqq \text{DLOG}(\xi_{j_i})$
return $\overrightarrow{y} = \overrightarrow{x}$

Oracle CHAL()

$j \coloneqq j + 1\,;\ n \coloneqq n + 1$
$x_n \xleftarrow{\$} \mathbb{Z}_p\,;\ j_n \coloneqq j$
$P_j \coloneqq \mathsf{X}_n$
return ENC()

Oracle DLOG(ξ)

if $\xi \notin \{\xi_i\}_{i \in [0,j]}$ **then return** $\perp$
$i \coloneqq \min\{k \in [0, j] \mid \xi = \xi_k\}$
$q \coloneqq q + 1$
$v \coloneqq P_i(\overrightarrow{x})$ $\boxed{\ ;\ v \xleftarrow{\$} \mathbb{Z}_p}$
if $P_i \in \mathsf{Span}(1, L)$ **then**
  let $(\alpha_k)_{k=0}^{q-1} \in \mathbb{Z}_p^q$ **s.t.** $P_i = \alpha_0 + \sum_{k=1}^{q-1} \alpha_k Q_k$
  $v \coloneqq \alpha_0$
$Q_q \coloneqq P_i - v\,;\ L = L \cup \{P_i - v\}$
// Abort condition in Game_3 and Game_4 only
if $\exists (i_1, i_2) \in [0, j]^2$ **:** $P_{i_1} - P_{i_2} \in \mathsf{Span}(L)$
  and $\xi_{i_1} \neq \xi_{i_2}$
  then abort game
return v

ENC() // outputs ξ_j which encodes P_j

// Abort condition in Game_2 only
if $\exists i \in [0, j-1]$ **:** $P_j(\overrightarrow{x}) = P_i(\overrightarrow{x})$
  and $P_j - P_i \notin \mathsf{Span}(L)$
  then abort game
if $\exists i \in [0, j-1] : P_j - P_i \in \mathsf{Span}(L)$
then $\xi_j \coloneqq \xi_i$
else
  $\xi_j \xleftarrow{\$} \{0,1\}^{\log(p)} \setminus \{\xi_i\}_{i \in [0, j-1]}$
return ξ_j

Oracle GCMP(ξ, ξ', b)

if $\xi \notin \{\xi_i\}_{i \in [0,j]}$ **or** $\xi' \notin \{\xi_i\}_{i \in [0,j]}$
  then return $\perp$
$i \coloneqq \min\{k \in [0, j] \mid \xi = \xi_k\}$
$i' \coloneqq \min\{k \in [0, j] \mid \xi' = \xi_k\}$
$j \coloneqq j + 1$
$P_j \coloneqq P_i + (-1)^b P_{i'}$
return ENC()

Figure 3.3: In Game_3 we move the abort condition from ENC to the oracle DLOG, so it can be checked without using scalars. The only remaining use is then "$v \coloneqq P_i(\overrightarrow{x})$" in oracle DLOG. Game_4 instead pick the output x uniformly at random.

Proof [Proof of Theorem 3.2] The proof will proceed as follows: we first compute the statistical distance between Game_0, which is OMDLGGM, and Game_1 (Figure 3.2); we then show that Game_1, Game_2, Game_3 and Game_4 (Figures 3.2 and 3.3) are equivalently distributed; and finally we upper-bound the probability of winning Game_4 (Figure 3.4).

Game_4

$j \coloneqq 0\,;\ q \coloneqq 0\,;\ n \coloneqq 0$
$P_0 \coloneqq 1\,;\ L \coloneqq \emptyset$
$\overrightarrow{y} \leftarrow \mathcal{A}^{\textsc{Chal},\textsc{DLog},\textsc{GCmp}}(\textsc{Enc}())$
if $q \geq n$ **then return** 0
for $i \in [1,n]$
  $x_i \coloneqq \textsc{DLog}(\xi_{j_i})$
return $\overrightarrow{y} = \overrightarrow{x}$

Oracle $\textsc{Chal}()$

$j \coloneqq j+1\,;\ n \coloneqq n+1$
$P_j \coloneqq \mathsf{X}_n\,;\ j_n \coloneqq j$
return $\textsc{Enc}()$

Oracle $\textsc{DLog}(\xi)$

if $\xi \notin \{\xi_i\}_{i\in[0,j]}$ **then return** $\perp$
$i \coloneqq \min\{k \in [0,j] \mid \xi = \xi_k\}$
$q \coloneqq q+1\,;\ v \xleftarrow{\$} \mathbb{Z}_p$
if $P_i \in \mathsf{Span}(1, L)$ **then**
  let $(\alpha_k)_{k=0}^{q-1} \in \mathbb{Z}_p^q$ **s.t.** $P_i = \alpha_0 + \sum_{k=1}^{q-1} \alpha_k Q_k$
  $v \coloneqq \alpha_0$
$Q_q \coloneqq P_i - v\,;\ L = L \cup \{P_i - v\}$
if $\exists (i_1, i_2) \in [0,j]^2 : P_{i_1} - P_{i_2} \in \mathsf{Span}(L)$
  and $\xi_{i_1} \neq \xi_{i_2}$
  then abort game
return v

$\textsc{Enc}()$ // outputs ξ_j which encodes P_j

if $\exists i \in [0, j-1] : P_j - P_i \in \mathsf{Span}(L)$
 then $\xi_j \coloneqq \xi_i$
else
  $\xi_j \xleftarrow{\$} \{0,1\}^{\log(p)} \setminus \{\xi_i\}_{i\in[0,j-1]}$
return ξ_j

Oracle $\textsc{GCmp}(\xi, \xi', b)$

if $\xi \notin \{\xi_i\}_{i\in[0,j]}$ **or** $\xi' \notin \{\xi_i\}_{i\in[0,j]}$
  then return $\perp$
$i \coloneqq \min\{k \in [0,j] \mid \xi = \xi_k\}$
$i' \coloneqq \min\{k \in [0,j] \mid \xi' = \xi_k\}$
$j \coloneqq j+1$
$P_j \coloneqq P_i + (-1)^b P_{i'}$
return $\textsc{Enc}()$

Figure 3.4: Final game Game_4 does not use $\overrightarrow{x}$ in the oracles anymore. It defines the challenge $\overrightarrow{x}$ after $\mathcal{A}$ gave its output and this is what makes it simple for us to prove it is hard to win for $\mathcal{A}$.

PRELIMINARY RESULTS. We start with proving three useful invariants of the polynomials P_i and the set L which are introduced in Game_1. The first one is:

$$\forall\, i \in [0,j] :\ P_i(\overrightarrow{x}) = a_i\ . \tag{3.7}$$

This holds in Game_1 and justifies replacing all occurrences of a_i by $P_i(\overrightarrow{x})$ in Game_2 in Figure 3.3. To prove this, we show that each time the games introduce a new polynomial P_j, we have $P_j(\overrightarrow{x}) = a_j$.

We prove this by induction. Initially, $P_0 = 1$ and $a_0 = 1$ so the statement holds for $j = 0$. Now suppose it is true for all $i \in [0, j-1]$. We show it is true for j. Polynomial P_j can be built either by oracle $\textsc{Chal}$ or by oracle $\textsc{GCmp}$:

- In oracle $\textsc{Chal}$, $P_j \coloneqq \mathsf{X}_n$ and $a_j \coloneqq x_n$ so we have $P_j(\overrightarrow{x}) = x_n = a_j$.
- In oracle $\textsc{GCmp}$, $P_j \coloneqq P_i + (-1)^b P_{i'}$ and $a_j \coloneqq a_i + (-1)^b a_{i'}$ so we have $P_j(\overrightarrow{x}) \coloneqq P_i(\overrightarrow{x}) + (-1)^b P_{i'}(\overrightarrow{x}) = a_i + (-1)^b a_{i'} = a_j$.

This proves (3.7).

We next show that the following holds in Game_1, Game_2 and Game_3:

$$\forall\, Q \in \mathsf{Span}(L),\ Q(\overrightarrow{x}) = 0 \tag{3.8}$$

(in the other games either L or $\overrightarrow{x}$ are not defined). For $L = \{Q_1, \ldots, Q_q\}$ if $Q \in \mathsf{Span}(L)$ then $Q = \sum_{k=1}^{q} \alpha_k Q_k$. To show (3.8), it suffices to show that for all $k \in [1, q]$ we have $Q_k(\overrightarrow{x}) = 0$.

For $k \in [0, q]$, Q_k is defined during the k-th call to DLOG on some input ξ. In Game_1, the oracle finds i such that $\xi_i = \xi$ and sets $v := a_i$ and $Q_k := P_i - v$, so we get $Q_k(\overrightarrow{x}) = P_i(\overrightarrow{x}) - a_i$. Using the first result (3.7), we get that (3.8) holds. In Game_2 and Game_3 the oracle sets $v := P_i(\overrightarrow{x})$ so we directly get $Q_k(\overrightarrow{x}) = P_i(\overrightarrow{x}) - P_i(\overrightarrow{x}) = 0$

The third result we will use holds (assuming the game did not abort) in Game_1, Game_2, Game_3 and Game_4:

$$\forall j \geq 1 \, \forall i \in [0, j-1] : \xi_j = \xi_i \Leftrightarrow P_j - P_i \in \mathsf{Span}(L) \ . \tag{3.9}$$

We first prove

$$\forall j \geq 1 \, \forall i \in [0, j-1] : \ \xi_j = \xi_i \Rightarrow P_j - P_i \in \mathsf{Span}(L)$$

by induction. We show that this holds for $j = 1$ and all other $j > 0$ and suppose that for some $i^* \in [0, j-1]$, $\xi_j = \xi_{i^*}$. We show that $P_j - P_{i^*} \in \mathsf{Span}(L)$.

- In Game_2, Game_3 and Game_4, since ξ_j is not a new random string when it is defined it means that for some $i_1 \in [0, j-1]$ we had $P_j - P_{i_1} \in \mathsf{Span}(L)$ and thus the game defined $\xi_j := \xi_{i_1}$. This implies that $\xi_{i_1} = \xi_{i^*}$, and since $i_1 < j$, using the induction hypothesis, we get that $P_{i_1} - P_{i^*} \in \mathsf{Span}(L)$ and furthermore

  $$P_j - P_{i^*} = (P_j - P_{i_1}) - (P_{i_1} - P_{i^*}) \in \mathsf{Span}(L) \ .$$

  Now the situation is more simple when $j = 1$: we must have $i_1 = i^* = 0$ so

  $$P_j - P_{i_1} = P_j - P_{i^*} = P_1 - P_0 \in \mathsf{Span}(L) \ .$$

- In Game_1 the proof is almost the same: since ξ_j is not a new random string it means that for some $i_1 \in [0, j-1]$ we had $P_j(\overrightarrow{x}) = P_{i_1}(\overrightarrow{x})$ and thus the game defined $\xi_j := \xi_{i_1}$. Since the game did not abort we don't have "$P_j(\overrightarrow{x}) = P_{i_1}(\overrightarrow{x})$ **and** $P_j - P_{i_1} \notin \mathsf{Span}(L)$", and thus $P_j - P_{i_1} \in \mathsf{Span}(L)$. From here the proof proceeds as for the other games above, and thus $P_j - P_{i^*} \in \mathsf{Span}(L)$.

  When $j = 1$, we have $i^* = 0$ and $P_1 - P_0 \in \mathsf{Span}(L)$, otherwise the game aborts.

We now prove the other implication:

$$\forall j \geq 1 \, \forall i \in [0, j-1] : P_j - P_i \in \mathsf{Span}(L) \Rightarrow \xi_j = \xi_i \ ,$$

again by induction. Using the same method as before we can argue that this is true for $j = 1$. For $j > 1$, when ENC() defines ξ_j, if for some $i^* \in [0, j-1]$ we have $P_j - P_{i^*} \in \mathsf{Span}(L)$ then we show that ξ_j is assigned $\xi_j = \xi_{i^*}$.

- In Game_2, Game_3 and Game_4, since for some $i_1 \in [0, j-1] : P_j - P_{i_1} \in \mathsf{Span}(L)$, the game defines $\xi_j := \xi_{i_1}$. And since

  $$P_{i^*} - P_{i_1} = (P_{i^*} - P_j) + (P_j - P_{i_1}) \in \mathsf{Span}(L) \ ,$$

  by induction we get $\xi_{i_1} = \xi_{i^*}$ which yields $\xi_j = \xi_{i^*}$.

- In Game_1, since we know that $(P_j - P_{i^*})(\overrightarrow{x}) = 0$ from the previous result (3.8), we get that for some $i_1 \in [0, j-1] : P_j(\overrightarrow{x}) = P_{i_1}(\overrightarrow{x})$. Since the game did not abort, we know that $P_j - P_{i_1} \in \mathsf{Span}(L)$, so using the same argument as before, we get $\xi_j = \xi_{i^*}$.

Game_0 TO Game_1. We now compare Game_0 to Game_1. The only difference between the two is when Game_1 aborts in the procedure ENC() on event

$$\exists i \in [0, j-1] \textbf{ such that } P_j(\overrightarrow{x}) = P_i(\overrightarrow{x}) \textbf{ and } P_j - P_i \notin \mathsf{Span}(L) \ . \tag{3.10}$$

We call this event F. Since ENC is called at most m times, we get:

$$\mathbf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}} \leq \mathbf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}} + m \cdot \Pr[F] \ . \tag{3.11}$$

We now upper-bound $\Pr[F]$. Before a call to ENC, the oracle defines P_j. Consider a fixed $i \in [0, j-1]$ and define $P \coloneqq P_j - P_i$. We will upper-bound the probability that

$$P_j(\overrightarrow{x}) - P_i(\overrightarrow{x}) = P(\overrightarrow{x}) = 0$$

with $P \coloneqq P_j - P_i \notin \mathsf{Span}(L)$.

Since $\mathcal{A}$ does not know $\overrightarrow{x}$ one might consider applying the Schwartz-Zippel lemma. But we cannot, since $\mathcal{A}$ knows information on $\overrightarrow{x}$. From $\mathcal{A}$'s point of view, $\overrightarrow{x}$ is not uniformly chosen over $\mathbb{Z}_p^n$, since it satisfies $Q(\overrightarrow{x}) = 0$ for all $Q \in L$ (using (3.8)). We write $L = \{Q_1, \ldots, Q_q\}$, now using the notation from Lemma 3.1 with $Q_{q+1} \coloneqq P$.

$\mathcal{A}$ also knows that if for some indexes i_1, i_2 it was given $\xi_{i_1} \neq \xi_{i_2}$ then $P_{i_1}(\overrightarrow{x}) \neq P_{i_2}(\overrightarrow{x})$. We can reformulate this by writing $D_{\overrightarrow{\imath}} = P_{i_1} - P_{i_2}$ for $\overrightarrow{\imath} \in I \coloneqq \{(i_1, i_2) \in [0, j-1]^2 \,|\, \xi_{i_1} \neq \xi_{i_2}\}$. $\mathcal{A}$ knows that $D_{\overrightarrow{\imath}}(\overrightarrow{x}) \neq 0$. Using the notation of Lemma 3.1 we get that

$$\overrightarrow{x} \in \mathcal{C} \coloneqq \Big(\bigcap_{j \in [1,q]} \mathcal{Q}_j \Big) \setminus \Big(\bigcup_{i \in I} \mathcal{D}_i \Big) \ .$$

Our goal is to apply Lemma 3.1 to upper-bound $\Pr_{\overrightarrow{x} \leftarrow \mathcal{C}}[P(\overrightarrow{x}) = 0]$. We need to verify that the three premises of the lemma are satisfied, which are: from $\mathcal{A}$'s point of view, $\overrightarrow{x} \in \mathcal{C}$ is picked uniformly at random, $\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$ and $\overrightarrow{Q}_{q+1}$ is independent of $(\overrightarrow{Q}_i)_{i \in [1,q]}$.

$\overrightarrow{x}$ IS CHOSEN UNIFORMLY IN $\mathcal{C}$. To show this, we fix the randomness (of the challenger and the adversary) of the game (which means the order in which the ξ_i are picked is deterministic) and we consider the transcript $\pi(\overrightarrow{x})$ of what $\mathcal{A}$ sees during the game when the secret is chosen as $\overrightarrow{x}$: $\pi(\overrightarrow{x}) = (\xi_0, \ldots, \xi_{j-1}, v_1, \ldots, v_q)$ (In this transcript, the strings ξ_i are ordered and so are the v_i, but we implicitly suppose that before the query v_k there was a query v_{k-1} or ξ_{i_k} and after the query v_k there was either a query v_{k+1} or $\xi_{i'_k}$. We do not formalize this.)

The transcript π corresponds to all the output of the oracles that were given to $\mathcal{A}$: The ξ_i are the outputs of GCMP and CHAL, and the v_i are the outputs of DLOG. The transcript $\pi(\overrightarrow{x})$ only depends on the challenge $\overrightarrow{x}$. What is important to notice is that for all $\overrightarrow{y} \in \mathcal{C}$: $\pi(\overrightarrow{y}) = \pi(\overrightarrow{x})$. Indeed, if we call $\pi(\overrightarrow{y}) = (\xi'_0, \ldots, \xi'_{j-1}, v'_1, \ldots, v'_q)$ we can show by induction that $\xi'_i = \xi_i$ and $v'_k = v_k$ for all $i \in [1, j-1]$ and $k \in [1, q]$.

- Let $k \in [1, q]$; we show that $v_k = v'_k$: in both challenges $\overrightarrow{x}$ and $\overrightarrow{y}$, since the transcript $\mathcal{A}$ received is the same by the induction hypothesis, it behaves the same way and calls DLOG on input ξ. The oracle DLOG then picks $i = \min\{j \,|\, \xi_j = \xi\}$ which is the same in both cases by the induction hypothesis. DLOG computes $v_k = P_i(\overrightarrow{x})$ and defines $Q_i \coloneqq P_i - v_k$ for the challenge $\overrightarrow{x}$ while it computes $v'_k = P_i(\overrightarrow{y})$ and $Q'_i \coloneqq P_i - v'_k$ for the challenge $\overrightarrow{y}$. Now Since $\overrightarrow{y} \in \mathcal{C}$, we have in particular $\overrightarrow{y} \in \mathcal{Q}_i$, so we know that $Q_k(\overrightarrow{y}) = P_i(\overrightarrow{y}) - v_k = 0$. This gives $P_i(\overrightarrow{y}) = v'_k = v_k$ and $Q'_k = Q_k$.

- Let $k \in [1, j-1]$; we show that $\xi_k = \xi'_k$: for both challenges $\overrightarrow{x}$ and $\overrightarrow{y}$, since the transcript $\mathcal{A}$ received is the same by induction hypothesis, it behaves the same way and calls either CHAL or GCMP. In both cases the the game creates a polynomial P_k and calls the procedure ENC(), for which there are two cases:

1: $\forall i \in [0, k-1] : P_k(\overrightarrow{x}) \neq P_i(\overrightarrow{x})$. The game with challenge $\overrightarrow{x}$ outputs a new random ξ_k, which means $\xi_k \neq \xi_i$ for $i \in [1, k-1]$. Since $\overrightarrow{y} \in \mathcal{C}$, we know that for all $i \in [0, k-1]$, $\overrightarrow{y} \notin \mathcal{D}_{i,k} = \{\overrightarrow{z} \ : \ (P_i - P_k)(\overrightarrow{z}) = 0 \textbf{ and } \xi_i \neq \xi_k\}$ This means that for all $i \in [0, k-1]$, since $\xi_i \neq \xi_k$, we have $P_i(\overrightarrow{y}) \neq P_k(\overrightarrow{y})$, so the game also chooses ξ'_k as a new random string. Since we fixed the randomness of the game, we get $\xi_k = \xi'_k$.

2: $\exists i^* \in [0, k-1] : P_k(\overrightarrow{x}) = P_{i^*}(\overrightarrow{x})$. The game defines $\xi_k \coloneqq \xi_i$ for the challenge $\overrightarrow{x}$. Since the game did not abort for $k < j$, we know that $P_k - P_{i^*} \in \mathsf{Span}(L)$. Now since $L = (Q_i)_i$ and $\overrightarrow{y} \in \bigcap_{i\in[1,q]} \mathcal{Q}_i$, we also get $(P_k - P_{i^*})(\overrightarrow{y}) = 0$. So the game defines $\xi'_k \coloneqq \xi'_i = \xi_i = \xi_k$, by the induction hypothesis and the preliminary result (3.9).

In both cases we get that $\xi_k = \xi'_k$.

Since the transcript that $\mathcal{A}$ sees is the same for all elements in $\mathcal{C}$, $\mathcal{A}$ can only make a uniform guess on which element of $\mathcal{C}$ is the challenge. Thus from $\mathcal{A}$'s point of view, $\overrightarrow{x}$ is chosen uniformly at random in $\mathcal{C}$.

$\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$. Since $\mathcal{Q}_{q+1} = \{\overrightarrow{x} \in \mathbb{Z}_p : P(\overrightarrow{x}) = 0\}$, if we had $\mathcal{C} \cap \mathcal{Q}_{q+1} = \emptyset$, then $P(\overrightarrow{x}) \neq 0$ for all $\overrightarrow{x} \in \mathcal{C}$, and thus $\Pr_{\overrightarrow{x} \xleftarrow{\$} \mathcal{C}}[P(\overrightarrow{x}) = 0] = 0$. In this case, there is no need to upper-bound the probability, which is why we assume that $\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$.

$\overrightarrow{Q}_{q+1}$ IS INDEPENDENT OF $(\overrightarrow{Q}_i)_{i\in[1,q]}$. Recall that $\overrightarrow{P} = (p_k)_{k\in[1,n]}$ is the vector representing the polynomial $P - P(\overrightarrow{0}) = \sum_{k=1}^{n} p_k \mathsf{X}_k$. We assume that $\overrightarrow{Q}_{q+1}$ is dependent of $(\overrightarrow{Q}_i)_{i\in[1,q]}$ and then show that this contradicts the previous premise $\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$. Assume thus that for some α:

$$Q_{q+1} - Q_{q+1}(\overrightarrow{0}) = \sum_{k=1}^{q} \alpha_k (Q_k - Q_k(\overrightarrow{0})) \ .$$

With $\alpha \coloneqq Q_{q+1}(\overrightarrow{0}) + \sum_{k=1}^{q} \alpha_k Q_k(\overrightarrow{0})$ and $Q \coloneqq \sum_{k=1}^{q} \alpha_k Q_k$, we can write this as $Q_{q+1} = \alpha + Q$ with $\alpha \in \mathbb{Z}_p$ and $Q \in \mathsf{Span}(L)$. Now since we are in event F, defined in (3.10), we have $Q_{q+1} = P \notin \mathsf{Span}(L)$, which implies $\alpha \neq 0$ (otherwise $P = Q \in \mathsf{Span}(L)$). Since $\mathcal{C} \subseteq \mathcal{Q}_i$ we have that for all $i \in [1, q]$ and all $\overrightarrow{x} \in \mathcal{C}$: $Q_i(\overrightarrow{x}) = 0$, and thus $Q(\overrightarrow{x}) = 0$. From this, we have $Q_{q+1}(\overrightarrow{x}) = \alpha + Q(\overrightarrow{x}) = \alpha$. Thus, $Q_{q+1}(\overrightarrow{x}) \neq 0$ for all $\overrightarrow{x} \in \mathcal{C}$, which implies $\mathcal{C} \cap \mathcal{Q}_{q+1} = \emptyset$, which contradicts the previous assumption. We thus proved that $\overrightarrow{Q}_{q+1}$ is independent of $(\overrightarrow{Q}_i)_{i\in[1,q]}$.

APPLYING LEMMA 3.1. Since all its premises are satisfied, we can apply Lemma 3.1 and obtain:

$$\Pr_{\overrightarrow{x} \leftarrow \mathcal{C}}[P(\overrightarrow{x}) = 0] = \Pr_{\overrightarrow{x} \leftarrow \mathcal{C}}[Q_{q+1}(\overrightarrow{x}) = 0] \leq \frac{1}{p - |I|} \ ,$$

with $|I| \leq j^2 \leq m^2$. Since we need to test this with $P = P_j - P_i$ for all $i \in [0, j-1]$, we get $\Pr[F] \leq \dfrac{m}{p - m^2}$ and from (3.11):

$$\mathbf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}} \leq \mathbf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}} + \frac{m^2}{p - m^2} \ . \tag{3.12}$$

Game_1 TO Game_2. There are three changes in Game_2, which we show do not affect the distributions of the game. First, we replace a_i by $P_i(\overrightarrow{x})$ in oracle DLOG, which is equivalent by (3.7).

Second, in ENC, we replace the condition

$$\textbf{if } \exists i \in [0, j-1] : P_j(\overrightarrow{x}) = P_i(\overrightarrow{x}) \textbf{ then } \xi_j \coloneqq \xi_i$$

by

$$\textbf{if } \exists i \in [0, j-1] : P_j - P_i \in \mathsf{Span}(L) \textbf{ then } \xi_j \coloneqq \xi_i \ .$$

We show that this new condition does not affect the output of ENC(). There are two cases for $P_j(\overrightarrow{x})$:

Case 1: $\exists i^* \in [0, j-1] : P_j(\overrightarrow{x}) = P_{i^*}(\overrightarrow{x})$. We have either

- $P_j - P_{i^*} \in \mathsf{Span}(L)$, and in this case Game_1 and Game_2 both set $\xi_j = \xi_{i^*}$ and output ξ_j using (3.9); or
- $P_j - P_{i^*} \notin \mathsf{Span}(L)$, meaning that both Game_1 and Game_2 abort since "$P_j - P_{i^*} \notin \mathsf{Span}(L)$ **and** $P_j(\overrightarrow{x}) = P_{i^*}(\overrightarrow{x})$" is the abort condition.

Case 2: $\forall i \in [0, j-1]\ P_j(\overrightarrow{x}) \neq P_i(\overrightarrow{x})$. Since, by 3.8, all polynomials in $\mathsf{Span}(L)$ vanish at $\overrightarrow{x}$, this implies $\forall i \in [0, j-1] : P_j - P_i \notin \mathsf{Span}(L)$. In this case both Game_1 and Game_2 output a random new string ξ_j.

The third change in Game_2, in the oracle DLOG, does not change the output either: in Game_1 the DLOG oracle always outputs $a_i = P_i(\overrightarrow{x})$. In Game_2, when $P_i \in \mathsf{Span}(L)$, the game uses the decomposition $P_i = \alpha_0 + \sum_{k=1}^{q-1} \alpha_k Q_k$, and since $Q_k(\overrightarrow{x}) = 0$ by (3.8), it outputs $P_i(\overrightarrow{x}) = \alpha_0$.

Together this yields:

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_1} = \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_2} \ . \tag{3.13}$$

Game_2 TO Game_3. In this game hop we move the abort condition from the procedure ENC to the oracle DLOG. We show that the two abort conditions are equivalent, by showing the two implications of the equivalence:

IF Game_2 ABORTS THEN Game_3 ALSO ABORTS. If Game_2 aborts, it means that for a fixed index j^* the game found $i^* \in [0, j^*-1]$ such that $P_{j^*} - P_{i^*} \notin \mathsf{Span}(L)$ and $P_{j^*}(\overrightarrow{x}) = P_{i^*}(\overrightarrow{x})$. We show that Game_3 also aborts in this situation. Let $P := P_{j^*} - P_{i^*}$. At the end of Game_3 the challenger makes calls to DLOG on each challenge $P_{j_i} = \mathsf{X}_i$. This adds the corresponding polynomials $\mathsf{X}_i - x_i$ to L for all $i \in [1, n]$. With $P = P(\overrightarrow{0}) + \sum_{k=1}^{n} p_k \mathsf{X}_k$, we can write

$$P = \sum_{k=1}^{n} p_k(\mathsf{X}_k - x_k) + P(\overrightarrow{0}) + \sum_{k=1}^{n} p_k x_k \ .$$

Since $P(\overrightarrow{x}) = P_{j^*}(\overrightarrow{x}) - P_{i^*}(\overrightarrow{x})$, we have $P(\overrightarrow{x}) = 0$. On the other hand, by (3.8), we have $P(\overrightarrow{x}) = P(\overrightarrow{0}) + \sum_{k=1}^{n} p_k x_k$. Together, this yields $P = \sum_{k=1}^{n} p_k(\mathsf{X}_k - x_k)$, which means $P \in \mathsf{Span}(L)$ at the end of the game. At the time when Game_2 would have aborted, we had $P \notin \mathsf{Span}(L)$ and thus the game attributed two different strings $\xi_{i^*} \neq \xi_{j^*}$ to P_{i^*} and P_{j^*}, respectively. But at the end of Game_3, when L contains all $\mathsf{X}_i - x_i$ for $i \in [1, n]$, we have $P \in \mathsf{Span}(L)$. This means that one call to DLOG updated L so that $P \in \mathsf{Span}(L)$ and when this happened, since $\xi_{i^*} \neq \xi_{j^*}$, the abort condition in DLOG was satisfied and the game aborted

IF Game_3 ABORTS THEN Game_2 ALSO ABORTS. If Game_3 aborts, then on a call to DLOG we have $\exists (i_1, i_2) \in [0, j]^2$ such that $P_{i_1} - P_{i_2} \in \mathsf{Span}(L)$ and $\xi_{i_1} \neq \xi_{i_2}$. From $P_{i_1} - P_{i_2} \in \mathsf{Span}(L)$, using (3.8) we get $P_{i_1}(\overrightarrow{x}) = P_{i_2}(\overrightarrow{x})$. Suppose $i_1 < i_2$. The challenger in Game_2 used the procedure ENC() when the counter j was equal to i_2 to compute $\xi_{i_2} \neq \xi_{i_1}$. This means that at that moment, L contained fewer elements and we had $P_{i_2} - P_{i_1} \notin \mathsf{Span}(L)$. Since Game_2 aborts when $P_{i_1}(\overrightarrow{x}) = P_{i_2}(\overrightarrow{x})$ and $P_{i_2} - P_{i_1} \notin \mathsf{Span}(L)$, thus Game_2 aborts in this case.

Combining both implications yields

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_2} = \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_3} \ . \tag{3.14}$$

Game_3 TO Game_4. The only difference between these games is in the oracle DLOG. Instead of computing $v \coloneqq P_i(\overrightarrow{x})$, Game_4 picks a random $v \xleftarrow{\$} \mathbb{Z}_p$. We prove that after this modification, the distribution of the outputs of oracle DLOG remains the same. The difference between the two games occurs only when $P_i \notin \mathsf{Span}(1, L)$. Let us bound $\Pr_{\overrightarrow{x} \leftarrow \mathcal{C}}\left[P_i(\overrightarrow{x}) = v \textbf{ in } \mathsf{Game}_3\right]$, where $\overrightarrow{x} \in \mathcal{C}$ represents the information that $\mathcal{A}$ knows about $\overrightarrow{x}$, which we previously used in the first game hop.

We apply Lemma 3.1 again to bound this probability. Now since the game does not abort immediately when the inconsistency $P_{i_1}(\overrightarrow{x}) = P_{i_2}(\overrightarrow{x})$ and $\xi_{i_1} \neq \xi_{i_2}$ occurs, the inequalities on the strings level do not give $\mathcal{A}$ any information on what the evaluation $P_i(\overrightarrow{x})$ cannot be. This means that $\mathcal{C}$ is simpler than in the first game hop, namely

$$\mathcal{C} = \bigcap_{i \in [1,q]} \mathcal{Q}_i \ .$$

We define $Q_{q+1} \coloneqq P_i - v$ and show that once again the three premises of Lemma 3.1 hold: $\overrightarrow{x} \in \mathcal{C}$ is picked uniformly at random, $\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$ and $\overrightarrow{Q}_{q+1}$ is independent of $(\overrightarrow{Q}_i)_{i \in [1,q]}$.

$\overrightarrow{x}$ IS CHOSEN UNIFORMLY IN $\mathcal{C}$. To show this, we again fix the randomness of the game and consider the transcript π that $\mathcal{A}$ sees during the game if a particular $\overrightarrow{x}$ is chosen: $\pi(\overrightarrow{x}) = (\xi_0, \ldots, \xi_{j-1}, v_1, \ldots, v_q)$, which contains all oracle outputs given to $\mathcal{A}$. We show that for all $\overrightarrow{y} \in \mathcal{C} : \pi(\overrightarrow{y}) = \pi(\overrightarrow{x})$. Indeed, for $\pi(\overrightarrow{y}) =: (\xi'_0, \ldots, \xi'_{j-1}, v'_1, \ldots, v'_q)$ we show by induction that $\xi'_i = \xi_i$ and $v'_k = v_k$ for all $i \in [1, j-1]$ and $k \in [1, q]$.

- Let $k \in [1, q]$; then $v_k = v'_k$ is showed exactly as in the first game hop (on page 28).
- Let $k \in [1, j-1]$; we show that $\xi_k = \xi'_k$: for both challenges $\overrightarrow{x}$ and $\overrightarrow{y}$, since the transcript $\mathcal{A}$ received is the same by induction hypothesis, $\mathcal{A}$ behaves the same way and calls either CHAL or GCMP. In both cases the game creates a polynomial P_k and calls ENC(), for which there are two cases:

  1: $\forall\, i \in [0, k-1] : P_k - P_i \notin \mathsf{Span}(L)$. Since this condition is independent of $\overrightarrow{x}$ and $\overrightarrow{y}$, for both the game outputs a new random string ξ_k and ξ'_k. Since we fixed the randomness of the game, we get $\xi_k = \xi'_k$.

  2: $\exists i^* \in [0, k-1] : P_k - P_{i^*} \in \mathsf{Span}(L)$. In this case the game defines $\xi_k \coloneqq \xi_i$ and $\xi'_k \coloneqq \xi'_i$ for both challenge $\overrightarrow{x}$ and $\overrightarrow{y}$. We get $\xi'_k \coloneqq \xi'_i = \xi_i = \xi_k$ by the induction hypothesis and (3.9).

  In both cases we thus have $\xi_k = \xi'_k$.

As in first game hop, we conclude that $\mathcal{A}$ cannot distinguish between two different values $\overrightarrow{x} \in \mathcal{C}$ and so we can consider $\overrightarrow{x}$ to be chosen uniformly at random in $\mathcal{C}$.

$\overrightarrow{Q}_{q+1}$ IS LINEARLY INDEPENDENT OF $(\overrightarrow{Q}_i)_{i \in [1,q]}$. Recall that $P_i \notin \mathsf{Span}(1, L)$ and $Q_{q+1} \coloneqq P_i - v$. If $\overrightarrow{Q}_{q+1}$ were linearly dependent of $(\overrightarrow{Q}_i)_{i \in [0,j]}$, then (using the same method as in the first game hop) we would have $Q_{q+1} = P_i - v = \alpha + Q$ with $\alpha \in \mathbb{Z}_p$ and $Q \in \mathsf{Span}(L)$. As this contradicts $P_i \notin \mathsf{Span}(1, L)$, we conclude that $\overrightarrow{Q}_{q+1}$ is linearly independent of $(\overrightarrow{Q}_i)_{i \in [1,q]}$.

$\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$. $\mathcal{C} = \bigcup_{i \in [1,q]} \mathcal{Q}_i$ is an affine space and $\overrightarrow{Q}_{q+1}$ is linearly independent of $(\overrightarrow{Q}_i)_{i \in [0,j]}$. This implies that $\mathcal{Q}_{q+1} \cap \mathcal{C}$ has dimension $\dim(\mathcal{C}) - 1$ and thus $\mathcal{Q}_{q+1} \cap \mathcal{C} \neq \emptyset$.

Applying Lemma 3.1. Since its three premises are satisfied, Lemma 3.1 with $M \coloneqq 0$ yields:

$$\Pr[\,Q_{q+1}(\overrightarrow{x}) = 0\,]_{\overrightarrow{x} \leftarrow \mathcal{C}} = \Pr_{\overrightarrow{x} \leftarrow \mathcal{C}}\left[P_i(\overrightarrow{x}) = v \textbf{ in } \mathsf{Game}_3\right] = \frac{1}{p}\ .$$

This means that in Game_3 the distribution of $P_i(\overrightarrow{x})$ is uniform, so the change we make does not affect the overall distribution of the game. We thus have

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_3} = \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_4}\ . \tag{3.15}$$

Analysis of Game_4. We prove that $\mathcal{A}$ wins Game_4 at most with negligible probability $\frac{1}{p}$. To do this, we prove that at least one component of the vector $\overrightarrow{x}$ is picked uniformly at random *after* $\mathcal{A}$ outputs $\overrightarrow{y}$.

When $\mathcal{A}$ outputs $\overrightarrow{y}$, L contains q elements, so $\mathsf{dim}(\mathsf{Span}(L)) \leq q$. Since $q < n$, $\mathsf{Span}(1, L)$ has dimension at most $q+1$ and therefore at most n when the adversary outputs the vector $\overrightarrow{y}$. Since the dimension of $\mathsf{Span}(\mathsf{X}_1, \ldots, \mathsf{X}_n)$ is n and $1 \notin \mathsf{Span}(\mathsf{X}_1, \ldots, \mathsf{X}_n)$, we get that $\mathsf{Span}(\mathsf{X}_1, \ldots, \mathsf{X}_n)$ is not contained in $\mathsf{Span}(1, L)$. This means that there will be at least one index $i \in [1, n]$ such that $\mathsf{X}_i \notin \mathsf{Span}(1, L)$. We choose the smallest index i that verifies this. Then the oracle DLog outputs a randomly sampled value x_i when called on ξ_{j_i}. This x_i is sampled randomly after the i-th coefficient of vector $\overrightarrow{y}$ output by $\mathcal{A}$ and we obtain: $\Pr[\overrightarrow{x} = \overrightarrow{y}] \leq \frac{1}{p}$. This yields:

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_4} \leq \frac{1}{p}. \tag{3.16}$$

The theorem now follows from Equations (3.12), (3.13), (3.14), (3.15), and (3.16)

Chapter 4

One More-Computationnal Diffie-Hellman security in the Generic Group Model

4.1 OMCDH in the GGM

The OMCDH assumption (defined in Figure 2.2), though, similar to the OMDL assumption, is slightly more complex. In OMDL the adversary has access to a DLog oracle and must solve DLog challenges; in OMCDH the adversary has access to a CDH oracle and must solve CDH challenges. In particular, the CDH oracle in OMCDH enables the adversary to construct (encodings of) group elements that correspond to high-degree polynomials since on input $(\Xi(x), \Xi(y))$, the oracle returns $\Xi(xy)$, which in the "ideal" game is encoded as the polynomial XY of degree 2 by the challenger. This makes using known proof techniques in the GGM impossible, since if $\mathcal{A}$ is not constrained in terms of the degree, it can build non-zero polynomials that evaluate to zero on the challenge with non-negligible probability. (E.g., $\mathsf{X}^p - \mathsf{X}$ evaluates to 0 everywhere in $\mathbb{Z}_p$.)

Given this situation, we can neither use the Schwartz-Zippel lemma (this lemma would yield a non-negligible bound on the adversary's advantage) nor Lemma 3.1 (since it only applies for polynomials of degree 1). In fact, some cryptanalysis in the literature (e.g., the attacks by Maurer and Wolf [MW96, Mau99]) use high-degree polynomials to break DL in group of smooth order when given a CDH oracle.

Since the generic group model does not handle high-degree polynomials well, in order to analyze the hardness of OMCDH, we decided to consider a *stronger* assumption instead, which we call OMCDH^{DL} and define in Figure 4.1. This problem is analog to OMCDH, except that the CDH oracle is replaced by a DLog oracle. As the adversary has access to the same oracles as in the game OMDL, as seen in the OMDL proof, it can only build polynomials of degree at most 1. Actually, as we show in Supplementary Material 4.2, OMCDH^{DL} implies OMDL (Property 4.5) and we also prove that (modulo a polynomial number of group operations) OMCDH^{DL} implies OMCDH (Property 4.1).

In Supplementary Material 4.3 we formally prove the hardness of OMCDH^{DL} in the generic group model. This is done following the same strategy as for OMDL in Theorem 3.2 (section 3.1); the games hops are the same, only the final analysis of the last game is different, since the winning condition is different. This leads to a different winning probability at the end. This is summarized in Theorem 4.7 below.

Property 4.1 (OMCDH^{DL} implies OMCDH)**.** *In a cyclic group of order p, let $\mathcal{A}$ be an adversary that solves OMCDH by using at most m group operations and q calls to DLog. We can build an*

Game $\mathrm{OMCDHDL}^{\mathcal{A}}_{\mathsf{GrGen}}(\lambda)$	Oracle $\textsc{Chal}()$	Oracle $\mathrm{DLog}(X)$
$(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$	$x \xleftarrow{\$} \mathbb{Z}_p$; $X \coloneqq xG$	$q \coloneqq q + 1$
$\overrightarrow{Z} \coloneqq ()$; $q \coloneqq 0$	$y \xleftarrow{\$} \mathbb{Z}_p$; $Y \coloneqq yG$	$x \coloneqq \log_G(X)$
$\overrightarrow{Z'} \leftarrow \mathcal{A}^{\textsc{Chal},\mathrm{DLog}}(p, \mathbb{G}, G)$	$\overrightarrow{Z} \coloneqq \overrightarrow{Z} \,\|\, (xyG)$	**return** x
return $\left(\overrightarrow{Z} = \overrightarrow{Z'} \ \wedge \ q < \lvert\overrightarrow{x}\rvert\right)$	**return** (X, Y)	

Figure 4.1: The $\mathrm{OMCDH}^{\mathrm{DL}}$ problem.

adversary $\mathcal{B}$ that solves $OMCDH^{DL}$ using at most $m + 2q\lceil\log(p)\rceil$ group operations.

The proof is straightforward by answering the CDH oracle queries by making queries to the DLog oracle. It is formally given as Property 4.6 in Supplementary Material 4.2.

Theorem 4.2. *Let $\mathcal{A}$ be an adversary that solves $OMCDH^{DL}$ in a generic group of order p, making at most m oracle queries. Then*

$$\boldsymbol{Adv}^{\mathit{OMCDH\ GGM}}_{\mathcal{A}} \leq \frac{1}{p-1} + \frac{2m}{p} + \frac{m^2}{p-m^2} \ .$$

A formal proof of the theorem can be found in Supplementary Material 4.3. Combining this with Property 4.1, we obtain the following corollary, which proves the security of OMCDH in the generic group model.

Corollary 4.3. *Let $\mathcal{A}$ be an adversary that solves $OMCDH^{DL}$ in a generic group of order p, making at most m oracle queries and q CDH oracle queries. Then*

$$\boldsymbol{Adv}^{\mathit{OMCDH\ GGM}}_{\mathcal{A}} \leq \frac{1}{p-1} + \frac{2(m + 2q\lceil\log(p)\rceil)}{p} + \frac{(m + 2q\lceil\log(p)\rceil)^2}{p - (m + 2q\lceil\log(p)\rceil)^2} \ .$$

Adversary $\mathcal{B}_1^{\mathcal{A}_1,\textsc{Chal},\textsc{DLog}}(p,\mathbb{G},G)$	Oracle $\textsc{Chal}'()$
$\overrightarrow{Y} \coloneqq () \,;\ \overrightarrow{x} \leftarrow \mathcal{A}_1^{\textsc{Chal}',\textsc{DLog}}(1^\lambda)$	$X, Y \coloneqq \textsc{Chal}()$
return $\big(\textsc{SqMul}(x_1,Y_1),\ldots,\textsc{SqMul}(x_n,Y_n)\big)$	$\overrightarrow{Y}, \coloneqq \overrightarrow{Y} \,\|\, (Y)$
	return X

Figure 4.2: Adversary $\mathcal{B}_1$ against OMCDH$^{\text{DL}}$.

4.2 Comparison of OMCDH$^{\text{DL}}$ to Other Assumptions

We want to prove that OMCDH$^{\text{DL}}$ is easier than OMDL and OMCDH. To show these two properties, we will use the well-known square-and-multiply algorithm.

Lemma 4.4. *Let $\mathbb{G}$ be a group of order p. The square-and-multiply SqMul algorithm in $\mathbb{G}$ takes as input a scalar $a \in \mathbb{Z}_p$, and a group element $X \in G$, and returns aX after computing at most $2\lceil\log(p)\rceil$ group operations.*

Property 4.5 (OMCDH$^{\text{DL}}$ is easier than OMDL)**.** *In a cyclic group of order p, let $\mathcal{A}_1$ be an adversary that solves OMDL using at most m group operations and n challenge oracle calls. Then we can build an adversary $\mathcal{B}_1$ solving OMCDH^{DL} using at most $m + 2n\lceil\log(p)\rceil$ group operations.*

Proof We define $\mathcal{B}_1$ in Figure 4.2. $\mathcal{A}_1$ plays in OMDL and $\mathcal{B}_1$ in OMCDH$^{\text{DL}}$. $\mathcal{B}_1$ gives $\mathcal{A}_1$ access to its own oracle DLog. Then for the oracle $\textsc{Chal}'$, $\mathcal{B}_1$ makes one query to its own oracle Chal, receiving the pair (X, Y), and outputs the challenge X to $\mathcal{A}_1$. Using the oracles DLog and $\textsc{Chal}'$, $\mathcal{A}_1$ can play in OMDL.

Let q be the number of queries made by $\mathcal{A}_1$ to the oracle DLog. Then $\mathcal{B}_1$ also uses the DLog oracle q times. When $\mathcal{A}_1$ wins this game we have $n > q$ and $\overrightarrow{x} = (x_1, \ldots, x_n)$ with $x_i = \log_G(X_i)$. Since n is the number of queries made by $\mathcal{A}_1$ to $\textsc{Chal}'$, it is also the number of queries made by $\mathcal{B}_1$ to Chal. Thus, $\mathcal{B}_1$ outputs

$$(x_1Y_1, \ldots, x_nY_n) = (x_1y_1G, \ldots, x_ny_nG) \ ,$$

which is the right answer to win OMCDH$^{\text{DL}}$. And since moreover $n > q$, it implies $\mathcal{B}_1$ wins the game OMCDH$^{\text{DL}}$. We upper-bound the number of group operations of $\mathcal{B}_1$ by noting that $\mathcal{B}_1$ uses group operations only trough $\mathcal{A}_1$ and SqMul and by combining Lemma 4.4 and the fact that the number of executions of SqMul is equal to n.

Property 4.6 (OMCDH$^{\text{DL}}$ is easier than OMCDH)**.** *In a cyclic group of order p, let $\mathcal{A}_2$ be an adversary that solves OMCDH by using at most m group operations and q DLog oracle calls. Then we can build an adversary $\mathcal{B}_2$ that solves OMCDH^{DL} using at most $m + 2q\lceil\log(p)\rceil$ group operations.*

Proof We show in Figure 4.3 how to build $\mathcal{B}_2$ using $\mathcal{A}_2$. The adversary $\mathcal{B}_2$ plays in OMCDH$^{\text{DL}}$ and gives $\mathcal{A}_2$ access to the oracles CDH and Chal in order to simulate game OMCDH to $\mathcal{A}_2$. The oracle Chal is the same for $\mathcal{A}_2$, and the oracle CDH$'$ is defined using the oracle DLog that $\mathcal{B}_2$ has an access to. On input (X, Y), with $X \coloneqq xG$ and $Y \coloneqq yG$, the oracle CDH$'$ should output $Z \coloneqq xyG$. It therefore uses the oracle DLog to compute x and then uses SqMul to compute $Z \coloneqq xY$.

Let n be the number of queries made by $\mathcal{A}_2$ to Chal. Then $\mathcal{B}_2$ also calls the Chal oracle n times. When $\mathcal{A}_2$ wins this game, we have $n > q$ and $\overrightarrow{Z'} = (\text{CDH}(X_1, Y_1), \ldots, \text{CDH}(X_1, Y_n))$. Thus $\mathcal{B}_2$ wins the game OMCDH$^{\text{DL}}$.

Adversary $\mathcal{B}_2^{\mathcal{A}_2,\text{CHAL},\text{DLOG}}(p,\mathbb{G},G)$	Oracle $\text{CDH}'(X,Y)$
$\overrightarrow{Z'} \leftarrow \mathcal{A}_2^{\text{CHAL},\text{CDH}'}(1^\lambda)$	$x \coloneqq \text{DLOG}(X)$
return $(\overrightarrow{Z'})$	$Z \coloneqq \text{SQMUL}(x,Y)$
	return Z

Figure 4.3: Adversary $\mathcal{B}_2$ against OMCDH^{DL}.

We upper-bound the number of group operations of $\mathcal{B}_2$ by noting that $\mathcal{B}_2$ uses group operations only via $\mathcal{A}_2$ and SQMUL and by combining Lemma 4.4 and the fact that the number of executions of SQMUL is equal to q.

4.3 OMCDH^{DL} in the Generic Group Model

In this section we show that OMCDH^{DL} is hard in the GGM. We first argue that Game_0, defined in Figure 4.4, describes OMCDH^{DL} in the GGM. As with OMDL, we show that all the modifications we made for convenience in Game_0 do not affect the result of the game and its behavior. In fact the only important modification in Game_0 for the initial OMCDH^{DL} is at the end of the game, after the adversary outputs $\overrightarrow{Z'}$.

We see that since j_n is defined in the oracle CHAL such that $a_{j_n} = x_n$ and $a_{j_n+1} = y_n$, the final calls to DLOG:

$$x_i \coloneqq \text{DLOG}(\xi_{j_i}) \textbf{ and } y_i \coloneqq \text{DLOG}(\xi_{j_i})$$

do not change the values of x_i and y_i for all $i \in [1,n]$.

The game also defines $z_i \coloneqq x_i y_i$ and $z'_i \coloneqq \text{DLOG}(Z'_i)$, which does not change its behavior since those are new elements.

The last change from the initial OMCDH^{DL} in Game_0 is the output of the game: instead of outputting the winning condition $\overrightarrow{Z'} = (x_i y_i G)_{i\in[1,n]}$, the game outputs the condition $\overrightarrow{z'} = \overrightarrow{z}$. But those conditions are the same since for a fixed generator in a cyclic group of prime order p, comparing the discrete logarithm of two group elements is equivalent to comparing the group elements themselves.

Since $z'_i \coloneqq \text{DLOG}(Z'_i)$, we get $z'_i = a_k$ such that $\Xi(a_k) = Z'_i$ which means $\Xi(z'_i) = Z'_i$. So when we compare $\overrightarrow{z'} = \overrightarrow{z}$, we do the same comparison as $\overrightarrow{Z'} = (x_i y_i G)_{i\in[1,n]}$ but on discrete-logarithm level. This means the answer is the same. This will help us prove the following theorem:

Theorem 4.7. *Let $\mathcal{A}$ be an adversary that solves OMCDH^{DL} in a generic group of order p, making at most m oracle queries. Then*

$$\mathbf{Adv}_{\mathcal{A}}^{\text{OMCDH GGM}} \leq \frac{1}{p-1} + \frac{2m}{p} + \frac{m^2}{p-m^2} .$$

Proof The proof follows exactly the same method as the proof of OMDL in Section 3.1. It makes the same game hops with the same boundaries and uses the Lemma 3.1 the same way. So we skip all the game hops and proceed directly to the analysis of the final game, which we call Game_4 in Figure 4.5

The various game hops give us the following bound:

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_0} \leq \mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_4} + \frac{m^2}{p-m^2} . \tag{4.1}$$

Game_0

$a_0 \coloneqq 1$
$j \coloneqq 0\,;\ q \coloneqq 0\,;\ n \coloneqq 0$
$\overrightarrow{Z}' \leftarrow \mathcal{A}^{\textsc{Chal},\textsc{DLog},\textsc{GCmp}}(\textsc{Enc}())$
if $q < n$ **then return** 0
for $i \in [1,n]$
  $z'_i \coloneqq \textsc{DLog}(Z'_i)$
for $i \in [1,n]$
  $x_i \coloneqq \textsc{DLog}(\xi_{j_i})$
  $y_i \coloneqq \textsc{DLog}(\xi_{j_i+1})$
  $z_i \coloneqq x_i y_i$
return $\overrightarrow{z}' = \overrightarrow{z}$

Oracle $\textsc{Chal}()$

$j \coloneqq j+1\,;\ n \coloneqq n+1$
$x_n \xleftarrow{\$} \mathbb{Z}_p\,;\ a_j \coloneqq x_n$
$j_n \coloneqq j$
$j \coloneqq j+1$
$y_n \xleftarrow{\$} \mathbb{Z}_p\,;\ a_j \coloneqq y_n$
return $\textsc{Enc}()$

Oracle $\textsc{DLog}(\xi)$

if $\xi \notin \{\xi_i\}_{i\in[0,j]}$
  then return $\perp$
$i \coloneqq \mathsf{min}(k \in [0,j] \mid \xi = \xi_k)$
$q = q+1\,;\ v = a_i$
return v

$\textsc{Enc}(\ )$ // outputs $\xi_j \coloneqq \Xi(a_j)$

if $\exists i \in [0, j-1] : a_j = a_i$
  then $\xi_j \coloneqq \xi_i$
else
  $\xi_j \xleftarrow{\$} \{0,1\}^{\log(p)} \setminus \{\xi_i\}_{i\in[0,j-1]}$
return ξ_j

Oracle $\textsc{GCmp}(\xi, \xi', b)$

if $\xi \notin \{\xi_i\}_{i\in[0,j]}$ **or** $\xi' \notin \{\xi_i\}_{i\in[0,j]}$
  then return $\perp$
$i \coloneqq \mathsf{min}(\ k \in [0,j] \mid \xi = \xi_k)$
$i' \coloneqq \mathsf{min}(\ k \in [0,j] \mid \xi' = \xi_k)$
$j \coloneqq j+1\,;\ a_j \coloneqq a_i + (-1)^b a_{i'}$
return $\textsc{Enc}()$

Figure 4.4: Game_0 represents the GGM version of the game OMCDH$^{\mathrm{DL}}$, with small modifications inspired by the proof of OMDL, which are useful in the proof of Theorem 4.7.

Now we analyze Game_4 described in Figure 4.5. This game looks a lot like the final game of OMDL (Figure 3.4): the challenges x_i and y_i are no longer defined, until the adversary makes some calls to DLog or until the game defines them at the end. We show that the adversary cannot predict some of the challenges that are assigned randomly at the end.

In this analysis we need to suppose that all the challenges x_i, y_i will be different from 0. Note that in OMCDH$^{\mathrm{DL}}$ and also in OMCDH, if we have $x_i = 0$ or $y_i = 0$, it becomes easy for the adversary to win since $x_i y_i = 0$ and the adversary can output $1_{\mathbb{G}}$. Since in Game_4 all the discrete logarithms are picked uniformly at random, we can deduce that x_i and y_i have a probability at most $\frac{1}{p}$ to be equal to zero. Another way to obtain this result would be to consider the fact that the distribution of the games are not modified and by doing the game hops from Game_1 to Game_4, as in OMDL. This way, in Game_4 the distribution of the challenge is the same as in Game_1, which is the same as Game_0 and in those games, x_i and y_i are defined uniformly at random, which means they are equal to 0 with probability $\frac{1}{p}$. We call E the event "$\exists i \in [1,n]$ **such that** $x_i = 0$ **or** $y_i = 0$".

Since there are n challenges x_i and n challenges y_i, we get:

$$\Pr[E] \leq 1 - \left(1 - \frac{1}{p}\right)^{2n} \leq 1 - \left(1 - \frac{1}{p}\right)^{2m},$$

and since $\frac{2m}{p}$ is small, using $\left(1 - \frac{1}{p}\right)^{2m} \geq 1 - \frac{2m}{p}$, we get:

$$Pr[E] \leq \frac{2m}{p}\,. \tag{4.2}$$

Now we suppose that we are not in event E. Since $q < n$, we know that L contains at most $n-1$ elements when $\mathcal{A}$ outputs its answers. Then the challenger adds n polynomials to L by making n queries to the oracle DLog, which are $z'_i \coloneqq \textsc{DLog}(Z'_i)$.

This means that just before defining the challenges x_i and y_i, L contains at most $2n-1$ elements, which yields that $\mathsf{Span}(1, L)$ has dimension at most $2n$. But since the dimension of

Game_4

$j \coloneqq 0\,;\ q \coloneqq 0\,;\ n \coloneqq 0$
$P_0 \coloneqq 1\,;\ L \coloneqq \emptyset$
$\overrightarrow{Z}' \leftarrow \mathcal{A}^{\textsc{Chal},\textsc{DLog},\textsc{GCmp}}(\textsc{Enc}())$
if $q < n$ **then return** 0
for $i \in [1,n]$
  $z'_i \coloneqq \textsc{DLog}(Z'_i)$
for $i \in [1,n]$
  $x_i \coloneqq \textsc{DLog}(\xi_{j_i})$
  $y_i \coloneqq \textsc{DLog}(\xi_{j_i})$
  $z_i \coloneqq x_i y_i\,;$
return $\overrightarrow{z'} = \overrightarrow{z}$

Oracle $\textsc{Chal}()$

$j \coloneqq j+1\,;\ n \coloneqq n+1$
$P_j \coloneqq \mathsf{X}_n\,;\ j_n \coloneqq j$
$\xi \coloneqq \textsc{Enc}()$
$j \coloneqq j+1\,;\ P_j \coloneqq \mathsf{Y}_n$
$\xi' \coloneqq \textsc{Enc}()$
return ξ, ξ'

Oracle $\textsc{DLog}(\xi)$

if $\xi \notin \{\xi_i\}_{i\in[0,j]}$
  then return $\perp$
$i \coloneqq \mathsf{min}(k \in [0,j] \mid \xi = \xi_k)$
$q = q+1\,;\ v \xleftarrow{\$} \mathbb{Z}_p$
if $P_i \in \mathsf{Span}(1, L)$ **then**
  get $(\alpha_k)_k \in \mathbb{Z}_p^q$ **s.t.**
  $P_i = \alpha_0 + \sum_{k=1}^{q-1} \alpha_k L_k$
  $x \coloneqq \alpha_0$
$L_q \coloneqq P_i - v\,;\ L = L \cup \{P_i - v\}$
if $\exists (i_1, i_2) \in [0,j]^2$:
  $P_{i_1} - P_{i_2} \in \mathsf{Span}(L)$
  and $\xi_{i_1} \neq \xi_{i_2}$
  then abort game
return v

$\textsc{Enc}(\ )$

if $\exists i \in [0, j-1] : P_j - P_i \in \mathsf{Span}(L)$
 then $\xi_j \coloneqq \xi_i$
else
  $\xi_j \xleftarrow{\$} \{0,1\}^{\log(p)} \setminus \{\xi_i\}_{i\in[0,j-1]}$
return ξ_j

Oracle $\textsc{GCmp}(\xi, \xi', b)$

if $\xi \notin \{\xi_i\}_{i\in[0,j]}$ **or** $\xi' \notin \{\xi_i\}_{i\in[0,j]}$
  then return $\perp$
$i \coloneqq \mathsf{min}(\ k \in [0,j] \mid \xi = \xi_k)$
$i' \coloneqq \mathsf{min}(\ k \in [0,j] \mid \xi' = \xi_k)$
$j \coloneqq j+1$
$P_j \coloneqq P_i + (-1)^b P_{i'}$
return $\textsc{Enc}()$

Figure 4.5: Game_4 is the final game in which the challenger simulates OMCDH^{DL} to the adversary by using only polynomials.

$\mathsf{Span}(\mathsf{X}_1, \ldots, \mathsf{X}_n, \mathsf{Y}_1, \ldots, \mathsf{Y}_n)$ is exactly $2n$ and $1 \notin \mathsf{Span}(\mathsf{X}_1, \ldots, \mathsf{X}_n, \mathsf{Y}_1, \ldots, \mathsf{Y}_n)$, there must exist an index i such that $\mathsf{X}_i \notin \mathsf{Span}(1, L)$ or $\mathsf{Y}_i \notin \mathsf{Span}(1, L)$. Without loss of generality, we suppose that $\mathsf{X}_i \notin \mathsf{Span}(1, L)$.

This means that on the call $x_i \coloneqq \textsc{DLog}(\xi_{j_i})$, x_i is picked uniformly at random. Since z'_i is fixed before that we have either:

- y_i was fixed by the game with the adversary and we get $\Pr[x_i = \frac{z'_i}{y_i}] \leq \frac{1}{p-1}$ (it is an equality when $z'_i \neq 0$). (We know that $y_i \neq 0$ since we are not on event E and we get $p-1$ instead of p since we know that x_i can't be 0). Or:
- y_i is defined uniformly at random as x_i, which implies the product $x_i y_i$ is also picked uniformly at random over $\mathbb{Z}_p^*$ and we obtain $\Pr[\, x_i y_i = z'_i \,] \leq \frac{1}{p-1}$ as before.

In the end, we get:

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{Game}_4} \leq \Pr[\, E \,] + \frac{1}{p-1}\ . \tag{4.3}$$

The equations (4.1), (4.2) and (4.3) together give the result of the theorem.

Chapter 5

Blind Schnorr signatures in the Algebraic Group Model

In this Chapter we first define the Schnorr signatures, and prove their unforgeability by using a reduction to the DL assumption in the AGM. This proof is simple but will give an insight on how to build the security proofs for Blind Schnorr signatures and Clause Blind Schnorr Signature which are more complex.

Then we define the Blind Schnorr signatures and the ROS assumption and we give a proof that the unforgeability of this signature scheme can be reduced to the ROS assumption and the OMDL problem in the AGM with the ROM (Random Oracle Model).

5.1 Schnorr Signatures

Definitions

A signature scheme SIG consists of the following algorithms:

- $par \leftarrow \mathsf{SIG.Setup}(1^\lambda)$: the setup algorithm takes as input the security parameter λ in unary and outputs public parameters par;
- $(sk, pk) \leftarrow \mathsf{SIG.KeyGen}(par)$: the key generation algorithm takes parameters par and outputs a secret key sk and a public key pk;
- $\sigma \leftarrow \mathsf{SIG.Sign}(sk, m)$: the signing algorithm takes as input a secret key sk and a message $m \in \{0,1\}^*$ and outputs a signature σ;
- $b \leftarrow \mathsf{SIG.Ver}(pk, m, \sigma)$: the (deterministic) verification algorithm takes a public key pk, a message m, and a signature σ; it returns 1 if σ is valid and 0 otherwise.

Correctness requires that for any λ and any message m, when running $par \leftarrow \mathsf{SIG.Setup}(1^\lambda)$, $(sk, pk) \leftarrow \mathsf{SIG.KeyGen}(par)$, $\sigma \leftarrow \mathsf{SIG.Sign}(sk, m)$, and $b \leftarrow \mathsf{SIG.Ver}(pk, m, \sigma)$, one has $b = 1$ with probability 1. The standard security notion for a signature scheme is *existential unforgeability under chosen-message attack* (EUF-CMA), formalized via game EUF-CMA, which we recall in Figure 5.1. The Schnorr signature scheme [Sch91] is specified in Figure 5.2.

Security of Schnorr Signatures in the AGM

As a warm-up and to introduce some of the techniques used later, we reduce security of Schnorr signatures to hardness of DL in the AGM+ROM.

Game $\text{EUF-CMA}^{\mathcal{A}}_{\mathsf{SIG}}(\lambda)$	Oracle $\textsc{Sign}(m)$
$par \leftarrow \mathsf{SIG.Setup}(1^\lambda)$	$\sigma \leftarrow \mathsf{SIG.Sign}(sk, m)$
$(sk, pk) \leftarrow \mathsf{SIG.KeyGen}(par)$; $\mathsf{Q} := ()$	$\mathsf{Q} := \mathsf{Q} \,\|\|\, (m)$
$(m^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sign}}(pk)$	**return** σ
return $\big(m^* \notin \mathsf{Q} \wedge \mathsf{SIG.Ver}(pk, m^*, \sigma^*)\big)$	

Figure 5.1: The EUF-CMA security game for a signature scheme SIG.

$\mathsf{Sch.Setup}(1^\lambda)$	$\mathsf{Sch.KeyGen}(par)$
$(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$	$(p, \mathbb{G}, G, \mathsf{H}) := par$; $x \xleftarrow{\$} \mathbb{Z}_p$; $X := xG$
Select $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_p$	$sk := (par, x)$; $pk := (par, X)$
return $par := (p, \mathbb{G}, G, \mathsf{H})$	**return** (sk, pk)
$\mathsf{Sch.Sign}(sk, m)$	$\mathsf{Sch.Ver}(pk, m, \sigma)$
$(p, \mathbb{G}, G, \mathsf{H}, x) := sk$; $r \xleftarrow{\$} \mathbb{Z}_p$; $R := rG$	$(p, \mathbb{G}, G, \mathsf{H}, X) := pk$; $(R, s) := \sigma$
$c := \mathsf{H}(R, m)$; $s := r + cx \bmod p$	$c := \mathsf{H}(R, m)$
return $\sigma := (R, s)$	**return** $(sG = R + cX)$

Figure 5.2: The Schnorr signature scheme $\mathsf{Sch}[\mathsf{GrGen}]$ based on a group generator GrGen.

Theorem 5.1. *Let* GrGen *be a group generator. Let* $\mathcal{A}_{\text{alg}}$ *be an algebraic adversary against the EUF-CMA security of the Schnorr signature scheme* $\mathsf{Sch}[\mathsf{GrGen}]$ *running in time at most* τ *and making at most* q_{s} *signature queries and* q_{h} *queries to the random oracle. Then there exists an algorithm* $\mathcal{B}$ *solving the DL problem w.r.t.* GrGen*, running in time at most* $\tau + O(q_{\text{s}} + q_{\text{h}})$*, such that*

$$\boldsymbol{Adv}^{euf\text{-}cma}_{\mathsf{Sch}[\mathsf{GrGen}],\mathcal{A}_{\text{alg}}} \leq \boldsymbol{Adv}^{DL}_{\mathsf{GrGen},\mathcal{B}} + \frac{q_{\text{s}}(q_{\text{s}} + q_{\text{h}}) + 1}{2^{\lambda-1}} \ .$$

We start with some intuition for the proof. In the random oracle model, Schnorr signatures can be simulated without knowledge of the secret key by choosing random c and s, setting $R := sG - cX$ and then programming the random oracle so that $\mathsf{H}(R, m) = c$. On the other hand, an adversary that returns a signature forgery $(m^*, (R^*, s^*))$ can be used to compute the discrete logarithm of the public key X. In the ROM this can be proved by rewinding the adversary and using the Forking Lemma [PS96b, PS00], which entails a security loss.

In the AGM+ROM, extraction is straight-line and the security proof thus tight. After querying the signing oracle on messages $m_1, \ldots, m_{q_{\text{s}}}$, the adversary obtains $(R_i, s_i)_{1 \leq i \leq q_{\text{s}}}$ that also verify $R_i = s_i G - c_i X$ with $c_i = \mathsf{H}(R_i, m_i)$. A valid forgery satisfies $R^* = s^* G - c^* X$, with $c^* := \mathsf{H}(R^*, m^*)$. On the other hand, since the adversary is algebraic, when it made its first query $\mathsf{H}(R^*, m^*)$, it provided a representation of R^* in basis $(G, X, \overrightarrow{R})$ with $\overrightarrow{R} = (R_i)_{1 \leq i \leq q_{\text{s}}}$, that is, $(\gamma^*, \xi^*, \overrightarrow{\rho^*})$ with $R^* = \gamma^* G + \xi^* X + \sum_{i=1}^{q_{\text{s}}} \rho_i^* R_i = \gamma^* G + \xi^* X + \sum_{i=1}^{q_{\text{s}}} s_i G - \sum_{i=1}^{q_{\text{s}}} c_i X$. Together, these equations yield

$$(c^* + \xi^* - \textstyle\sum_{i=1}^{q_{\text{s}}} \rho_i^* c_i) X = (s^* - \gamma^* - \sum_{i=1}^{q_{\text{s}}} \rho_i^* s_i) G \ .$$

Since c^* was chosen at random *after* the adversary chose $\xi^*, \rho_1^*, \ldots$ and $\rho_{q_{\text{s}}}^*$, the probability that $c^* + \xi^* - \sum_{i=1}^{q_{\text{s}}} \rho_i^* c_i \not\equiv_p 0$ is overwhelming, in which case we can compute the discrete logarithm of X from the above equation.

Proof

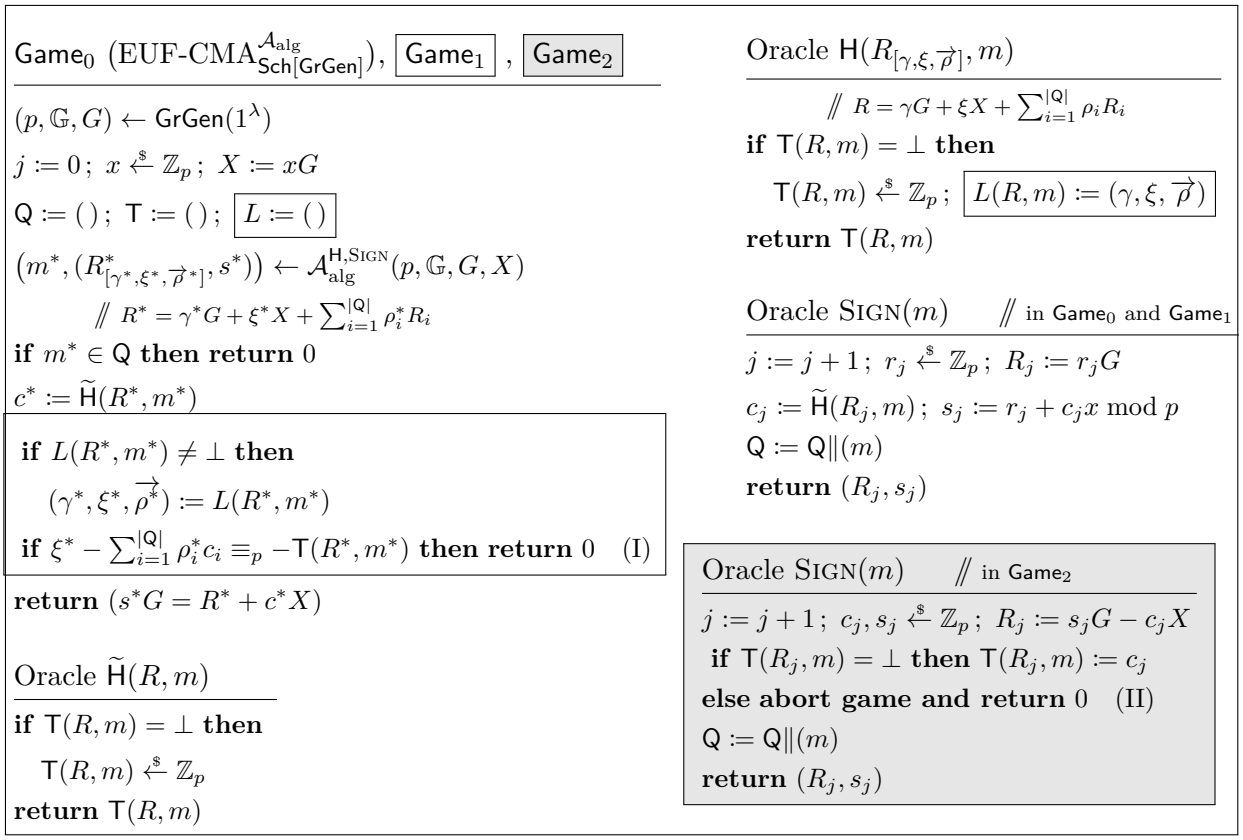

Figure 5.3: Games in the proof of Theorem 5.1. Game$_0$ is defined by ignoring all boxes; boxes are included in Game$_1$ and Game$_2$; Gray boxes are only included in Game$_2$.

[Proof of Theorem 5.1] Let $\mathcal{A}_{\text{alg}}$ be an algebraic adversary in EUF-CMA$_{\text{Sch[GrGen]}}$ that makes at most q_{s} signature queries and q_{h} RO queries. We proceed by a sequence of games specified in Figure 5.3.

Game$_0$. The first game is EUF-CMA (Figure 5.1) for the Schnorr signature scheme (Figure 5.2) with a random oracle H. The game maintains a list Q of queried messages and T of values sampled for H. To prepare the change to Game$_1$, we have written the finalization of the game in an equivalent way: it first checks that $m^* \notin \mathsf{Q}$ and then runs $\mathsf{Sch.Ver}(pk, m^*, (R^*, s^*))$, which we have written explicitly. Since the adversary is algebraic, it must provide a representation $(\gamma^*, \xi^*, \overrightarrow{\rho^*})$ for its forgery $(m^*, (R^*_{[\gamma^*,\xi^*,\overrightarrow{\rho^*}]}, s^*)$ such that $R^* = \gamma^* G + \xi^* X + \sum_{i=1}^{|\mathsf{Q}|} \rho_i^* R_i$, and similarly for each RO query $\mathsf{H}(R_{[\gamma,\xi,\overrightarrow{\rho}]}, m)$. By definition,

$$\mathbf{Adv}^{\text{Game}_0}_{\mathcal{A}_{\text{alg}}} = \mathbf{Adv}^{euf\text{-}cma}_{\text{Sch[GrGen]},\mathcal{A}_{\text{alg}}} \,. \tag{5.1}$$

Game$_1$. In Game$_1$ we introduce an auxiliary table L that for each query $\mathsf{H}(R_{[\gamma,\xi,\overrightarrow{\rho}]}, m)$ stores the representation $(\gamma, \xi, \overrightarrow{\rho})$ of R. Second, when the adversary returns its forgery $(m^*, (R^*_{[\gamma^*,\xi^*,\overrightarrow{\rho^*}]}, s^*))$ and previously made a query $\mathsf{H}(R^*_{[\gamma',\xi',,\overrightarrow{\rho'}]}, m^*)$ for some $(\gamma', \xi', \overrightarrow{\rho'})$, then we consider this previous representation of R^*, that is, we set $(\gamma^*, \xi^*, \overrightarrow{\rho^*}) := (\gamma', \xi', \overrightarrow{\rho'})$. The only actual difference to Game$_0$ is that Game$_1$ returns 0 in case $\xi^* - \sum_{i=1}^{|\mathsf{Q}|} \rho_i^* c_i \equiv_p -\mathsf{T}(R^*, m^*)$ (line (I)).

We show that this happens with probability $1/p \le 1/2^{\lambda-1}$. First note that line (I) is only executed if $m^* \notin \mathsf{Q}$, as otherwise the game would already have returned 0. Hence $\mathsf{T}(R^*, m^*)$ can only have been defined either (1) during a call to H by the adversary or (2), if it is still undefined when $\mathcal{A}_{\text{alg}}$ stops, by the game when defining c^*. In both cases the probability of returning 0 in line (I) is $1/p$:

(1) If $\mathsf{T}(R^*, m^*)$ was defined during a H query of the form $\mathsf{H}(R^*_{[\gamma',\xi',\overrightarrow{\rho'}]}, m^*)$ then $\mathsf{T}(R^*, m^*)$ is drawn uniformly at random and independently from ξ', $\overrightarrow{\rho'}$ and values $(c_i)_{1\le i\le|\mathsf{Q}|}$. Since then $L(R^*, m^*) \neq \bot$, the game sets $\xi^* := \xi'$, $\overrightarrow{\rho'} := \overrightarrow{\rho^*}$ and hence $\xi^* - \sum_{i=1}^{|\mathsf{Q}|} \rho_i^* c_i \equiv_p -\mathsf{T}(R^*, m^*)$ holds with probability exactly $1/p$. (2) If $\mathsf{T}(R^*, m^*)$ is only defined after the adversary output ξ^* and $\overrightarrow{\rho^*}$ then again we have $\xi^* - \sum_{i=1}^{|\mathsf{Q}|} \rho_i^* c_i \equiv_p -\mathsf{T}(R^*, m^*)$ with probability $1/p$. Hence,

$$\mathbf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}_{\text{alg}}} \ge \mathbf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}_{\text{alg}}} - \frac{1}{2^{\lambda-1}} \ . \tag{5.2}$$

$\underline{\mathsf{Game}_2}$. In the final game we use the standard method for Schnorr signatures of simulating the SIGN oracle without the secret key x by programming the random oracle. Game_1 and Game_2 are identical unless Game_2 returns 0 in line (II). For each signature query, R_j is uniformly random, and the size of table T is at most $q_{\text{s}} + q_{\text{h}}$, hence the game aborts in line (II) with probability at most $(q_{\text{s}} + q_{\text{h}})/p \le (q_{\text{s}} + q_{\text{h}})/2^{\lambda-1}$. By summing over the at most q_{s} signature queries, we have

$$\mathbf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}_{\text{alg}}} \ge \mathbf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}_{\text{alg}}} - \frac{q_{\text{s}}(q_{\text{s}} + q_{\text{h}})}{2^{\lambda-1}} \ . \tag{5.3}$$

$\underline{\text{REDUCTION TO DL}}$. We now construct an adversary $\mathcal{B}$ solving DL with the same probability as $\mathcal{A}_{\text{alg}}$ wins Game_2. On input group description $(p, \mathbb{G}, G)$ and X, the adversary runs $\mathcal{A}_{\text{alg}}$ on input $(p, \mathbb{G}, G, X)$ and simulates Game_2, which can be done without knowledge of $\log_G(X)$. Assume that the adversary wins Game_2 by returning (m^*, R^*, s^*) and let $c^* := \mathsf{T}(R^*, m^*)$ and $(\gamma^*, \xi^*, \overrightarrow{\rho^*})$ be defined as in the game. Thus, $\xi^* - \sum_{i=1}^{|\mathsf{Q}|} \rho_i^* c_i \neq -c^* \bmod p$ and $R^* = \gamma^* G + \xi^* X + \sum_{i=1}^{|\mathsf{Q}|} \rho_i^* R_i$; moreover, validity of the forgery implies that $s^* G = R^* + c^* X$. Hence, $(s^* - \gamma^* - \sum_{i=1}^{|\mathsf{Q}|} \rho_i^* s_i)G = (\xi^* - \sum_{i=1}^{|\mathsf{Q}|} \rho_i^* c_i + c^*)X$ and $\mathcal{B}$ can compute

$$\log X = (s^* - \gamma^* - \textstyle\sum_{i=1}^{|\mathsf{Q}|} \rho_i^* s_i)(\xi^* - \sum_{i=1}^{|\mathsf{Q}|} \rho_i^* c_i + c^*)^{-1} \bmod p \ .$$

Combining this with Eqs. (5.1)–(5.3), we have

$$\mathbf{Adv}^{DL}_{\mathsf{GrGen},\mathcal{B}} = \mathbf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}_{\text{alg}}} \ge \mathbf{Adv}^{euf\text{-}cma}_{\mathsf{Sch}[\mathsf{GrGen}],\mathcal{A}_{\text{alg}}} - \frac{q_{\text{s}}(q_{\text{s}} + q_{\text{h}}) + 1}{2^{\lambda-1}} \ .$$

Assuming that scalar multiplications in $\mathbb{G}$ and assignments in tables T and L take unit time, the running time of $\mathcal{B}$ is $\tau + O(q_{\text{s}} + q_{\text{h}})$.

5.2 Blind Schnorr Signatures

Definitions

We start with defining the syntax and security of blind signature schemes and focus on schemes with a 2-round (i.e., 4 messages) signing protocol for concreteness.

Game $\mathrm{UNF}^{\mathcal{A}}_{\mathsf{BS}}(\lambda)$

$par \leftarrow \mathsf{BS.Setup}(1^\lambda)$
$(sk, pk) \leftarrow \mathsf{BS.KeyGen}(par)$
$\mathsf{ctr}_1 \coloneqq 0\,;\ \mathsf{ctr}_2 \coloneqq 0\,;\ \mathcal{S} \coloneqq \emptyset$
$(m_i^*, \sigma_i^*)_{i\in[n]} \leftarrow \mathcal{A}^{\textsc{Sign}_1,\textsc{Sign}_2}(pk)$
return $\big(\mathsf{ctr}_2 < n$
$\quad \wedge\ \forall i \neq j \in [n] : (m_i^*, \sigma_i^*) \neq (m_j^*, \sigma_j^*)$
$\quad \wedge\ \forall i \in [n] : \mathsf{BS.Ver}(pk, m_i^*, \sigma_i^*) = 1\big)$

Oracle $\textsc{Sign}_1(msg)$

$\mathsf{ctr}_1 \coloneqq \mathsf{ctr}_1 + 1$ // session id
$(msg', state_{\mathsf{ctr}_1}) \leftarrow \mathsf{BS.Sign}_1(sk, msg)$
$\mathcal{S} \coloneqq \mathcal{S} \cup \{\mathsf{ctr}_1\}$ // open sessions
return (ctr_1, msg')

Oracle $\textsc{Sign}_2(j, msg)$

if $j \notin \mathcal{S}$ **then return** $\perp$
$(msg', b) \leftarrow \mathsf{BS.Sign}_2(state_j, msg)$
if $b = 1$ **then** $\mathcal{S} \coloneqq \mathcal{S} \setminus \{j\}\,;\ \mathsf{ctr}_2 \coloneqq \mathsf{ctr}_2 + 1$
return msg'

Figure 5.4: The (strong) unforgeability game for a blind signature scheme BS with a 2-round signing protocol.

Syntax. A blind signature scheme BS consists of the following algorithms:

- $par \leftarrow \mathsf{BS.Setup}(1^\lambda)$: the setup algorithm takes the security parameter λ in unary and returns public parameters par;
- $(sk, pk) \leftarrow \mathsf{BS.KeyGen}(par)$: the key generation algorithm takes the public parameters par and returns a secret/public key pair (sk, pk);
- $(b, \sigma) \leftarrow \langle \mathsf{BS.Sign}(sk), \mathsf{BS.User}(pk, m)\rangle$: an interactive protocol is run between the signer with private input a secret key sk and the user with private input a public key pk and a message m; the signer outputs $b = 1$ if the interaction completes successfully and $b = 0$ otherwise, while the user outputs a signature σ if it terminates correctly, and $\perp$ otherwise. For a 2-round protocol the interaction can be realized by the following algorithms:

$$\begin{aligned}(msg_{U,0}, state_{U,0}) &\leftarrow \mathsf{BS.User}_0(pk, m)\\ (msg_{S,1}, state_S) &\leftarrow \mathsf{BS.Sign}_1(sk, msg_{U,0})\\ (msg_{U,1}, state_{U,1}) &\leftarrow \mathsf{BS.User}_1(state_{U,0}, msg_{S,1})\\ (msg_{S,2}, b) &\leftarrow \mathsf{BS.Sign}_2(state_S, msg_{U,1})\\ \sigma &\leftarrow \mathsf{BS.User}_2(state_{U,1}, msg_{S,2})\end{aligned}$$

  (Typically, $\mathsf{BS.User}_0$ just initiates the session, and thus $msg_{U,0} = (\,)$ and $state_{U,0} = (pk, m)$.)
- $b \leftarrow \mathsf{BS.Ver}(pk, m, \sigma)$: the (deterministic) verification algorithm takes a public key pk, a message m, and a signature σ, and returns 1 if σ is valid on m under pk and 0 otherwise.

Correctness requires that for any λ and any message m, when running $par \leftarrow \mathsf{BS.Setup}(1^\lambda)$, $(sk, pk) \leftarrow \mathsf{BS.KeyGen}(par)$, $(b, \sigma) \leftarrow \langle \mathsf{BS.Sign}(sk), \mathsf{BS.User}(pk, m)\rangle$, and $b' \leftarrow \mathsf{BS.Ver}(pk, m, \sigma)$, we have $b = 1 = b'$ with probability 1.

Unforgeability. The standard security notion for blind signatures demands that no user, after interacting arbitrary many times with a signer and k of these interactions were considered successful by the signer, can produce more than k signatures. Moreover, the adversary can schedule and interleave its sessions with the signer in any arbitrary way.

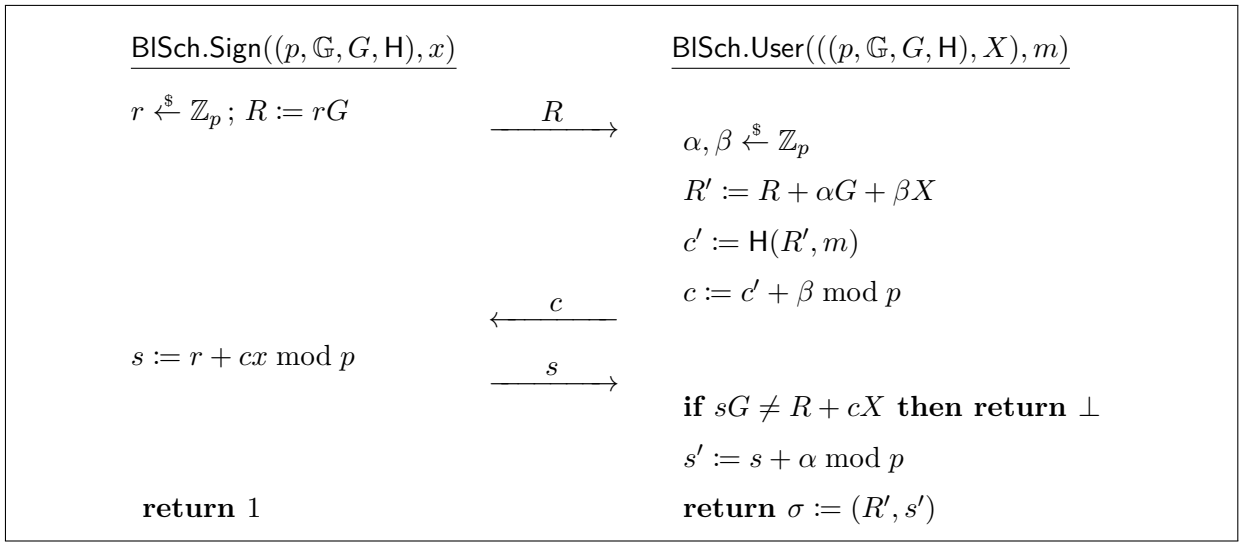

Figure 5.5: The signing protocol of the blind Schnorr signature scheme.

In game $\text{UNF}^{\mathcal{A}}_{\mathsf{BS}}$ defined in Figure 5.4 the adversary has access to two oracles SIGN_1 and SIGN_2 corresponding to the two phases of the interactive protocol. The game maintains two counters ctr_1 and ctr_2 (initially set to 0), where ctr_1 is used as session identifier, and a set $\mathcal{S}$ of "open" sessions. Oracle SIGN_1 takes the user's first message (which for blind Schnorr signatures is empty), increments ctr_1, adds ctr_1 to $\mathcal{S}$ and runs the first round on the signer's side, storing its state as $state_{\mathsf{ctr}_1}$. Oracle SIGN_2 takes as input a session identifier j and a user message; if $j \in \mathcal{S}$, it runs the second round on the signer's side; if successful, it removes j from $\mathcal{S}$ and increments ctr_2, which thus represents the number of successful interactions.

We say that BS satisfies unforgeability if $\mathbf{Adv}^{UNF}_{\mathsf{BS},\mathcal{A}}$ is negligible for all p.p.t. adversaries $\mathcal{A}$. Note that we consider "strong" unforgeability, which only requires that all pairs (m_i^*, σ_i^*) returned by the adversary (rather than all messages m_i^*) are distinct.

BLINDNESS. Blindness requires that a signer cannot link a message/signature pair to a particular execution of the signing protocol. Formally, the adversary chooses two messages m_0 and m_1 and the experiment runs the signing protocol acting as the user with the adversary, first obtaining a signature σ_b on m_b and then σ_{1-b} on m_{1-b} for a random bit b. If both signatures are valid, the adversary is given (σ_0, σ_1) and must determine the value of b. A formal definition can be found in section 6.4

BLIND SCHNORR SIGNATURES. A blind signature scheme BlSch is obtained from the scheme Sch in Figure 5.2 by replacing $\mathsf{Sch.Sign}$ with the interactive protocol specified in Figure 5.5 (the first message $msg_{U,0}$ from the user to the signer is empty and is not depicted). Correctness follows since a signature (R', s') obtained by the user after interacting with the signer satisfies $\mathsf{Sch.Ver}$:

$$\begin{aligned} s'G = sG + \alpha G = (r + cx)G + \alpha G = R + \alpha G + \beta X + (-\beta + c)X \\ = R' + c'X = R' + \mathsf{H}(R', m)\, X \ . \end{aligned}$$

Moreover, Schnorr signatures achieve perfect blindness [CP93].

The ROS Problem

The security of blind Schnorr signatures is related to the ROS (Random inhomogeneities in an Overdetermined, Solvable system of linear equations) problem, which was introduced by

Game $\mathrm{ROS}^{\mathcal{A}}_{\mathsf{GrGen},\ell,\Omega}(\lambda)$

$(p,\mathbb{G},G) \leftarrow \mathsf{GrGen}(1^\lambda)\,;\ \mathsf{T}_{\mathrm{ros}} := ()$
$\big((\overrightarrow{\rho}_i, \mathsf{aux}_i)_{i\in[\ell+1]}, (c_j)_{j\in[\ell]}\big) \leftarrow \mathcal{A}^{\mathsf{H}_{\mathrm{ros}}}(p)$
$\quad /\!/\ \overrightarrow{\rho}_i = (\rho_{i,1},\ldots,\rho_{i,\ell})$
return $\big(\forall\, i \neq i' \in [\ell+1] : (\overrightarrow{\rho}_i, \mathsf{aux}_i) \neq (\overrightarrow{\rho}_{i'}, \mathsf{aux}_{i'})$
$\quad \wedge\ \forall\, i \in [\ell+1] : \sum_{j=1}^{\ell} \rho_{i,j} c_j \equiv_p \mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho_i}, \mathsf{aux}_i)\big)$

Oracle $\mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}, \mathsf{aux})$

if $\mathsf{T}_{\mathrm{ros}}(\overrightarrow{\rho}, \mathsf{aux}) = \bot$ **then**
$\quad \mathsf{T}_{\mathrm{ros}}(\overrightarrow{\rho}, \mathsf{aux}) \xleftarrow{\$} \mathbb{Z}_p$
return $\mathsf{T}_{\mathrm{ros}}(\overrightarrow{\rho}, \mathsf{aux})$

Figure 5.6: The ROS game, where $\mathsf{H}_{\mathrm{ros}} \colon (\mathbb{Z}_p)^\ell \times \Omega \to \mathbb{Z}_p$ is a random oracle.

Schnorr [Sch01]. Consider the game $\mathrm{ROS}_{\mathsf{GrGen},\ell,\Omega}$ in Figure 5.6, parameterized by a group generator GrGen,[1] an integer $\ell \geq 1$, and a set Ω (we omit GrGen and Ω from the notation when they are clear from context). The adversary $\mathcal{A}$ receives a prime p and has access to a random oracle $\mathsf{H}_{\mathrm{ros}}$ taking as input $(\overrightarrow{\rho}, \mathsf{aux})$ where $\overrightarrow{\rho} \in (\mathbb{Z}_p)^\ell$ and $\mathsf{aux} \in \Omega$. Its goal is to find $\ell+1$ distinct pairs $(\overrightarrow{\rho}_i, \mathsf{aux}_i)_{i\in[\ell+1]}$ together with a solution $(c_j)_{j\in[\ell]}$ to the linear system $\sum_{j=1}^{\ell} \rho_{i,j} c_j \equiv_p \mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho_i}, \mathsf{aux}_i)$, $i \in [\ell+1]$.[2]

The lemma below, which refines Schnorr's observation [Sch01], shows how an algorithm $\mathcal{A}$ solving the ROS_ℓ problem can be used to break the one-more unforgeability of blind Schnorr signatures.

Lemma 5.2. *Let* GrGen *be a group generator. Let* $\mathcal{A}$ *be an algorithm for game* $\mathrm{ROS}_{\mathsf{GrGen},\ell,\Omega}$*, where* $\Omega = (\mathbb{Z}_p)^2 \times \{0,1\}^*$*, running in time at most* τ *and making at most* q_{h} *random oracle queries. Then there exists an (algebraic) adversary* $\mathcal{B}$ *running in time at most* $\tau + O(\ell + q_{\mathrm{h}})$*, making at most* ℓ *queries to* $\textsc{Sign}_1$ *and* $\textsc{Sign}_2$ *and* q_{h} *random oracle queries, such that*

$$\boldsymbol{Adv}^{unf}_{\mathsf{BlSch}[\mathsf{GrGen}],\mathcal{B}} \geq \boldsymbol{Adv}^{ros}_{\mathsf{GrGen},\ell,\Omega,\mathcal{A}} - \frac{q_{\mathrm{h}}^2 + (\ell+1)^2}{2^{\lambda-1}}\ .$$

Proof : We first consider a slightly modified game $\mathrm{ROS}'_{\mathsf{GrGen},\ell,\Omega}$, which differs from ROS in that it first draws $x, r_1, \ldots, r_\ell \xleftarrow{\$} \mathbb{Z}_p$ and returns 0 if one of the following two events occurs:

- E_1: when $\mathcal{A}$ queries $\mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}, (\gamma, \xi, m))$, there has been a previous query $\mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}\,', (\gamma', \xi', m'))$ such that $m = m'$ and
$$\gamma + \xi x + \textstyle\sum_{j=1}^{\ell} \rho_j r_j \equiv_p \gamma' + \xi' x + \sum_{j=1}^{\ell} \rho'_j r_j\ ;$$
- E_2: when $\mathcal{A}$ returns $\big((\overrightarrow{\rho}_i, (\gamma_i, \xi_i, m_i))_{i\in[\ell+1]}, (c_j)_{j\in[\ell]}\big)$, there exists $i \neq i' \in [\ell+1]$ such that $m_i = m_{i'}$ and
$$\gamma_i + \xi_i x + \textstyle\sum_{j=1}^{\ell} \rho_{i,j} r_j \equiv_p \gamma_{i'} + \xi_{i'} x + \sum_{j=1}^{\ell} \rho_{i',j} r_j\ .$$

Games ROS and ROS′ are identical unless E_1 or E_2 occurs in ROS′. Note that we could defer the random selection of $x, r_1, \ldots, r_\ell$ and the check whether E_1 or E_2 occurred to the very end of the game. Consider two *distinct* random oracle queries $(\overrightarrow{\rho}, (\gamma, \xi, m))$ and $(\overrightarrow{\rho}\,', (\gamma', \xi', m'))$; if $m \neq m'$ then E_1 cannot occur; if $m = m'$, then $(\gamma, \xi, \overrightarrow{\rho}) \neq (\gamma', \xi', \overrightarrow{\rho}\,')$ and by the Schwartz-Zippel Lemma,

$$(\gamma - \gamma') + (\xi - \xi')x + \textstyle\sum_{j=1}^{\ell} (\rho_j - \rho'_j) r_j \equiv_p 0$$

[1] The group generator GrGen is only used to generate a prime p of length λ; the group $\mathbb{G}$ is not used in the game.
[2] The original definition of the problem by Schnorr [Sch01] sets $\Omega := \emptyset$. Our more general definition does not seem to significantly modify the hardness of the problem while allowing to prove Theorem 5.4.

with probability $1/p \le 1/2^{\lambda-1}$ over the draw of $x, r_1, \ldots, r_\ell$. Hence, event E_1 occurs with probability at most $q_{\mathrm{h}}^2/2^{\lambda-1}$. Similarly, event E_2 occurs with probability at most $(\ell+1)^2/2^{\lambda-1}$. Hence,

$$\mathbf{Adv}^{ros'}_{\mathsf{GrGen},\ell,\Omega,\mathcal{A}} \ge \mathbf{Adv}^{ros}_{\mathsf{GrGen},\ell,\Omega,\mathcal{A}} - \frac{q_{\mathrm{h}}^2 + (\ell+1)^2}{2^{\lambda-1}} \ . \tag{5.4}$$

We now construct an adversary $\mathcal{B}$ for the game $\mathrm{UNF}_{\mathsf{BlSch}[\mathsf{GrGen}]}$ as follows. Adversary $\mathcal{B}$, which takes as input $(p, \mathbb{G}, G, X)$ and has access to random oracle H and signing oracles $\textsc{Sign}_1$ and $\textsc{Sign}_2$, simulates game ROS′ as follows. First, $\mathcal{B}$ initiates ℓ parallel instances of the protocol by querying $(j, R_j) \leftarrow \textsc{Sign}_1()$ for $j \in [\ell]$. Then, it runs $\mathcal{A}(p)$. When $\mathcal{A}$ queries $\mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}, (\gamma, \xi, m))$ where $\overrightarrow{\rho} = (\rho_j)_{j\in[\ell]} \in (\mathbb{Z}_p)^\ell$ and $(\gamma, \xi, m) = \mathsf{aux} \in (\mathbb{Z}_p)^2 \times \{0,1\}^*$, $\mathcal{B}$ computes $R \coloneqq \gamma G + \xi X + \sum_{j=1}^{\ell} \rho_j R_j$ and returns $\mathsf{H}(R, m) + \xi$, unless there has been a previous query $\mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}\,', (\gamma', \xi', m'))$ with $m = m'$ and $R = \gamma' G + \xi' X + \sum_{j=1}^{\ell} \rho'_j R_j$, in which case $\mathcal{B}$ aborts. It is easy to see that $\mathcal{B}$ perfectly simulate game ROS′. Eventually, $\mathcal{A}$ returns $((\overrightarrow{\rho}\,_i, (\gamma_i, \xi_i, m_i^*))_{i\in[\ell+1]}, (c_j)_{j\in[\ell]})$. Then $\mathcal{B}$ closes all signing sessions by calling $s_j \leftarrow \textsc{Sign}_2(j, c_j)$ for $j \in [\ell]$. Finally, for $i \in [\ell+1]$, it computes

$$\begin{aligned} R_i^* &\coloneqq \gamma_i G + \xi_i X + \textstyle\sum_{j=1}^{\ell} \rho_{i,j} R_j \\ s_i^* &\coloneqq \gamma_i + \textstyle\sum_{j=1}^{\ell} \rho_{i,j} s_j \bmod p \end{aligned}$$

and returns $\ell+1$ forgeries $(m_i^*, (R_i^*, s_i^*))_{i\in[\ell+1]}$.

Assume that $\mathcal{A}$ wins game ROS′. Then, in particular, (i) all pairs (m_i^*, R_i^*) are distinct and (ii) for all $i \in [\ell+1]$,

$$\textstyle\sum_{j=1}^{\ell} \rho_{i,j} c_j \equiv_p \mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}\,_i, (\gamma_i, \xi_i, m_i^*)) \equiv_p \mathsf{H}(R_i^*, m_i^*) + \xi_i \ ,$$

where the second equality follows from the way $\mathcal{B}$ answers $\mathcal{A}$'s queries to $\mathsf{H}_{\mathrm{ros}}$. While (i) implies that all forgeries $(m_i^*, (R_i^*, s_i^*))$ are distinct, (ii) implies that all forgeries are valid since for all $i \in [\ell+1]$,

$$s_i^* G = \gamma_i G + \sum_{j=1}^{\ell} \rho_{i,j}(r_j + c_j x) G = \underbrace{\gamma_i G + \sum_{j=1}^{\ell} \rho_{i,j} R_j}_{R_i^* - \xi_i X} + \underbrace{\left(\sum_{j=1}^{\ell} \rho_{i,j} c_j\right)}_{\mathsf{H}(R_i^*, m_i^*) + \xi_i} X = R_i^* + \mathsf{H}(R_i^*, m_i^*) X \ .$$

Thus, $\mathcal{B}$ successfully breaks unforgeability of $\mathsf{BlSch}[\mathsf{GrGen}]$, and thus

$$\mathbf{Adv}^{unf}_{\mathsf{BlSch}[\mathsf{GrGen}],\mathcal{B}} = \mathbf{Adv}^{ros'}_{\mathsf{GrGen},\ell,\Omega,\mathcal{A}} \ . \tag{5.5}$$

Clearly, $\mathcal{B}$ runs in time at most $\tau + O(\ell + q_{\mathrm{h}})$ and makes at most ℓ queries to $\textsc{Sign}_1$ and $\textsc{Sign}_2$ and q_{h} random oracle queries. Combining Eqs. (5.4) and (5.5) concludes the proof.

The hardness of the ROS problem critically depends on ℓ. In particular, for small values of ℓ, the ROS problem is statistically hard, as captured by the following lemma.

Lemma 5.3. *Let* GrGen *be a group generator,* $\ell \ge 1$*, and* Ω *be some arbitrary set. Then for any adversary* $\mathcal{A}$ *making at most* q_{h} *queries to* $\mathsf{H}_{\mathrm{ros}}$*,*

$$\boldsymbol{Adv}^{ros}_{\mathsf{GrGen},\ell,\Omega,\mathcal{A}} \le \frac{\binom{q_{\mathrm{h}}}{\ell+1} + 1}{2^{\lambda-1}} \ .$$

PROOF Consider a modified game $\mathrm{ROS}^*_{\mathsf{GrGen},\ell,\Omega}$ that is identical to ROS, except that it returns 0 when the adversary outputs $((\overrightarrow{\rho}\,_i, \mathsf{aux}_i)_{i\in[\ell+1]}, (c_j)_{j\in[\ell]})$ such that for some $i \in [\ell+1]$ it has not made the query $\mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}\,_i, \mathsf{aux}_i)$. Games ROS and ROS* are identical unless in game ROS the adversary

wins and has not made the query $\mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}_i, \mathsf{aux}_i)$ for some i, which happens with probability at most $1/p \leq 1/2^{\lambda-1}$. Hence,

$$\mathbf{Adv}^{ros}_{\mathsf{GrGen},\ell,\Omega,\mathcal{A}} \leq \mathbf{Adv}^{ros^*}_{\mathsf{GrGen},\ell,\Omega,\mathcal{A}} + \frac{1}{2^{\lambda-1}} \ .$$

In order to win the modified game ROS*, $\mathcal{A}$ must in particular make $\ell+1$ distinct random oracle queries $(\overrightarrow{\rho}_i, \mathsf{aux}_i)_{i\in[\ell+1]}$ such that the system

$$\textstyle\sum_{j=1}^{\ell} \rho_{i,j} c_j \equiv_p \mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho_i}, \mathsf{aux}_i), \quad i \in [\ell+1] \tag{5.6}$$

with unknowns $c_1, \ldots, c_\ell$ has a solution. Consider any subset of $\ell+1$ queries $(\overrightarrow{\rho}_i, \mathsf{aux}_i)_{i\in[\ell+1]}$ made by the adversary to the random oracle and let M denote the $(\ell+1)\times\ell$ matrix whose i-th row is $\overrightarrow{\rho}_i$ and let $t \leq \ell$ denote its rank. Then, Equation 5.6 has a solution if and only if the row vector $\overrightarrow{h} := (\mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}_i, \mathsf{aux}_i))^{\mathrm{T}}_{i\in[\ell+1]}$ is in the span of the columns of M. Since $\overrightarrow{h}$ is uniformly random, this happens with probability $p^t/p^{\ell+1} \leq 1/p \leq 1/2^{\lambda-1}$. By the union bound,

$$\mathbf{Adv}^{ros^*}_{\mathsf{GrGen},\ell,\Omega,\mathcal{A}} \leq \frac{\binom{q_{\mathrm{h}}}{\ell+1}}{2^{\lambda-1}} \ ,$$

which concludes the proof.

On the other hand, the ROS_ℓ problem can be reduced the $(\ell+1)$-sum problem, for which Wagner's generalized birthday algorithm [Wag02, MS12, NS15] can be used. More specifically, consider the $(\ell+1)\times\ell$ matrix

$$(\rho_{i,j}) = \begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ & & \ddots & \\ 0 & \cdots & 0 & 1 \\ 1 & \cdots & \cdots & 1 \end{pmatrix}$$

and let $\overrightarrow{\rho}_i$ denote its i-th line, $i \in [\ell+1]$. Let $q := 2^{\lambda/(1+\lfloor \lg(\ell+1)\rfloor)}$. The solving algorithm builds lists $L_i = (\mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}_i, \mathsf{aux}_{i,k}))_{k\in[q]}$ for $i \in [\ell]$ and $L_{\ell+1} = (-\mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}_{\ell+1}, \mathsf{aux}_{\ell+1,k}))_{k\in[q]}$ for arbitrary values $\mathsf{aux}_{i,k}$ and uses Wagner's algorithm to find an element e_i in each list L_i such that $\sum_{i=1}^{\ell+1} e_i \equiv_p 0$. Then, it is easily seen that $((\overrightarrow{\rho}_i, \mathsf{aux}_i)_{i\in[\ell+1]}, (e_j)_{j\in[\ell]})$, where aux_i is such that $e_i = \mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}_i, \mathsf{aux}_i)$, is a solution to the ROS problem. This algorithm makes $q_{\mathrm{h}} = (\ell+1)2^{\lambda/(1+\lfloor \lg(\ell+1)\rfloor)}$ random oracle queries, runs in time an space $O((\ell+1)2^{\lambda/(1+\lfloor \lg(\ell+1)\rfloor)})$, and succeeds with constant probability.

Security of Blind Schnorr Signatures

We now formally prove that blind Schnorr signatures are unforgeable assuming the hardness of the one-more discrete logarithm problem and the ROS problem.

Theorem 5.4. *Let* GrGen *be a group generator. Let* $\mathcal{A}_{\mathrm{alg}}$ *be an algebraic adversary against the UNF security of the blind Schnorr signature scheme* $\mathsf{BlSch}[\mathsf{GrGen}]$ *running in time at most* τ *and making at most* q_{s} *queries to* $\textsc{Sign}_1$ *and* q_{h} *queries to the random oracle. Then there exist an algorithm* $\mathcal{B}_{\mathrm{ros}}$ *for the* $\mathrm{ROS}_{q_{\mathrm{s}}}$ *problem making at most* $q_{\mathrm{h}} + q_{\mathrm{s}} + 1$ *random oracle queries and an algorithm* $\mathcal{B}_{\mathrm{omdl}}$ *for the OMDL problem w.r.t.* GrGen *making at most* q_{s} *queries to its oracle* $\textsc{DLog}$*, both running in time at most* $\tau + O(q_{\mathrm{s}} + q_{\mathrm{h}})$*, such that*

$$\boldsymbol{Adv}^{unf}_{\mathsf{BlSch}[\mathsf{GrGen}],\mathcal{A}_{\mathrm{alg}}} \leq \boldsymbol{Adv}^{omdl}_{\mathsf{GrGen},\mathcal{B}_{\mathrm{omdl}}} + \boldsymbol{Adv}^{ros}_{\ell,\mathcal{B}_{\mathrm{ros}}} \ .$$

We start with explaining the proof idea. Consider an adversary in the unforgeability game, let X be the public key and $R_1, \ldots, R_\ell$ be the elements returned by the oracle SIGN_1 and let (R_i^*, s_i^*) be the adversary's forgeries on messages m_i^*. As $\mathcal{A}_{\text{alg}}$ is algebraic, it must also output a representation $(\gamma_i, \xi_i, \overrightarrow{\rho}_i)$ for R_i^* w.r.t. the group elements received from the game: $R_i^* = \gamma_i G + \xi_i X + \sum_{j=1}^{\ell} \rho_{i,j} R_j$. Validity of the forgeries implies another representation, namely $R_i^* = s_i^* G - c_i^* X$ with $c_i^* = \mathsf{H}(R_i^*, m_i^*)$. Together, these yield

$$(c_i^* + \xi_i^*)X + \textstyle\sum_{j=1}^{\ell} \rho_{i,j}^* R_j = (s_i^* - \gamma_i^*)G \ , \tag{5.7}$$

which intuitively can be used to compute $\log X$.

However, the reduction also needs to simulate SIGN_2 queries, for which, contrary to the proof for standard Schnorr signatures (Theorem 5.1), it cannot rely on programming the random oracle. In fact, the reduction can only win OMDL, which is an *easier* game than DL. In particular, the reduction obtains $X, R_1, \ldots, R_q$ from its challenger and must compute their logarithms. It can make q logarithm queries, which it uses to simulate the SIGN_2 oracle: on input (j, c_j), it simply returns $s_j \leftarrow \text{DLOG}(R_j + c_j X)$.

But this means that in Equation 5.7 the reduction does not know the logarithms of the R_j's; all it knows is $R_j = s_j G - c_j X$, which, when plugged into Equation 5.7 yields

$$(\underbrace{c_i^* + \xi_i^* - \textstyle\sum_{j=1}^{\ell} \rho_{i,j}^* c_j}_{=:\chi_i})X = (s_i^* - \gamma_i^* - \textstyle\sum_{j=1}^{\ell} \rho_{i,j}^* s_j)G \ .$$

Thus, if for some i, $\chi_i \neq 0$, the reduction can compute $x = \log X$, from which it can derive $r_j = \log R_j = s_j - c_j\, x$. Together, $x, r_1, \ldots, r_q$ constitute an OMDL solution.

On the other hand, we can show that if $\chi_i = 0$ for *all* i, then the adversary has actually found a solution to the ROS problem (Figure 5.6): A reduction to ROS would answer the adversary's queries $\mathsf{H}(R_{[\gamma,\xi,\overrightarrow{\rho}]}, m)$ by $\mathsf{H}_{\text{ros}}(\overrightarrow{\rho}, (\gamma, \xi, m)) - \xi$; then $\chi_i = 0$ implies (recall that $c_i^* = \mathsf{H}(R_i^*, m^*)$)

$$0 = \chi_i = \mathsf{H}(R_i^*, m_i^*) + \xi_i^* - \textstyle\sum_{j=1}^{\ell} \rho_{i,j}^* c_j = \mathsf{H}_{\text{ros}}(\overrightarrow{\rho}^*_i, (\gamma_i^*, \xi_i^*, m_i^*)) - \sum_{j=1}^{\ell} \rho_{i,j}^* c_j \ ,$$

meaning $((\overrightarrow{\rho}^*_i, (\gamma_i^*, \xi_i^*, m_i^*))_i, (c_j)_j)$ is a solution to ROS.

To simplify the proof we first show the following lemma.

Lemma 5.5. *Let* GrGen *be a group generator and let* $\mathcal{A}$ *be an adversary against the UNF security of the blind Schnorr signature scheme* $\mathsf{BlSch}[\mathsf{GrGen}]$ *running in time at most* τ *and making at most* q_{s} *queries to* SIGN_1 *and* q_{h} *queries to the random oracle. Then there exists an adversary* $\mathcal{B}$ *that makes* exactly q_{s} *queries to* SIGN_1 *and* q_{s} *queries to* SIGN_2 *that do not return* $\perp$*, and returns* $q_{\text{s}} + 1$ *forgeries, running in time at most* $\tau + O(q_{\text{s}})$*, such that*

$$\boldsymbol{Adv}^{unf}_{\mathsf{BlSch}[\mathsf{GrGen}],\mathcal{A}} = \boldsymbol{Adv}^{unf}_{\mathsf{BlSch}[\mathsf{GrGen}],\mathcal{B}} \ .$$

PROOF We construct the following adversary that plays game UNF (Figure 5.4). On input pk, adversary $\mathcal{B}$ runs $\mathcal{A}(pk)$ and relays all oracle queries and responses between its challenger and $\mathcal{A}$. Let q be the number of $\mathcal{A}$'s SIGN_1 queries, let $R_1, \ldots, R_q$ be the answers, and let $\mathcal{C}$ be the completed sessions, that is, the set of values j such that $\mathcal{A}$ queried SIGN_2 on some input $(j, *)$ and SIGN_2 did not reply $\perp$. Let $(m_i^*, (R_i^*, s_i^*))_{i \in [n]}$ be $\mathcal{A}$'s output, for which we must have $k = |\mathcal{C}| < n$ when $\mathcal{A}$ wins.

$\mathcal{B}$ then makes $q_{\text{s}} - q$ queries to SIGN_1 to receive $R_{q+1}, \ldots, R_{q_{\text{s}}}$. Next, $\mathcal{B}$ completes all $q_{\text{s}} - k$ open signing sessions for distinct messages by following the protocol in Figure 5.5: for every $j \in \mathcal{S} := [1, \ldots, q_{\text{s}}] \setminus \mathcal{C}$, adversary $\mathcal{B}$ picks a fresh message $m_j \notin \{m_i^*\}_{i \in [n]} \cup \{m_i\}_{i \in \mathcal{S} \setminus [j]}$ and $\alpha_j, \beta_j \xleftarrow{\$} \mathbb{Z}_p$, computes $R'_j := R_j + \alpha_j G + \beta_j X$, queries $\mathsf{H}(R', m_j)$ to get c'_j, computes $c_j := c'_j + \beta_j \bmod p$

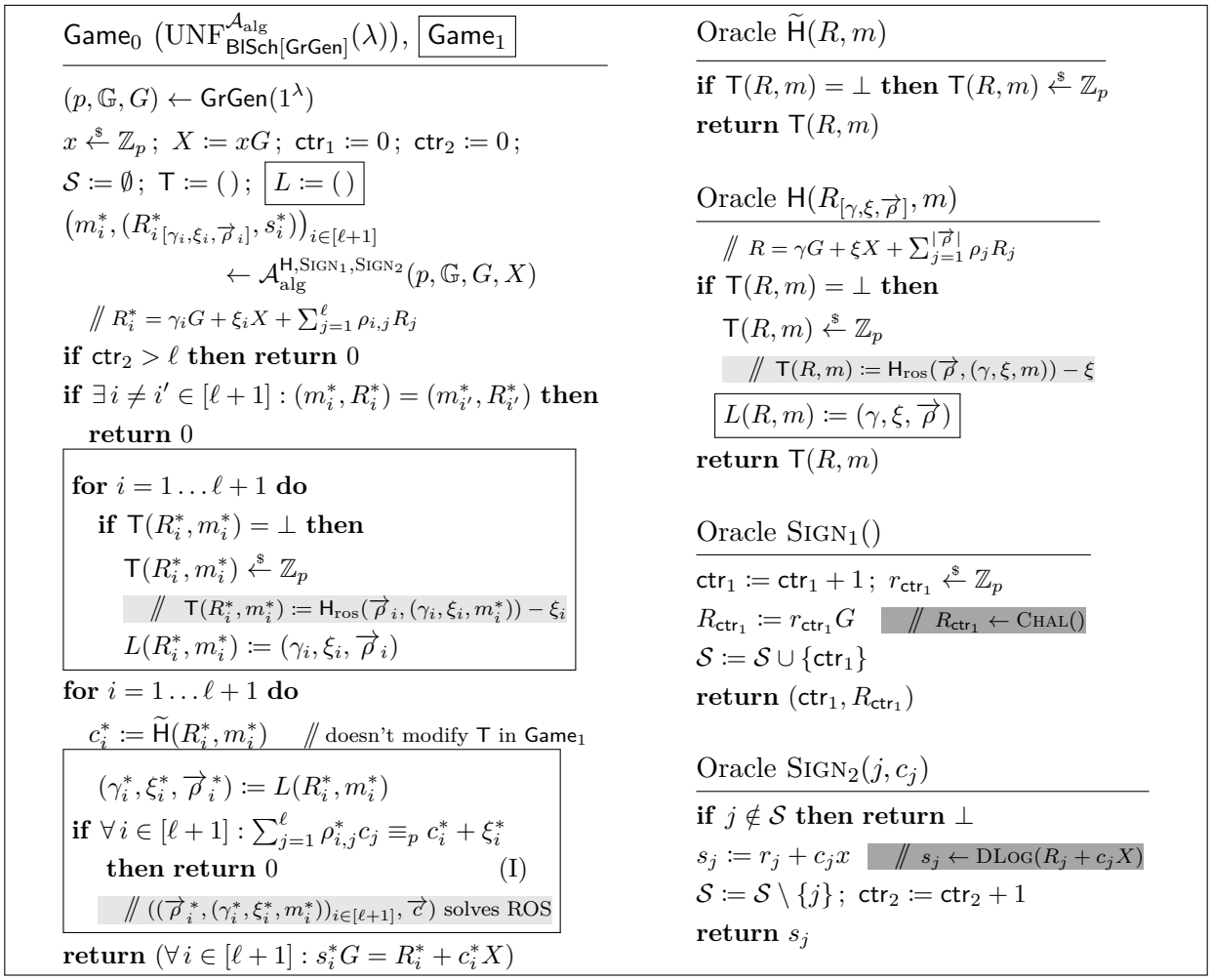

Figure 5.7: Games used in the proof of Theorem 5.4. Game$_0$ ignores all boxes. The light-gray comments in Game$_1$ and oracle H show how reduction $\mathcal{B}_{\mathrm{ros}}$ solves ROS; the dark-gray comments in the Sign oracles show how $\mathcal{B}_{\mathrm{omdl}}$ embeds its challenges and simulates Game$_1$.

and queries (j, c_j) to Sign_2. Upon receiving s_j, $\mathcal{B}$ computes $s'_j := s_j + \alpha_j \bmod p$, which yields a signature (R'_j, s'_j) on message m_j.

Finally, $\mathcal{B}$ concatenates $\mathcal{A}$'s output with $q_{\mathrm{s}} + 1 - n \leq q_{\mathrm{s}} - k$ signatures: let $\mathcal{S} = \{j_1, \ldots, j_{q_{\mathrm{s}}-k}\}$; then $\mathcal{B}$ returns $(m_i^*, (R_i^*, s_i^*))_{i\in[n]} \,\|\, (m_{j_i}, (R'_{j_i}, s'_{j_i}))_{i\in[q_{\mathrm{s}}+1-n]}$. When $\mathcal{A}$ wins the game, all tuples $(m_i^*, (R_i^*, s_i^*))$ are different; as all remaining messages also differ, all tuples output by $\mathcal{B}$ are distinct. By correctness of the scheme, $\mathcal{B}$'s signatures are valid. Thus whenever $\mathcal{A}$ wins, then so does $\mathcal{B}$.

PROOF [Proof of Theorem 5.4] Let $\mathcal{A}_{\mathrm{alg}}$ be an algebraic adversary making at most q_{s} queries to Sign_1 and q_{h} random oracle queries. By the above lemma, we can assume that $\mathcal{A}_{\mathrm{alg}}$ makes exactly $\ell := q_{\mathrm{s}}$ queries to Sign_1, closes all sessions, and returns $\ell + 1$ valid signatures. We proceed with a sequence of games defined in Figure 5.7.

Game$_0$. The first game is the UNF game (Figure 5.4) for scheme BISch[GrGen] played with $\mathcal{A}_{\mathrm{alg}}$ in the random oracle model. We have written the finalization of the game in a different but equivalent way. In particular, instead of checking that $(m_i^*, (R_i^*, s_i^*)) \neq (m_{i'}^*, (R_{i'}^*, s_{i'}^*))$ for all $i \neq i' \in [\ell+1]$,

we simply check that $(m_i^*, R_i^*) \neq (m_{i'}^*, R_{i'}^*)$. This is equivalent since for any pair (m, R), there is a single $s \in \mathbb{Z}_p$ such that (R, s) is a valid signature for m. Hence, if the adversary returns $(m_i^*, (R_i^*, s_i^*))$ and $(m_{i'}^*, (R_{i'}^*, s_{i'}^*))$ with $(m_i^*, R_i^*) = (m_{i'}^*, R_{i'}^*)$ and $s_i^* \neq s_{i'}^*$, at least one of the two forgeries is invalid. Thus,

$$\mathbf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}_{\mathrm{alg}}} = \mathbf{Adv}^{unf}_{\mathsf{BISch}[\mathsf{GrGen}],\mathcal{A}_{\mathrm{alg}}} \ . \tag{5.8}$$

$\underline{\mathsf{Game}_1}$. In Game_1, we make the following changes (which are analogous to those in the proof of Theorem 5.1). First, we introduce an auxiliary table L that for each query $\mathsf{H}(R_{[\gamma,\xi,\overrightarrow{\rho}]}, m)$ stores the representation $(\gamma, \xi, \overrightarrow{\rho})$ of R. Second, when the adversary returns its forgeries $(m_i^*, (R^*_{i\,[\gamma_i,\xi_i,\overrightarrow{\rho}_i]}, s_i^*))_{i\in[\ell+1]}$, then for each $i \in [\ell+1]$ for which $\mathsf{T}(R_i^*, m_i^*)$ is undefined, we emulate a call to $\mathsf{H}(R^*_{i\,[\gamma_i,\xi_i,\overrightarrow{\rho}_i]}, m_i^*)$. Again, this does not change the output of the game, since in Game_0, the value $\mathsf{T}(R_i^*, m_i^*)$ would be randomly assigned when the game calls $\widetilde{\mathsf{H}}$ to check the signature. Finally, for each $i \in [\ell+1]$, we retrieve $(\gamma_i^*, \xi_i^*, \overrightarrow{\rho}_i^*) := L(R_i^*, m_i^*)$ (which is necessarily defined at this point) and return 0 if $\sum_{i=1}^{\ell} \rho_{i,j}^* c_j \equiv_p c_i^* + \xi_i^*$ for all $i \in [\ell+1]$, where c_j is the (unique) value submitted to $\textsc{Sign}_2$ together with j and not answered by $\perp$.

Game_0 and Game_1 are identical unless Game_1 returns 0 in line (I). We reduce indistinguishability of the games to ROS by constructing an algorithm $\mathcal{B}_{\mathrm{ros}}$ solving the ROS_ℓ problem whenever Game_1 stops in line (I). Algorithm $\mathcal{B}_{\mathrm{ros}}$, which has access to oracle $\mathsf{H}_{\mathrm{ros}}$, runs $\mathcal{A}_{\mathrm{alg}}$ and simulates Game_1 in a straightforward way, except for using its $\mathsf{H}_{\mathrm{ros}}$ oracle to define the entries of T.

In particular, consider a query $\mathsf{H}(R_{[\gamma,\xi,\overrightarrow{\rho}]}, m)$ by $\mathcal{A}_{\mathrm{alg}}$ such that $\mathsf{T}(R, m) = \perp$. Then $\mathcal{B}_{\mathrm{ros}}$ pads the vector $\overrightarrow{\rho}$ with 0's to make it of length ℓ (at this point, not all $R_1, \ldots, R_\ell$ are necessarily defined, so $\overrightarrow{\rho}$ might not be of length ℓ), and assigns $\mathsf{T}(R, m) := \mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}, (\gamma, \xi, m)) - \xi$ (cf. comments in Figure 5.7). Similarly, when $\mathcal{A}_{\mathrm{alg}}$ returns its forgeries $(m_i^*, (R^*_{i\,[\gamma_i,\xi_i,\overrightarrow{\rho}_i]}, s_i^*))_{i\in[\ell+1]}$, then for each $i \in [\ell+1]$ with $\mathsf{T}(R_i^*, m_i^*) = \perp$, reduction $\mathcal{B}_{\mathrm{ros}}$ assigns $\mathsf{T}(R_i^*, m_i^*) := \mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}_i, (\gamma_i, \xi_i, m_i^*)) - \xi_i$. Since $\mathsf{H}_{\mathrm{ros}}$ returns uniformly random elements in $\mathbb{Z}_p$, the simulation is perfect.

If Game_1 aborts in line (I), then $\mathcal{B}_{\mathrm{ros}}$ returns $((\overrightarrow{\rho}_i^*, (\gamma_i^*, \xi_i^*, m_i^*))_{i\in[\ell+1]}, (c_j)_{j\in[\ell]})$, where $(\gamma_i^*, \xi_i^*, \overrightarrow{\rho}_i^*) := L(R_i^*, m_i^*)$. We show that this is a valid ROS solution.

First, for all $i \neq i' \in [\ell+1]$: $(\overrightarrow{\rho}_i^*, (\gamma_i^*, \xi_i^*, m_i^*)) \neq (\overrightarrow{\rho}_{i'}^*, (\gamma_{i'}^*, \xi_{i'}^*, m_{i'}^*))$. Indeed, otherwise we would have $(m_i^*, R_i^*) = (m_{i'}^*, R_{i'}^*)$ and the game would have returned 0 earlier. Second, since the game returns 0 in line (I), we have $\sum_{j=1}^{\ell} \rho_{i,j}^* c_j \equiv_p c_i^* + \xi_i^*$ for all $i \in [\ell+1]$. Hence, to show that the ROS solution is valid, it is sufficient to show that for all $i \in [\ell+1]$, $c_i^* = \mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}_i^*, (\gamma_i^*, \xi_i^*, m_i^*)) - \xi_i^*$. This is clearly the case if $\mathsf{T}(R_i^*, m_i^*) = \perp$ when the adversary returns its forgeries. Indeed, in that case $(\gamma_i^*, \xi_i^*, \overrightarrow{\rho}_i^*) = (\gamma_i, \xi_i, \overrightarrow{\rho}_i)$ and

$$c_i^* = \mathsf{T}(R_i^*, m_i^*) = \mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}_i, (\gamma_i, \xi_i, m_i^*)) - \xi_i = \mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}_i^*, (\gamma_i^*, \xi_i^*, m_i^*)) - \xi_i^* \ .$$

Otherwise, $\mathsf{T}(R_i^*, m_i^*)$ was necessarily assigned during a call to H, and this call was of the form $\mathsf{H}(R^*_{i\,[\gamma_i^*,\xi_i^*,\overrightarrow{\rho}_i^*]}, m_i^*)$, which implies that $c_i^* = \mathsf{T}(R_i^*, m^*) = \mathsf{H}_{\mathrm{ros}}(\overrightarrow{\rho}_i^*, (\gamma_i^*, \xi_i^*, m_i^*)) - \xi_i^*$. Hence,

$$\mathbf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}_{\mathrm{alg}}} \geq \mathbf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}_{\mathrm{alg}}} - \mathbf{Adv}^{ros}_{\ell,\mathcal{B}_{\mathrm{ros}}} \ . \tag{5.9}$$

Moreover, it is easy to see that $\mathcal{B}_{\mathrm{ros}}$ makes at most $q_{\mathrm{h}} + \ell + 1$ queries to $\mathsf{H}_{\mathrm{ros}}$ and runs in time at most $\tau + O(\ell + q_{\mathrm{h}})$, assuming scalar multiplications in $\mathbb{G}$ and table assignments take unit time.

$\underline{\text{Reduction to OMDL}}$. In our last step, we construct an algorithm $\mathcal{B}_{\mathrm{omdl}}$ solving the OMDL problem whenever $\mathcal{A}_{\mathrm{alg}}$ wins Game_1. Algorithm $\mathcal{B}_{\mathrm{omdl}}$, which has access to two oracles $\textsc{Chal}$ and $\textsc{DLog}$ (see Figure 2.1) takes as input a group description $(p, \mathbb{G}, G)$, makes a first query $X \leftarrow \textsc{Chal}()$, and runs $\mathcal{A}_{\mathrm{alg}}$ on input $(p, \mathbb{G}, G, X)$, simulating Game_1 as follows (cf. comments in Figure 5.7). Each time $\mathcal{A}_{\mathrm{alg}}$ makes a $\textsc{Sign}_1()$ query, $\mathcal{B}_{\mathrm{omdl}}$ queries its $\textsc{Chal}$ oracle to obtain R_j. It simulates $\textsc{Sign}_2(j, c)$ without knowledge of x and r_j by querying $s_j \leftarrow \textsc{DLog}(R_j + cX)$.

Assume that Game_1 returns 1, which implies that all forgeries (R_i^*, s_i^*) returned by $\mathcal{A}_{\text{alg}}$ are valid. We show how $\mathcal{B}_{\text{omdl}}$ solves OMDL. First, note that $\mathcal{B}_{\text{omdl}}$ made exactly ℓ calls to its oracle DLog in total (since it makes exactly one call for each (valid) Sign_2 query made by $\mathcal{A}_{\text{alg}}$).

Since Game_1 did not return 0 in line (I), there exists $i \in [\ell + 1]$ such that

$$\textstyle\sum_{j=1}^{\ell} \rho_{i,j}^* c_j \not\equiv_p c_i^* + \xi_i^* \ . \tag{5.10}$$

For all i, the adversary returned a representation $(\gamma_i^*, \xi_i^*, \overrightarrow{\rho}_i^*)$ of R_i^*, thus

$$R_i^* = \gamma_i^* G + \xi_i^* X + \textstyle\sum_{j=1}^{\ell} \rho_{i,j}^* R_j \ . \tag{5.11}$$

On the other hand, validity of the i-th forgery yields another representation: $R_i^* = s_i^* G + c_i^* X$. Combining these two, we get

$$(c_i^* + \xi_i^*)X + \textstyle\sum_{j=1}^{\ell} \rho_{i,j}^* R_j = (s_i^* - \gamma_i^*)G \ . \tag{5.12}$$

Finally, for each $j \in [\ell]$, s_j was computed with a call $s_j \leftarrow \text{DLog}(R_j + c_j X)$, hence

$$R_j = s_j G - c_j X \ . \tag{5.13}$$

Injecting Equation 5.13 in Equation 5.12, we obtain

$$\left(c_i^* + \xi_i^* - \textstyle\sum_{j=1}^{\ell} \rho_{i,j}^* c_j\right) X = \left(s_i^* - \gamma_i^* - \textstyle\sum_{j=1}^{\ell} \rho_{i,j}^* s_j\right) G \ . \tag{5.14}$$

Since by Equation 5.10 the coefficient in front of X is non-zero, this allows $\mathcal{B}_{\text{omdl}}$ to compute $x := \log X$. Furthermore, from Equation 5.13 we have $r_j := \log R_j = s_j - c_j x$ for all $j \in [\ell]$. By returning $(x, r_1, \ldots, r_\ell)$, $\mathcal{B}_{\text{omdl}}$ solves the OMDL problem whenever $\mathcal{A}_{\text{alg}}$ wins Game_1, which implies

$$\mathbf{Adv}^{omdl}_{\mathsf{GrGen},\mathcal{B}_{\text{omdl}}} = \mathbf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}_{\text{alg}}} \ . \tag{5.15}$$

The theorem now follows from Equations (5.8), (5.9) and (5.15).

Chapter 6

Clause Blind Schnorr signatures

6.1 Definition of the Clause Blind Schnorr Signature Scheme

We present a variation of the blind Schnorr signature scheme that only modifies the signing protocol. The scheme thus does not change the signatures themselves, meaning that it can be very smoothly integrated in existing applications.

The signature issuing protocol is changed so that it prevents the adversary from attacking the scheme by solving the ROS problem using Wagner's algorithm [Wag02, MS12]. The reason is that, as we show in Theorem 6.1, the attacker must now solve a *modified* ROS problem, which we define in Figure 6.2.

We start with explaining the modified signing protocol, formally defined in Figure 6.1. In the first round the signer and the user execute two parallel runs of the blind signing protocol from Figure 5.5, of which the signer only finishes one at random in the last round, that is, it finishes $(\text{Run}_1 \vee \text{Run}_2)$: the clause from which the scheme takes its name.

This minor modification has major consequences. Recall that in the attack against the standard blind signature scheme from subsection 5.2, the adversary opens ℓ signing sessions, receiving $R_1, \ldots, R_\ell$, then searches a solution $\overrightarrow{c}$ to the ROS problem and closes the signing sessions by sending $c_1, \ldots, c_\ell$. Our modified signing protocol prevents this attack, as now for every opened session the adversary must *guess* which of the two challenges the signer will reply to. Only if all its guesses are correct is the attack successful. As the attack only works for large values of ℓ, this probability vanishes exponentially.

In Theorem 6.1 we make this intuition formal; that is, we define a modified ROS game, which we show any successful attacker (which does not solve OMDL) must solve.

We have used two parallel executions of the basic protocol for the sake of simplicity, but the idea can be straightforwardly generalized to $t > 2$ parallel runs, of which the signer closes only one at random in the last round, that is, it closes $(\text{Run}_1 \vee \ldots \vee \text{Run}_t)$. This decreases the probability that the user correctly guesses which challenges will be answered by the signer in ℓ concurrent sessions.

6.2 The Modified ROS Problem

Consider Figure 6.2. The difference to the original ROS problem (Figure 5.6) is that the queries to the H_{ros} oracle consist of *two* vectors $\overrightarrow{\rho}_0, \overrightarrow{\rho}_1$ and additional aux information. Analogously, the adversary's task is to return $\ell + 1$ tuples $(\overrightarrow{\rho}_{i,0}, \overrightarrow{\rho}_{i,1}, \mathsf{aux}_i)$, except that the ROS solution $c_1^*, \ldots, c_\ell^*$ is selected as follows: for every index $j \in [\ell]$ the adversary must query an additional oracle $\textsc{Select}(j, c_{j,0}, c_{j,1})$, which flips a random bit b_j and sets the j-th coordinate of the solution to $c_j^* := c_{j,b_j}$.

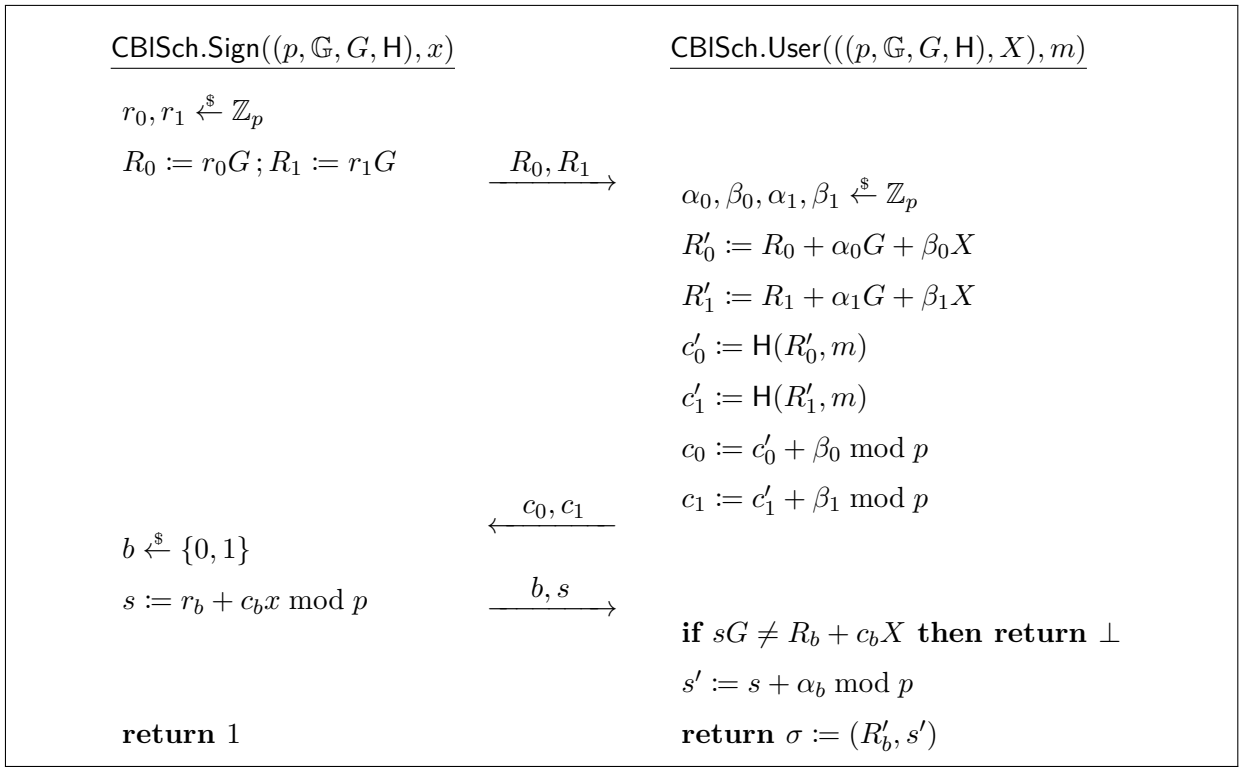

Figure 6.1: The clause blind Schnorr signing protocol.

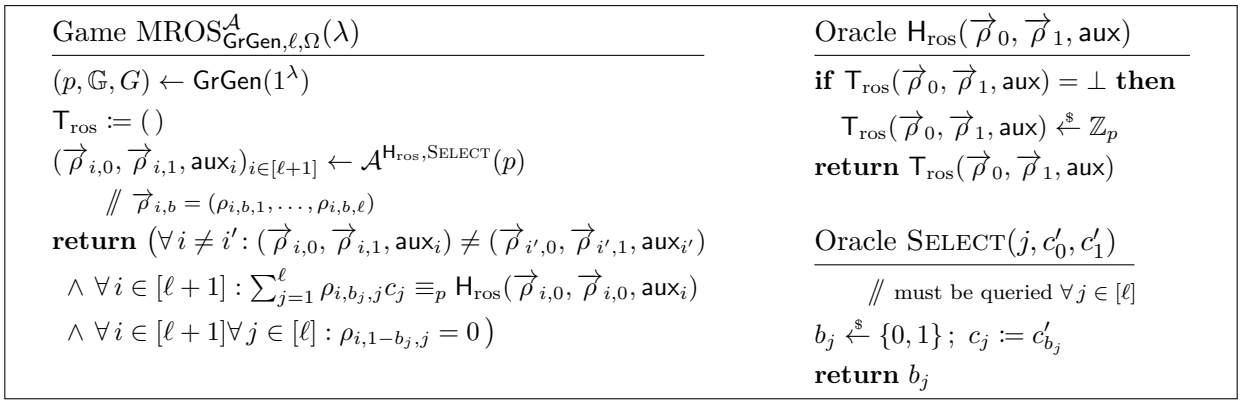

Figure 6.2: The modified ROS problem.

Up to now, nothing really changed, as an adversary could always choose $\overrightarrow{\rho}_{i,0} = \overrightarrow{\rho}_{i,1}$ and $c_{j,0} = c_{j,1}$ for all indices, and solve the standard ROS problem. What complicates the task for the adversary considerably is the additional winning condition, which demands that in *all* tuples returned by the adversary, the ρ values that correspond to the complement of the selected bit must be zero, that is, for all $i \in [\ell + 1]$ and all $j \in [\ell]$: $\rho_{i,1-b_j,j} = 0$. The adversary thus must commit to the solution coordinate c_j^* before it learns b_j, which then restricts the format of its ρ values.

We conjecture that the best attack against this modified ROS problem is to guess the ℓ bits b_j and to solve the standard ROS problem based on this guess using Wagner's algorithm. Hence, the complexity of the attack is increased by a factor 2^ℓ and requires time

$$O\big(2^\ell \cdot (\ell+1) 2^{\lambda/(1+\lfloor \lg(\ell+1) \rfloor)}\big) \ .$$

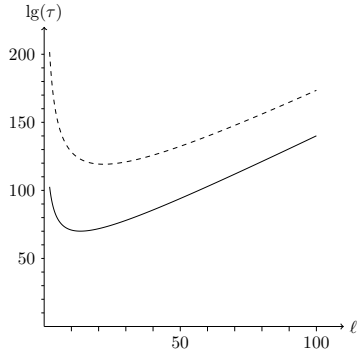

Figure 6.3: Estimated complexity τ of conjectured best attack against the modified ROS problem as a function of parameter ℓ for $\lambda = 256$ (solid line) and $\lambda = 512$ (dashed line).

This estimated complexity is plotted for $\lambda \in \{256, 512\}$ in Figure 6.3. This should be compared to the standard Wagner attack with $\ell + 1 = 2^{\sqrt{\lambda}}$ running in time 2^{32} and 2^{45}, respectively, for the same values of the security parameter.

6.3 Unforgeability of the Clause Blind Schnorr Signature Scheme

We now prove that the Schnorr signature scheme from Figure 5.2, with the signing algorithm replaced by the protocol in Figure 6.1 is secure under the OMDL assumption for the underlying group and hardness of the modified ROS problem.

Theorem 6.1. *Let* GrGen *be a group generator. Let* $\mathcal{A}_{\mathrm{alg}}$ *be an algebraic adversary against the UNF security of the clause blind Schnorr signature scheme* $\mathsf{CBlSch}[\mathsf{GrGen}]$ *running in time at most* τ *and making at most* q_{s} *queries to* $\textsc{Sign}_1$ *and* q_{h} *queries to the random oracle. Then there exist an algorithm* $\mathcal{B}_{\mathrm{mros}}$ *for the* $\mathit{MROS}_{q_{\mathrm{s}}}$ *problem making at most* $q_{\mathrm{h}} + q_{\mathrm{s}} + 1$ *random oracle queries and an algorithm* $\mathcal{B}_{\mathrm{omdl}}$ *for the OMDL problem w.r.t.* GrGen *making at most* q_{s} *queries to its oracle* $\textsc{DLog}$*, both running in time at most* $\tau + O(q_{\mathrm{s}} + q_{\mathrm{h}})$*, such that*

$$\boldsymbol{Adv}^{unf}_{\mathsf{BlSch}[\mathsf{GrGen}],\mathcal{A}_{\mathrm{alg}}} \leq \boldsymbol{Adv}^{omdl}_{\mathsf{GrGen},\mathcal{B}_{\mathrm{omdl}}} + \boldsymbol{Adv}^{mros}_{\ell,\mathcal{B}_{\mathrm{mros}}} \ .$$

19em20.5em

The theorem follows by adapting the proof of Theorem 5.4; we therefore discuss the changes and refer to Figure 6.4, which compactly presents all the details.

The proof again proceeds by one game hop, where an adversary behaving differently in the two games is used to break the modified ROS problem; the only change to the proof of Theorem 5.4 is that when simulating $\textsc{Sign}_2$, the reduction $\mathcal{B}_{\mathrm{mros}}$ calls $\textsc{Select}(j, c_{j,0}, c_{j,1})$ to obtain bit b instead of choosing it itself. By definition, Game_1 aborts in line (I) if and only if $\mathcal{B}_{\mathrm{mros}}$ has found a solution for MROS.

The difference in the reduction to OMDL of the modified game is that the adversary can fail to solve MROS in two ways: (1) its values $((\rho_{i,b_j,j})_{i,j}, (c_j)_j)$ are not a ROS solution; in this case the reduction can solve OMDL as in the proof of Theorem 5.4; (2) these values *are* a ROS solution, but for some i, j, we have $\rho_{i,1-b_j,j} \neq 0$. We show that in this case the OMDL reduction can compute the discrete logarithm of one of the values $R_{j,1-b_j}$.

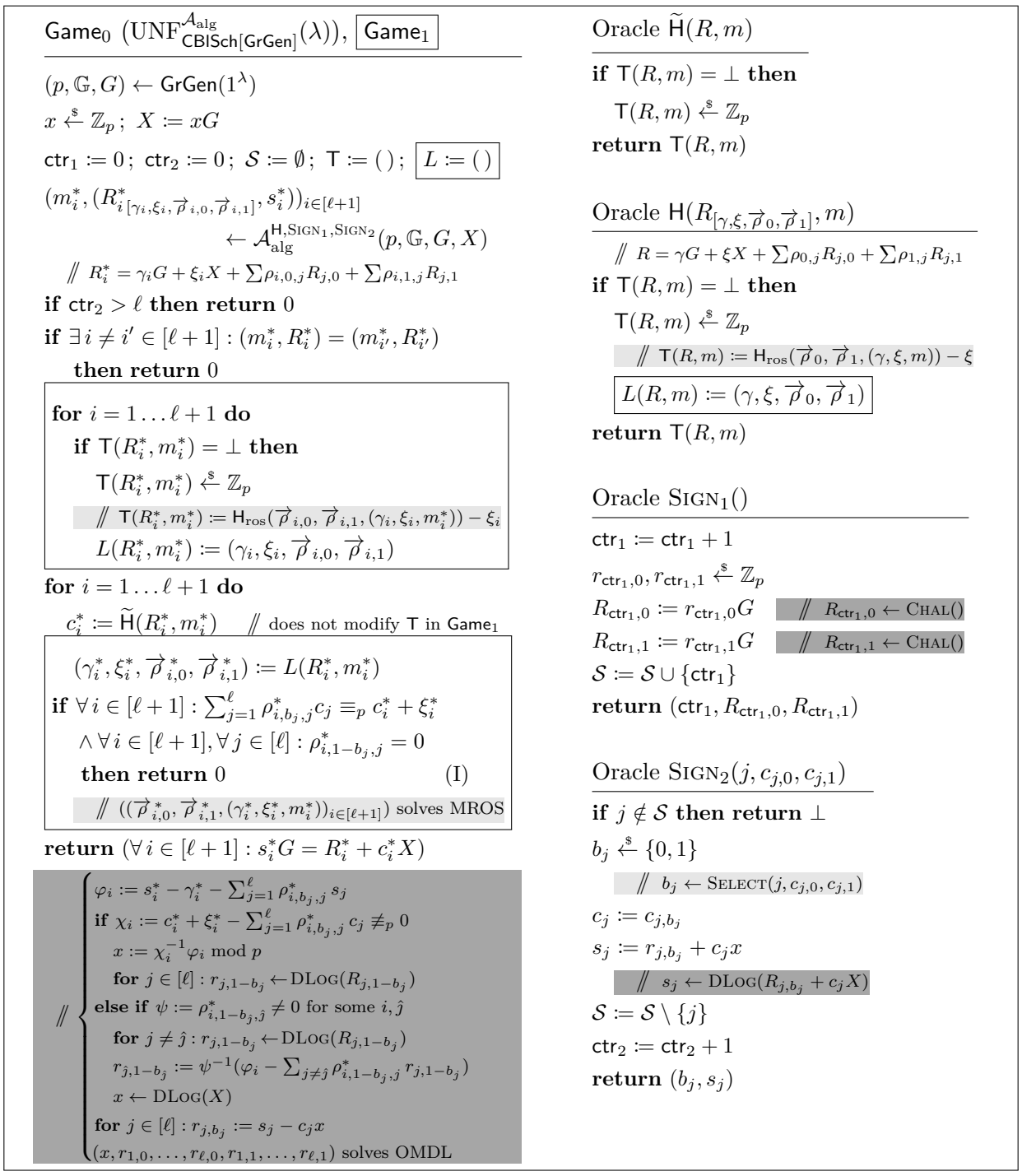

Figure 6.4: Games used in the proof of Theorem 6.1. Game_0 is the unforgeability game for the clause blind Schnorr signature scheme in the ROM for an algebraic adversary $\mathcal{A}_{\mathrm{alg}}$. The comments in light gray show how $\mathcal{B}_{\mathrm{mros}}$ solves MROS; the dark comments show how $\mathcal{B}_{\mathrm{omdl}}$ solves OMDL.

More in detail, the main difference to Theorem 5.4 is that the representation of the values R_i^* in the adversary's forgery depend on both the $R_{j,0}$ and the $R_{j,1}$ values; we can thus write them as

$$R_i^* = \gamma_i^* G + \xi_i^* X + \sum_{j=1}^{\ell} \rho_{i,b_j,j}^* R_{j,b_j} + \sum_{j=1}^{\ell} \rho_{i,1-b_j,j}^* R_{j,1-b_j}$$

(this corresponds to Equation 5.11 in the proof of Theorem 5.4). Validity of the forgery implies $R_i^* = s_i^* G - c_i^* X$, which together with the above yields

$$(c_i^* + \xi_i^*)X + \sum_{j=1}^{\ell} \rho_{i,b_j,j}^* R_{j,b_j} = (s_i^* - \gamma_i^*)G - \sum_{j=1}^{\ell} \rho_{i,1-b_j,j}^* R_{j,1-b_j}$$

(cf. Equation 5.12). By definition of s_j, we have $R_{j,b_j} = s_j G - c_j X$ for all $j \in [\ell]$; the above equation becomes thus

$$(c_i^* + \xi_i^* - \sum_{j=1}^{\ell} \rho_{i,b_j,j}^* c_j)X = (s_i^* - \gamma_i^* - \sum_{j=1}^{\ell} \rho_{i,b_j,j}^* s_j)G - \sum_{j=1}^{\ell} \rho_{i,1-b_j,j}^* R_{j,1-b_j} \tag{6.1}$$

(which corresponds to Equation 5.14 in Theorem 5.4). In Theorem 5.4, not solving ROS implied that for some i, the coefficient of X in the above equation was non-zero, which allowed computation of $\log X$.

However, if the adversary sets all these coefficients to 0, it could still fail to solve MROS if $\rho_{i^*,1-b_{j^*},j^*}^* \neq 0$ for some i^*, j^* (this is case (2) defined above). In this case Game_1 does not abort and the OMDL reduction $\mathcal{B}_{\text{omdl}}$ must succeed. Since in this case the left-hand side of Equation 6.1 is then 0, $\mathcal{B}_{\text{omdl}}$ can, after querying $\text{DLOG}(R_{j,1-b_j})$ for all $j \neq j^*$, compute $\text{DLOG}(R_{j^*,1-b_{j^*}})$, which breaks OMDL.

We finally note that the above case distinction was merely didactic, as the same OMDL reduction can handle both cases simultaneously, which means that our reduction does not introduce any additional security loss. In particular, the reduction obtains X and all values $(R_{j,0}, R_{j,1})$ from its OMDL challenger, then handles case (2) as described, and case (1) by querying $R_{1,1-b_1}, \ldots, R_{\ell,1-b_\ell}$ to its DLOG oracle. In both cases it made 2ℓ queries to DLOG and computed the discrete logarithms of all $2\ell + 1$ challenges.

Figure 6.4 presents the unforgeability game and Game_1, which aborts if the adversary solved MROS. The gray and dark gray comments also precisely define how a reduction $\mathcal{B}_{\text{mros}}$ solves MROS whenever Game_1 aborts in line (I), and how a reduction $\mathcal{B}_{\text{omdl}}$ solves OMDL whenever $\mathcal{A}_{\text{alg}}$ wins Game_1.

Blindness of Clause Blind Schnorr Signatures Blindness of the "clause" variant in Figure 6.1 follows via a hybrid argument from blindness of the standard scheme (Figure 5.5). In the game defining blindness (see Figure 6.5 in section 6.4), the adversary impersonates a signer and selects two messages m_0 and m_1. The game flips a bit b, runs the signing protocol with the adversary for m_b and then for m_{1-b}. If both sessions terminate, the adversary is given the resulting signatures and must determine b.

In the blindness game for scheme CBISch, the challenger runs *two* instances of the issuing protocol from BISch for m_b of which the signer finishes one, as determined by its message (β_b, s_b) in the third round (β_b corresponds to b in Figure 6.1), and then *two* instances for m_{1-b}.

If $b = 0$, the challenger thus asks the adversary for signatures on m_0, m_0, m_1 and then m_1. We define a hybrid game where the order of the messages is $\underline{m_1}, m_0, \underline{m_0}, m_1$; this game thus lies between the blindness games for $b = 0$ and $b = 1$, where the messages are m_1, m_1, m_0, m_0. The original games differ from the hybrid game by exactly one message pair; intuitively, they are thus indistinguishable by blindness of BISch.

A technical detail is that the above argument only works when $\beta_0 = \beta_1$, as otherwise in the reduction to BISch blindness, both reductions (between each original game and the hybrid game)

abort one session and do not get any signatures from its challenger. The reductions thus guess the values β_0 and β_1 (and return a random bit if the guess turns out wrong). The hybrid game then replaces the β_0-th message of the first two and the β_1-th of the last two (as opposed to the ones underlined as above). Following this argument, in section 6.4 we prove the following:

Theorem 6.2. *Let $\mathcal{A}$ be a p.p.t. adversary against blindness of the scheme* CBISch*. Then there exist two p.p.t. algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ against blindness of* BISch *such that*

$$\boldsymbol{Adv}^{blind}_{\mathsf{CBISch},\mathcal{A}} \leq 4 \cdot (\boldsymbol{Adv}^{blind}_{\mathsf{BISch},\mathcal{B}_1} + \boldsymbol{Adv}^{blind}_{\mathsf{BISch},\mathcal{B}_2}) \ .$$

Since the (standard) blind Schnorr signature scheme is perfectly blind [CP93], by the above, our variant also satisfies perfect blindness.

Analyzing $\ell = 1$. The modified ROS problem for $\ell = 1$ is as follows (dropping index j). The adversary can query $\mathsf{H}_{\mathrm{ros}}(\rho_0, \rho_1, aux)$, where $aux = (\gamma, \xi, m)$. At some point it queries $\textsc{Select}(c_0, c_1)$ and gets $b \xleftarrow{\$} \{0,1\}$. It can keep making hash queries. Eventually, the adversary returns

$$((\rho_{1,0}, \rho_{1,1}, aux_1), (\rho_{2,0}, \rho_{2,1}, aux_2)).$$

The adversary wins if

$$\begin{aligned}
\rho_{1,b} c_b &= \mathsf{H}_{\mathrm{ros}}(\rho_{1,0}, \rho_{1,1}, aux_1) \\
\rho_{2,b} c_b &= \mathsf{H}_{\mathrm{ros}}(\rho_{2,0}, \rho_{2,1}, aux_2) \\
\rho_{1,b-1} &= \rho_{2,b-1} = 0.
\end{aligned}$$

First, we can assume $(\rho_{1,0}, \rho_{1,1}) \neq (0,0)$ and $(\rho_{2,0}, \rho_{2,1}) \neq (0,0)$. Otherwise, this is a standard Schnorr forgery and we can solve DL: The adversary submits a hash query $H(R^*, m^*)$ with a representation (γ, ξ) and returns a forgery $(m^*, (R^*, s^*))$. Hence, we can assume the adversary makes no query of the form $\mathsf{H}_{\mathrm{ros}}(0, 0, aux)$.

Let q'_{h}, resp. q''_{h}, be the number of hash queries before, resp. after the call to Select. After the call to Select, hash queries don't help much since c_b is fixed. Hence, $\mathsf{H}_{\mathrm{ros}}(\rho_{i,1}, \rho_{i,2}, aux_i) = \rho_{i,b} c_b$ with probability $1/p$. Hence, the adversary finds two hash queries satisfying the system with probability at most $(q''_{\mathrm{h}}/p)^2$.

Consider now the hash queries before the call to Select. The adversary's output must be such that $\rho_{1,0} = \rho_{2,0} = 0$ or $\rho_{1,1} = \rho_{2,1} = 0$ as otherwise it cannot win. So the best strategy seems to be to guess b first, and only make queries of the form $\mathsf{H}_{\mathrm{ros}}(0, \rho_1, aux)$ if guess $b = 0$ and $\mathsf{H}_{\mathrm{ros}}(\rho_0, 0, aux)$ if guess $b = 1$. Then, the probability to find two hash queries such that the system has a solution is at most $(q'_{\mathrm{h}})^2/p$. So one should be able to prove that the best advantage is $\max\{(q'_{\mathrm{h}})^2/(2p), (q''_{\mathrm{h}})^2/p^2\}$.

Arbitrary ℓ. Assume to simplify that the adversary makes all its hash queries, and then all its Select queries. We simplify the game for the adversary. At the end of its hash queries, we draw $b_j \xleftarrow{\$} \{0,1\}$ for $j \in [\ell]$ and consider any subset of $\ell + 1$ out of the q_{h} queries. If the system $\sum_{j=1}^{\ell} \rho_{i,j,b_j} c_j = \mathsf{H}_{\mathrm{ros}}(\gamma_i, \xi_i, \overrightarrow{\rho_i}, m_i) \bmod p$ admits a solution and $\forall i \in [\ell+1], \forall j \in [\ell] : \rho_{i,j,1-b_j} = 0$, then we say the adversary has won. Consider any subset of $\ell + 1$ hash queries. Let k be the number of integers $j \in [\ell]$ such that $\forall i \in [\ell+1]$, $\rho_{i,j,0} = \rho_{i,j,1} = 0$. (This is the number of columns that are simultaneously zero in both matrices $\hat{\rho}_0 = (\rho_{i,j,0})$ and $\hat{\rho}_1 = (\rho_{i,j,1})$. For these columns, the random draw of b_j "does not count".) Then, for $\ell - k$ integers j' in $[\ell]$, the j'-th column of either $\hat{\rho}_0$ or $\hat{\rho}_1$ is non-zero, so that

$$\Pr\left[b_1, \ldots, b_\ell \xleftarrow{\$} \{0,1\} : \forall i \in [\ell+1], \forall j \in [\ell], \rho_{i,j,1-b_j} = 0\right] \leq \frac{1}{2^{\ell-k}}.$$

(This probability could be zero if for some the j'-th column in both $\hat{\rho}_0$ and $\hat{\rho}_1$ is non-zero for some j'.) now consider the matrix $\hat{\rho} = (\rho_{i,j,b_j})$. This $(\ell+1) \times \ell$ matrix has at least k zero columns, hence its rank is at most $\ell - k$. This implies that, over the draw of $\mathsf{H}_{\mathrm{ros}}$, the probability that the system $\sum_{j=1}^{\ell} \rho_{i,j,b_j} c_j = \mathsf{H}_{\mathrm{ros}}(\gamma_i, \xi_i, \overrightarrow{\rho_i}, m_i) \bmod p$ admits a solution is at most $1/p^{k+1}$. (This is the probability that the vector of inhomogeneities is in the span of the columns. If the rank of the matrix is s, this probability is $p^s/p^{\ell+1} \leq p^{\ell-k}/p^{\ell+1}$.) Hence, the adversary wins for any subset of $\ell+1$ queries with probability at most

$$\max_{k \in [0..\ell]} \frac{1}{2^{\ell-k}} \cdot \frac{1}{p^{k+1}} = \frac{1}{p \cdot 2^{\ell}} \leq \frac{1}{2^{\lambda+\ell-1}}.$$

By the union bound, the adversary wins with probability at most

$$\frac{\binom{q_{\mathrm{h}}}{\ell+1}}{2^{\lambda+\ell-1}}.$$

6.4 Blindness of the Clause Blind Schnorr Signatures

In this section we formally prove blindness of the clause blind Schnorr signature scheme CBISch, whose signing protocol is defined in Figure 6.1, by reducing it to blindness of the (standard) blind Schnorr signature scheme BISch (Figure 5.5).

In the game defining blindness for BISch, the adversary plays the role of the signer and interacts with oracles that simulate a user running two signing sessions. Oracle U_1 reproduces the first interaction $\mathsf{BISch.User}_1$ of session i, in which the user sends a challenge c. Oracle U_2 is the second interaction $\mathsf{BISch.User}_2$, which, once both sessions are finished, outputs the resulting signatures.

The formal game $\mathrm{BLIND}_{\mathsf{BISch}}$ for adversary $\mathcal{B}$ is specified in Figure 6.5, where we follow the definition from Hauk, Kiltz and Loss [HKL19]. As usual, $\mathcal{B}$'s advantage is defined as $\mathbf{Adv}^{blind}_{\mathsf{BISch},\mathcal{B}} := 2 \cdot \Pr\left[1 \leftarrow \mathrm{BLIND}^{\mathcal{B}}_{\mathsf{BISch}}(\lambda)\right] - 1$.

Proof of Theorem 6.2 Figure 6.5 shows the blindness game for clause blind Schnorr signatures, where we have replaced $\mathsf{CBISch.User}_1$ and $\mathsf{CBISch.User}_2$ by their instantiations in terms of $\mathsf{BISch.User}_1$ and $\mathsf{BISch.User}_2$: the user first runs two instances of $\mathsf{BISch.User}_1$, and the signer calls U_2 with an additional input β, which specifies which instance the signer completes.

To reduce blindness of CBISch to blindness of BISch, we will guess the bits β_0 and β_1 that the adversary will use in its calls to U_2: game $\mathrm{G}^{\mathcal{A}}(\lambda)$, specified in Figure 6.5, is defined like $\mathrm{BLIND}^{\mathcal{A}}_{\mathsf{CBISch}}$, except that it picks two random bits $\hat{\beta}_0$ and $\hat{\beta}_1$ and aborts if its guess was wrong. (We also make a syntactical change in that U_2 continues session $\hat{\beta}_i$ instead of β_i; when $\hat{\beta}_i \neq \beta_i$, the simulation is not correct, but the game ignores $\mathcal{A}$'s output anyway.) When $\hat{\beta}_0 \neq \beta_0$ or $\hat{\beta}_1 \neq \beta_1$, the bit b' is random, so we have

$$\Pr\left[1 \leftarrow \mathrm{G}^{\mathcal{A}}(\lambda) \,\middle|\, \hat{\beta}_0 \neq \beta_0 \vee \hat{\beta}_1 \neq \beta_1\right] = \tfrac{1}{2} \ . \tag{6.2}$$

On the other hand, when $\hat{\beta}_0 = \beta_0$ and $\hat{\beta}_1 = \beta_1$, the game is the same as the original blindness game, whose output is independent of the guess, which yields

$$\Pr\left[1 \leftarrow \mathrm{G}^{\mathcal{A}}(\lambda) \,\middle|\, \hat{\beta}_0 = \beta_0 \wedge \hat{\beta}_1 = \beta_1\right] = \Pr\left[1 \leftarrow \mathrm{BLIND}^{\mathcal{A}}_{\mathsf{CBISch}}(\lambda)\right] \ . \tag{6.3}$$

From Eqs. (6.2) and (6.3), we have

$$\Pr\left[1 \leftarrow \mathrm{G}^{\mathcal{A}}(\lambda)\right] = \tfrac{1}{2} \cdot \tfrac{3}{4} + \Pr\left[1 \leftarrow \mathrm{BLIND}^{\mathcal{A}}_{\mathsf{CBISch}}(\lambda)\right] \cdot \tfrac{1}{4} \ ,$$

and thus

$$\mathbf{Adv}^{\mathrm{G}}_{\mathcal{A}} = 2 \cdot \Pr\left[1 \leftarrow \mathrm{G}^{\mathcal{A}}(\lambda)\right] - 1 = \tfrac{1}{4} \cdot \mathbf{Adv}^{blind}_{\mathsf{CBISch},\mathcal{A}} \ . \tag{6.4}$$

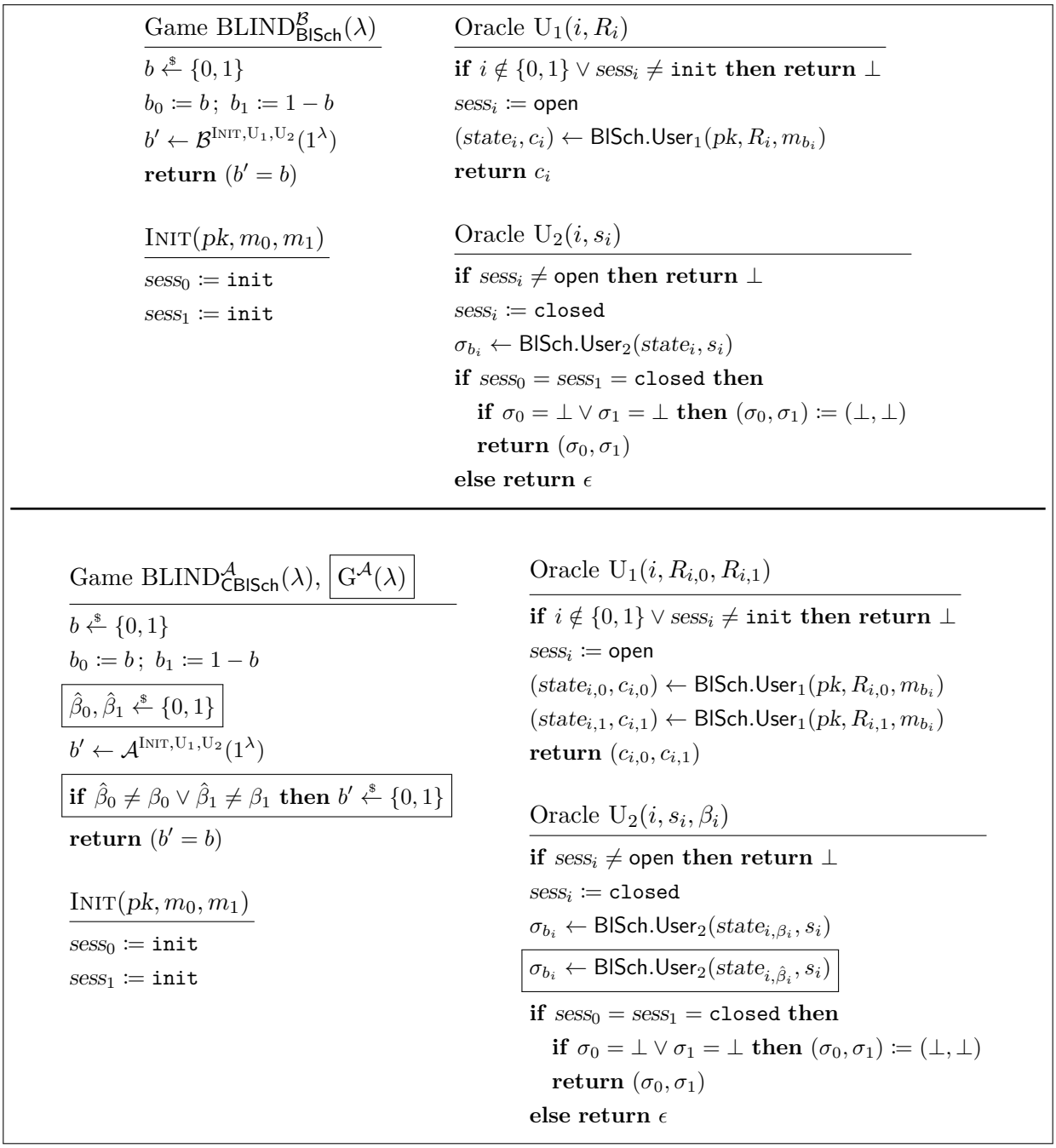

Figure 6.5: The blindness game for the blind Schnorr signature scheme BISch (top) and (bottom) for the clause blind Schnorr signature scheme CBISch (ignoring boxes) and game G (including the boxes) used in the proof of Theorem 6.2.

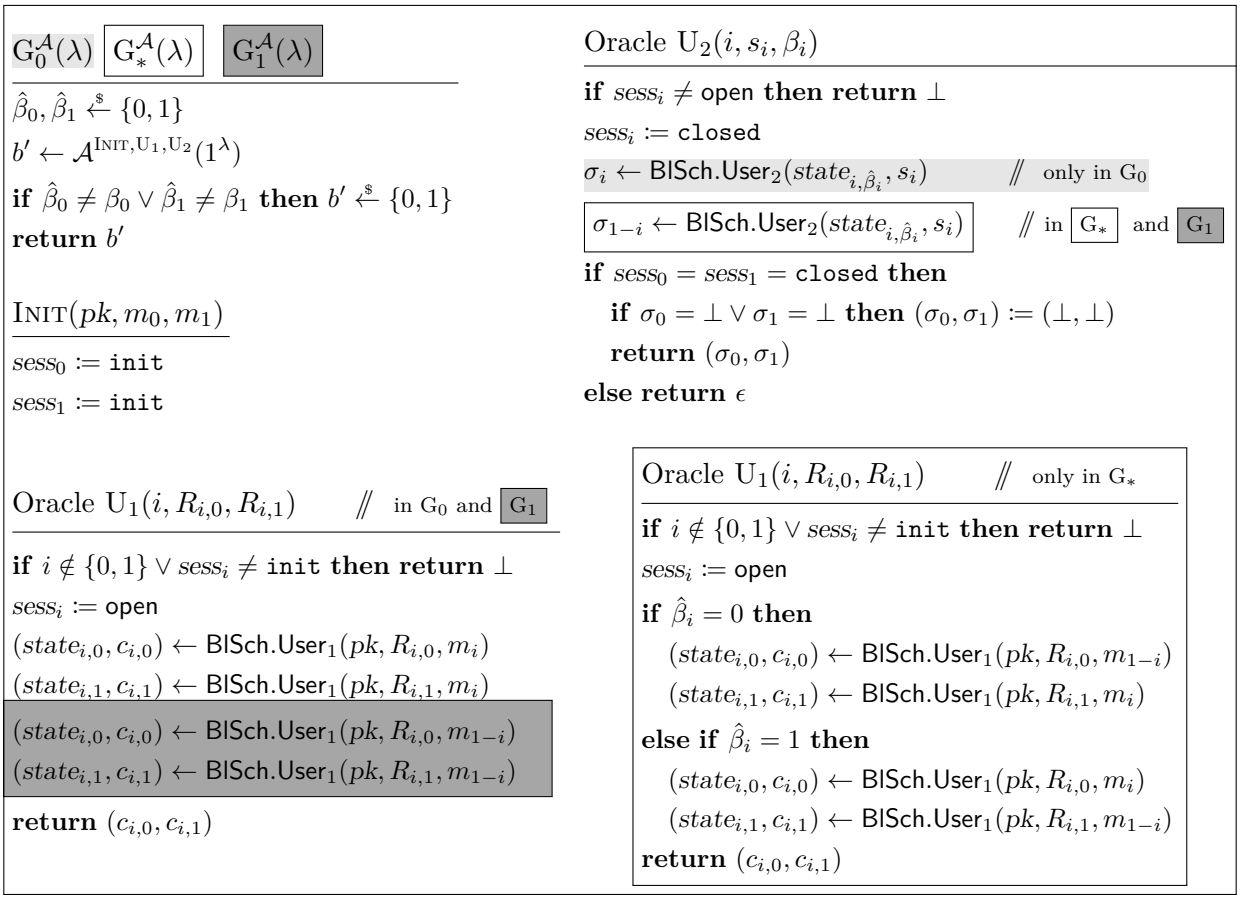

Figure 6.6: Description of the games G_0 and G_1 which fix the bit b in game G from Figure 6.5. G_* is a hybrid game that makes the transition between G_0 and G_1.

In the remainder of the proof, we will show that the adversary's behavior only changes negligibly when the bit b changes from 0 to 1. To do so, we define $\mathrm{G}_0^{\mathcal{A}}$ and $\mathrm{G}_1^{\mathcal{A}}$ by modifying $\mathrm{G}^{\mathcal{A}}$ as follows: the bit b is fixed to 0 and 1, respectively, and the game *directly* outputs bit b'. The games are specified in Figure 6.6 and we define $\text{BLIND}^{\mathcal{B}}_{0,\mathsf{BISch}}$ and $\text{BLIND}^{\mathcal{B}}_{1,\mathsf{BISch}}$ analogously. We have:

$$\begin{aligned}\mathbf{Adv}^{\mathrm{G}}_{\mathcal{A}} &= \Pr\left[1 \leftarrow \mathrm{G}^{\mathcal{A}}(\lambda) \,|\, b = 1\right] + \Pr\left[1 \leftarrow \mathrm{G}^{\mathcal{A}}(\lambda) \,|\, b = 0\right] - 1 \\ &= \Pr\left[1 \leftarrow \mathrm{G}_1^{\mathcal{A}}(\lambda)\right] - \Pr\left[1 \leftarrow \mathrm{G}_0^{\mathcal{A}}(\lambda)\right] . \end{aligned} \tag{6.5}$$

We now define a hybrid game G_* which lies "between" G_0 and G_1 and is also specified in Figure 6.6. It differs from G_0 in the $\hat{\beta}_i$-th message used in signing session i and from G_1 in the $(1-\hat{\beta}_i)$-th message. Since

$$\mathbf{Adv}^{\mathrm{G}}_{\mathcal{A}} = \Pr[1 \leftarrow \mathrm{G}_1^{\mathcal{A}}(\lambda)] - \Pr[1 \leftarrow \mathrm{G}_*^{\mathcal{A}}(\lambda)] + \Pr[1 \leftarrow \mathrm{G}_*^{\mathcal{A}}(\lambda)] - \Pr[1 \leftarrow \mathrm{G}_0^{\mathcal{A}}(\lambda)] , \tag{6.6}$$

it suffices to bound these two differences. For the first, we construct an adversary $\mathcal{B}_1$ playing game $\text{BLIND}_{\mathsf{BISch}}$ and simulating G to $\mathcal{A}$ so that if $\mathcal{B}_1$ plays $\text{BLIND}_{0,\mathsf{BISch}}$, it simulates G_0 to $\mathcal{A}$; whereas if it plays $\text{BLIND}_{1,\mathsf{BISch}}$, it simulates G_* to $\mathcal{A}$. Adversary $\mathcal{B}_1$ thus embeds its interaction with its challenger as the two sessions that $\mathcal{A}$ will conclude (provided that $\hat{\beta}_0$ and $\hat{\beta}_1$ are guessed correctly); it is specified in Figure 6.7. By inspection, we have

$$\begin{aligned}\Pr\left[1 \leftarrow \text{BLIND}^{\mathcal{B}_1}_{0,\mathsf{BISch}}(\lambda)\right] &= \Pr\left[1 \leftarrow \mathrm{G}_0^{\mathcal{A}}(\lambda)\right] \text{ and} \\ \Pr\left[1 \leftarrow \text{BLIND}^{\mathcal{B}_1}_{1,\mathsf{BISch}}(\lambda)\right] &= \Pr\left[1 \leftarrow \mathrm{G}_*^{\mathcal{A}}(\lambda)\right] . \end{aligned} \tag{6.7}$$

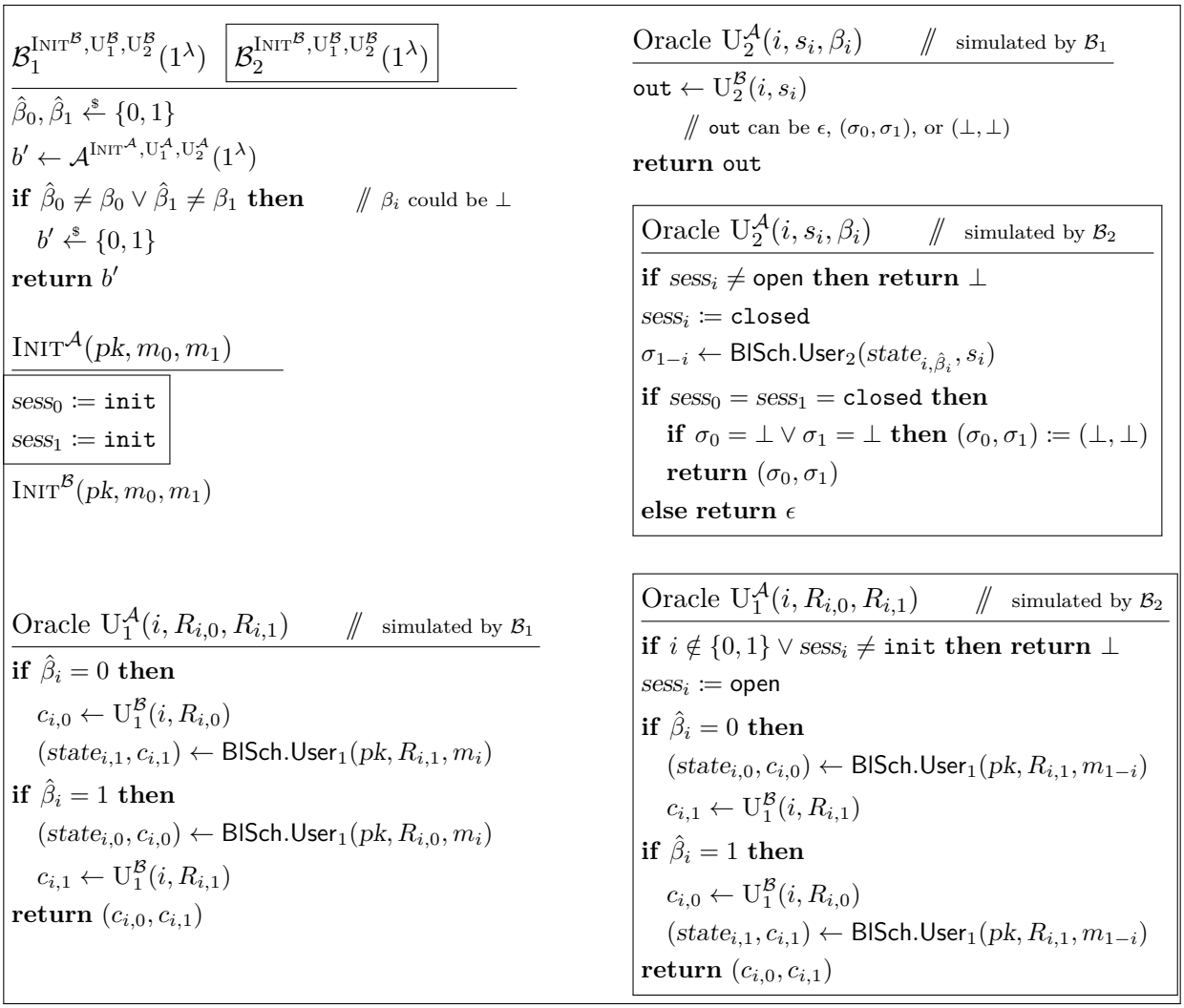

Figure 6.7: Description of adversaries $\mathcal{B}_1$ and $\mathcal{B}_2$ in the proof of Theorem 6.2.

We also construct an adversary $\mathcal{B}_2$ that simulates game $\text{G}_*^{\mathcal{A}}(\lambda)$ or $\text{G}_1^{\mathcal{A}}(\lambda)$. It embeds its interaction as the sessions that $\mathcal{A}$ will abort and executes the concluding sessions (which are the same in G_* and G_1) on its own. Adversary $\mathcal{B}_2$ is also specified in Figure 6.7 (note that in its simulation of U_2, the variable $state_{i,\hat{\beta}_i}$ is always defined because of our syntactical change in Figure 6.5). We have

$$\begin{aligned} \Pr[1 \leftarrow \text{BLIND}^{\mathcal{B}_2}_{0,\mathsf{BlSch}}(\lambda)] &= \Pr[1 \leftarrow \text{G}_*^{\mathcal{A}}(\lambda)] \quad \text{and} \\ \Pr[1 \leftarrow \text{BLIND}^{\mathcal{B}_2}_{1,\mathsf{BlSch}}(\lambda)] &= \Pr[1 \leftarrow \text{G}_1^{\mathcal{A}}(\lambda)] \ . \end{aligned} \tag{6.8}$$

From Eqs. (6.6) – (6.8) we get

$$\mathbf{Adv}^{\text{G}}_{\mathcal{A}} = \mathbf{Adv}^{blind}_{\mathsf{BlSch},\mathcal{B}_1} + \mathbf{Adv}^{blind}_{\mathsf{BlSch},\mathcal{B}_2} \ ,$$

which, together with Equation 6.4, concludes the proof.

Chapter 7

ElGamal Public Key Encapsulation Scheme in the AGM

7.1 Schnorr-Signed ElGamal Encryption

A public key for the ElGamal public-key encryption (PKE) scheme is a group element $Y \in \mathbb{G}$. Messages are group elements $M \in \mathbb{G}$ and to encrypt M under Y, one samples a random $x \in \mathbb{Z}_p$ and derives an ephemeral key $K := xY$ to blind the message: $C := xY + M$. Given in addition the value $X := xG$, the receiver that holds $y = \log Y$ can derive $K := yX$ and recover $M := C - K$.

Under the decisional Diffie-Hellman (DDH) assumption (see Figure 7.1), ciphertexts of different messages are computationally indistinguishable: replacing K by a random value K' makes the ciphertext C perfectly hide the message. In the AGM, ElGamal, viewed as a key-encapsulation mechanism (KEM) was shown to satisfy CCA1-security (where the adversary can only make decryption queries before seeing the challenge key) under a parametrized variant of DDH [FKL18].

The idea of *Schnorr-signed* ElGamal is to accompany the ciphertext by a proof of knowledge of the randomness $x = \log X$ used to encrypt, in particular, a Schnorr signature on the pair (X, C) under the public key X. The scheme is detailed in Figure 7.2. (Note that we changed the argument order in the hash function call compared to section 5.1 so that it is the same as in ciphertexts.)

The strongest security notion for PKE is indistinguishability of ciphertexts under adaptive chosen-ciphertext attack (IND-CCA2), where the adversary can query decryptions of ciphertexts of its choice even after receiving the challenge. The (decisional) game IND-CCA2 is defined in Figure 7.3.

When ephemeral keys are hashed (that is, defined as $k := \mathsf{H}'(xY)$) and the scheme is viewed as a KEM, then CCA2-security can be reduced to the *strong* Diffie-Hellman (SDH) assumption[1] [ABR01, CS03] in the ROM. In section 7.3 we show that when key hashing is applied to the Schnorr-signed ElGamal scheme from Figure 7.2, then in the AGM+ROM we can directly reduce CCA2 security of the corresponding KEM to the DL assumption (Figure 2.1); in particular, we do so using a *tight* security proof (note that SDH is equivalent to DL in the AGM [FKL18] but the reduction from DL to SDH is non-tight). Here we prove that the Schnorr-signed ElGamal PKE is IND-CCA2-secure in the AGM+ROM under the DDH assumption.

Theorem 7.1. *Let* GrGen *be a group generator. Let* $\mathcal{A}_{\mathrm{alg}}$ *be an algebraic adversary against the IND-CCA2 security of the Schnorr-signed ElGamal PKE scheme* $\mathsf{SEG}[\mathsf{GrGen}]$ *making at most* q_{d} *decryption queries and* q_{h} *queries to the random oracle. Then there exist two algorithms* $\mathcal{B}_1$ *and* $\mathcal{B}_2$

[1] SDH states that given $X = xG$ and Y it is infeasible to compute xY even when given access to an oracle which on input (Y', Z') returns 1 if $Z' = xY'$ and 0 otherwise.

Game $\mathrm{DDH}^{\mathcal{A}}_{\mathsf{GrGen}}(\lambda)$

$(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$; $b \xleftarrow{\$} \{0,1\}$; $x, y, z \xleftarrow{\$} \mathbb{Z}_p$
$X := xG$; $Y := yG$; $Z_0 := xyG$; $Z_1 := zG$
$b' \leftarrow \mathcal{A}(p, \mathbb{G}, G, X, Y, Z_b)$
return $(b = b')$

Figure 7.1: The DDH problem.

$\mathsf{SEG.Setup}(\lambda)$

$(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
Select $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_p$
return $par := (p, \mathbb{G}, G, \mathsf{H})$

$\mathsf{SEG.KeyGen}(par)$

$(p, \mathbb{G}, G, \mathsf{H}) := par$; $y \xleftarrow{\$} \mathbb{Z}_p$; $Y := yG$
$sk := (par, y)$; $pk := (par, Y)$
return (sk, pk)

$\mathsf{SEG.Enc}(pk, M)$

$(p, \mathbb{G}, G, \mathsf{H}, Y) := pk$; $x, r \xleftarrow{\$} \mathbb{Z}_p$
$X := xG$; $R := rG$; $C := xY + M$
$s := r + \mathsf{H}(X, C, R) \cdot x \bmod p$
return (X, C, R, s)

$\mathsf{SEG.Dec}(sk, (X, C, R, s))$

$(p, \mathbb{G}, G, \mathsf{H}, y) := sk$
if $sG \neq R + \mathsf{H}(X, C, R) \cdot X$ **then**
  return $\bot$
return $M := C - yX$

Figure 7.2: The Schnorr-Signed ElGamal PKE scheme $\mathsf{SEG}[\mathsf{GrGen}]$.

Game $\text{IND-CCA2}^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)$

$par \leftarrow \mathsf{PKE.Setup}(\lambda)$
$(pk, sk) \leftarrow \mathsf{PKE.KeyGen}(par)$
$b \xleftarrow{\$} \{0,1\}$
$b' \leftarrow \mathcal{A}^{\mathrm{Enc}, \mathrm{Dec}}(pk)$
return $(b = b')$

Oracle $\mathrm{Enc}(m_0, m_1)$ // one time

$c^* \leftarrow \mathsf{PKE.Enc}(pk, m_b)$
return c^*

Oracle $\mathrm{Dec}(c)$

if $c = c^*$ **then return** $\bot$
return $\mathsf{PKE.Dec}(sk, c)$

Figure 7.3: The IND-CCA2 security game for a PKE scheme PKE.

solving respectively the DL problem and the DDH problem w.r.t. GrGen*, such that*

$$\boldsymbol{Adv}^{\textit{ind-cca2}}_{\mathsf{SEG}[\mathsf{GrGen}], \mathcal{A}_{\mathrm{alg}}} \leq 2 \cdot \boldsymbol{Adv}^{\textit{ddh}}_{\mathsf{GrGen}, \mathcal{B}_2} + \boldsymbol{Adv}^{\textit{DL}}_{\mathsf{GrGen}, \mathcal{B}_1} + \frac{q_{\mathrm{d}} + \frac{1}{2^{\lambda-1}}(q_{\mathrm{d}} + q_{\mathrm{h}})}{2^{\lambda-1}} \ .$$

We start with the proof idea. The full proof can be found in section 7.2. Let Y be the public key, let P_0 and P_1 denote the challenge plaintexts, and let $(X^* = x^*G, C^* = x^*Y + P_b, R^*, s^*)$ be the challenge ciphertext. Under the DDH assumption, given Y and X^*, the value x^*Y looks random. We can thus replace x^*Y by a random group element Z^*, which perfectly hides P_b and leads to a game where the adversary gains no information about the challenge bit b.

It remains to show how the reduction can simulate the game without knowledge of $\log X^*$ (needed to sign the challenge ciphertext) and $\log Y$ (needed to answer decryption queries). The Schnorr signature under X^* contained in the challenge ciphertext can be simulated by programming the random oracle H as for Theorem 5.1.

Decryption queries leverage the fact that the Schnorr signature contained in a queried ciphertext (X, C, R, s) proves knowledge of x with $X = xG$. Thus, intuitively, the reduction should be able to answer a query by extracting x and returning $M = C - xY$. However, this extraction is a lot trickier than in the proof of Theorem 5.1: During the game the adversary obtains group elements Y, X^*, C^*, and R^*, as well as the answers $M_1, \ldots, M_{q_\mathrm{d}}$ to its queries to Dec. The adversary's representations of group elements can thus depend on all these elements. In particular, since Dec on input $(X, C, \ldots)$ computes $M := C - yX$, by successive calls to Dec, the adversary can obtain arbitrary powers of y.

In our proof we first show that from a representation given by the adversary, we can always (efficiently) derive a representation in basis

$$(G, X^*, Y = yG, \ldots, y^{q_\mathrm{d}+1}G, x^*yG, \ldots, x^*y^{q_\mathrm{d}+1}G) \ .$$

Now consider a decryption query (X, C, R, s), each group element represented as

$$X = \gamma_x G + \xi_x X^* + \textstyle\sum_{i=1}^{q_d+1} \upsilon_x^{(i)} y^i G + \sum_{i=1}^{q_d+1} \zeta_x^{(i)} x^* y^i G \ , \qquad R = \gamma_r G + \ldots \tag{7.1}$$

We show that each query falls into one of three categories:
(1) The choice of $c = \mathsf{H}(X, C, R)$ was unlucky, which only happens with negligible probability (this corresponds to an abort in line (I) in Figure 7.4 in section 7.2).
(2) The representation of X is independent of Y, that is, $X = \gamma_x G + \xi_x X^*$. Then xY (and hence the answer $M = C - xY$ to the query) can be computed as $xY := \gamma_x Y + \xi_x Z^*$ (where $Z^* := x^*Y$ is known by the reduction).
(3) Otherwise we show that the adversary has actually computed $\log Y$ (corresponding to an abort in line (III) in Figure 7.4): If the Dec query was valid then $sG = R + cX$, which, by plugging in the representations (7.1) yields

$$0 = (\gamma_r + c\gamma_x - s)G + (\xi_r + c\xi_x)X^* + \sum_{i=1}^{q_d+1} \Big(\underbrace{(\upsilon_r^{(i)} + x^*\zeta_r^{(i)}) + c\overbrace{(\upsilon_x^{(i)} + x^*\zeta_x^{(i)})}^{=:\beta^{(i)}}}_{=:\alpha^{(i)}} \Big) y^i G$$

If $\beta^{(i)} \equiv_p 0$ for all i, we are in case (2). If $\beta^{(j)} \not\equiv_p 0$ for some j and $\alpha^{(i)} \equiv_p 0$ for all i, then $c \equiv_p -(\upsilon_r^{(j)} + x^*\zeta_r^{(j)}) \cdot (\beta^{(j)})^{-1}$ was an unlucky choice (made *after* the adversary chose its representations from (7.1)) (case (1)). Otherwise $\alpha^{(j)} \equiv_p 0$ for some j and

$$0 = \gamma_r + c\gamma_x - s + (\xi_r + c\xi_x)x^* + \textstyle\sum_{i=1}^{q_d+1} \alpha^{(i)} y^i$$

can be solved for y. (Note that the reduction to DL chooses x^* itself.)

7.2 Proof of the Theorem

Consider games Game_0–Game_4 in Figure 7.4, where Game_0 is $\text{IND-CCA2}^{\mathcal{A}_\mathrm{alg}}_{\mathsf{SEG}[\mathsf{GrGen}]}$ and the adversary's advantage in Game_4 is 0. We prove the theorem by bounding the probability that the adversary behaves differently in two consecutive games Game_i and Game_{i+1}.

First, we establish some notation regarding the representation of group elements. Let (P_0, P_1) be the two messages of the adversary's call to Enc. At the beginning of the experiment, the only group-element inputs to $\mathcal{A}_\mathrm{alg}$ are G and the challenge public key Y. When the adversary queries its Enc oracle, it receives three more group elements (X^*, C^*, R^*) in the answer. Furthermore, as the adversary queries its Dec oracle, it receives additional group elements $M_1, \ldots, M_{q_d}$ in response. All in all, representations provided by the adversary are w.r.t. $(G, Y, X^*, C^*, R^*, M_1, \ldots, M_{q_d})$ and for a group element A we write

$$A = \gamma_a G + \upsilon_a Y + \xi_a X^* + \kappa_a C^* + \rho_a R^* + \sum_{j=1}^{q_d} \mu_{a,j} M_i \ ,$$

with the convention that $(\xi_a, \kappa_a, \rho_a) = (0,0,0)$ before the call to Enc and $\mu_{a,j} = 0$ before the j-th call to Dec.

Claim Consider a query $\mathsf{H}(X_{[\gamma_x,\upsilon_x,\xi_x,\kappa_x,\rho_x,(\mu_{x,j})_j]}, C_{[\dots]}, R_{[\dots]})$ or $\textsc{Dec}(X_{[\dots]}, C_{[\dots]}, R_{[\dots]}, s)$ in Game_0. Then there exist efficiently computable coefficients $(\bar\gamma_r, \bar\xi_r, (\bar\upsilon_r^{(i)})_i, (\bar\zeta_r^{(i)})_i, \bar\gamma_x, \bar\xi_x, (\bar\upsilon_x^{(i)})_i, (\bar\zeta_x^{(i)})_i)$ which only depend on c^*, s^*, and on the coefficients of the representations of X, C, R and of the group elements contained in previous Dec queries such that

$$X = \bar\gamma_x G + \bar\xi_x X^* + \sum_{i=1}^{q_d+1} \bar\upsilon_x^{(i)} y^i G + \sum_{i=1}^{q_d+1} \bar\zeta_x^{(i)} x^* y^i G \tag{7.2}$$

$$R = \bar\gamma_r G + \bar\xi_r X^* + \sum_{i=1}^{q_d+1} \bar\upsilon_r^{(i)} y^i G + \sum_{i=1}^{q_d+1} \bar\zeta_r^{(i)} x^* y^i G \ . \tag{7.3}$$

Proof When the adversary queries H on a tuple (X, C, R) or Dec on a tuple (X, C, R, s), it provides a representation of X, C, and R in terms of the group elements received so far:

$$X = \gamma_x G + \upsilon_x Y + \xi_x X^* + \kappa_x C^* + \rho_x R^* + \textstyle\sum_{j=1}^{q_d} \mu_{x,j} M_j \tag{7.4}$$

$$C = \gamma_c G + \upsilon_c Y + \xi_c X^* + \kappa_c C^* + \rho_c R^* + \textstyle\sum_{j=1}^{q_d} \mu_{c,j} M_j \tag{7.5}$$

$$R = \gamma_r G + \upsilon_r Y + \xi_r X^* + \kappa_r C^* + \rho_r R^* + \textstyle\sum_{j=1}^{q_d} \mu_{r,j} M_j \ . \tag{7.6}$$

Let $\mathcal{B}_j = (G, X^*, yG, \dots, y^jG, x^*yG, \dots, x^*y^jG)$. We will show the following:

1. the j-th message M_j returned by Dec can be represented over $\mathcal{B}_{j+1}$;
2. C^* and R^* can be represented over $\mathcal{B}_{q_d+1}$.

Combined with (7.4) and (7.6), this will prove the claim.

We first show (i) for the Dec calls before the Enc query (if any). Consider the first Dec call before the Enc query and let (X_1, C_1, R_1, s_1) be its input. Then (7.4) and (7.5) simplify to $X_1 = \gamma_{x,1} G + \upsilon_{x,1} Y$ and $C_1 = \gamma_{c,1} G + \upsilon_{c,1} Y$. The output of Dec is thus

$$M_1 \coloneqq C_1 - yX_1 = \underbrace{\gamma_{c,1}}_{=:\gamma_{m,1}} G + \underbrace{(\upsilon_{c,1} - \gamma_{x,1})}_{=:\upsilon_{m,1}^{(1)}} Y + \underbrace{(-\upsilon_{x,1})}_{=:\upsilon_{m,1}^{(2)}} y^2 G \ .$$

In the second Dec call preceding the Enc query, the representation of the arguments (X_2, C_2, R_2, s_2) can also depend on M_1, so analogously, we can define coefficients $\gamma_{m,2}, \upsilon_{m,2}^{(1)}, \upsilon_{m,2}^{(2)}$ and $\upsilon_{m,2}^{(3)}$ such that the second output M_2 of Dec satisfies

$$M_2 = \gamma_{m,2} G + \upsilon_{m,2}^{(1)} Y + \upsilon_{m,2}^{(2)} y^2 G + \upsilon_{m,2}^{(3)} y^3 G \ .$$

More generally, the j-th message returned by the Dec oracle before the Enc query can be written as

$$M_j = \gamma_{m,j} G + \textstyle\sum_{i=1}^{j+1} \upsilon_{m,j}^{(i)} y^i G \ . \tag{7.7}$$

In the following, we let k denote the number of queries to Dec before the Enc call.

When the adversary queries its Enc oracle, it provides the representation of the challenge messages P_0 and P_1. We thus have for $b \in \{0,1\}$, $P_b = \gamma_{p,b} G + \upsilon_{p,b} Y + \sum_{j=1}^{k} \mu_{p,b,j} M_j$. Analogously to the above, using (7.7) we can define coefficients such that

$$P_b = \bar\gamma_{p,b} G + \textstyle\sum_{i=1}^{k+1} \bar\upsilon_{p,b}^{(i)} y^i G \ .$$

By inspection of the code of Enc, it follows that

$$C^* = x^*Y + P_b = x^*Y + \bar{\gamma}_{p,b}G + \textstyle\sum_{i=1}^{k+1} \bar{\upsilon}_{p,b}^{(i)} y^i G \quad \text{and} \tag{7.8}$$

$$R^* = s^*G - c^*X^* \ , \tag{7.9}$$

which proves (ii). Consider now the first Dec query after the Enc query. Plugging in (7.8) and (7.9) into Equation (7.4) yields

$$X = (\gamma_x + \kappa_x\bar{\gamma}_{p,b} + \rho_x s^*)G + (\xi_x - \rho_x c^*)X^* + (\upsilon_x + \kappa_x\bar{\upsilon}_{p,b}^{(1)})Y + \kappa_x x^*Y \\ + \textstyle\sum_{i=2}^{k+1} \kappa_x\bar{\upsilon}_{p,b}^{(i)} y^i G + \sum_{j=1}^{k} \mu_{x,j} M_j \ .$$

Moreover, the M_i's are still of the form as in (7.7), hence we obtain

$$X = \underbrace{(\gamma_x + \kappa_x\bar{\gamma}_{m,b} + \rho_x s^* + \textstyle\sum_{j=1}^{k} \mu_{x,j}\gamma_{m,j})}_{=:\bar{\gamma}_x} G + \underbrace{(\xi_x - \rho_x c^*)}_{=:\bar{\xi}_x} X^* \\ + \underbrace{(\upsilon_x + \kappa_x\bar{\upsilon}_{p,b}^{(1)} + \textstyle\sum_{j=1}^{k} \mu_{x,j}\upsilon_{m,j}^{(1)})}_{=:\bar{\upsilon}_x^{(1)}} Y + \textstyle\sum_{i=2}^{k+1} \underbrace{(\kappa_x\bar{\upsilon}_{p,b}^{(i)} + \sum_{j=1}^{k} \mu_{x,j}\upsilon_{m,j}^{(i)})}_{=:\bar{\upsilon}_x^{(i)}} y^i G + \underbrace{\kappa_x}_{=:\bar{\zeta}_x} x^*Y$$

and similarly for C. Hence, the output $M_{k+1} = C - yX$ of Dec is of the form

$$M_{k+1} = \gamma_{m,k+1}G + \xi_{m,k+1}X^* + \textstyle\sum_{i=1}^{k+2} \upsilon_{m,k+1}^{(i)} y^i G + \sum_{i=1}^{2} \zeta_{m,k+1}^{(i)} x^* y^i G \ .$$

More generally, the j-th message returned by Dec, $j > k$, can be written

$$M_j = \gamma_{m,j}G + \xi_{m,j}X^* + \textstyle\sum_{i=1}^{j+1} \upsilon_{m,j}^{(i)} y^i G + \sum_{i=1}^{j+1-k} \zeta_{m,j}^{(i)} x^* y^i G \ ,$$

which proves (i) for all j.

To prove Theorem 7.1, we start with the difference between Game_0 and Game_1. First note that at any point $\mathsf{T}(X^*, C^*, R^*)$ is the only value in T that might not have been set during an adversary's call to H or Dec, and that could thus does *not* have a corresponding entry in L. Moreover, if Dec does not reply $\perp$, we must have $(X, C, R) \neq (X^*, C^*, R^*)$, since otherwise by the 3rd line in Dec, we have $s = \log R^* + \mathsf{T}(X^*, C^*, R^*) \log X^* = s^*$ and the oracle would have returned $\perp$ in the 1st line.

The values $(\bar{\gamma}_x, \bar{\xi}_x, (\bar{\upsilon}_x^{(i)})_i, (\bar{\zeta}_x^{(i)})_i, \bar{\gamma}_r, \bar{\xi}_r, (\bar{\upsilon}_r^{(i)})_i, (\bar{\zeta}_r^{(i)})_i)$ as defined in Dec were (implicitly) chosen by the adversary before $\mathsf{T}(X, C, R) = c$ was randomly drawn, which, as we argued above, must have been by a call from the adversary. The value c is thus independent of $(\bar{\gamma}_x, \ldots, (\bar{\zeta}_r^{(i)})_i)$.

Game_0 and Game_1 behave identically unless Game_1 aborts during a Dec call in line (I), thus in particular for some j: $\beta^{(j)} := \bar{\upsilon}_x^{(j)} + x^*\bar{\zeta}_x^{(j)} \not\equiv_p 0$ and $\alpha^{(j)} := (\bar{\upsilon}_r^{(j)} + x^*\bar{\zeta}_r^{(j)}) + c\,(\bar{\upsilon}_x^{(j)} + x^*\bar{\zeta}_x^{(j)}) \equiv_p 0$. By the above argument, the probability that c was chosen such as $c = -(\bar{\upsilon}_r^{(j)} + x^*\bar{\zeta}_r^{(j)}) \cdot (\beta^{(j)})^{-1} \bmod p$ is upper-bounded by $\frac{1}{2^{\lambda-1}}$.

Denoting by A the event that Game_1 aborts in line (I) during some Dec call, we have $\Pr[A] \leq \frac{q_\mathrm{d}}{2^{\lambda-1}}$. Since $\Pr[1 \leftarrow \mathsf{Game}_0 \,|\, \neg A] = \Pr[1 \leftarrow \mathsf{Game}_1 \,|\, \neg A]$, we have

$$\begin{aligned} \mathbf{Adv}_{\mathcal{A}_\mathrm{alg}}^{\mathsf{Game}_0} - \mathbf{Adv}_{\mathcal{A}_\mathrm{alg}}^{\mathsf{Game}_1} &= 2\Pr[A]\big(\Pr[1 \leftarrow \mathsf{Game}_0 \,|\, A] - \Pr[1 \leftarrow \mathsf{Game}_1 \,|\, A]\big) \\ &\leq \Pr[A] \ , \end{aligned}$$

as $\Pr[1 \leftarrow \mathsf{Game}_1 \,|\, A] = \frac{1}{2}$. We thus have

$$\mathbf{Adv}_{\mathcal{A}_\mathrm{alg}}^{\mathsf{Game}_1} \geq \mathbf{Adv}_{\mathcal{A}_\mathrm{alg}}^{\mathsf{Game}_0} - \frac{q_\mathrm{d}}{2^{\lambda-1}} \ . \tag{7.10}$$

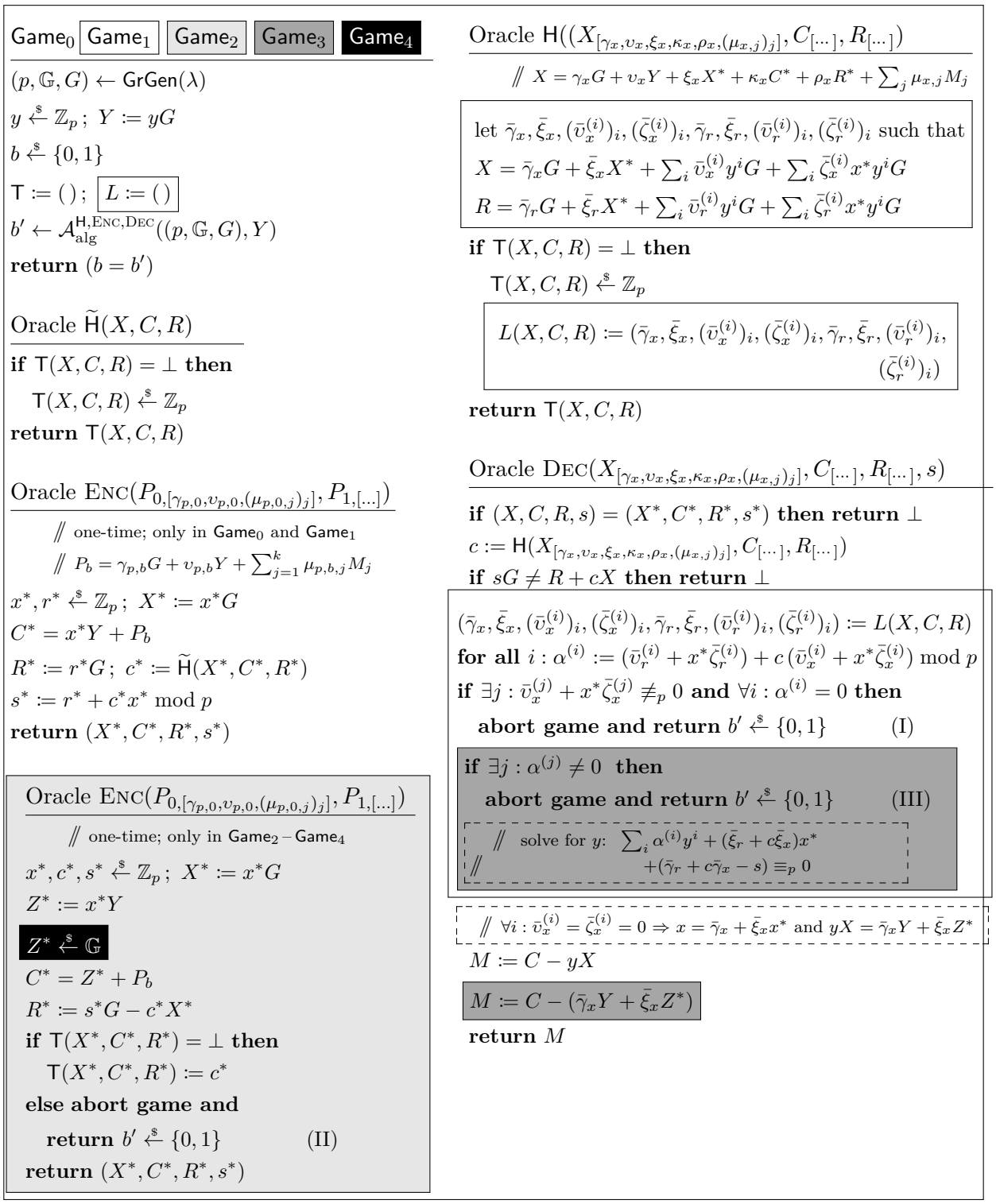

Figure 7.4: The IND-CCA2 game for the Schnorr-Signed ElGamal scheme (Game$_0$) and games Game$_1$–Game$_4$ used in the proof. The comments in dashed boxes show how the DL reduction simulates Game$_3$ without knowledge of y.

Game_1 and Game_2 behave identically unless oracle Enc generates values (X^*, C^*, R^*) that have already been assigned a value in the table T. The values X^* and R^* are uniformly random in $\mathbb{G}$. Thus, after the adversary has made q_h queries to H and q_d to Dec, at most $q_\mathrm{h} + q_\mathrm{d}$ values in T are assigned. Thus the probability that (X^*, C^*, R^*) collides with one of the entries is bounded by $\frac{q_\mathrm{h}+q_\mathrm{d}}{(2^{\lambda-1})^2}$, and we thus have

$$\mathbf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}_\mathrm{alg}} \geq \mathbf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}_\mathrm{alg}} - \frac{q_\mathrm{d} + q_\mathrm{h}}{(2^{\lambda-1})^2} \ . \tag{7.11}$$

Game_2 and Game_3 behave identically unless for some j: $\alpha^{(j)} := (\bar{\upsilon}_r^{(j)} + x^* \bar{\zeta}_r^{(j)}) + c\,(\bar{\upsilon}_x^{(j)} + x^* \bar{\zeta}_x^{(j)}) \not\equiv_p 0$. We show that when this happens, we can build a reduction $\mathcal{B}_1$ that computes the discrete logarithm of Y. The reason is that if Dec does not return $\perp$ then $sG = R + cX$, which, by plugging in (7.2) and (7.3), yields

$$\begin{aligned} 0 &= (\bar{\gamma}_r + c\bar{\gamma}_x - s)G + (\bar{\xi}_r + c\bar{\xi}_x)X^* + \sum_{i=1}^{q_d+1} (\bar{\upsilon}_r^{(i)} + c\bar{\upsilon}_x^{(i)}) y^i G + \sum_{i=1}^{q_d+1} (\bar{\zeta}_r^{(i)} + c\bar{\zeta}_x^{(i)}) x^* y^i G \\ &= (\bar{\gamma}_r + c\bar{\gamma}_x - s)G + (\bar{\xi}_r + c\bar{\xi}_x)X^* + \sum_{i=1}^{q_d+1} \alpha^{(i)} y^i G \ . \end{aligned} \tag{7.12}$$

If $\alpha^{(j)} \neq 0$ for some j, then we can solve the above for y.

In more detail, reduction $\mathcal{B}_1$ is given a challenge Y and sets it as the public key. It simulates Game_2 by choosing random values x^*, c^* and s^* during the Enc call. Moreover, it can simulate any Dec query before a potential abort in line (III) without knowledge of y as follows.

Consider a call $\mathrm{Dec}(X_{[\gamma_x, \upsilon_x, \xi_x, \kappa_x, \rho_x, (\mu_{x,i})_i]}, C_{[\dots]}, R_{[\dots]}, s)$. The oracle returns $\perp$ if $sG \neq R + \mathsf{H}(X, C, R)X$ or $(X, C, R, s) = (X^*, C^*, R^*, s^*)$. As s is determined by (X, C, R), the latter implies $(X, C, R) \neq (X^*, C^*, R^*)$ if Dec did not return $\perp$.

If furthermore Dec does not abort in line (I) or (III), then $\bar{\upsilon}_x^{(i)} + x^* \bar{\zeta}_x^{(i)} \equiv_p 0$ for all i; thus, by (7.2): $X = \bar{\gamma}_x G + \bar{\xi}_x X^*$. The reduction can thus compute $yX = \bar{\gamma}_x Y + \bar{\xi}_x x^* Y$ and return

$$M := C - (\bar{\gamma}_x + \bar{\xi}_x x^*)Y = C - xY = C - yX \ .$$

(Note that Game_3 also introduced a syntactical change by directly defining the response of Dec as $M := C - \bar{\gamma}_x Y + \bar{\xi}_x Z^* = (\bar{\gamma}_x + \bar{\xi}_x x^*)Y = C - yX$.)

This shows that $\mathcal{B}_1$ can simulate Game_3 until an abort in line (III). In this case, $\mathcal{B}_1$ returns y, the solution of the following equation (cf. (7.12)):

$$\textstyle\sum_{i=1}^{q_d+1} \alpha^{(i)} y^i + \bar{\gamma}_r + c\bar{\gamma}_x - s + (\bar{\xi}_r + c\bar{\xi}_x)x^* \equiv_p 0 \ .$$

and thus solves the DL challenge Y. This yields

$$\mathbf{Adv}^{\mathsf{Game}_3}_{\mathcal{A}_\mathrm{alg}} \geq \mathbf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}_\mathrm{alg}} - \mathbf{Adv}^{dl}_{\mathsf{GrGen}, \mathcal{B}_1} \ . \tag{7.13}$$

Finally, Game_3 and Game_4 only differ in the definition of Z^*, which in Game_3 is the CDH of X^* and Y, whereas in Game_4 it is random. We build a reduction $\mathcal{B}_2$ to DDH, which is given a DDH challenge (X^*, Y, Z^*) and uses these values to simulate Game_3 (when Z^* is x^*Y) or Game_4 (when it is random). Note that the games can be simulated without knowledge of x^* and y; in particular, the abort condition for (III) can be checked as

$$0 \stackrel{?}{=} \alpha^{(j)} G = (\bar{\upsilon}_r^{(j)} + c\bar{\upsilon}_x^{(j)})G + (\bar{\zeta}_r^{(j)} + c\bar{\zeta}_x^{(j)})X^*$$

and likewise for the condition of abort (I). We thus have

$\underline{\mathsf{SEGK.Setup}(\lambda)}$

$(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
Select $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_p$
Select $\mathsf{H}'\colon \{0,1\}^* \to \mathcal{K}$
return $par := (p, \mathbb{G}, G, \mathsf{H}, \mathsf{H}')$

$\underline{\mathsf{SEGK.KeyGen}(par)}$

$(p, \mathbb{G}, G, \mathsf{H}, \mathsf{H}') := par$
$y \xleftarrow{\$} \mathbb{Z}_p\,;\ Y := yG$
$sk := (par, y)\,;\ pk := (par, Y)$
return (sk, pk)

$\underline{\mathsf{SEGK.Enc}(pk)}$

$(p, \mathbb{G}, G, \mathsf{H}, \mathsf{H}', Y) := pk$
$x, r \xleftarrow{\$} \mathbb{Z}_p\,;\ X := xG\,;\ R := rG$
$k := \mathsf{H}'(xY)\,;\ s := r + \mathsf{H}(R, X) \cdot x \bmod p$
return $(k, (X, R, s))$

$\underline{\mathsf{SEGK.Dec}(sk, (X, R, s))}$

$(p, \mathbb{G}, G, \mathsf{H}, \mathsf{H}', y) := sk$
if $sG \neq R + \mathsf{H}(R, X) \cdot X$ **then**
  return $\bot$
return $k := \mathsf{H}'(yX)$

Figure 7.5: The Schnorr-signed ElGamal KEM scheme $\mathsf{SEGK}[\mathsf{GrGen}]$ for key space $\mathcal{K}$.

$$\mathbf{Adv}^{\mathsf{Game}_4}_{\mathcal{A}_{\mathrm{alg}}} \geq \mathbf{Adv}^{\mathsf{Game}_3}_{\mathcal{A}_{\mathrm{alg}}} - 2 \cdot \mathbf{Adv}^{ddh}_{\mathsf{GrGen},\mathcal{B}_2}\ . \tag{7.14}$$

Inspecting Game_4, we note that $\mathcal{A}$'s output is independent of b, because the random group element Z^* completely hides the message P_b. And whenever the game aborts, it outputs a random bit; we thus have:

$$\mathbf{Adv}^{\mathsf{Game}_4}_{\mathcal{A}_{\mathrm{alg}}} = 2 \cdot \Pr[1 \leftarrow \mathsf{Game}_4] - 1 = 0\ . \tag{7.15}$$

The theorem now follows from Equations (7.10), (7.11), (7.13), (7.14) and (7.15).

7.3 Schnorr-Signed Hashed ElGamal KEM

A public key for the ElGamal key-encapsulation mechanism (KEM) is a group element $Y \in \mathbb{G}$. To encrypt a message under Y, one samples a random $x \in \mathbb{Z}_p$ and derives an ephemeral key $K := xY$ to encrypt the message. Given the *encapsulation* $X := xG$, the receiver that holds $y = \log Y$ can derive the same key as $K := yX$. Under the decisional Diffie-Hellman assumption (DDH), this scheme is IND-CPA-secure. In the AGM, it was shown to satisfy CCA1 security (where the adversary can only make decryption queries before it has seen the challenge key) under a parameterized variant of DDH [FKL18].

By hashing the key, that is, defining $k := \mathsf{H}(xY)$, the assumption for proving CPA, resp. CCA2 security, can be relaxed to CDH, resp. strong Diffie-Hellman (SDH), in the random-oracle model.

Exactly as in section 7.1, the idea of *Schnorr-signed* hashed ElGamal is that, in addition to X, the encapsulation contains a proof of knowledge of the used randomness $x = \log X$, in the form of a Schnorr signature on message X under the public key X. The scheme is detailed in Figure 7.5.

The strongest security notion for KEM schemes is indistinguishability of ciphertexts under chosen-ciphertext attack (IND-CCA2), where the adversary can query decryptions of encapsulations of its choice even after receiving the challenge. The (decisional) game IND-CCA2 is defined in Figure 7.6.

We now prove that the Schnorr-signed hashed ElGamal KEM is tightly IND-CCA2-secure in the AGM+ROM under the discrete logarithm assumption.

Theorem 7.2. *Let* GrGen *be a group generator. Let* $\mathcal{A}_{\mathrm{alg}}$ *be an algebraic adversary against the IND-CCA2 security of the Schnorr-signed ElGamal KEM scheme* $\mathsf{SEGK}[\mathsf{GrGen}]$ *making at most* q_{d}

Game IND-CCA2$^{\mathcal{A}}_{\mathsf{KEM}}(\lambda)$	Oracle ENC() // one time
$par \leftarrow \mathsf{KEM.Setup}(\lambda)$	$(k_0, c^*) \leftarrow \mathsf{KEM.Enc}(pk)$; $k_1 \xleftarrow{\$} \mathcal{K}$
$(pk, sk) \leftarrow \mathsf{KEM.KeyGen}(par)$	**return** (k_b, c^*)
$b \xleftarrow{\$} \{0,1\}$	Oracle DEC(c)
$b' \leftarrow \mathcal{A}^{\text{ENC},\text{DEC}}(pk)$	**if** $c = c^*$ **then return** $\perp$
return $(b = b')$	**return** $\mathsf{KEM.Dec}(sk, c)$

Figure 7.6: The IND-CCA2 security game for a KEM scheme KEM.

decryption queries and q_h queries to both random oracles. Then there exists an algorithm $\mathcal{B}$ solving the DL problem w.r.t. GrGen*, such that*

$$\boldsymbol{Adv}^{\textit{ind-cca2}}_{\mathsf{SEGK[GrGen]},\mathcal{A}_{\mathrm{alg}}} \leq \boldsymbol{Adv}^{DL}_{\mathsf{GrGen},\mathcal{B}} + \frac{q_\mathrm{d} + \frac{1}{2^{\lambda-1}}(q_\mathrm{d} + q_\mathrm{h})}{2^{\lambda-1}} \ .$$

We start with the proof idea. Let Y be the public key and let $(X^* = x^*G, R^*, s^*)$ be the challenge ciphertext. If the adversary never queries $\mathsf{H}'(x^*Y)$ then it has no information about the challenge key k_b; but in order to query $K^* := x^*Y$, the adversary must solve the CDH problem for (Y, X^*). A CDH solution cannot be recognized by the reduction, so it would have to guess one of $\mathcal{A}$'s H' queries, which would make the proof non-tight.

In the AGM we can give a *tight* reduction to a *weaker* assumption, namely DL: Given a DL challenge Y, we set it as the public key, pick a random z and set $X^* := zY$. If the adversary makes the query $\mathsf{H}'(K^*)$ then we have $K^* = zy^2G$. On the other hand, the adversary must provide a representation $(\gamma, \upsilon, \xi, \rho)$ of K^* w.r.t. (G, Y, X^*, R^*), and thus

$$K^* = \gamma G + \upsilon Y + \xi X^* + \rho R^* = (\gamma + \upsilon y + \xi zy + \rho s^* - \rho c^* zy)G \ , \tag{7.16}$$

using the fact that $R^* = s^*G - c^*X^*$. Setting these two representations of $\log K^*$ equal yields the following quadratic equation in y:

$$zy^2 - (\upsilon + \xi z - \rho c^* z)y \equiv_p \gamma + \rho s^* \ .$$

If one of the solutions is the DL of Y, we are done; otherwise, the adversary's query was not of the form K^* and the challenge bit remains information-theoretically hidden.

The rest of the game is simulated without knowledge of $\log X^*$ and $\log Y$ as follows: The Schnorr signature under X^* contained in the challenge encapsulation can be simulated by programming the random oracle H as in the proof of Theorem 5.1. Decryption queries leverage the fact that the Schnorr signature contained in an encapsulation (X, R, s) proves knowledge of x with $X = xG$. By extracting x, the reduction can answer the query with $k = \mathsf{H}'(xY)$, but this extraction is trickier than in the proof of Theorem 5.1, since both X and R can also depend on Y, X^* and R^* (if the query is made *after* seeing the challenge ciphertext, which is the harder case).

In more detail, given the representations $(\gamma, \upsilon, \xi, \rho)$ and $(\gamma', \upsilon', \xi', \rho')$ of R and X provided by the adversary, we can write (analogously to Equation 7.16):

$$\begin{aligned} r = \log R &\equiv_p \gamma + \upsilon y + \xi zy + \rho s^* - \rho c^* zy \quad \equiv_p \alpha y + (\gamma + \rho s^*) \quad \text{and} \\ x = \log X &\equiv_p \gamma' + \upsilon' y + \xi' zy + \rho' s^* - \rho' c^* zy \equiv_p \alpha' y + (\gamma' + \rho' s^*) \end{aligned} \tag{7.17}$$

with $\alpha := \upsilon + (\xi - \rho c^*)z \bmod p$ and $\alpha' := \upsilon' + (\xi' - \rho' c^*)z \bmod p$. Since the signature (R, s) contained in the query must be valid, we have $s \equiv_p r + cx$ with $c = \mathsf{H}(R, X)$. Plugging the above

two equations into the latter yields

$$(\alpha + \alpha' c)y \equiv_p s - (\gamma + \rho s^*) - (\gamma' + \rho' s^*)c \ .$$

If $\alpha + \alpha' c \not\equiv_p 0$ then solving the above for y solves the challenge DL and the reduction can stop. Since $c = \mathsf{H}(R, X)$ was chosen by the experiment after the adversary provided representations of R and X, which define α and α', we have that $\alpha + \alpha' c \equiv_p 0$ happens with probability $\frac{1}{p}$, unless $\alpha' = 0$.

In the latter case however, from Equation 7.17 we have $x = \gamma' + \rho' s^* \bmod p$, meaning the reduction can compute x and can therefore answer the decryption query by returning $\mathsf{H}'(xY) = \mathsf{H}'(yX)$.

Proof

[Proof of Theorem 7.2] Consider the games Game_0 through Game_3 in Figure 7.7, where in Game_3 the adversary's advantage is 0. Game_0 has the same behavior as $\text{IND-CCA2}^{\mathcal{A}_{\text{alg}}}_{\mathsf{SEGK}[\mathsf{GrGen}]}$; the only syntactical change is that the value X^* used in the ENC oracle is already set before running $\mathcal{A}$ (which ensures that in later games it is defined in the abort conditions for lines (I), (III) and (IV) even when ENC has not been called yet). We prove the theorem by bounding the probability that the adversary behaves differently in two consecutive games Game_i and Game_{i+1}.

We start with the difference between Game_0 and Game_1, which consists in a possible abort in line (I) in oracle DEC. This happens when the experiment randomly chooses c as one particular value. (Note that Game_1 sets $c^* := 0$, so the value is defined in DEC when ENC has not been called yet.)

Observe that at any point $\mathsf{T}(R^*, X^*)$ is the only value in T that might not have been set during an adversary's call to H or Dec, and that could *not* have a corresponding entry in L. Moreover, if a call (X, R, s) to DEC is not answered by $\perp$, we must have $(X, R) \neq (X^*, R^*)$, since otherwise by the 3rd line $s = \log R^* + \mathsf{T}(R^*, S^*) \log X^* = s^*$ and the oracle would have returned $\perp$ in the 1st line.

Game_1 sets the values $(\gamma, \upsilon, \xi, \rho, \gamma', \upsilon', \xi', \rho')$ that were given as the representation of X and R when $\mathsf{T}(X, R) = c$ was randomly drawn. As we argued above, this must have been during a call from the adversary. The value c is thus independent of $(\gamma, \ldots, \rho')$, the values that define α and α'.

The two games Game_0 and Game_1 behave identically unless Game_1 aborts in line (I), that is, if $\alpha + c\alpha' \equiv_p 0$ and $\alpha' \neq 0$. By the above argument, the probability that c was chosen such as $c = -\alpha \cdot (\alpha')^{-1} \bmod p$ is upper-bounded by $\frac{1}{2^{\lambda-1}}$. Denoting by A the event that Game_1 aborts in line (I) during some DEC call, we have $\Pr[A] \leq \frac{q_{\text{d}}}{2^{\lambda-1}}$. Since $\Pr[1 \leftarrow \mathsf{Game}_0 \,|\, \neg A] = \Pr[1 \leftarrow \mathsf{Game}_1 \,|\, \neg A]$, we have

$$\begin{aligned} \mathbf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}_{\text{alg}}} - \mathbf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}_{\text{alg}}} &= 2\Pr[A](\Pr[1 \leftarrow \mathsf{Game}_0 \,|\, A] - \Pr[1 \leftarrow \mathsf{Game}_1 \,|\, A]) \\ &\leq \Pr[A] \ , \end{aligned}$$

as $\Pr[1 \leftarrow \mathsf{Game}_1 \,|\, A] = \frac{1}{2}$. We thus have:

$$\mathbf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}_{\text{alg}}} \geq \mathbf{Adv}^{\mathsf{Game}_0}_{\mathcal{A}_{\text{alg}}} - \frac{q_{\text{d}}}{2^{\lambda-1}} \ . \tag{7.18}$$

The two games Game_1 and Game_2 behave identically unless oracle ENC generates values (R^*, X^*) that have already been assigned a value in the table T. The values R^* and X^* are uniformly random in $\mathbb{G}$. Moreover, after the adversary has made q_{h} queries to H and q_{d} to DEC, at most $q_{\text{h}} + q_{\text{d}}$ values in T are assigned. Thus, the probability that (R^*, X^*) collides with one of the entries is bounded by $\frac{q_{\text{h}}+q_{\text{d}}}{(2^{\lambda-1})^2}$, and we thus have

$$\mathbf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}_{\text{alg}}} \geq \mathbf{Adv}^{\mathsf{Game}_1}_{\mathcal{A}_{\text{alg}}} - \frac{(q_{\text{d}} + q_{\text{h}})}{(2^{\lambda-1})^2} \ . \tag{7.19}$$

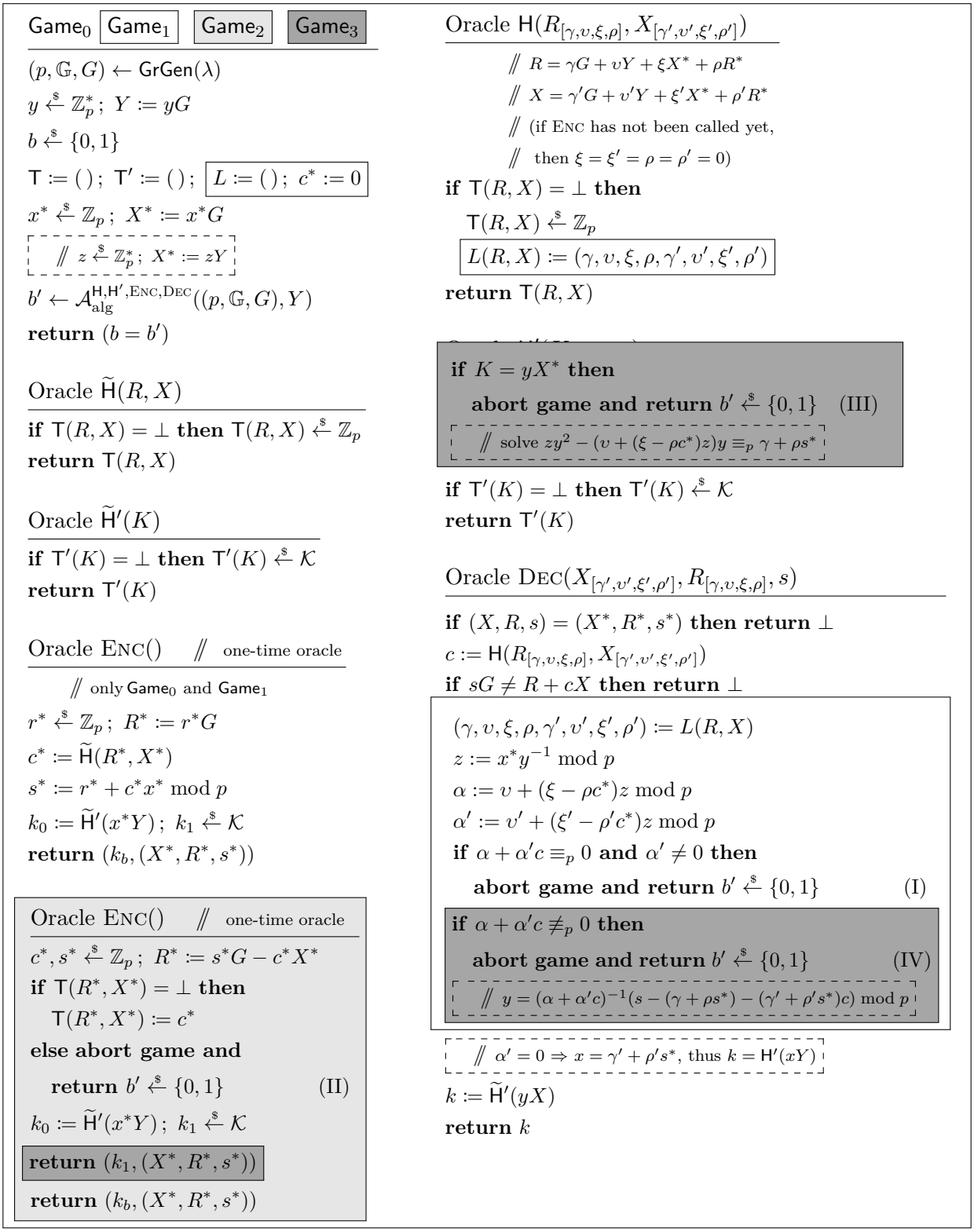

Figure 7.7: The IND-CCA2 security game $\text{IND-CCA2}^{\mathcal{A}_{\text{alg}}}_{\text{SEGK}[\mathsf{GrGen}]}$ (Game_0) for the Schnorr-Signed hashed ElGamal KEM scheme and games Game_1–Game_3 used in the proof of Theorem 7.2. The comments in dashed boxes show how the DL reduction simulates Game_3 without knowledge of x^* and y.

REDUCTION TO DL. We now construct an adversary $\mathcal{B}$ solving DL whenever Game_3 differs from Game_2, that is, when there is an abort in line (III) or (IV). Given a DL challenge Y, the reduction $\mathcal{B}$ sets Y as the public key, chooses a random $z \xleftarrow{\$} \mathbb{Z}_p^*$ and sets $X^* := zY$. It simulates ENC() by computing (R^*, s^*) as prescribed by the oracle, but setting $k_b := k_1$, a random key. (We argue below that when this introduces an inconsistency, the game aborts in line (III) anyway). $\mathcal{B}$ simulates the other oracles in Game_3 for $\mathcal{A}$ without knowledge of y and x^* as follows (cf. the comments in dashed boxes in Figure 7.7):

- Queries to H': whenever $\mathcal{A}$ queries $K_{[\gamma,\upsilon,\xi,\rho]}$ with

  $$K = \gamma G + \upsilon Y + \xi X^* + \rho R^* = (\gamma + \upsilon y + \xi z y + \rho s^* - \rho c^* z y)G \ ,$$

  $\mathcal{B}$ checks whether $K = yX^*$ (which equals zy^2G) by solving the following equation for y

  $$zy^2 - (\upsilon + \xi z - \rho c^* z)y \equiv_p \gamma + \rho s^*$$

  and checking whether some solution y satisfies $Y = yG$ (in this case Game_3 would abort in line (III)); if so, $\mathcal{B}$ stops and returns y.

  Note that otherwise, $K \neq yX^*$ and thus k_1 still perfectly simulates $\mathsf{H}'(yX^*) = k_b$.

- Queries to DEC: when queried (X, R, s), DEC returns $\perp$ if $sG \neq R + \mathsf{H}(R, X)X$ or $(X, R, s) = (X^*, R^*, s^*)$. Since s is determined by (R, X), the latter implies $(R, X) \neq (R^*, X^*)$ if DEC did not return $\perp$. By the first lines in the box in DEC, we have that $(\gamma, \upsilon, \xi, \rho)$ and $(\gamma', \upsilon', \xi', \rho')$ are such that

  $$\begin{aligned} R &= \gamma G + \upsilon Y + \xi X^* + \rho R^* \ = \ (\gamma + \upsilon y + \xi z y + \rho s^* - \rho c^* z y)G \\ X &= \gamma' G + \upsilon' Y + \xi' X^* + \rho' R^* = (\gamma' + \upsilon' y + \xi' z y + \rho' s^* - \rho' c^* z y)G \ . \end{aligned} \tag{7.20}$$

  As in DEC, we let $\alpha := \upsilon + (\xi - \rho c^*)z \bmod p$ and $\alpha' := \upsilon' + (\xi' - \rho' c^*)z \bmod p$ and thus from Equation 7.20 we have

  $$\begin{aligned} r &:= \log R = \gamma + \rho s^* + \alpha y \bmod p \\ x &:= \log X = \gamma' + \rho' s^* + \alpha' y \bmod p \ . \end{aligned} \tag{7.21}$$

  Since the oracle did not return $\perp$ in the 3rd line, we have $s \equiv_p r + cx$, and thus, by substituting r and x from Equation 7.21:

  $$(\alpha + \alpha' c)y \equiv_p s - (\gamma + \rho s^*) - (\gamma' + \rho' s^*)c \ .$$

  If $\alpha + \alpha' c \not\equiv_p 0$ then Game_3 would abort in line (IV); in this case $\mathcal{B}$ returns the DL solution $y = (\alpha + \alpha' c)^{-1}(s - (\gamma + \rho s^*) - (\gamma' + \rho' s^*)c) \bmod p$.

  If $\alpha + \alpha' c \equiv_p 0$ and $\alpha' \neq 0$ then both Game_2 and Game_3 (and thus $\mathcal{B}$) abort in line (I). Otherwise we must have $\alpha' = 0$ and, from Equation 7.21: $x = \gamma' + \rho' s^* \bmod p$. The reduction can thus simulate the decryption query by returning $\mathsf{H}'(xY)$ (which might define a new H' value or not).

This shows that whenever Game_3 differs from Game_2 (in lines (III) or (IV)), reduction $\mathcal{B}$ solves the DL problem, which yields:

$$\mathbf{Adv}^{\mathsf{Game}_3}_{\mathcal{A}_{\mathrm{alg}}} \geq \mathbf{Adv}^{\mathsf{Game}_2}_{\mathcal{A}_{\mathrm{alg}}} - \mathbf{Adv}^{dl}_{\mathsf{GrGen},\mathcal{B}} \ . \tag{7.22}$$

Inspecting Game_3, we note that $\mathcal{A}$'s output is independent of b and if the game aborts it outputs a random bit; we thus have:

$$\mathbf{Adv}^{\mathsf{Game}_3}_{\mathcal{A}_{\mathrm{alg}}} = 2 \cdot \Pr[1 \leftarrow \mathsf{Game}_3] - 1 = 0 \ . \tag{7.23}$$

The theorem now follows from Equations (7.18), (7.19), (7.22) and (7.23).

Bibliography

[Abe01] M. Abe. A secure three-move blind signature scheme for polynomially many signatures. In *EUROCRYPT 2001*, pages 136–151.

[ABM15] M. Abdalla, F. Benhamouda, and P. MacKenzie. Security of the J-PAKE password-authenticated key exchange protocol. In *2015 IEEE Symposium on Security and Privacy*, pages 571–587.

[ABR01] M. Abdalla, M. Bellare, and P. Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In *CT-RSA 2001*, pages 143–158.

[BB08] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, 2008.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *EUROCRYPT 2005*, LNCS 3494, pages 440–456. Springer, 2005.

[BCC04] E. F. Brickell, J. Camenisch, and L. Chen. Direct anonymous attestation. In *ACM CCS 2004*, pages 132–145.

[BCC+09] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Randomizable proofs and delegatable anonymous credentials. In *CRYPTO 2009*, pages 108–125.

[BCJ08] Ali Bagherzandi, Jung Hee Cheon, and Stanislaw Jarecki. Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *ACM CCS 2008*, pages 449–458. ACM Press, 2008.

[BDL+12] Daniel J. Bernstein, Niels Duif, Tanja Lange, Peter Schwabe, and Bo-Yin Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, 2012.

[BDN18] Dan Boneh, Manu Drijvers, and Gregory Neven. Compact multi-signatures for smaller blockchains. In Thomas Peyrin and Steven Galbraith, editors, *ASIACRYPT 2018, Part II*, LNCS 11273, pages 435–464. Springer, 2018.

[BFL20] Balthazar Bauer, Georg Fuchsbauer, and Julian Loss. A classification of computational assumptions in the algebraic group model. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, LNCS 12171, pages 121–151. Springer, 2020.

[BFPV13] O. Blazy, G. Fuchsbauer, D. Pointcheval, and D. Vergnaud. Short blind signatures. *Journal of Computer Security*, 21(5):627–661, 2013.

[BFW16] D. Bernhard, M. Fischlin, and B. Warinschi. On the hardness of proving CCA-security of signed ElGamal. In *PKC 2016, Part I*, pages 47–69.

[BL13a] F. Baldimtsi and A. Lysyanskaya. Anonymous credentials light. In *ACM CCS 2013*, pages 1087–1098.

[BL13b] F. Baldimtsi and A. Lysyanskaya. On the security of one-witness blind signature schemes. In *ASIACRYPT 2013, Part II*, pages 82–99.

[BLS04] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 2004.

[BLOR20] Fabrice Benhamouda, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. Cryptology ePrint Archive, Report 2020/945, 2020.

[BMV08] Emmanuel Bresson, Jean Monnerat and Damien Vergnaud. Separation Results on the"One-More" Computational Problems. Topics in cryptology - CT-RSA 2008, Lect. Notes Comput. Sci., T. Malkin ed.,Springer, vol. 4964, 2008, p. 71-87.

[BN06] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006*, pages 390–399. ACM Press, 2006.

[BNN04] Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security proofs for identity-based identification and signature schemes. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, LNCS 3027, pages 268–286. Springer, 2004.

[BNPS03] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-RSA-inversion problems and the security of Chaum's blind signature scheme. *Journal of Cryptology*, 16(3):185–215, 2003.

[BNW17] D. Bernhard, N. K. Nguyen, and B. Warinschi. Adaptive proofs have straightline extractors (in the random oracle model). In *ACNS 17*, pages 336–353.

[Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, LNCS 2567, pages 31–46. Springer, 2003.

[Boy08] Xavier Boyen. The uber-assumption family (invited talk). In Steven D. Galbraith and Kenneth G. Paterson, editors, *PAIRING 2008*, LNCS 5209, pages 39–56. Springer, 2008.

[BP02] Mihir Bellare and Adriana Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In Moti Yung, editor, *CRYPTO 2002*, LNCS 2442, pages 162–177. Springer, 2002.

[BS07] Mihir Bellare and Sarah Shoup. Two-tier signatures, strongly unforgeable signatures, and Fiat-Shamir without random oracles. In Tatsuaki Okamoto and Xiaoyun Wang, editors, *PKC 2007*, volume 4450, pages 201–216. Springer, 2007.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *ACM CCS 93*, pages 62–73.

[BR95] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *EUROCRYPT'94*, pages 92–111.

[Bra94] S. Brands. Untraceable off-line cash in wallets with observers (extended abstract). In *CRYPTO'93*, pages 302–318.

[Bro07] Daniel R. L. Brown. Irreducibility to the One-More Evaluation Problems Cryptology ePrint Archive, Report 2007/435, 3 November 2007 (last revised 2010).

[CDG18] Sandro Coretti, Yevgeniy Dodis, and Siyao Guo. Non-uniform bounds in the random-permutation, ideal-cipher, and generic-group models. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, LNCS 10991, pages 693–721. Springer, 2018.

[CFN90] D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *CRYPTO'88*, pages 319–327.

[Cha82] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO'82*, pages 199–203. Plenum Press, New York, USA, 1982.

[CHL05] J. Camenisch, S. Hohenberger, and A. Lysyanskaya. Compact e-cash. In *EUROCRYPT 2005*, pages 302–321.

[CL01] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT 2001*, pages 93–118.

[CS03] R. Cramer and V. Shoup. Design and analysis of practical public-key encryption schemes secure against adaptive chosen ciphertext attack. *SIAM Journal on Computing*, 33(1):167–226, 2003.

[CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, LNCS 740, pages 89–105. Springer, 1993.

[DEF+19] Manu Drijvers, Kasra Edalatnejad, Bian Ford, Eike Kiltz, Julian Loss, Gregory Neven, and Igor Stepanovs. On the security of two-round multi-signatures. *IEEE S&P*, page 1084–110, 2019.

[DL77] DeMillo Richard A. and Lipton Richard J. A Probabilistic Remark on Algebraic Program Testing. Technical report, Georgia Inst of Tech Antlanta School of information and computer science. 1977

[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[FF13] Marc Fischlin and Nils Fleischhacker. Limitations of the meta-reduction technique: The case of Schnorr signatures. In Thomas Johansson and Phong Q. Nguyen, editors, *EUROCRYPT 2013*, LNCS 7881, pages 444–460. Springer, 2013.

[FHS15] G. Fuchsbauer, C. Hanser, and D. Slamanig. Practical round-optimal blind signatures in the standard model. In *CRYPTO 2015, Part II*, pages 233–253.

[FH21] Masayuki Fukumitsu and Shingo Hasegawa. Impossibility on the Schnorr signature from the one-more DL assumption in the non-programmable random oracle model. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 2021.

[FJS14] Nils Fleischhacker, Tibor Jager, and Dominique Schröder. On tight security proofs for Schnorr signatures. In Palash Sarkar and Tetsu Iwata, editors, *ASIACRYPT 2014, Part I*, LNCS 8873, pages 512–531. Springer, 2014.

[FJS19] N. Fleischhacker, T. Jager, and D. Schröder. On tight security proofs for Schnorr signatures. *Journal of Cryptology*, 32(2):566–599, April 2019.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, LNCS 10992, pages 33–62. Springer, 2018.

[FOO93] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections.

In *AUSCRYPT'92*, pages 244–251.

[FPS20] Georg Fuchsbauer, Antoine Plouviez, and Yannick Seurin. Blind schnorr signatures and signed ElGamal encryption in the algebraic group model. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part II*, LNCS 12106, pages 63–95. Springer, 2020.

[FPV09] G. Fuchsbauer, D. Pointcheval, and D. Vergnaud. Transferable constant-size fair e-cash. In *CANS'09*, pages 226–247.

[FS10] Marc Fischlin and Dominique Schröder. On the impossibility of three-move blind signature schemes. In Henri Gilbert, editor, *EUROCRYPT 2010*, LNCS 6110, pages 197–215. Springer, 2010.

[Fuc11] G. Fuchsbauer. Commuting signatures and verifiable encryption. In *EUROCRYPT 2011*, pages 224–245.

[GBL08] Sanjam Garg, Raghav Bhaskar, and Satyanarayana V. Lokam. Improved bounds on security reductions for discrete log based signatures. In David Wagner, editor, *CRYPTO 2008*, LNCS 5157, pages 93–107. Springer, 2008.

[GG14] S. Garg and D. Gupta. Efficient round optimal blind signatures. In *EUROCRYPT 2014*, pages 477–495.

[GLSY04] Rosario Gennaro, Darren Leigh, R. Sundaram, and William S. Yerazunis. Batching Schnorr identification scheme with applications to privacy-preserving authorization and low-bandwidth communication devices. In Pil Joong Lee, editor, *ASIACRYPT 2004*, LNCS 3329, pages 276–292. Springer, 2004.

[GRS+11] S. Garg, V. Rao, A. Sahai, D. Schröder, and D. Unruh. Round optimal blind signatures. In *CRYPTO 2011*, pages 630–648.

[HHK10] J. Herranz, D. Hofheinz, and E. Kiltz. Some (in)sufficient conditions for secure hybrid encryption. *Inf. Comput.*, 208(11):1243–1257, 2010.

[HKL19] E. Hauck, E. Kiltz, and J. Loss. A modular treatment of blind signatures from identification schemes. In *EUROCRYPT 2019, Part III*, pages 345–375.

[IN83] K. Itakura and K. Nakamura. A public-key cryptosystem suitable for digital multisignatures. *NEC Research and Development*, 71:1–8, 1983.

[Jak98] M. Jakobsson. A practical mix. In *EUROCRYPT'98*, pages 448–461.

[KM07] Neal Koblitz and Alfred J. Menezes. Another look at "provable security". *Journal of Cryptology*, 20(1):3–37, 2007.

[KM08] Neal Koblitz and Alfred Menezes. Another look at non-standard discrete log and Diffie-Hellman problems. *J. Math. Cryptol.*, 2(4):311–326, 2008.

[KM10] Neal Koblitz and Alfred Menezes. The brave new world of bodacious assumptions in cryptography. *Notices of the American Mathematical Society*, 57(3):357–365, 2010.

[LRSW99] Anna Lysyanskaya, Ronald L. Rivest, Amit Sahai, and Stefan Wolf. Pseudonym systems. In Howard M. Heys and Carlisle M. Adams, editors, *SAC 1999*, LNCS 1758, pages 184–199. Springer, 1999.

[Mau99] Ueli M. Maurer. Information-theoretic cryptography (invited lecture). In Michael J. Wiener, editor, *CRYPTO'99*, LNCS 1666, pages 47–64. Springer, 1999.

[Mau05] Ueli M. Maurer. Abstract models of computation in cryptography (invited paper). In Nigel P. Smart, editor, *IMA Cryptography and Coding*, LNCS 3796, pages 1–12. Springer, 2005.

[MPSW19] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple Schnorr multi-signatures with applications to Bitcoin. *Des. Codes Cryptogr.*, 87(9):2139–2164, 2019.

[MR02] Silvio Micali and Ronald L. Rivest. Transitive signature schemes. In Bart Preneel, editor, *CT-RSA 2002*, LNCS 2271, pages 236–243. Springer, 2002.

[MS12] L. Minder and A. Sinclair. The extended k-tree algorithm. *Journal of Cryptology*, 25(2):349–382, April 2012.

[MW96] Ueli M. Maurer and Stefan Wolf. Diffie-Hellman oracles. In Neal Koblitz, editor, *CRYPTO'96*, LNCS 1109, pages 268–282. Springer, 1996.

[MWLD10] Changshe Ma, Jian Weng, Yingjiu Li, and Robert H. Den. Efficient discrete logarithm based multi-signature scheme in the plain public key mode. *Des. Codes Cryptography*, 54(2):121–133, 2010.

[Nec94] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55(2):165–172, 1994.

[Nic19] Jonas Nick. Blind signatures in scriptless scripts. Presentation given at *Building on Bitcoin*, 2019. https://jonasnick.github.io/blog/2018/07/31/blind-signatures-in-scriptless-scripts/.

[NIS19] NIST. Digital signature standard (DSS), FIPS PUB 186-5 (draft), 2019. https://csrc.nist.gov/publications/detail/fips/186/5/draft.

[NRS20] Jonas Nick, Tim Ruffing, and Yannick Seurin. MuSig2: Simple two-round Schnorr multi-signatures. Cryptology ePrint Archive, Report 2020/1261, 2020. https://eprint.iacr.org/2020/1261.

[NS15] I. Nikolic and Y. Sasaki. Refinements of the k-tree algorithm for the generalized birthday problem. In *ASIACRYPT 2015, Part II*, pages 683–703.

[OO92] T. Okamoto and K. Ohta. Universal electronic cash. In *CRYPTO'91*, pages 324–337.

[OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In Kwangjo Kim, editor, *PKC 2001*, LNCS 1992, pages 104–118. Springer, 2001.

[Pas11] Rafael Pass. Limits of provable security from standard assumptions. In Lance Fortnow and Salil P. Vadhan, editors, *43rd ACM STOC*, pages 109–118. ACM Press, 2011.

[PS96a] D. Pointcheval and J. Stern. Provably secure blind signature schemes. In *ASIACRYPT'96*, pages 252–265.

[PS96b] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *EUROCRYPT'96*, pages 387–398.

[PS00] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

[PV05] Pascal Paillier and Damien Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In Bimal K. Roy, editor, *ASIACRYPT 2005*, LNCS 3788, pages 1–20. Springer, 2005.

[Sch90] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO'89*, pages 239–252.

[Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, 1991.

[Sch01] Claus-Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In Sihan Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *ICICS 01*, LNCS 2229, pages 1–12. Springer, 2001.

[Seu12] Yannick Seurin. On the exact security of Schnorr-type signatures in the random oracle model. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT 2012*, LNCS 7237, pages 554–571. Springer, 2012.

[SG02] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 15(2):75–96, March 2002.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *EUROCRYPT'97*, LNCS 1233, pages 256–266. Springer, 1997.

[SJ99] Claus-Peter Schnorr and Markus Jakobsson. Security of discrete log cryptosystems in the random oracle and the generic model, 1999. Available at https://core.ac.uk/download/pdf/14504220.pdf.

[SJ00] C.-P. Schnorr and M. Jakobsson. Security of signed ElGamal encryption. In *ASIACRYPT 2000*, pages 73–89.

[STV+16] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. Keeping authorities "honest or bust" with decentralized witness cosigning. In *2016 IEEE S&P*, pages 526–545. IEEE Computer Society Press, 2016.

[ST13] Y. Seurin and J. Treger. A robust and plaintext-aware variant of signed ElGamal encryption. In *CT-RSA 2013*, pages 68–83.

[TY98] Y. Tsiounis and M. Yung. On the security of ElGamal based encryption. In *PKC'98*, pages 117–134.

[Wag02] David Wagner. A generalized birthday problem. In Moti Yung, editor, *CRYPTO 2002*, LNCS 2442, pages 288–303. Springer, 2002.

[Wik08] D. Wikström. Simplified submission of inputs to protocols. In *SCN 08*, pages 293–308.

[Wui18] Pieter Wuille. Schnorr signatures for secp256k1. Bitcoin Improvement Proposal, 2018. See https://github.com/sipa/bips/blob/bip-schnorr/bip-schnorr.mediawiki.

[Yun15] Aaram Yun. Generic hardness of the multiple discrete logarithm problem. In Elisabeth Oswald and Marc Fischlin, editors, *EUROCRYPT 2015, Part II*, LNCS 9057, pages 817–836. Springer, 2015.

Papers from this work:

- Blind Schnorr Signatures and Signed ElGamal Encryption in the Algebraic Group Model, Georg Fuchsbauer, Antoine Plouviez and Yannick Seurin. *Eurocrypt 2020*
- The One-More Discrete Logarithm Assumption in the Generic Group Model, Balthazar Bauer, Georg Fuchsbauer and Antoine Plouviez. *Asiacrypt 2021*

RÉSUMÉ

L'hypothèse de sécurité appelée "one more-discrete logarithm" (OMDL) est centrale dans l'analyse de sécurité des protocoles d'identifiactions, les signatures aveugles ainsi que les récentes multi-signatures MuSig2. Malgré sa grande utilisation, il est surprenant que OMDL n'a jamais été rigoureusement analysée. Dans cette thèse, nous donnons une preuve rigoureuse de OMDL dans le Modèle du Groupe Générique (GGM) ainsi que d'autres hypothèses en lien.

Les signatures de Schnorr aveugles sont un protocole qui permet d'envoyer à l'aveugle des signatures de Schnorr, qui sont parmi les signatures les plus utilisées. De même que OMDL, en dépit de leur utilité pratique, l'analyse de sécurité (basée sur OMDL) de ces signatures est insatisfaisant. Nous analysons la sécurité de ces protocoles dans le Modèle du Groupe Algébrique (AGM), qui est un modèle idéalisé plus proche du modèle standard que du GGM. Pour le renforcer, nous proposons une modification simple du protocole de signature, qui laisse les signatures inchangées.

MOTS CLÉS

Signatures de Schnorr aveugles, One More Discrete Logarithm, Modèle du Groupe Générique, Modèle du Groupe Algèbrique.

ABSTRACT

The one more-discrete logarithm assumption (OMDL) is central to the security analysis of identification protocols, blind signatures and multi-signature schemes, most notably blind Schnorr signatures and the recent MuSig2 multi-signatures. Despite its wide use, surprisingly, OMDL is lacking any rigorous analysis. In this work we give rigorous proofs in the Generic Group Model of OMDL and a related assumption.

The Schnorr blind signing protocol allows blind issuing of Schnorr signatures, one of the most widely used signatures. As for OMDL, despite its practical relevance, its security analysis (based on OMDL) is unsatisfactory. We analyze the security of these schemes in the algebraic group model (AGM), an idealized model closer to the standard model than the GGM.

KEYWORDS

Blind Schnorr signatures, One More Discrete Logarithm, Generic Group Model, Algebraic Group Model