

# Blind Schnorr Signatures in the Algebraic Group Model

Georg Fuchsbauer<sup>1,2</sup>, Antoine Plouviez<sup>1,2</sup>, and Yannick Seurin<sup>3</sup>

<sup>1</sup> Inria, Paris, France

<sup>2</sup> École normale supérieure, CNRS, PSL, Paris, France

`first.last@ens.fr`

<sup>3</sup> ANSSI, Paris, France

`first.last@m4x.org`

July 30, 2019

**Abstract.** We study the security of schemes related to Schnorr signatures in the algebraic group model (AGM) proposed by Fuchsbauer, Kiltz, and Loss (CRYPTO 2018), where the adversary can only compute new group elements by applying the group operation. Schnorr signatures can be proved secure in the random oracle model (ROM) under the discrete logarithm assumption (DL) by rewinding the adversary; but this security proof is loose. We start with giving a tight security proof under DL in the combination of the AGM and the ROM. Our main focus are blind Schnorr signatures, whose only known security proof is in the combination of the ROM and the generic group model, under the assumption that the so-called ROS problem is hard. We show that in the AGM+ROM the scheme is secure assuming hardness of the one-more discrete logarithm problem and the ROS problem. As the latter can be solved in sub-exponential time using Wagner’s algorithm, this is not entirely satisfying. Hence, we then propose a very simple modification of the scheme (which leaves key generation and signature verification unchanged) and show that, instead of ROS, its security relies on the hardness of a related problem which appears much harder than ROS. Finally, we give a tight reduction of the CCA2 security of Schnorr-signed ElGamal key encapsulation to DL, again in the AGM+ROM.

**Keywords:** Schnorr signatures, blind signatures, algebraic group model, ElGamal encryption

## 1 Introduction

IDEALIZED MODELS. Cryptosystems are often proved secure via a security reduction, which turns any adversary  $\mathcal{A}$  against the system into an algorithm solving a (presumably hard) computational problem. When this does not appear possible, one can resort to *idealized models*.

The *generic group model* (GGM) [Nec94, Sho97] is an idealized model for the security analysis of cryptosystems that are defined over cyclic groups. Instead of being given concrete group elements, the adversary only receives “handles” for them and has access to an oracle that performs the group operation (which we denote by addition) on handles. This implies that if the adversary is given a list of (handles of) group elements  $(X_1, \dots, X_n)$  and later returns a (handle of a) group element  $Z$ , by inspecting its oracle calls one can derive a “representation”  $\vec{z} = (z_1, \dots, z_n)$  such that  $Z = \sum_{i=1}^n z_i X_i$ .

The *algebraic group model* (AGM) [FKL18] lies between the standard model and the GGM. On the one hand, the adversary has direct access to group elements, but on the other hand, it is assumed to only produce new group elements by applying the group operation to received group elements; in particular, with every group element  $Z$  that the adversary outputs, it also

gives a representation  $\vec{z}$  of  $Z$  with respect to the group elements it has received so far. While the GGM allows for proving information-theoretic guarantees, security results in the AGM are proved via reductions to computationally hard problems, like in the standard model.

The *random-oracle model* (ROM) [BR93] replaces cryptographic hash functions by truly random functions. In security games the adversary is given oracle access to such a function, which is implemented by lazy sampling.

The results in this paper are given in the combination AGM+ROM, as already considered when the AGM was first defined [FKL18]. Adversaries are assumed to be algebraic w.r.t. the group and they are given access to a random oracle  $H$ . When  $H$  takes group elements as inputs, as the adversary is algebraic, any of its queries  $Z$  must be accompanied by a representation  $\vec{z}$  of  $Z$ . The security results in this paper will be by reduction to either the discrete logarithm problem or a variant of it, the *one-more discrete logarithm* (OMDL) problem.

**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<sup>+</sup>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 proven in the ROM under the discrete logarithm (DL) assumption by Pointcheval and Stern [PS96b, PS00]. 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].

Our starting point is the observation that in the AGM+ROM, we can give a reduction which is *straight-line*, that is, it runs the adversary only once, resulting in a *tight* security proof for Schnorr signatures under the DL assumption.<sup>4</sup> We then turn to two schemes related to Schnorr signatures whose security in the standard model remains elusive: 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 the message  $m$  remains unknown to 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 credentials [Bra94, CL01, BCC<sup>+</sup>09, 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,

---

<sup>4</sup> 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, with an extractor working straight-line. Our proof is much simpler and gives a concrete security statement.

[Abe01, Bol03, HKL19] to very theoretical schemes based on generic assumptions and provably secure in the standard model [GRS<sup>+</sup>11, BFPV13, GG14, FHS15].

The blind Schnorr signature scheme derives quite naturally from the standard Schnorr signature scheme [CP93]. While hardness of the discrete logarithm problem in the underlying group  $\mathbb{G}$  is obviously a necessary condition for the scheme to be secure against one-more forgeries, 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  $\text{ROS}_\ell$  problem, parameterized by an integer  $\ell$ , 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  $c_1, \dots, c_\ell$  over  $\mathbb{Z}_p$

$$\sum_{j=1}^{\ell} \rho_{i,j} c_j = H_{\text{ros}}(\vec{\rho}_i), \quad i \in [\ell + 1]$$

has a solution, where  $H_{\text{ros}}: (\mathbb{Z}_p)^\ell \rightarrow \mathbb{Z}_p$  is a random oracle. Schnorr showed that an attacker able to solve the  $\text{ROS}_\ell$  problem can produce  $\ell + 1$  valid signatures while interacting (concurrently) only  $\ell$  times with the signer. Slightly later, Wagner [Wag02] showed that the  $\text{ROS}_\ell$  problem can be reduced to the  $(\ell + 1)$ -sum problem, which can be solved with time and space complexity  $O((\ell + 1)2^{\lambda/(1+\lceil \lg(\ell+1) \rceil)})$ , where  $\lambda$  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.

From a provable-security point of view, a number of results [FS10, Pas11, BL13] 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 GGM+ROM assuming hardness of the ROS problem.

**OUR RESULTS ON BLIND SCHNORR SIGNATURES.** 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  $\ell$  interactions with the signer must either solve the one-more discrete logarithm problem or the  $\text{ROS}_\ell$  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, the proofs in [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, the exact specification of which is central for a rigorous security proof) and carefully work out the details of the security reductions to both the OMDL and the ROS problem.

Nevertheless, the threat constituted by Wagner’s attack for standard-size group orders remains. In order to remedy this situation, we propose a very 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. 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, at least for large values of  $\ell$  for which the standard ROS problem becomes tractable.

CHOSEN-CIPHERTEXT-SECURE ELGAMAL ENCRYPTION. Recall the ElGamal public-key encryption (PKE) scheme [EIG85]: given a cyclic group  $\mathbb{G}$  (denoted additively) 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  $x \leftarrow_s \mathbb{Z}_p$ , computes  $X = xG$ , and outputs ciphertext  $(X, M + xY)$ . This scheme is IND-CPA-secure 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 the adversary can only make decryption queries before receiving the challenge ciphertext) in the AGM under a “ $q$ -type” variant of DDH [FKL18].<sup>5</sup>

In order to make ElGamal encryption IND-CCA2-secure, a natural solution is to add a proof of knowledge of the randomness used to encrypt. Intuitively, this would make 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  $(X = xG, Z = M + xY)$  from the proof of knowledge and return the plaintext  $M = Z - xY$ .) 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.

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. 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 another proof under a non-standard “knowledge assumption” about Schnorr signatures, which amounts to assuming that they 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].

A second solution is to switch to the “hashed” variant of the scheme (also known as DHIES) [ABR01], in which a key is derived as  $k = H(xY)$ . In the ROM, the corresponding key-encapsulation mechanism (KEM) is IND-CCA2-secure under the gap 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.

RELEVANCE OF OUR RESULTS. We conclude with discussing the relevance of our results in particular for blind Schnorr signatures. While the initial proposal is arguably one of the

<sup>5</sup> [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 that the PKE scheme is also IND-CCA1-secure.

simplest and most efficient blind signature schemes, Wagner’s attack represents a serious weakness, which can be thwarted by using our modification of the signing protocol.

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 the major motivation for studying blind Schnorr signatures in its real-world relevance for protocols that use Schnorr signatures or will in the near future. The prime example is Bitcoin, for which developers are already actively exploring the use of blind Schnorr signatures for *blind* coin swaps, trustless tumbler services, and more [Nic19].

## 2 Preliminaries

GENERAL NOTATION. We denote the (closed) integer interval from  $a$  to  $b$  by  $[a, b]$ . We use  $[b]$  as shorthand for  $[1, b]$ . A function  $\mu: \mathbb{N} \rightarrow [0, 1]$  is *negligible* (denoted  $\mu = \text{negl}$ ) 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  $\nu$  is *overwhelming* if  $1 - \nu = \text{negl}$ . We let  $\lg$  denote the logarithm in base 2 and write  $x \equiv_p y$  for  $x \equiv y \pmod{p}$ .

Given a non-empty finite set  $S$ , we let  $x \leftarrow_s S$  denote the operation of sampling an element  $x$  from  $S$  uniformly at random. All algorithms are probabilistic unless stated otherwise. By  $y \leftarrow \mathcal{A}(x_1, \dots, x_n)$  we denote the operation of running algorithm  $\mathcal{A}$  on inputs  $(x_1, \dots, x_n)$  and uniformly random coins and letting  $y$  denote the output. If  $\mathcal{A}$  has oracle access to some algorithm ORACLE, we write  $y \leftarrow \mathcal{A}^{\text{ORACLE}}(x_1, \dots, x_n)$ . A list  $\vec{z} = (z_1, \dots, z_n)$ , also denoted  $(z_i)_{i \in [n]}$ , is a finite sequence. The length of a list  $\vec{z}$  is denoted  $|\vec{z}|$ . The empty list is denoted  $()$ .

A *security game*  $\text{GAME}_{\text{par}}(\lambda)$  indexed by a set of parameters  $\text{par}$  consists of a main procedure and a collection of oracle procedures. The main procedure, on input the security parameter  $\lambda$ , 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  $b$  (usually a single bit), which we write  $b \leftarrow \text{GAME}_{\text{par}}^{\mathcal{A}}(\lambda)$ . When the game outputs the truth value of a predicate, we identify **false** with 0 and **true** with 1. Games are either computational or decisional. The *advantage* of  $\mathcal{A}$  in  $\text{GAME}_{\text{par}}(\lambda)$  is defined as  $\text{Adv}_{\text{par}, \mathcal{A}}^{\text{game}}(\lambda) := \Pr[1 \leftarrow \text{GAME}_{\text{par}}^{\mathcal{A}}(\lambda)]$  if the game is computational and as  $\text{Adv}_{\text{par}, \mathcal{A}}^{\text{game}}(\lambda) := 2 \cdot \Pr[1 \leftarrow \text{GAME}_{\text{par}}^{\mathcal{A}}(\lambda)] - 1$  if it is decisional, where the probability is taken over the random coins of the game and the adversary. We say that  $\text{GAME}_{\text{par}}$  is *hard* if for any probabilistic polynomial-time (p.p.t.) adversary  $\mathcal{A}$ ,  $\text{Adv}_{\text{par}, \mathcal{A}}^{\text{game}}(\lambda) = \text{negl}(\lambda)$ . All games considered in this paper are computational unless stated otherwise (we only consider decisional games in Section 6 and Appendix B.)

ALGEBRAIC ALGORITHMS. A *group description* is a tuple  $\Gamma = (p, \mathbb{G}, G)$  where  $p$  is an odd prime,  $\mathbb{G}$  is an abelian group of order  $p$ , and  $G$  is a generator of  $\mathbb{G}$ . We will use additive notation for the group law throughout this paper, and denote group elements (including the generator  $G$ ) with italic uppercase letters. We assume the existence of a p.p.t. algorithm GrGen which, on input the security parameter  $1^\lambda$  in unary, outputs a group description  $\Gamma = (p, \mathbb{G}, G)$  where  $p$  is of bit-length  $\lambda$ . Given an element  $X \in \mathbb{G}$ , we let  $\log_G(X)$  denote the discrete logarithm of  $X$  in base  $G$ , i.e., the unique  $x \in \mathbb{Z}_p$  such that  $X = xG$ . We write  $\log X$  when  $G$  is clear from context.

Game $\text{DL}_{\text{GrGen}}^{\mathcal{A}}(\lambda)$	Game $\text{OMDL}_{\text{GrGen}}^{\mathcal{A}}(\lambda)$	Oracle $\text{CHAL}()$	Oracle $\text{DLOG}(X)$
$(p, \mathbb{G}, G) \leftarrow \text{GrGen}(1^\lambda)$	$(p, \mathbb{G}, G) \leftarrow \text{GrGen}(1^\lambda)$	$x \leftarrow \$_{\mathbb{Z}_p}; X := xG$	$q := q + 1$
$x \leftarrow \$_{\mathbb{Z}_p}; X := xG$	$\vec{x} := (); q := 0$	$\vec{x} := \vec{x} \parallel (x)$	$x := \log_G(X)$
$y \leftarrow \mathcal{A}(p, \mathbb{G}, G, X)$	$\vec{y} \leftarrow \mathcal{A}^{\text{CHAL, DLOG}}(p, \mathbb{G}, G)$	<b>return</b> $X$	<b>return</b> $x$
<b>return</b> $(y = x)$	<b>return</b> $(\vec{y} = \vec{x} \wedge q <  \vec{x} )$		

**Fig. 1.** The DL and OMDL problems.

An *algebraic security game* (w.r.t.  $\text{GrGen}$ ) is a game  $\text{GAME}_{\text{GrGen}}^{\mathcal{A}}(\lambda)$  that (among other things) runs  $\Gamma \leftarrow \text{GrGen}(1^\lambda)$  and gives  $\Gamma = (p, \mathbb{G}, G)$  as input to  $\mathcal{A}$  (see for example games DL and OMDL in Figure 1). An algorithm  $\mathcal{A}_{\text{alg}}$  executed in an algebraic game  $\text{GAME}_{\text{GrGen}}^{\mathcal{A}_{\text{alg}}}(\lambda)$  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  $\vec{X} = (X_0, \dots, X_n) \in \mathbb{G}^{n+1}$  is the list of all group elements that  $\mathcal{A}_{\text{alg}}$  has received so far (where by convention we let  $X_0 = G$ ), then  $\mathcal{A}_{\text{alg}}$  must output  $Z$  together with  $\vec{z} = (z_0, \dots, z_n) \in (\mathbb{Z}_p)^{n+1}$  such that  $Z = \sum_{i=0}^n z_i X_i$ . We let  $Z_{[\vec{z}]}$  denote such an augmented output. When writing the vector  $\vec{z}$  explicitly, we simply write  $Z_{[z_0, \dots, z_n]}$  (rather than  $Z_{[(z_0, \dots, z_n)]}$ ) to lighten the notation.

ALGEBRAIC ALGORITHMS IN THE RANDOM ORACLE MODEL. The original paper [FKL18] considered the algebraic group model augmented by a random oracle and proved tight security of BLS signatures [BLS04] in this AGM+ROM model. The random oracle in that work is of type  $\text{H}: \{0, 1\}^* \rightarrow \mathbb{G}$ , 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  $\text{H}: \mathbb{G} \times \{0, 1\}^* \rightarrow \mathbb{Z}_p$ . Thus, an algebraic adversary querying  $\text{H}$  on some input  $(Z, m)$  must also provide a representation  $\vec{z}$  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  $\tilde{\text{H}}$ , which is used by the game itself (and thus does not take representations of group elements as input) and implements the same function as  $\text{H}$ .

THE (ONE-MORE) DISCRETE LOGARITHM PROBLEM. We recall the discrete logarithm (DL) problem in Figure 1. The one-more discrete logarithm (OMDL) problem, also defined in Figure 1, is an extension of the DL problem and consists in finding the discrete logarithm of  $q$  group elements by making strictly less than  $q$  calls to an oracle solving the discrete logarithm problem. It was introduced in [BNPS03] and used for example to prove the security of the Schnorr identification protocol against active and concurrent attacks [BP02].

### 3 Schnorr Signatures

#### 3.1 Definitions

A signature scheme  $\text{SIG}$  consists of the following algorithms:

- $par \leftarrow \text{SIG.Setup}(1^\lambda)$ : the setup algorithm takes as input the security parameter  $\lambda$  in unary and outputs public parameters  $par$ ;

<p style="text-align: center; border-bottom: 1px solid black; margin-bottom: 5px;">Game EUF-CMA<sub>SIG</sub><sup>A</sup>(<math>\lambda</math>)</p> <p><math>par \leftarrow \text{SIG.Setup}(1^\lambda)</math>  <math>(sk, pk) \leftarrow \text{SIG.KeyGen}(par)</math>; <math>Q := ()</math>  <math>(m^*, \sigma^*) \leftarrow \mathcal{A}^{\text{SIGN}}(pk)</math>  <b>return</b> <math>(m^* \notin Q \wedge \text{SIG.Ver}(pk, m^*, \sigma^*))</math></p>	<p style="text-align: center; border-bottom: 1px solid black; margin-bottom: 5px;">Oracle SIGN(<math>m</math>)</p> <p><math>\sigma \leftarrow \text{SIG.Sign}(sk, m)</math>  <math>Q := Q \parallel (m)</math>  <b>return</b> <math>\sigma</math></p>
--	---

**Fig. 2.** The EUF-CMA security game for a signature scheme SIG.

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

Correctness requires that for any  $\lambda$  and any message  $m$ , when running  $par \leftarrow \text{SIG.Setup}(1^\lambda)$ ,  $(sk, pk) \leftarrow \text{SIG.KeyGen}(par)$ ,  $\sigma \leftarrow \text{SIG.Sign}(sk, m)$ , and  $b \leftarrow \text{SIG.Ver}(pk, m, \sigma)$ , 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 2. The Schnorr signature scheme [Sch91] is specified in Figure 3.

### 3.2 Security of Schnorr Signatures in the AGM

**Theorem 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 Sch[GrGen] running in time at most  $\tau$  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  $\tau + O(q_s + q_h)$ , such that*

$$\text{Adv}_{\text{Sch[GrGen]}, \mathcal{A}_{\text{alg}}}^{\text{euf-cma}}(\lambda) \leq \text{Adv}_{\text{GrGen}, \mathcal{B}}^{\text{dl}}(\lambda) + \frac{q_s(q_s + q_h) + 1}{2^{\lambda-1}}.$$

<p style="text-align: center; border-bottom: 1px solid black; margin-bottom: 5px;">Sch.Setup(<math>1^\lambda</math>)</p> <p><math>(p, \mathbb{G}, G) \leftarrow \text{GrGen}(1^\lambda)</math>  Select <math>H: \{0, 1\}^* \rightarrow \mathbb{Z}_p</math>  <b>return</b> <math>par := (p, \mathbb{G}, G, H)</math></p>	<p style="text-align: center; border-bottom: 1px solid black; margin-bottom: 5px;">Sch.KeyGen(<math>par</math>)</p> <p><math>(p, \mathbb{G}, G, H) := par</math>; <math>x \leftarrow \mathbb{Z}_p</math>; <math>X := xG</math>  <math>sk := (par, x)</math>; <math>pk := (par, X)</math>  <b>return</b> <math>(sk, pk)</math></p>
<p style="text-align: center; border-bottom: 1px solid black; margin-bottom: 5px;">Sch.Sign(<math>sk, m</math>)</p> <p><math>(p, \mathbb{G}, G, H, x) := sk</math>  <math>r \leftarrow \mathbb{Z}_p</math>; <math>R := rG</math>  <math>c := H(R, m)</math>; <math>s := r + cx \text{ mod } p</math>  <b>return</b> <math>\sigma := (R, s)</math></p>	<p style="text-align: center; border-bottom: 1px solid black; margin-bottom: 5px;">Sch.Ver(<math>pk, m, \sigma</math>)</p> <p><math>(p, \mathbb{G}, G, H, X) := pk</math>  <math>(R, s) := \sigma</math>  <math>c := H(R, m)</math>  <b>return</b> <math>(sG = R + cX)</math></p>

**Fig. 3.** The Schnorr signature scheme Sch[GrGen] based on a group generator GrGen.

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  $H(R, m) = c$ . 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: a valid forgery satisfies  $R^* = s^*G - c^*X$ , with  $c^* := H(R^*, m^*)$ ; on the other hand, since the adversary is algebraic, when it made its first query  $H(R^*, m^*)$ , it provided a representation of  $R^*$  in basis  $(G, X)$ , that is  $(\gamma^*, \xi^*)$  with  $R^* = \gamma^*G + \xi^*X$ . Together, these equations yield

$$(\xi^* + c^*)X = (s^* - \gamma^*)G .$$

Since  $c^*$  was chosen at random *after* the adversary chose  $\xi^*$ , the probability that  $\xi^* + c^* \not\equiv_p 0$  is overwhelming, in which case we can compute the discrete logarithm of  $X$  from the above equation.

*Proof.* Let  $\mathcal{A}_{\text{alg}}$  be an algebraic adversary playing  $\text{EUF-CMA}_{\text{Sch}[\text{GrGen}]}$  and making at most  $q_s$  signature queries and  $q_h$  RO queries. We proceed by a sequence of games specified in Figure 4.

**Game<sub>0</sub>.** The first game is the EUF-CMA game (Figure 2) for the Schnorr signature scheme (Figure 3) in the random oracle model for  $H$ . The game maintains a list  $Q$  of queried messages and  $T$  of values sampled for  $H$ . To prepare the change to **Game<sub>1</sub>**, we have written the finalization of the game in an equivalent way: it first checks that  $m^* \notin Q$  and then runs  $\text{Sch.Ver}(pk, m^*, (R^*, s^*))$ , which we have written explicitly. Note that since the adversary is algebraic, it must provide a representation  $(\gamma^*, \xi^*)$  for its forgery  $(m^*, (R_{[\gamma^*, \xi^*]}^*, s^*))$  such that  $R^* = \gamma^*G + \xi^*X$ , and similarly for each RO query  $H(R_{[\gamma, \xi]}, m)$ . By definition,

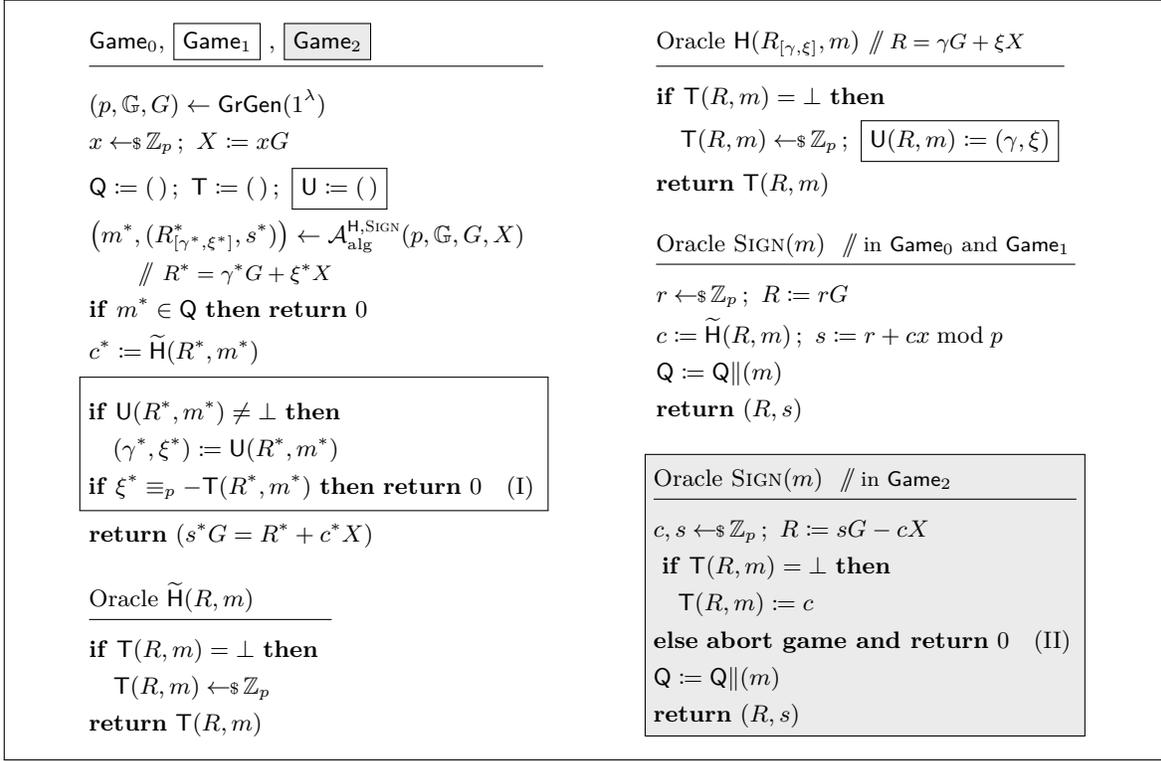
$$\text{Adv}_{\mathcal{A}_{\text{alg}}}^{\text{game}_0}(\lambda) = \text{Adv}_{\text{Sch}[\text{GrGen}], \mathcal{A}_{\text{alg}}}^{\text{euf-cma}}(\lambda) .$$

**Game<sub>1</sub>.** In **Game<sub>1</sub>**, we make the following changes. First, we introduce an auxiliary table  $U$  that for each query  $H(R_{[\gamma, \xi]}, m)$  stores the representation  $(\gamma, \xi)$  of  $R$ . Clearly, this does not modify the probability that the game returns 1. Second, when the adversary returns its forgery  $(m^*, (R_{[\gamma^*, \xi^*]}^*, s^*))$  and had previously made a query  $H(R_{[\gamma', \xi']}^*, m^*)$  for some  $(\gamma', \xi')$ , then we consider this previous representation of  $R^*$ , that is, we set  $(\gamma^*, \xi^*) := (\gamma', \xi')$ . The only actual difference in **Game<sub>1</sub>** is that it returns 0 in case  $\xi^* \equiv_p -T(R^*, m^*)$  (line (I)); otherwise **Game<sub>0</sub>** and **Game<sub>1</sub>** are identical.

We show that this happens with probability exactly  $1/p \leq 1/2^{\lambda-1}$ . First note that line (I) is only executed if  $m^* \notin Q$ , as otherwise the game would already have returned 0; hence  $T(R^*, m^*)$  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  $c^*$ . We show that in both cases the probability of returning 0 in line (I) is  $1/p$ :

- (1) If  $T(R^*, m^*)$  was defined during a  $H$  query of the form  $H(R_{[\gamma', \xi']}^*, m^*)$  then  $T(R^*, m^*)$  is drawn uniformly at random and independently from  $\xi'$ . Since then  $U(R^*, m^*) \neq \perp$ , the game sets  $\xi^* := \xi'$  and hence  $\xi^* \equiv_p -T(R^*, m^*)$  holds with probability exactly  $1/p$ .
- (2) If  $T(R^*, m^*)$  is only defined after the adversary output  $\xi^*$  then again we have  $\xi^* \equiv_p -T(R^*, m^*)$  with probability  $1/p$ . Hence,

$$\text{Adv}_{\mathcal{A}_{\text{alg}}}^{\text{game}_1}(\lambda) \geq \text{Adv}_{\mathcal{A}_{\text{alg}}}^{\text{game}_0}(\lambda) - \frac{1}{2^{\lambda-1}} .$$



**Fig. 4.** Games used in the proof of [Theorem 1](#). **Game<sub>0</sub>**, defined by ignoring all boxes, is the EUF-CMA security game for the Schnorr signature scheme  $\text{Sch}[\text{GrGen}]$  with algebraic adversary  $\mathcal{A}_{\text{alg}}$ . As we work in the ROM,  $\mathcal{A}_{\text{alg}}$  has oracle access to  $\text{H}$  while the game uses a local function  $\tilde{\text{H}}$ , both implemented via lazy sampling with the same table  $\text{T}$  (see [Section 2](#)). Boxed statements are only included in **Game<sub>1</sub>** and **Game<sub>2</sub>**; Gray boxes are only included in **Game<sub>2</sub>**.

**Game<sub>2</sub>**. In the final game we use the standard strategy for Schnorr signatures of simulating the  $\text{SIGN}$  oracle without the secret key  $x$  by programming the random oracle. Clearly, **Game<sub>1</sub>** and **Game<sub>2</sub>** are identical unless **Game<sub>2</sub>** returns 0 in line (II). For each signature query,  $R$  is uniformly random, and the size of table  $\text{T}$  is at most  $q_s + q_h$ , hence the game aborts in line (II) with probability at most  $(q_s + q_h)/p \leq (q_s + q_h)/2^{\lambda-1}$ . By summing over the at most  $q_s$  signature queries, we have

$$\text{Adv}_{\mathcal{A}_{\text{alg}}}^{\text{game}_2}(\lambda) \geq \text{Adv}_{\mathcal{A}_{\text{alg}}}^{\text{game}_1}(\lambda) - \frac{q_s(q_s + q_h)}{2^{\lambda-1}}.$$

**REDUCTION TO DL.** We now construct an adversary  $\mathcal{B}$  solving DL with the same probability as  $\mathcal{A}_{\text{alg}}$  wins **Game<sub>2</sub>**. 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<sub>2</sub>**, which can be done without knowledge of  $\log_G(X)$ . Assume that the adversary wins **Game<sub>2</sub>** by returning  $(m^*, R^*, s^*)$  and let  $c^* := \text{T}(R^*, m^*)$  and  $(\gamma^*, \xi^*)$  be defined as in the game. Thus,  $\xi^* \neq -c^* \text{ mod } p$  and  $R^* = \gamma^*G + \xi^*X$ ; moreover, validity of the forgery implies that  $s^*G = R^* + c^*X$ . Hence,  $(s^* - \gamma^*)G = (\xi^* + c^*)X$  and  $\mathcal{B}$  can compute  $\log X = (s^* - \gamma^*)(\xi^* + c^*)^{-1} \text{ mod } p$ . Combining previous inequalities, we have

$$\text{Adv}_{\text{GrGen}, \mathcal{B}}^{\text{dl}}(\lambda) = \text{Adv}_{\mathcal{A}_{\text{alg}}}^{\text{game}_2}(\lambda) \geq \text{Adv}_{\text{Sch}[\text{GrGen}], \mathcal{A}_{\text{alg}}}^{\text{euf-cma}}(\lambda) - \frac{q_s(q_s + q_h) + 1}{2^{\lambda-1}}.$$

Assuming that scalar multiplications in  $\mathbb{G}$  and assignments in tables  $T$  and  $U$  take unit time, the running time of  $\mathcal{B}$  is  $\tau + O(q_s + q_h)$ .  $\square$

## 4 Blind Schnorr Signatures

### 4.1 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.

**SYNTAX.** A blind signature scheme  $\text{BS}$  consists of the following algorithms:

- $par \leftarrow \text{BS.Setup}(1^\lambda)$ : the setup algorithm takes the security parameter  $\lambda$  in unary and returns public parameters  $par$ ;
- $(sk, pk) \leftarrow \text{BS.KeyGen}(par)$ : the key generation algorithm takes the public parameters  $par$  and returns a secret/public key pair  $(sk, pk)$ ;
- $(b, \sigma) \leftarrow \langle \text{BS.Sign}(sk), \text{BS.User}(pk, m) \rangle$ : 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  $\sigma$  if it terminates correctly, and  $\perp$  otherwise. For a 2-round protocol the interaction can be realized by the following algorithms:

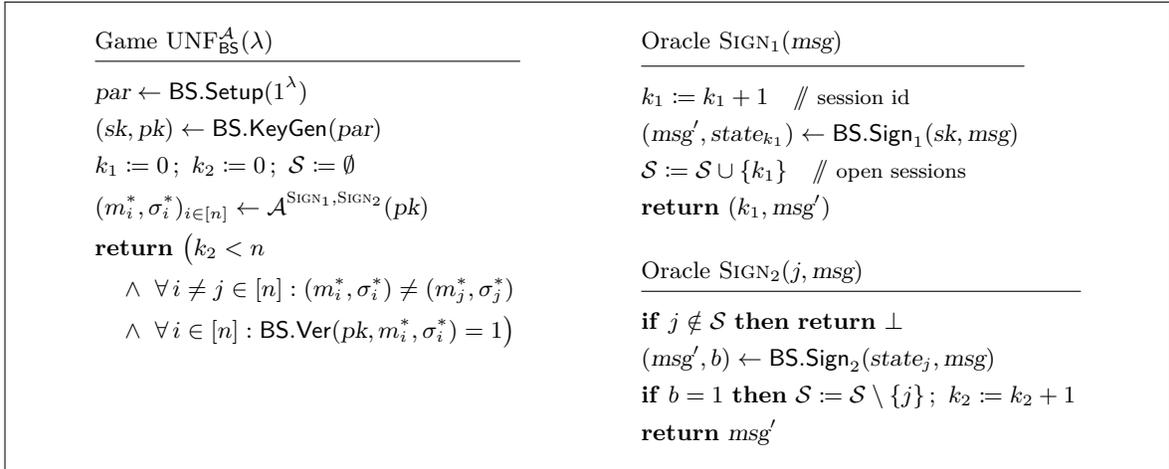
$$\begin{aligned}
 (msg_{U,0}, state_{U,0}) &\leftarrow \text{BS.User}_0(pk, m) \\
 (msg_{S,1}, state_S) &\leftarrow \text{BS.Sign}_1(sk, msg_{U,0}) \\
 (msg_{U,1}, state_{U,1}) &\leftarrow \text{BS.User}_1(state_{U,0}, msg_{S,1}) \\
 (msg_{S,2}, b) &\leftarrow \text{BS.Sign}_2(state_S, msg_{U,1}) \\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

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

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

In game  $\text{UNF}_{\text{BS}}^A(\lambda)$  defined in [Figure 5](#) the adversary has access to two oracles  $\text{SIGN}_1$  and  $\text{SIGN}_2$  corresponding to the two phases of the interactive protocol. The game maintains two counters  $k_1$  and  $k_2$  (initially set to 0), where  $k_1$  is used as session identifier, and a set  $\mathcal{S}$  of “open” sessions. Oracle  $\text{SIGN}_1$  takes the user’s first message (which for blind Schnorr signatures is empty), increments  $k_1$ , adds  $k_1$  to  $\mathcal{S}$  and runs the first round on the signer’s side, storing its state as  $state_{k_1}$ . Oracle  $\text{SIGN}_2$  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  $k_2$ , which thus represents the number of successful interactions.

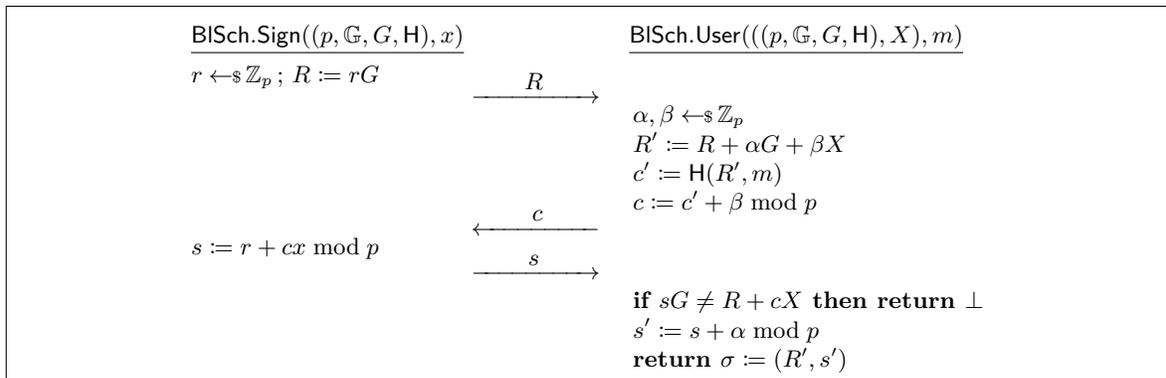


**Fig. 5.** The (strong) unforgeability game for a blind signature scheme BS with a 2-round signing protocol.

We say that BS satisfies unforgeability if  $\text{Adv}_{\text{BS}, \mathcal{A}}^{\text{unf}}(\lambda)$  is negligible for all p.p.t. adversaries  $\mathcal{A}$ . Note that we consider “strong” unforgeability, which only requires that all pairs  $(m_i^*, \sigma_i^*)$  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  $\sigma_b$  on  $m_b$  and then  $\sigma_{1-b}$  on  $m_{1-b}$  for a random bit  $b$ . If both signatures are valid, the adversary is given  $(\sigma_0, \sigma_1)$  and must determine the value of  $b$ . A formal definition can be found in [Appendix B](#).

**BLIND SCHNORR SIGNATURES.** A blind signature scheme **BISch** is obtained from the scheme **Sch** in [Figure 3](#) by replacing **Sch.Sign** with the interactive protocol specified in [Figure 6](#) (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



**Fig. 6.** The signing protocol of the blind Schnorr signature scheme.

Game $\text{ROS}_{\text{GrGen}, \ell, \Omega}^{\mathcal{A}}(\lambda)$	Oracle $\text{H}_{\text{ros}}(\vec{\rho}, \text{aux})$
$(p, \mathbb{G}, G) \leftarrow \text{GrGen}(1^\lambda)$	<b>if</b> $\text{T}_{\text{ros}}(\vec{\rho}, \text{aux}) = \perp$ <b>then</b>
$\text{T}_{\text{ros}} := ()$	$\text{T}_{\text{ros}}(\vec{\rho}, \text{aux}) \leftarrow \$_\mathbb{Z}_p$
$((\vec{\rho}_i, \text{aux}_i)_{i \in [\ell+1]}, (c_j)_{j \in [\ell]}) \leftarrow \mathcal{A}^{\text{H}_{\text{ros}}}(p)$	<b>return</b> $\text{T}_{\text{ros}}(\vec{\rho}, \text{aux})$
$\parallel \vec{\rho}_i = (\rho_{i,1}, \dots, \rho_{i,\ell})$	
<b>return</b> $(\forall i \neq i' \in [\ell+1] : (\vec{\rho}_i, \text{aux}_i) \neq (\vec{\rho}_{i'}, \text{aux}_{i'}))$	
$\wedge \forall i \in [\ell+1] : \sum_{j=1}^{\ell} \rho_{i,j} c_j \equiv_p \text{H}_{\text{ros}}(\vec{\rho}_i, \text{aux}_i)$	

**Fig. 7.** The ROS game, where  $\text{H}_{\text{ros}} : (\mathbb{Z}_p)^\ell \times \Omega \rightarrow \mathbb{Z}_p$  is a random oracle.

Sch.Ver:

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

Moreover, Schnorr signatures have been shown to achieve perfect blindness [CP93].

## 4.2 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 Schnorr [Sch01]. Consider the game  $\text{ROS}_{\text{GrGen}, \ell, \Omega}$  in Figure 7, parameterized by a group generator  $\text{GrGen}$ ,<sup>6</sup> an integer  $\ell \geq 1$ , and a set  $\Omega$  (we omit  $\text{GrGen}$  and  $\Omega$  from the notation when they are clear from context). The adversary  $\mathcal{A}$  receives a prime  $p$  and has access to a random oracle  $\text{H}_{\text{ros}}$  taking as input  $(\vec{\rho}, \text{aux})$  where  $\vec{\rho} \in (\mathbb{Z}_p)^\ell$  and  $\text{aux} \in \Omega$ . Its goal is to find  $\ell + 1$  distinct pairs  $(\vec{\rho}_i, \text{aux}_i)_{i \in [\ell+1]}$  together with a solution  $(c_j)_{j \in [\ell]}$  to the linear system  $\sum_{j=1}^{\ell} \rho_{i,j} c_j \equiv_p \text{H}_{\text{ros}}(\vec{\rho}_i, \text{aux}_i)$ ,  $i \in [\ell + 1]$ .<sup>7</sup>

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

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

$$\text{Adv}_{\text{BISch}[\text{GrGen}], \mathcal{B}}^{\text{unf}}(\lambda) \geq \text{Adv}_{\text{GrGen}, \ell, \Omega, \mathcal{A}}^{\text{ros}}(\lambda) - \frac{q_h^2 + (\ell + 1)^2}{2^{\lambda-1}} .$$

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

<sup>6</sup> The group generator  $\text{GrGen}$  is only used to generate a prime  $p$  of length  $\lambda$ ; the group  $\mathbb{G}$  is not used in the game.

<sup>7</sup> 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 2.

- $E_1$ : when  $\mathcal{A}$  queries  $\mathbf{H}_{\text{ros}}(\vec{\rho}, (\gamma, \xi, m))$ , there has been a previous query  $\mathbf{H}_{\text{ros}}(\vec{\rho}', (\gamma', \xi', m'))$  such that  $m = m'$  and

$$\gamma + \xi x + \sum_{j=1}^{\ell} \rho_j r_j \equiv_p \gamma' + \xi' x + \sum_{j=1}^{\ell} \rho'_j r_j ;$$

- $E_2$ : when  $\mathcal{A}$  returns  $((\vec{\rho}_i, (\gamma_i, \xi_i, m_i))_{i \in [\ell+1]}, (c_j)_{j \in [\ell]})$ , there exists  $i \neq i' \in [\ell+1]$  such that  $m_i = m_{i'}$  and

$$\gamma_i + \xi_i x + \sum_{j=1}^{\ell} \rho_{i,j} r_j \equiv_p \gamma_{i'} + \xi_{i'} x + \sum_{j=1}^{\ell} \rho_{i',j} r_j .$$

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

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

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

$$\text{Adv}_{\text{GrGen}, \ell, \Omega, \mathcal{A}}^{\text{ros}'}(\lambda) \geq \text{Adv}_{\text{GrGen}, \ell, \Omega, \mathcal{A}}^{\text{ros}}(\lambda) - \frac{q_{\text{h}}^2 + (\ell+1)^2}{2^{\lambda-1}} . \quad (1)$$

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

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

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  $i \in [\ell+1]$ ,

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

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

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

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

$$\text{Adv}_{\text{BISch}[\text{GrGen}], \mathcal{B}}^{\text{unf}}(\lambda) = \text{Adv}_{\text{GrGen}, \ell, \Omega, \mathcal{A}}^{\text{ros}' }(\lambda) . \quad (2)$$

Clearly,  $\mathcal{B}$  runs in time at most  $\tau + O(\ell + q_h)$  and makes at most  $\ell$  queries to  $\text{SIGN}_1$  and  $\text{SIGN}_2$  and  $q_h$  random oracle queries. Combining Eqs. (1) and (2) concludes the proof.  $\square$

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

**Lemma 2.** *Let  $\text{GrGen}$  be a group generator,  $\ell \geq 1$ , and  $\Omega$  be some arbitrary set. Then for any adversary  $\mathcal{A}$  making at most  $q_h$  queries to  $\text{H}_{\text{ros}}$ ,*

$$\text{Adv}_{\text{GrGen}, \ell, \Omega, \mathcal{A}}^{\text{ros}}(\lambda) \leq \frac{\binom{q_h}{\ell+1} + 1}{2^{\lambda-1}} .$$

*Proof.* Consider a modified game  $\text{ROS}_{\text{GrGen}, \ell, \Omega}^*$  that is identical to ROS, except that it returns 0 when the adversary outputs  $((\vec{\rho}_i, \text{aux}_i)_{i \in [\ell+1]}, (c_j)_{j \in [\ell]})$  such that for some  $i \in [\ell+1]$  it has not made the query  $\text{H}_{\text{ros}}(\vec{\rho}_i, \text{aux}_i)$ . Games ROS and  $\text{ROS}^*$  are identical unless in game ROS the adversary wins and has not made the query  $\text{H}_{\text{ros}}(\vec{\rho}_i, \text{aux}_i)$  for some  $i$ , which happens with probability at most  $1/p \leq 1/2^{\lambda-1}$ . Hence,

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

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

$$\sum_{j=1}^{\ell} \rho_{i,j} c_j \equiv_p \text{H}_{\text{ros}}(\vec{\rho}_i, \text{aux}_i), \quad i \in [\ell+1] \quad (3)$$

with unknowns  $c_1, \dots, c_{\ell}$  has a solution. Consider any subset of  $\ell+1$  queries  $(\vec{\rho}_i, \text{aux}_i)_{i \in [\ell+1]}$  made by the adversary to the random oracle and let  $M$  denote the  $(\ell+1) \times \ell$  matrix whose  $i$ -th row is  $\vec{\rho}_i$  and let  $t \leq \ell$  denote its rank. Then, Eq. (3) has a solution if and only if the row vector  $\vec{h} := (\text{H}_{\text{ros}}(\vec{\rho}_i, \text{aux}_i))_{i \in [\ell+1]}^{\text{T}}$  is in the span of the columns of  $M$ . Since  $\vec{h}$  is uniformly random, this happens with probability  $p^t/p^{\ell+1} \leq 1/p \leq 1/2^{\lambda-1}$ . By the union bound,

$$\text{Adv}_{\text{GrGen}, \ell, \Omega, \mathcal{A}}^{\text{ros}^*}(\lambda) \leq \frac{\binom{q_h}{\ell+1}}{2^{\lambda-1}} ,$$

which concludes the proof.  $\square$

On the other hand, the  $\text{ROS}_{\ell}$  problem can be reduced to 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

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

and let  $\vec{\rho}_i$  denote its  $i$ -th line,  $i \in [\ell + 1]$ . Let  $q := 2^{\lambda/(1+\lceil \lg(\ell+1) \rceil)}$ . The solving algorithm builds lists  $L_i = (\mathbf{H}_{\text{ros}}(\vec{\rho}_i, \text{aux}_{i,k}))_{k \in [q]}$  for  $i \in [\ell]$  and  $L_{\ell+1} = (-\mathbf{H}_{\text{ros}}(\vec{\rho}_{\ell+1}, \text{aux}_{\ell+1,k}))_{k \in [q]}$  for arbitrary values  $\text{aux}_{i,k}$  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  $((\vec{\rho}_i, \text{aux}_i)_{i \in [\ell+1]}, (e_j)_{j \in [\ell]})$ , where  $\text{aux}_i$  is such that  $e_i = \mathbf{H}_{\text{ros}}(\vec{\rho}_i, \text{aux}_i)$ , is a solution to the ROS problem. This algorithm makes  $q_h = (\ell+1)2^{\lambda/(1+\lceil \lg(\ell+1) \rceil)}$  random oracle queries, runs in time and space  $O((\ell+1)2^{\lambda/(1+\lceil \lg(\ell+1) \rceil)})$ , and succeeds with constant probability.

### 4.3 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 2.** *Let GrGen be a group generator. Let  $\mathcal{A}_{\text{alg}}$  be an algebraic adversary against the UNF security of the blind Schnorr signature scheme  $\text{BISch}[\text{GrGen}]$  running in time at most  $\tau$  and making at most  $q_s$  queries to  $\text{SIGN}_1$  and  $q_h$  queries to the random oracle. Then there exist an algorithm  $\mathcal{B}_{\text{ros}}$  for the  $\text{ROS}_{q_s}$  problem making at most  $q_h + q_s + 1$  random oracle queries and an algorithm  $\mathcal{B}_{\text{omdl}}$  for the OMDL problem w.r.t. GrGen making at most  $q_s$  queries to its oracle DLOG, both running in time at most  $\tau + O(q_s + q_h)$ , such that*

$$\text{Adv}_{\text{BISch}[\text{GrGen}], \mathcal{A}_{\text{alg}}}^{\text{unf}}(\lambda) \leq \text{Adv}_{\text{GrGen}, \mathcal{B}_{\text{omdl}}}^{\text{omdl}}(\lambda) + \text{Adv}_{\ell, \mathcal{B}_{\text{ros}}}^{\text{ros}}(\lambda) .$$

We start with explaining the proof idea. Consider an adversary in the unforgeability game, let  $X$  be the public key and  $R_1, \dots, R_\ell$  be the elements returned by the oracle  $\text{SIGN}_1$  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  $(\gamma_i, \xi_i, \vec{\rho}_i)$  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  $c_i^* = \mathbf{H}(R_i^*, m_i^*)$ . Together, these yield

$$(c_i^* + \xi_i^*)X + \sum_{j=1}^{\ell} \rho_{i,j}^* R_j = (s_i^* - \gamma_i^*)G , \quad (4)$$

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

However, the reduction also needs to simulate  $\text{SIGN}_2$  queries, for which, contrary to the proof for standard Schnorr signatures ([Theorem 1](#)), it cannot rely on programming the random oracle. In fact, the reduction can only break OMDL, which is an *easier* game than DL. In particular, the reduction obtains  $X, R_1, \dots, R_q$  from its challenger and must compute their logarithms. It can make  $q$  logarithm queries, which it uses to simulate the  $\text{SIGN}_2$  oracle: on input  $(j, c_j)$ , it simply returns  $s_j \leftarrow \text{DLOG}(R_j + c_j X)$ .

But this means that in [Eq. \(4\)](#) 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 [Eq. \(4\)](#) yields

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

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,  $x, r_1, \dots, r_q$  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 (see [Figure 7](#)): A reduction to ROS would answer the

adversary's queries  $H(R_{[\gamma, \xi, \bar{\rho}]}, m)$  by  $H_{\text{ros}}(\bar{\rho}, (\gamma, \xi, m)) - \xi$ ; then  $\chi_i = 0$  implies (recall that  $c_i^* = H(R_i^*, m^*)$ )

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

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

To simplify the proof we first show the following lemma.

**Lemma 3.** *Let GrGen be a group generator and let  $\mathcal{A}$  be an adversary against the UNF security of the blind Schnorr signature scheme  $\text{BISch}[\text{GrGen}]$  running in time at most  $\tau$  and making at most  $q_s$  queries to  $\text{SIGN}_1$  and  $q_h$  queries to the random oracle. Then there exists an adversary  $\mathcal{B}$  that makes exactly  $q_s$  queries to  $\text{SIGN}_1$  and  $q_s$  queries to  $\text{SIGN}_2$  that do not return  $\perp$ , and returns  $q_s + 1$  forgeries, running in time at most  $\tau + O(q_s)$ , such that*

$$\text{Adv}_{\text{BISch}[\text{GrGen}], \mathcal{A}}^{\text{unf}}(\lambda) = \text{Adv}_{\text{BISch}[\text{GrGen}], \mathcal{B}}^{\text{unf}}(\lambda) .$$

*Proof.* We construct the following adversary that plays game UNF (Figure 5). 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  $\text{SIGN}_1$  queries, let  $R_1, \dots, R_q$  be the answers, and let  $\mathcal{C}$  be the completed sessions, that is, the set of values  $j$  such that  $\mathcal{A}$  queried  $\text{SIGN}_2$  on some input  $(j, *)$  and  $\text{SIGN}_2$  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.

$\mathcal{B}$  now makes  $q_s - q$  queries to  $\text{SIGN}_1$  to receive  $R_{q+1}, \dots, R_{q_s}$ . Next,  $\mathcal{B}$  completes all  $q_s - k$  open signing sessions for distinct messages by following the protocol in Figure 6: for every  $j \in \mathcal{S} := [1, \dots, q_s] \setminus \mathcal{C}$ , 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  $\alpha_j, \beta_j \leftarrow \mathbb{Z}_p$ , computes  $R'_j := R_j + \alpha_j G + \beta_j X$ , queries  $H(R'_j, m_j)$  to get  $c'_j$ , computes  $c_j := c'_j + \beta_j \bmod p$  and queries  $(j, c_j)$  to  $\text{SIGN}_2$ . 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,  $\mathcal{B}$  returns  $\mathcal{A}$ 's output concatenated with  $q_s + 1 - n \leq q_s - k$  signatures: let  $\mathcal{S} = \{j_1, \dots, j_{q_s - k}\}$ ; then  $\mathcal{B}$  returns  $(m_i^*, (R_i^*, s_i^*))_{i \in [n]} \parallel (m_{j_i}, (R'_{j_i}, s'_{j_i}))_{i \in [q_s + 1 - n]}$ .

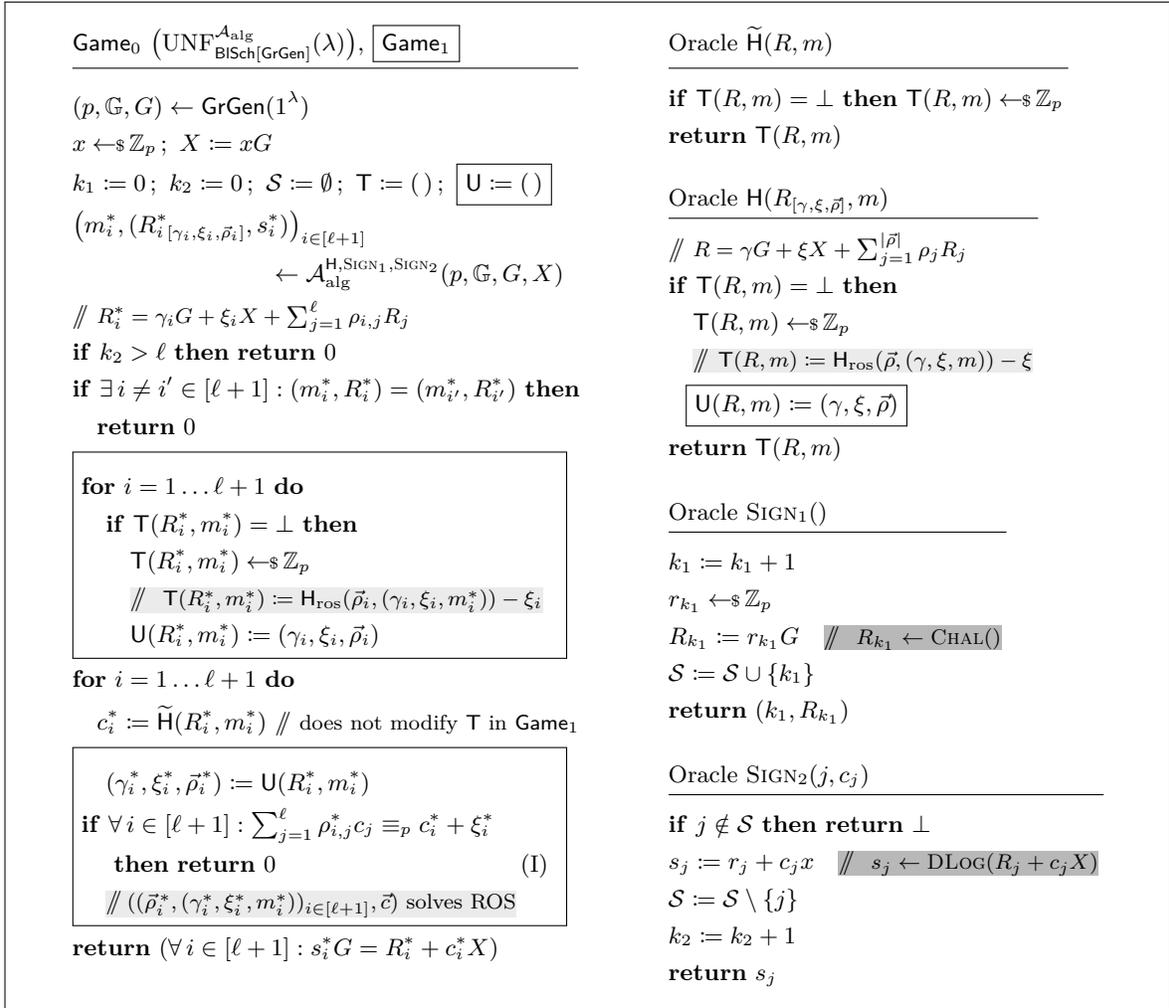
When  $\mathcal{A}$  wins the game, all tuples  $(m_i^*, (R_i^*, s_i^*))$  are different; since moreover all remaining messages are distinct, all tuples output by  $\mathcal{B}$  differ. By correctness of the scheme, all of  $\mathcal{B}$ 's signatures are valid. Thus whenever  $\mathcal{A}$  wins then so does  $\mathcal{B}$ .  $\square$

*Proof of Theorem 2.* Let  $\mathcal{A}_{\text{alg}}$  be an algebraic adversary making at most  $q_s$  queries to  $\text{SIGN}_1$  and  $q_h$  queries to the random oracle. By the above lemma, we can assume that  $\mathcal{A}_{\text{alg}}$  makes exactly  $\ell := q_s$  queries to  $\text{SIGN}_1$ , closes all sessions, and returns  $\ell + 1$  valid signatures. To prove the theorem under this assumption, it now suffices to construct  $\mathcal{B}_{\text{ros}}$  and  $\mathcal{B}_{\text{omdl}}$  and show

$$\text{Adv}_{\text{BISch}[\text{GrGen}], \mathcal{A}_{\text{alg}}}^{\text{unf}}(\lambda) \leq \text{Adv}_{\text{GrGen}, \mathcal{B}_{\text{omdl}}}^{\text{omdl}}(\lambda) + \text{Adv}_{\ell, \mathcal{B}_{\text{ros}}}^{\text{ros}}(\lambda) . \quad (5)$$

We proceed with a sequence of games defined in Figure 8.

Game<sub>0</sub>. The first game is the UNF game (see Figure 5) for scheme  $\text{BISch}[\text{GrGen}]$  played with  $\mathcal{A}_{\text{alg}}$  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



**Fig. 8.** Games used in the proof of [Theorem 2](#). **Game<sub>0</sub>** (ignore all boxes) is the unforgeability game adapted for the original blind Schnorr signature scheme  $\text{BISch}[\text{GrGen}]$  in the random oracle model for an algebraic adversary  $\mathcal{A}_{\text{alg}}$ . The light-gray comments in **Game<sub>1</sub>** and oracle **H** show how the reduction  $\mathcal{B}_{\text{ros}}$  solves ROS and the comments in the **SIGN** oracles show how  $\mathcal{B}_{\text{omdl}}$  embeds its challenges and simulates **Game<sub>1</sub>**.

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,

$$\text{Adv}_{\mathcal{A}_{\text{alg}}}^{\text{game}_0}(\lambda) = \text{Adv}_{\text{BISch}[\text{GrGen}], \mathcal{A}_{\text{alg}}}^{\text{unf}}(\lambda). \quad (6)$$

**Game<sub>1</sub>**. In **Game<sub>1</sub>**, we make the following changes (which are analogous to those in the proof of [Theorem 1](#)). First, we introduce an auxiliary table  $\mathsf{U}$  that for each query  $\text{H}(R_{[\gamma, \xi, \vec{\rho}]}, m)$  stores the representation  $(\gamma, \xi, \vec{\rho})$  of  $R$ . Clearly, this does not change the output of the game. Second, when the adversary returns its forgeries  $(m_i^*, (R_{i[\gamma_i, \xi_i, \vec{\rho}_i]}^*, s_i^*))_{i \in [\ell+1]}$ , then for each  $i \in [\ell+1]$  for which  $\mathsf{T}(R_i^*, m_i^*)$  is undefined, we emulate a call to  $\text{H}(R_{i[\gamma_i, \xi_i, \vec{\rho}_i]}^*, m_i^*)$ . Again, this does not change the probability that the game returns 1, since in **Game<sub>0</sub>**, the value  $\mathsf{T}(R_i^*, m_i^*)$  would be randomly assigned when the game calls  $\tilde{\text{H}}$  to check the signature. Finally, for each  $i \in [\ell+1]$ ,

we retrieve  $(\gamma_i^*, \xi_i^*, \vec{\rho}_i^*) := \mathsf{U}(R_i^*, m_i^*)$  (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  $\text{SIGN}_2$  together with  $j$  and not answered by  $\perp$ .

Clearly,  $\text{Game}_0$  and  $\text{Game}_1$  are identical unless  $\text{Game}_1$  returns 0 in line (I). We reduce indistinguishability of  $\text{Game}_0$  and  $\text{Game}_1$  to ROS by constructing an algorithm  $\mathcal{B}_{\text{ros}}$  solving the  $\text{ROS}_{\ell}$  problem whenever  $\text{Game}_1$  returns 0 in line (I). Algorithm  $\mathcal{B}_{\text{ros}}$ , which has access to oracle  $\mathsf{H}_{\text{ros}}$ , runs  $\mathcal{A}_{\text{alg}}$  and simulates  $\text{Game}_1$  in a straightforward way, except that it uses its  $\mathsf{H}_{\text{ros}}$  oracle to define the entries of  $\mathsf{T}$ .

In particular, consider a query  $\mathsf{H}(R_{[\gamma, \xi, \vec{\rho}]}, m)$  by  $\mathcal{A}_{\text{alg}}$  such that  $\mathsf{T}(R, m) = \perp$ . Then  $\mathcal{B}_{\text{ros}}$  pads the vector  $\vec{\rho}$  with 0's to make it of length  $\ell$  (at this point, not all  $R_1, \dots, R_{\ell}$  are necessarily defined, so  $\vec{\rho}$  might not be of length  $\ell$ ), and assigns  $\mathsf{T}(R, m) := \mathsf{H}_{\text{ros}}(\vec{\rho}, (\gamma, \xi, m)) - \xi$  (cf. comments in Figure 8). Similarly, when  $\mathcal{A}_{\text{alg}}$  has returned its forgeries  $(m_i^*, (R_{i, [\gamma_i, \xi_i, \vec{\rho}_i]}^*, s_i^*))_{i \in [\ell+1]}$ , then for each  $i \in [\ell+1]$  with  $\mathsf{T}(R_i^*, m_i^*) = \perp$ , reduction  $\mathcal{B}_{\text{ros}}$  assigns  $\mathsf{T}(R_i^*, m_i^*) := \mathsf{H}_{\text{ros}}(\vec{\rho}_i, (\gamma_i, \xi_i, m_i^*)) - \xi_i$ . Since  $\mathsf{H}_{\text{ros}}$  returns uniformly random elements in  $\mathbb{Z}_p$ , the simulation is perfect.

If  $\text{Game}_1$  aborts in line (I), then  $\mathcal{B}_{\text{ros}}$  returns  $((\vec{\rho}_i^*, (\gamma_i^*, \xi_i^*, m_i^*))_{i \in [\ell+1]}, (c_j)_{j \in [\ell]})$ , where  $(\gamma_i^*, \xi_i^*, \vec{\rho}_i^*) := \mathsf{U}(R_i^*, m_i^*)$ . We show that this is a valid ROS solution.

First, for all  $i \neq i' \in [\ell + 1]$ :  $(\vec{\rho}_i^*, (\gamma_i^*, \xi_i^*, m_i^*)) \neq (\vec{\rho}_{i'}^*, (\gamma_{i'}^*, \xi_{i'}^*, m_{i'}^*))$ . 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  $i \in [\ell + 1]$ ,  $c_i^* = \mathsf{H}_{\text{ros}}(\vec{\rho}_i^*, (\gamma_i^*, \xi_i^*, m_i^*)) - \xi_i^*$ . This is clearly the case if  $\mathsf{T}(R_i^*, m_i^*) = \perp$  when the adversary returns its forgeries. Indeed, in that case  $(\gamma_i^*, \xi_i^*, \vec{\rho}_i^*) = (\gamma_i, \xi_i, \vec{\rho}_i)$  and

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

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

$$\text{Adv}_{\mathcal{A}_{\text{alg}}}^{\text{game}_1}(\lambda) \geq \text{Adv}_{\mathcal{A}_{\text{alg}}}^{\text{game}_0}(\lambda) - \text{Adv}_{\ell, \mathcal{B}_{\text{ros}}}^{\text{ros}}(\lambda) . \quad (7)$$

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

**REDUCTION TO OMDL.** In our last step, we construct an algorithm  $\mathcal{B}_{\text{omdl}}$  solving the OMDL problem whenever  $\mathcal{A}_{\text{alg}}$  wins  $\text{Game}_1$ . Algorithm  $\mathcal{B}_{\text{omdl}}$ , which has access to two oracles  $\text{CHAL}$  and  $\text{DLOG}$  (see Figure 1 in Section 2) takes as input a group description  $(p, \mathbb{G}, G)$ , makes a first query  $X \leftarrow \text{CHAL}()$ , and runs  $\mathcal{A}_{\text{alg}}$  on input  $(p, \mathbb{G}, G, X)$ , simulating  $\text{Game}_1$  as follows (cf. comments in Figure 8). Each time  $\mathcal{A}_{\text{alg}}$  makes a  $\text{SIGN}_1()$  query,  $\mathcal{B}_{\text{omdl}}$  queries its  $\text{CHAL}$  oracle to obtain  $R_j$ . It simulates  $\text{SIGN}_2(j, c)$  without knowledge of  $x$  and  $r_j$  by querying  $s_j \leftarrow \text{DLOG}(R_j + cX)$ .

Assume that  $\text{Game}_1$  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  $\ell$  calls to its oracle  $\text{DLOG}$  in total (since it makes exactly one call for each (valid)  $\text{SIGN}_2$  query made by  $\mathcal{A}_{\text{alg}}$ ).

Since  $\text{Game}_1$  did not return 0 in line (I), there exists  $i \in [\ell + 1]$  such that

$$\sum_{j=1}^{\ell} \rho_{i,j}^* c_j \not\equiv_p c_i^* + \xi_i^* . \quad (8)$$

Since the  $i$ -th forgery is valid, we have

$$s_i^* G = R_i^* + c_i^* X . \quad (9)$$

On the other hand,  $(\gamma_i^*, \xi_i^*, \vec{\rho}_i^*)$  is a representation of  $R_i^*$ , i.e.,

$$R_i^* = \gamma_i^* G + \xi_i^* X + \sum_{j=1}^{\ell} \rho_{i,j}^* R_j . \quad (10)$$

Combining Equations (9) and (10), we get

$$(c_i^* + \xi_i^*) X + \sum_{j=1}^{\ell} \rho_{i,j}^* R_j = (s_i^* - \gamma_i^*) G . \quad (11)$$

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

$$R_j = s_j G - c_j X . \quad (12)$$

Injecting Eq. (12) in Eq. (11), we obtain

$$\left( c_i^* + \xi_i^* - \sum_{j=1}^{\ell} \rho_{i,j}^* c_j \right) X = \left( s_i^* - \gamma_i^* - \sum_{j=1}^{\ell} \rho_{i,j}^* s_j \right) G . \quad (13)$$

Since by Eq. (8) the coefficient in front of  $X$  is non-zero, this allows  $\mathcal{B}_{\text{omdl}}$  to compute  $x := \log X$ . Furthermore, from Eq. (12) we have  $r_j := \log R_j = s_j - c_j x$  for all  $j \in [\ell]$ . By returning  $(x, r_1, \dots, r_\ell)$ ,  $\mathcal{B}_{\text{omdl}}$  solves the OMDL problem whenever  $\mathcal{A}_{\text{alg}}$  wins  $\text{Game}_1$ , which implies

$$\text{Adv}_{\text{GrGen}, \mathcal{B}_{\text{omdl}}}^{\text{omdl}}(\lambda) = \text{Adv}_{\mathcal{A}_{\text{alg}}}^{\text{game}_1}(\lambda) . \quad (14)$$

Eq. (5) now follows from Equations (6), (7) and (14).  $\square$

## 5 The Clause Blind Schnorr Signature Scheme

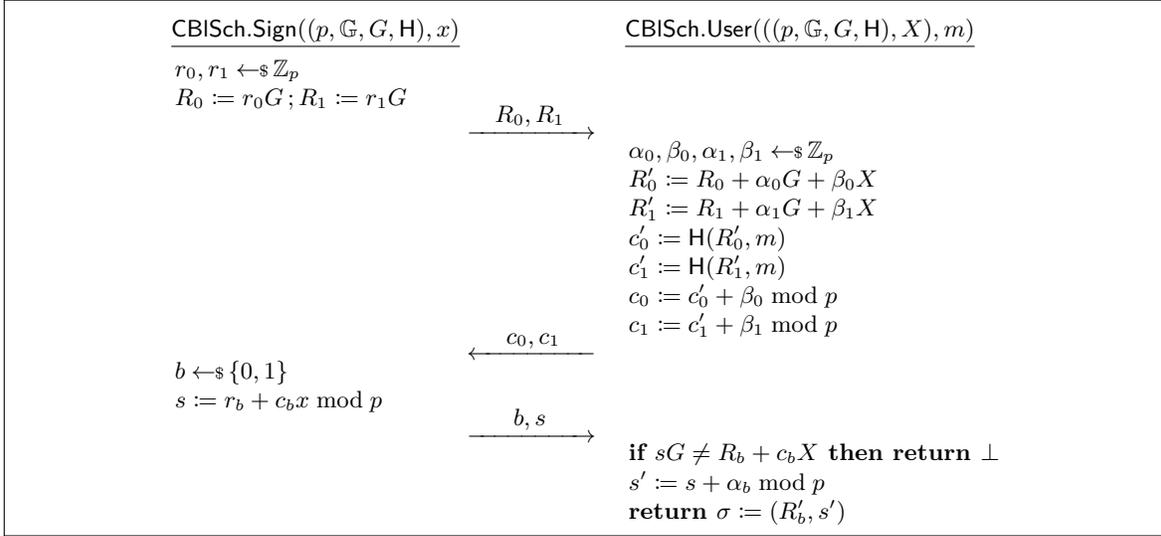
We present a variation of the blind Schnorr signature scheme that only modifies the signing protocol. The scheme thus does not change the signatures themselves, meaning that it can be very smoothly integrated in existing applications.

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 3, the attacker must now solve a *modified* ROS problem, which we define in Figure 10.

We start with explaining the modified signing protocol, which we formally define in Figure 9. The idea is that in the first round the signer and the user execute two parallel runs of the blind signing protocol from Figure 6, 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 Section 4.2, the adversary opens  $\ell$  signing sessions, receiving  $R_1, \dots, R_\ell$ , then searches a solution  $\vec{c}$  to the ROS problem and closes the signing sessions by sending  $c_1, \dots, c_\ell$ .

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  $\ell$ , this probability vanishes exponentially.



**Fig. 9.** The clause blind Schnorr signing protocol.

In [Theorem 3](#) 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  $(\text{Run}_1 \vee \dots \vee \text{Run}_t)$ . This decreases the probability that the user correctly guesses which challenges will be answered by the signer in  $\ell$  concurrent sessions.

**THE MODIFIED ROS PROBLEM.** Consider [Figure 10](#). The difference to the original ROS problem ([Figure 7](#)) is that the queries to the  $H_{\text{ROS}}$  oracle consist of *two* vectors  $\vec{\rho}_0, \vec{\rho}_1$  and additional *aux* information. Analogously, the adversary's task is to return  $\ell + 1$  tuples  $(\vec{\rho}_{i,0}, \vec{\rho}_{i,1}, \text{aux}_i)$ , except that the ROS solution  $c_1^*, \dots, c_\ell^*$  is selected as follows: for every index  $j \in [\ell]$  the adversary must query an additional oracle  $\text{SELECT}(j, c_{j,0}, c_{j,1})$ , which flips a random bit  $b_j$  and sets the  $j$ -th coordinate of the solution to  $c_j^* := c_{j,b_j}$ .

Up to now, nothing really changed, as an adversary could always choose  $\vec{\rho}_{i,0} = \vec{\rho}_{i,1}$  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  $\rho$  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  $\rho$  values.

We conjecture that the best attack against this modified ROS problem is to guess the  $\ell$  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  $2^\ell$  and requires time

$$O\left(2^\ell \cdot (\ell + 1)2^{\lambda/(1+\lceil \lg(\ell+1) \rceil)}\right).$$

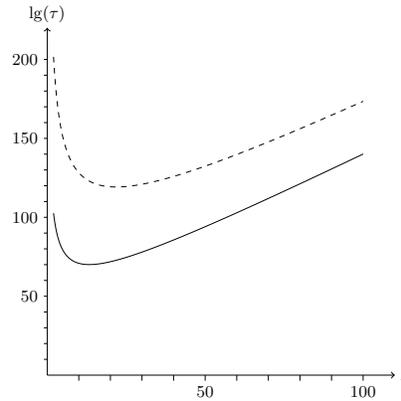
Game $\text{MROS}_{\text{GrGen}, \ell, \Omega}^A(\lambda)$	Oracle $\text{H}_{\text{ros}}(\vec{\rho}_0, \vec{\rho}_1, \text{aux})$
$(p, \mathbb{G}, G) \leftarrow \text{GrGen}(1^\lambda)$ $\text{T}_{\text{ros}} := ()$ $(\vec{\rho}_{i,0}, \vec{\rho}_{i,1}, \text{aux}_i)_{i \in [\ell+1]} \leftarrow \mathcal{A}^{\text{H}_{\text{ros}}, \text{SELECT}}(p)$ $\quad // \vec{\rho}_{i,b} = (\rho_{i,b,1}, \dots, \rho_{i,b,\ell})$ <b>return</b> $(\forall i \neq i' : (\vec{\rho}_{i,0}, \vec{\rho}_{i,1}, \text{aux}_i) \neq (\vec{\rho}_{i',0}, \vec{\rho}_{i',1}, \text{aux}_{i'}))$ $\quad \wedge \forall i \in [\ell+1] : \sum_{j=1}^{\ell} \rho_{i,b_j,j} c_j \equiv_p \text{H}_{\text{ros}}(\vec{\rho}_{i,0}, \vec{\rho}_{i,1}, \text{aux}_i)$ $\quad \wedge \forall i \in [\ell+1], \forall j \in [\ell] : \rho_{i,1-b_j,j} = 0$	<b>if</b> $\text{T}_{\text{ros}}(\vec{\rho}_0, \vec{\rho}_1, \text{aux}) = \perp$ <b>then</b> $\quad \text{T}_{\text{ros}}(\vec{\rho}_0, \vec{\rho}_1, \text{aux}) \leftarrow \$_{\mathbb{Z}_p}$ <b>return</b> $\text{T}_{\text{ros}}(\vec{\rho}_0, \vec{\rho}_1, \text{aux})$ <hr/> Oracle $\text{SELECT}(j, c'_0, c'_1)$ $\quad // \text{ must be queried } \forall j \in [\ell]$ $\quad b_j \leftarrow \$_{\{0,1\}}$ $\quad c_j := c'_{b_j}$ <b>return</b> $b_j$

**Fig. 10.** The modified ROS problem.

This estimated complexity is plotted for  $\lambda \in \{256, 512\}$  in [Figure 11](#). 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.

UNFORGEABILITY OF THE CLAUSE BLIND SCHNORR SIGNATURE SCHEME. We now prove that the Schnorr signature scheme from [Figure 3](#), with the signing algorithm replaced by the protocol in [Figure 9](#) is secure under the OMDL assumption for the underlying group and hardness of the modified ROS problem.

**Theorem 3.** *Let  $\text{GrGen}$  be a group generator. Let  $\mathcal{A}_{\text{alg}}$  be an algebraic adversary against the UNF security of the clause blind Schnorr signature scheme  $\text{CBISch}[\text{GrGen}]$  running in time at most  $\tau$  and making at most  $q_s$  queries to  $\text{SIGN}_1$  and  $q_h$  queries to the random oracle. Then there exist an algorithm  $\mathcal{B}_{\text{ros}}$  for the  $\text{MROS}_{q_s}$  problem making at most  $q_h + q_s + 1$  random oracle queries and an algorithm  $\mathcal{B}_{\text{omdl}}$  for the OMDL problem w.r.t.  $\text{GrGen}$  making at most  $q_s$*



**Fig. 11.** The estimated complexity  $\tau$  of the conjectured best attack against the modified ROS problem as a function of parameter  $\ell$  for  $\lambda = 256$  (solid line) and  $\lambda = 512$  (dashed line).

queries to its oracle DLOG, both running in time at most  $\tau + O(q_s + q_h)$ , such that

$$\text{Adv}_{\text{BISch}[\text{GrGen}], \mathcal{A}_{\text{alg}}}^{\text{unf}}(\lambda) \leq \text{Adv}_{\text{GrGen}, \mathcal{B}_{\text{omdl}}}^{\text{omdl}}(\lambda) + \text{Adv}_{\ell, \mathcal{B}_{\text{mros}}}^{\text{mros}}(\lambda).$$

The theorem follows by adapting the proof of [Theorem 2](#); we therefore discuss the changes and refer to [Figure 12](#), 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 2](#) is that when simulating  $\text{SIGN}_2$ , the reduction  $\mathcal{B}_{\text{mros}}$  calls  $\text{SELECT}(j, c_{j,0}, c_{j,1})$  to obtain bit  $b$  instead of choosing it itself. By definition,  $\text{Game}_1$  aborts in line (I) if and only if  $\mathcal{B}_{\text{mros}}$  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 2](#); (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  $R_{j,1-b_j}$ .

More in detail, the main difference to [Theorem 2](#) 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 [Eq. \(10\)](#) in [Theorem 2](#)). Validity of the forgery implies  $R_i^* = s_i^* G - c_i^* X$ , which together with the above yields

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

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

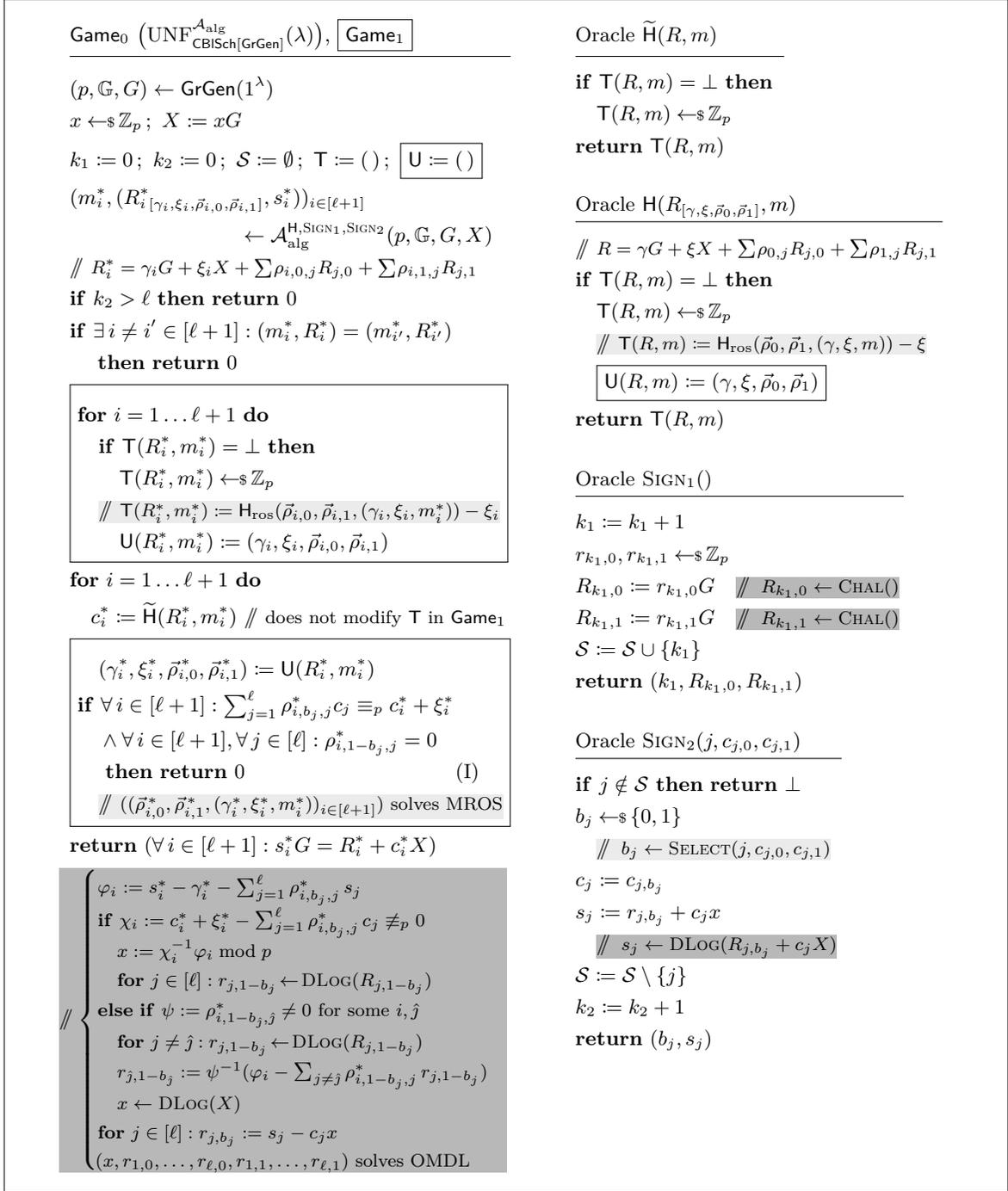
$$(c_i^* + \xi_i^* - \sum_{j=1}^{\ell} \rho_{i,b_j,j}^* c_j) X = (s_i^* - \gamma_i^* - \sum_{j=1}^{\ell} \rho_{i,b_j,j}^* s_j) G - \sum_{j=1}^{\ell} \rho_{i,1-b_j,j}^* R_{j,1-b_j} \quad (15)$$

(which corresponds to [Eq. \(13\)](#) in [Theorem 2](#)). In [Theorem 2](#), 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 exactly case (2) defined above). In this case  $\text{Game}_1$  does not abort and the OMDL reduction  $\mathcal{B}_{\text{omdl}}$  must succeed. Since in this case the left-hand side of [Eq. \(15\)](#) 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  $R_{1,1-b_1}, \dots, R_{\ell,1-b_\ell}$  to its DLOG oracle. In both cases it made  $2\ell$  queries to DLOG and computed the discrete logarithms of all  $2\ell + 1$  challenges.

[Figure 12](#) presents the unforgeability game and  $\text{Game}_1$ , 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  $\text{Game}_1$  aborts in line (I), and how a reduction  $\mathcal{B}_{\text{omdl}}$  solves OMDL whenever  $\mathcal{A}_{\text{alg}}$  wins  $\text{Game}_1$ .



**Fig. 12.** Games used in the proof of [Theorem 3](#). Game<sub>0</sub> is the unforgeability game for the clause blind Schnorr signature scheme in the ROM for an algebraic adversary  $\mathcal{A}_{\text{alg}}$ . The comments in light gray show how  $\mathcal{B}_{\text{mros}}$  solves MROS; the dark comments show how  $\mathcal{B}_{\text{omdl}}$  solves OMDL.

BLINDNESS OF THE CLAUSE BLIND SCHNORR SIGNATURE SCHEME. Blindness of the “clause” variant in Figure 9 follows via a hybrid argument from blindness of the standard scheme (Figure 6). In the game defining blindness 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  $(\beta_b, s_b)$  in the third round ( $\beta_b$  corresponds to  $b$  in Figure 9), 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 this 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  $\beta_0$  and  $\beta_1$  (and return a random bit if the guess turns out wrong). The hybrid game then replaces the  $\beta_0$ -th message of the first two and the  $\beta_1$ -th of the last two (as opposed to the ones underlined as above). Following this argument, in Appendix B, where we formally define the game  $\text{BLIND}_{\text{BS}}^A$ , we prove the following:

**Theorem 4.** *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*

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

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

## 6 Schnorr-Signed 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 a 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), keys are pseudorandom when given an encapsulation. By hashing the key, that is, defining  $k := \text{H}(xY)$ , the assumption can be relaxed to CDH in the random-oracle model. In the AGM, the (standard) ElGamal KEM 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].

The idea of *Schnorr-signed* 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 13.

The strongest security notion for KEM schemes is indistinguishability of ciphertexts under chosen-ciphertext attack (IND-CCA2), where the adversary can query decryptions of

<p><u>SEGK.Setup(<math>\lambda</math>)</u></p> <p><math>(p, \mathbb{G}, G) \leftarrow \text{GrGen}(1^\lambda)</math>          Select <math>H: \{0, 1\}^* \rightarrow \mathbb{Z}_p</math>          Select <math>H': \{0, 1\}^* \rightarrow \mathcal{K}</math>  <b>return</b> <math>par := (p, \mathbb{G}, G, H, H')</math></p> <hr/> <p><u>SEGK.Enc(<math>pk</math>)</u></p> <p><math>(p, \mathbb{G}, G, H, H', Y) := pk</math>  <math>x, r \leftarrow \mathbb{Z}_p</math>; <math>X := xG</math>; <math>R := rG</math>  <math>k := H'(xY)</math>; <math>s := r + H(R, X) \cdot x \bmod p</math>  <b>return</b> <math>(k, (X, R, s))</math></p>	<p><u>SEGK.KeyGen(<math>par</math>)</u></p> <p><math>(p, \mathbb{G}, G, H, H') := par</math>  <math>y \leftarrow \mathbb{Z}_p</math>; <math>Y := yG</math>  <math>sk := (par, y)</math>; <math>pk := (par, Y)</math>  <b>return</b> <math>(sk, pk)</math></p> <hr/> <p><u>SEGK.Dec(<math>sk, (X, R, s)</math>)</u></p> <p><math>(p, \mathbb{G}, G, H, H', y) := sk</math>  <b>if</b> <math>sG \neq R + H(R, X) \cdot X</math> <b>then</b>              <b>return</b> <math>\perp</math>  <b>return</b> <math>k := H'(yX)</math></p>
---	---

**Fig. 13.** The Schnorr-signed ElGamal KEM scheme  $\text{SEGK}[\text{GrGen}]$  for key space  $\mathcal{K}$ .

encapsulations of its choice even after receiving the challenge. The (decisional) game IND-CCA2 is defined in [Figure 14](#).

We now prove that the Schnorr-signed ElGamal KEM is tightly IND-CCA2-secure against algebraic adversaries in the random-oracle model under the discrete logarithm assumption.

**Theorem 5.** *Let  $\text{GrGen}$  be a group generator. Let  $\mathcal{A}_{\text{alg}}$  be an algebraic adversary against the IND-CCA2 security of the Schnorr-signed ElGamal KEM scheme  $\text{SEGK}[\text{GrGen}]$  making at most  $q_d$  decryption queries and  $q_h$  queries to both random oracles. Then there exists an algorithm  $\mathcal{B}$  solving the DL problem w.r.t.  $\text{GrGen}$ , such that*

$$\text{Adv}_{\text{SEGK}[\text{GrGen}], \mathcal{A}_{\text{alg}}}^{\text{ind-cca2}}(\lambda) \leq 2 \cdot \text{Adv}_{\text{GrGen}, \mathcal{B}}^{\text{dl}}(\lambda) + \frac{q_d + \frac{1}{2^{\lambda-1}}(q_d + q_h)}{2^{\lambda-2}}.$$

We give the proof idea here; a formal proof can be found in [Appendix A](#). Let  $Y$  be the public key and let  $(X^* = x^*G, R^*, s^*)$  be the challenge ciphertext. If the adversary never queries  $H'(x^*Y)$  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

<p><u>Game IND-CCA2<math>_{\text{KEM}}^{\mathcal{A}}</math>(<math>\lambda</math>)</u></p> <p><math>par \leftarrow \text{KEM.Setup}(\lambda)</math>  <math>(pk, sk) \leftarrow \text{KEM.KeyGen}(par)</math>  <math>b \leftarrow \mathbb{Z}_p</math>  <math>b' \leftarrow \mathcal{A}^{\text{Enc, Dec}}(pk)</math>  <b>return</b> <math>(b = b')</math></p>	<p><u>Oracle ENC() // one time</u></p> <p><math>(k_0, c^*) \leftarrow \text{KEM.Enc}(pk)</math>; <math>k_1 \leftarrow \mathbb{Z}_p</math>  <b>return</b> <math>(k_b, c^*)</math></p> <hr/> <p><u>Oracle DEC(<math>c</math>)</u></p> <p><b>if</b> <math>c = c^*</math> <b>then return</b> <math>\perp</math>  <b>return</b> <math>\text{KEM.Dec}(sk, c)</math></p>
--	---

**Fig. 14.** The IND-CCA2 security game for a KEM scheme KEM.

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  $H'(K^*)$  then we have  $K^* = zy^2G$ . On the other hand, the adversary must provide a representation  $(\gamma, v, \xi, \rho)$  of  $K^*$  w.r.t.  $(G, Y, X^*, R^*)$ , and thus

$$K^* = \gamma G + vY + \xi X^* + \rho R^* = (\gamma + vy + \xi zy + \rho s^* - \rho c^* zy)G, \quad (16)$$

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

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

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 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  $k = H'(xY)$ , but this extraction is trickier than in the proof of [Theorem 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  $(\gamma, v, \xi, \rho)$  and  $(\gamma', v', \xi', \rho')$  of  $R$  and  $X$  provided by the adversary, we can write (analogously to [Eq. \(16\)](#)):

$$\begin{aligned} r = \log R &\equiv_p \gamma + vy + \xi zy + \rho s^* - \rho c^* zy \equiv_p \alpha y + (\gamma + \rho s^*) \quad \text{and} \\ x = \log X &\equiv_p \gamma' + v'y + \xi' zy + \rho' s^* - \rho' c^* zy \equiv_p \alpha' y + (\gamma' + \rho' s^*) \end{aligned} \quad (17)$$

with  $\alpha := v + (\xi - \rho c^*)z \bmod p$  and  $\alpha' := v' + (\xi' - \rho' c^*)z \bmod p$ . Since the signature  $(R, s)$  contained in the query must be valid, we have  $s \equiv_p r + cx$ . 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  $c = H(R, X)$  was chosen by the experiment after the adversary provided representations of  $R$  and  $X$ , which define  $\alpha$  and  $\alpha'$ , we have that  $\alpha + \alpha' c \equiv_p 0$  happens with probability  $\frac{1}{p}$ , unless  $\alpha' = 0$ .

In the latter case however, from [Eq. \(17\)](#) we have  $x = \gamma' + \rho' s^* \bmod p$ , meaning the reduction can compute  $x$  and can therefore answer the decryption query by returning  $H'(xY) = H'(yX)$ .

**ACKNOWLEDGEMENTS.** The first author is supported by the French ANR *EfTrEC* project (ANR-16-CE39-0002). The first and the second authors are supported by the *MSR-Inria Joint Centre*.

## References

- [Abe01] Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 136–151. Springer, May 2001.
- [ABM15] Michel Abdalla, Fabrice Benhamouda, and Philip MacKenzie. Security of the J-PAKE password-authenticated key exchange protocol. In *2015 IEEE Symposium on Security and Privacy*, pages 571–587. IEEE Computer Society Press, May 2015.
- [ABR01] Michel Abdalla, Mihir Bellare, and Phillip Rogaway. The oracle Diffie-Hellman assumptions and an analysis of DHIES. In David Naccache, editor, *CT-RSA 2001*, volume 2020 of *LNCS*, pages 143–158. Springer, April 2001.
- [BCC04] Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 2004*, pages 132–145. ACM Press, October 2004.
- [BCC<sup>+</sup>09] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable proofs and delegatable anonymous credentials. In Shai Halevi, editor, *CRYPTO 2009*, volume 5677 of *LNCS*, pages 108–125. Springer, August 2009.
- [BDL<sup>+</sup>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, September 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*, volume 11273 of *LNCS*, pages 435–464. Springer, December 2018.
- [BFPV13] Olivier Blazy, Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Short blind signatures. *Journal of Computer Security*, 21(5):627–661, 2013.
- [BFW16] David Bernhard, Marc Fischlin, and Bogdan Warinschi. On the hardness of proving CCA-security of signed ElGamal. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part I*, volume 9614 of *LNCS*, pages 47–69. Springer, March 2016.
- [BL13] Foteini Baldimtsi and Anna Lysyanskaya. On the security of one-witness blind signature schemes. In Kazuo Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 82–99. Springer, December 2013.
- [BLS04] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. *Journal of Cryptology*, 17(4):297–319, September 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, June 2003.
- [Bol03] Alexandra Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In Yvo Desmedt, editor, *PKC 2003*, volume 2567 of *LNCS*, pages 31–46. Springer, January 2003.
- [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*, volume 2442 of *LNCS*, pages 162–177. Springer, August 2002.
- [BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby, editors, *ACM CCS 93*, pages 62–73. ACM Press, November 1993.
- [BR95] Mihir Bellare and Phillip Rogaway. Optimal asymmetric encryption. In Alfredo De Santis, editor, *EUROCRYPT’94*, volume 950 of *LNCS*, pages 92–111. Springer, May 1995.
- [Bra94] Stefan Brands. Untraceable off-line cash in wallets with observers (extended abstract). In Douglas R. Stinson, editor, *CRYPTO’93*, volume 773 of *LNCS*, pages 302–318. Springer, August 1994.
- [CFN90] David Chaum, Amos Fiat, and Moni Naor. Untraceable electronic cash. In Shafi Goldwasser, editor, *CRYPTO’88*, volume 403 of *LNCS*, pages 319–327. Springer, August 1990.
- [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] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. In Ronald Cramer, editor, *EUROCRYPT 2005*, volume 3494 of *LNCS*, pages 302–321. Springer, May 2005.

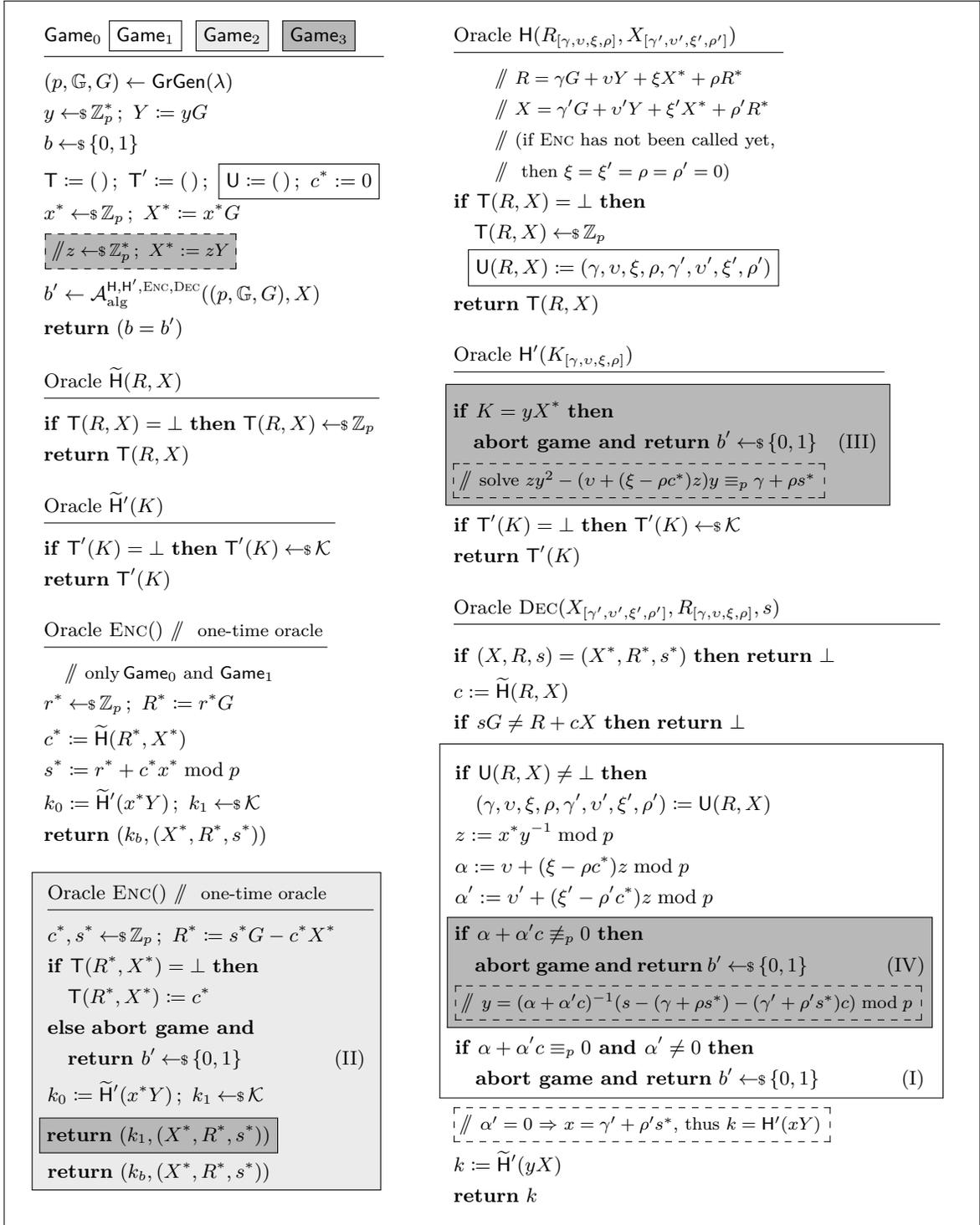
- [CL01] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer, May 2001.
- [CP93] David Chaum and Torben P. Pedersen. Wallet databases with observers. In Ernest F. Brickell, editor, *CRYPTO'92*, volume 740 of *LNCS*, pages 89–105. Springer, August 1993.
- [CS03] Ronald Cramer and Victor 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.
- [ELG85] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.
- [FHS15] Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Practical round-optimal blind signatures in the standard model. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part II*, volume 9216 of *LNCS*, pages 233–253. Springer, August 2015.
- [FJS19] Nils Fleischhacker, Tibor Jager, and Dominique 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*, volume 10992 of *LNCS*, pages 33–62. Springer, August 2018.
- [FOO93] Atsushi Fujioka, Tatsuaki Okamoto, and Kazuo Ohta. A practical secret voting scheme for large scale elections. In Jennifer Seberry and Yuliang Zheng, editors, *AUSCRYPT'92*, volume 718 of *LNCS*, pages 244–251. Springer, December 1993.
- [FPV09] Georg Fuchsbauer, David Pointcheval, and Damien Vergnaud. Transferable constant-size fair e-cash. In Juan A. Garay, Atsuko Miyaji, and Akira Otsuka, editors, *CANS 09*, volume 5888 of *LNCS*, pages 226–247. Springer, December 2009.
- [FS10] Marc Fischlin and Dominique Schröder. On the impossibility of three-move blind signature schemes. In Henri Gilbert, editor, *EUROCRYPT 2010*, volume 6110 of *LNCS*, pages 197–215. Springer, May / June 2010.
- [Fuc11] Georg Fuchsbauer. Commuting signatures and verifiable encryption. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 224–245. Springer, May 2011.
- [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*, volume 5157 of *LNCS*, pages 93–107. Springer, August 2008.
- [GG14] Sanjam Garg and Divya Gupta. Efficient round optimal blind signatures. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 477–495. Springer, May 2014.
- [GRS<sup>+</sup>11] Sanjam Garg, Vanishree Rao, Amit Sahai, Dominique Schröder, and Dominique Unruh. Round optimal blind signatures. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 630–648. Springer, August 2011.
- [HHK10] Javier Herranz, Dennis Hofheinz, and Eike Kiltz. Some (in)sufficient conditions for secure hybrid encryption. *Inf. Comput.*, 208(11):1243–1257, 2010.
- [HKL19] Eduard Hauck, Eike Kiltz, and Julian Loss. A modular treatment of blind signatures from identification schemes. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 345–375. Springer, May 2019.
- [Jak98] Markus Jakobsson. A practical mix. In Kaisa Nyberg, editor, *EUROCRYPT'98*, volume 1403 of *LNCS*, pages 448–461. Springer, May / June 1998.
- [MPSW19] Gregory Maxwell, Andrew Poelstra, Yannick Seurin, and Pieter Wuille. Simple Schnorr multi-signatures with applications to Bitcoin. *Designs, Codes and Cryptography*, 2019.
- [MS12] Lorenz Minder and Alistair Sinclair. The extended k-tree algorithm. *Journal of Cryptology*, 25(2):349–382, April 2012.
- [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*, 2019. Slides and video available at <https://jonasnick.github.io/blog/2018/07/31/blind-signatures-in-scriptless-scripts/>.
- [NS15] Ivica Nikolic and Yu Sasaki. Refinements of the k-tree algorithm for the generalized birthday problem. In Tetsu Iwata and Jung Hee Cheon, editors, *ASIACRYPT 2015, Part II*, volume 9453 of *LNCS*, pages 683–703. Springer, November / December 2015.

- [OO92] Tatsuaki Okamoto and Kazuo Ohta. Universal electronic cash. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 324–337. Springer, August 1992.
- [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, June 2011.
- [PS96a] David Pointcheval and Jacques Stern. Provably secure blind signature schemes. In Kwangjo Kim and Tsutomu Matsumoto, editors, *ASIACRYPT'96*, volume 1163 of *LNCS*, pages 252–265. Springer, November 1996.
- [PS96b] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In Ueli M. Maurer, editor, *EUROCRYPT'96*, volume 1070 of *LNCS*, pages 387–398. Springer, May 1996.
- [PS00] David Pointcheval and Jacques 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*, volume 3788 of *LNCS*, pages 1–20. Springer, December 2005.
- [Sch90] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, August 1990.
- [Sch91] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.
- [Sch01] Claus-Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In Sihang Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *ICICS 01*, volume 2229 of *LNCS*, pages 1–12. Springer, November 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*, volume 7237 of *LNCS*, pages 554–571. Springer, April 2012.
- [SG02] Victor Shoup and Rosario 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*, volume 1233 of *LNCS*, pages 256–266. Springer, May 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] Claus-Peter Schnorr and Markus Jakobsson. Security of signed ElGamal encryption. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 73–89. Springer, December 2000.
- [ST13] Yannick Seurin and Joana Treger. A robust and plaintext-aware variant of signed ElGamal encryption. In Ed Dawson, editor, *CT-RSA 2013*, volume 7779 of *LNCS*, pages 68–83. Springer, February / March 2013.
- [TY98] Yiannis Tsiounis and Moti Yung. On the security of ElGamal based encryption. In Hideki Imai and Yuliang Zheng, editors, *PKC'98*, volume 1431 of *LNCS*, pages 117–134. Springer, February 1998.
- [Wag02] David Wagner. A generalized birthday problem. In Moti Yung, editor, *CRYPTO 2002*, volume 2442 of *LNCS*, pages 288–303. Springer, August 2002.
- [Wui18] Pieter Wuille. Schnorr signatures for secp256k1. Bitcoin Improvement Proposal, 2018. See <https://github.com/sipa/bips/blob/bip-schnorr/bip-schnorr.mediawiki>.

## A Proof of Theorem 5

Consider the games  $\text{Game}_0$  through  $\text{Game}_3$  in Figure 15, where in  $\text{Game}_3$  the adversary's advantage is 0.  $\text{Game}_0$  has the same behavior as  $\text{IND-CCA2}_{\text{SEGK}[\text{GrGen}]_{\text{Alg}}^{\text{Alg}}}$ ; 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  $\text{Game}_i$  and  $\text{Game}_{i+1}$ .

We start with the difference between  $\text{Game}_0$  and  $\text{Game}_1$ , which consists in a possible abort in line (I) in oracle DEC. This happens when the experiment randomly chooses  $c$  as one



**Fig. 15.** The IND-CCA2 security game for the Schnorr-Signed ElGamal KEM scheme  $\text{IND-CCA2}_{\text{SEG}[\text{GrGen}]}^{\text{Alg}}(\lambda)$  (Game<sub>0</sub>) and games Game<sub>1</sub>–Game<sub>3</sub> used in the proof. The comments in dashed boxes show how the DL reduction simulates Game<sub>3</sub> without knowledge of  $x^*$  and  $y$ .

particular value. (Note that  $\text{Game}_1$  sets  $c^* := 0$ , so the value is defined in DEC when ENC has not been called yet.)

Observe that at any point  $\text{T}(R^*, X^*)$  is the only value in  $\text{T}$  that was not set during an adversary's call to H or Dec, and that thus does *not* have a corresponding entry in  $\text{U}$ . 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  $s = \log R^* + \text{T}(R^*, S^*) \log X^* = s^*$  and the oracle would have returned  $\perp$  in the 1st line.

$\text{Game}_1$  sets the values  $(\gamma, v, \xi, \rho, \gamma', v', \xi', \rho')$  that were given as the representation of  $X$  and  $R$  when  $\text{T}(X, R) = c$  was defined. As we argued above, this must have been during a call from the adversary. The value  $c$  is thus independent of  $(\gamma, \dots, \rho')$ , the values that define  $\alpha$  and  $\alpha'$ .

The two games  $\text{Game}_0$  and  $\text{Game}_1$  behave identically unless  $\text{Game}_1$  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  $\frac{1}{2^{\lambda-1}}$ . We thus have

$$\begin{aligned} \text{Adv}_{\mathcal{A}_{\text{alg}}}^{\text{game}_1}(\lambda) &= 2 \cdot \Pr[1 \leftarrow \text{Game}_1] - 1 \\ &\geq 2 \cdot (\Pr[1 \leftarrow \text{Game}_0] - q_d/2^{\lambda-1}) - 1 = \text{Adv}_{\mathcal{A}_{\text{alg}}}^{\text{game}_0}(\lambda) - \frac{q_d}{2^{\lambda-2}}. \end{aligned} \quad (18)$$

The two games  $\text{Game}_1$  and  $\text{Game}_2$  behave identically unless oracle ENC generates values  $(R^*, X^*)$  that have already been assigned a value in the table  $\text{T}$ . 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  $q_h + q_d$  values in  $\text{T}$  are assigned. Thus, the probability that  $(R^*, X^*)$  collides with one of the entries is bounded by  $\frac{q_h + q_d}{(2^{\lambda-1})^2}$ , and we thus have

$$\text{Adv}_{\mathcal{A}_{\text{alg}}}^{\text{game}_2}(\lambda) \geq \text{Adv}_{\mathcal{A}_{\text{alg}}}^{\text{game}_1}(\lambda) - 2 \cdot \frac{(q_d + q_h)}{(2^{\lambda-1})^2}. \quad (19)$$

**REDUCTION TO DL.** We now construct an adversary  $\mathcal{B}$  solving DL whenever  $\text{Game}_3$  differs from  $\text{Game}_2$ , 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  $z \leftarrow_{\$} \mathbb{Z}_p^*$  and sets  $X^* := zY$ . It simulates  $\boxed{\text{ENC}()}$  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  $\text{Game}_3$  for  $\mathcal{A}$  without knowledge of  $y$  and  $x^*$  as follows (cf. the comments in  $\boxed{\text{dashed boxes}}$  in Figure 15):

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

$$K = \gamma G + vY + \xi X^* + \rho R^* = (\gamma + vy + \xi zy + \rho s^* - \rho c^* zy)G,$$

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

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

and checking whether some solution  $y$  satisfies  $Y = yG$  (in this case  $\text{Game}_3$  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  $\text{H}'(yX^*) = k_b$ .

- Queries to DEC: when queried  $(X, R, s)$ , DEC returns  $\perp$  if  $sG \neq R + H(R, X)X$  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  $(\gamma, v, \xi, \rho)$  and  $(\gamma', v', \xi', \rho')$  are such that

$$\begin{aligned} R &= \gamma G + vY + \xi X^* + \rho R^* = (\gamma + vy + \xi zy + \rho s^* - \rho c^* zy)G \\ X &= \gamma' G + v'Y + \xi' X^* + \rho' R^* = (\gamma' + v'y + \xi' zy + \rho' s^* - \rho' c^* zy)G . \end{aligned} \quad (20)$$

As in DEC, we let  $\alpha := v + (\xi - \rho c^*)z \bmod p$  and  $\alpha' := v' + (\xi' - \rho' c^*)z \bmod p$  and thus from Eq. (20) we have

$$\begin{aligned} r &:= \log R = \gamma + \rho s^* + \alpha y \bmod p \\ x &:= \log X = \gamma' + \rho' s^* + \alpha' y \bmod p . \end{aligned} \quad (21)$$

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 Eq. (21):

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

If  $\alpha + \alpha' c \not\equiv_p 0$  then  $\text{Game}_3$  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  $\text{Game}_2$  and  $\text{Game}_3$  (and thus  $\mathcal{B}$ ) abort in line (I). Otherwise we must have  $\alpha' = 0$  and, from Eq. (21):  $x = \gamma' + \rho' s^* \bmod p$ . The reduction can thus simulate the decryption query by returning  $H'(xY)$  (which might define a new  $H'$  value or not).

This shows that whenever  $\text{Game}_3$  differs from  $\text{Game}_2$  (in lines (III) or (IV)), reduction  $\mathcal{B}$  solves the DL problem, which yields:

$$\text{Adv}_{\mathcal{A}_{\text{alg}}}^{\text{game}_3}(\lambda) \geq \text{Adv}_{\mathcal{A}_{\text{alg}}}^{\text{game}_2}(\lambda) - 2 \cdot \text{Adv}_{\text{GrGen}, \mathcal{B}}^{\text{dl}}(\lambda) . \quad (22)$$

Inspecting  $\text{Game}_3$ , we note that  $\mathcal{A}$ 's output is independent of  $b$  and if the game aborts it outputs a random bit; we thus have:

$$\text{Adv}_{\mathcal{A}_{\text{alg}}}^{\text{game}_3}(\lambda) = 2 \cdot \Pr[1 \leftarrow \text{Game}_3] - 1 = 0 . \quad (23)$$

The theorem now follows from Equations (18), (19), (22) and (23).

## B 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 9, by reducing it to blindness of the (standard) blind Schnorr signature scheme BISch (Figure 6).

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  $\text{BISch.User}_1$  of session  $i$ , in which the user sends a challenge  $c$ . Oracle  $U_2$  is the second interaction  $\text{BISch.User}_2$ , which, once both sessions are finished, outputs the resulting signatures.

The formal game  $\text{BLIND}_{\text{BISch}}$  for adversary  $\mathcal{B}$  is specified in Figure 16, where we follow the definition from Hauk, Kiltz and Loss [HKL19]. As usual,  $\mathcal{B}$ 's advantage is defined as  $\text{Adv}_{\text{BISch}, \mathcal{B}}^{\text{blind}}(\lambda) := 2 \cdot \Pr[1 \leftarrow \text{BLIND}_{\text{BISch}}^{\mathcal{B}}(\lambda)] - 1$ .

<p style="text-align: center; border-bottom: 1px solid black;">Game <math>\text{BLIND}_{\text{BISch}}^{\mathcal{B}}(\lambda)</math></p> <p><math>b \leftarrow_{\\$} \{0, 1\}</math>  <math>b_0 := b; b_1 := 1 - b</math>  <math>b' \leftarrow \mathcal{B}^{\text{INIT}, U_1, U_2}(1^\lambda)</math>  <b>return</b> (<math>b' = b</math>)</p> <p style="text-align: center; border-bottom: 1px solid black;">INIT(<math>pk, m_0, m_1</math>)</p> <p><math>\text{sess}_0 := \text{init}</math>  <math>\text{sess}_1 := \text{init}</math></p>	<p style="text-align: center; border-bottom: 1px solid black;">Oracle <math>U_1(i, R_i)</math></p> <p><b>if</b> <math>i \notin \{0, 1\} \vee \text{sess}_i \neq \text{init}</math> <b>then return</b> <math>\perp</math>  <math>\text{sess}_i := \text{open}</math>  <math>(\text{state}_i, c_i) \leftarrow \text{BISch.User}_1(pk, R_i, m_{b_i})</math>  <b>return</b> <math>c_i</math></p> <p style="text-align: center; border-bottom: 1px solid black;">Oracle <math>U_2(i, s_i)</math></p> <p><b>if</b> <math>\text{sess}_i \neq \text{open}</math> <b>then return</b> <math>\perp</math>  <math>\text{sess}_i := \text{closed}</math>  <math>\sigma_{b_i} \leftarrow \text{BISch.User}_2(\text{state}_i, s_i)</math>  <b>if</b> <math>\text{sess}_0 = \text{sess}_1 = \text{closed}</math> <b>then</b>        <b>if</b> <math>\sigma_0 = \perp \vee \sigma_1 = \perp</math> <b>then</b> <math>(\sigma_0, \sigma_1) := (\perp, \perp)</math>        <b>return</b> <math>(\sigma_0, \sigma_1)</math>  <b>else return</b> <math>\epsilon</math></p>
<hr style="border: 0; border-top: 1px solid black; margin: 0;"/>	
<p style="text-align: center; border-bottom: 1px solid black;">Game <math>\text{BLIND}_{\text{CBISch}}^{\mathcal{A}}(\lambda), \boxed{G^{\mathcal{A}}(\lambda)}</math></p> <p><math>b \leftarrow_{\\$} \{0, 1\}</math>  <math>b_0 := b; b_1 := 1 - b</math>  <math>\boxed{\hat{\beta}_0, \hat{\beta}_1 \leftarrow_{\\$} \{0, 1\}}</math>  <math>b' \leftarrow \mathcal{A}^{\text{INIT}, U_1, U_2}(1^\lambda)</math>  <math>\boxed{\text{if } \hat{\beta}_0 \neq \beta_0 \vee \hat{\beta}_1 \neq \beta_1 \text{ then } b' \leftarrow_{\\$} \{0, 1\}}</math>  <b>return</b> (<math>b' = b</math>)</p> <p style="text-align: center; border-bottom: 1px solid black;">INIT(<math>pk, m_0, m_1</math>)</p> <p><math>\text{sess}_0 := \text{init}</math>  <math>\text{sess}_1 := \text{init}</math></p>	<p style="text-align: center; border-bottom: 1px solid black;">Oracle <math>U_1(i, R_{i,0}, R_{i,1})</math></p> <p><b>if</b> <math>i \notin \{0, 1\} \vee \text{sess}_i \neq \text{init}</math> <b>then return</b> <math>\perp</math>  <math>\text{sess}_i := \text{open}</math>  <math>(\text{state}_{i,0}, c_{i,0}) \leftarrow \text{BISch.User}_1(pk, R_{i,0}, m_{b_i})</math>  <math>(\text{state}_{i,1}, c_{i,1}) \leftarrow \text{BISch.User}_1(pk, R_{i,1}, m_{b_i})</math>  <b>return</b> <math>(c_{i,0}, c_{i,1})</math></p> <p style="text-align: center; border-bottom: 1px solid black;">Oracle <math>U_2(i, s_i, \beta_i)</math></p> <p><b>if</b> <math>\text{sess}_i \neq \text{open}</math> <b>then return</b> <math>\perp</math>  <math>\text{sess}_i := \text{closed}</math>  <math>\sigma_{b_i} \leftarrow \text{BISch.User}_2(\text{state}_{i,\beta_i}, s_i)</math>  <math>\boxed{\sigma_{b_i} \leftarrow \text{BISch.User}_2(\text{state}_{i,\hat{\beta}_i}, s_i)}</math>  <b>if</b> <math>\text{sess}_0 = \text{sess}_1 = \text{closed}</math> <b>then</b>        <b>if</b> <math>\sigma_0 = \perp \vee \sigma_1 = \perp</math> <b>then</b> <math>(\sigma_0, \sigma_1) := (\perp, \perp)</math>        <b>return</b> <math>(\sigma_0, \sigma_1)</math>  <b>else return</b> <math>\epsilon</math></p>

**Fig. 16.** 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.

*Proof of Theorem 4.* Figure 16 shows the blindness game for clause blind Schnorr signatures, where we have replaced  $\text{CBISch.User}_1$  and  $\text{CBISch.User}_2$  by their instantiations in terms of  $\text{BISch.User}_1$  and  $\text{BISch.User}_2$ : the user first runs two instances of  $\text{BISch.User}_1$ , and the signer calls  $U_2$  with an additional input  $\beta$ , which specifies which instance the signer completes.

To reduce blindness of CBISch to blindness of BISch, we will guess the bits  $\beta_0$  and  $\beta_1$  that the adversary will use in its calls to  $U_2$ : game  $G^{\mathcal{A}}(\lambda)$ , specified in [Figure 16](#), is defined like  $\text{BLIND}_{\text{CBISch}}^{\mathcal{A}}$ , 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  $\beta_i$ ; 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

$$\Pr [1 \leftarrow G^{\mathcal{A}}(\lambda) \mid \hat{\beta}_0 \neq \beta_0 \vee \hat{\beta}_1 \neq \beta_1] = \frac{1}{2} . \quad (24)$$

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

$$\Pr [1 \leftarrow G^{\mathcal{A}}(\lambda) \mid \hat{\beta}_0 = \beta_0 \wedge \hat{\beta}_1 = \beta_1] = \Pr [1 \leftarrow \text{BLIND}_{\text{CBISch}}^{\mathcal{A}}(\lambda)] . \quad (25)$$

From Eqs. (24) and (25), we have

$$\Pr [1 \leftarrow G^{\mathcal{A}}(\lambda)] = \frac{1}{2} \cdot \frac{3}{4} + \Pr [1 \leftarrow \text{BLIND}_{\text{CBISch}}^{\mathcal{A}}(\lambda)] \cdot \frac{1}{4} ,$$

and thus

$$\text{Adv}_{\mathcal{A}}^{\mathcal{G}}(\lambda) = 2 \cdot \Pr [1 \leftarrow G^{\mathcal{A}}(\lambda)] - 1 = \frac{1}{4} \cdot \text{Adv}_{\text{CBISch}, \mathcal{A}}^{\text{blind}}(\lambda) . \quad (26)$$

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  $G_0^{\mathcal{A}}$  and  $G_1^{\mathcal{A}}$  by modifying  $G^{\mathcal{A}}$  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 17](#) and we define  $\text{BLIND}_{0, \text{BISch}}^{\mathcal{B}}$  and  $\text{BLIND}_{1, \text{BISch}}^{\mathcal{B}}$  analogously. We have:

$$\begin{aligned} \text{Adv}_{\mathcal{A}}^{\mathcal{G}}(\lambda) &= \Pr [1 \leftarrow G^{\mathcal{A}}(\lambda) \mid b = 1] + \Pr [1 \leftarrow G^{\mathcal{A}}(\lambda) \mid b = 0] - 1 \\ &= \Pr [1 \leftarrow G_1^{\mathcal{A}}(\lambda)] - \Pr [1 \leftarrow G_0^{\mathcal{A}}(\lambda)] . \end{aligned} \quad (27)$$

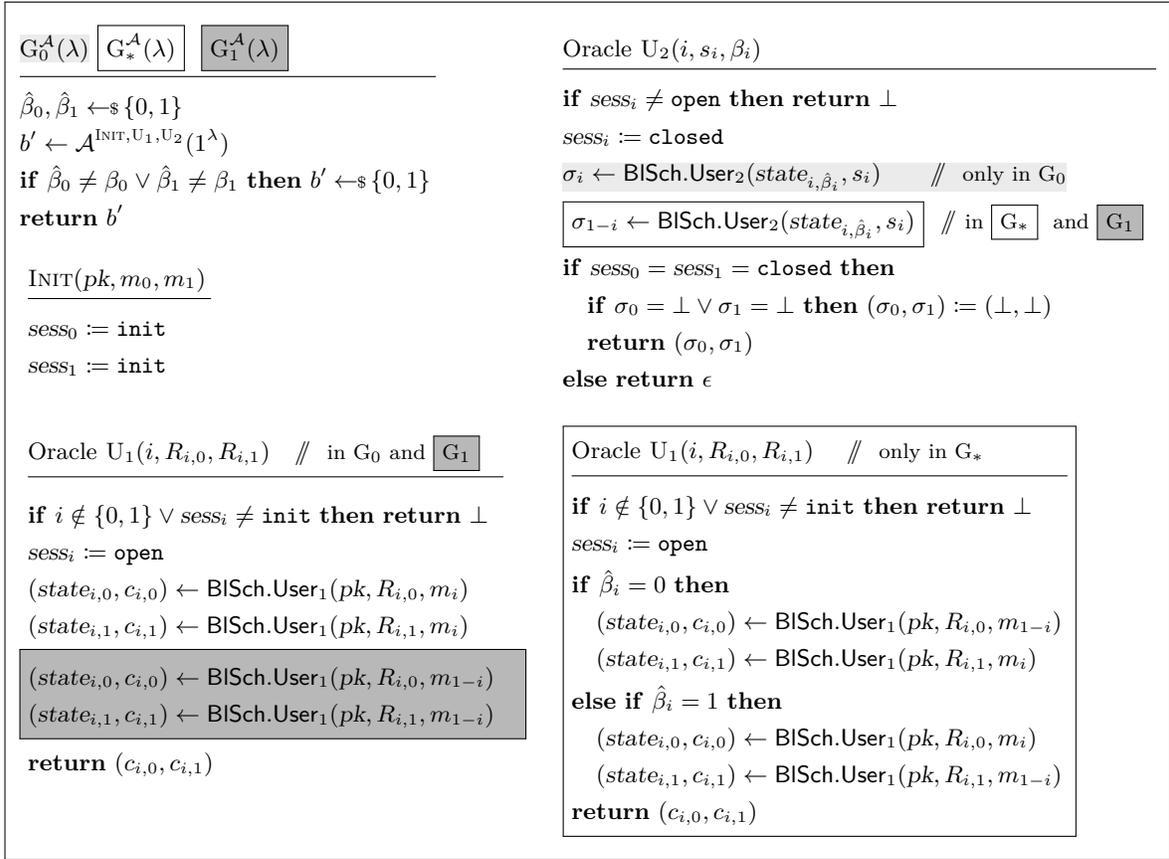
We now define a hybrid game  $G_*$  which lies “between”  $G_0$  and  $G_1$  and is also specified in [Figure 17](#). It differs from  $G_0$  in the  $\hat{\beta}_i$ -th message used in signing session  $i$  and from  $G_1$  in the  $(1 - \hat{\beta}_i)$ -th message. Since

$$\text{Adv}_{\mathcal{A}}^{\mathcal{G}}(\lambda) = \Pr [1 \leftarrow G_1^{\mathcal{A}}(\lambda)] - \Pr [1 \leftarrow G_*^{\mathcal{A}}(\lambda)] + \Pr [1 \leftarrow G_*^{\mathcal{A}}(\lambda)] - \Pr [1 \leftarrow G_0^{\mathcal{A}}(\lambda)] , \quad (28)$$

it suffices to bound these two differences. For the first, we construct an adversary  $\mathcal{B}_1$  playing game  $\text{BLIND}_{\text{BISch}}$  and simulating  $G$  to  $\mathcal{A}$  so that if  $\mathcal{B}_1$  plays  $\text{BLIND}_{0, \text{BISch}}$ , it simulates  $G_0$  to  $\mathcal{A}$ ; whereas if it plays  $\text{BLIND}_{1, \text{BISch}}$ , 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 18](#). By inspection, we have

$$\begin{aligned} \Pr [1 \leftarrow \text{BLIND}_{0, \text{BISch}}^{\mathcal{B}_1}(\lambda)] &= \Pr [1 \leftarrow G_0^{\mathcal{A}}(\lambda)] \quad \text{and} \\ \Pr [1 \leftarrow \text{BLIND}_{1, \text{BISch}}^{\mathcal{B}_1}(\lambda)] &= \Pr [1 \leftarrow G_*^{\mathcal{A}}(\lambda)] . \end{aligned} \quad (29)$$

We also construct an adversary  $\mathcal{B}_2$  that simulates game  $G_*^{\mathcal{A}}(\lambda)$  or  $G_1^{\mathcal{A}}(\lambda)$ . 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 18](#) (note that in



**Fig. 17.** Description of the games  $G_0$  and  $G_1$  which fix the bit  $b$  in game  $G$  from Figure 16.  $G_*$  is a hybrid game that makes the transition between  $G_0$  and  $G_1$ .

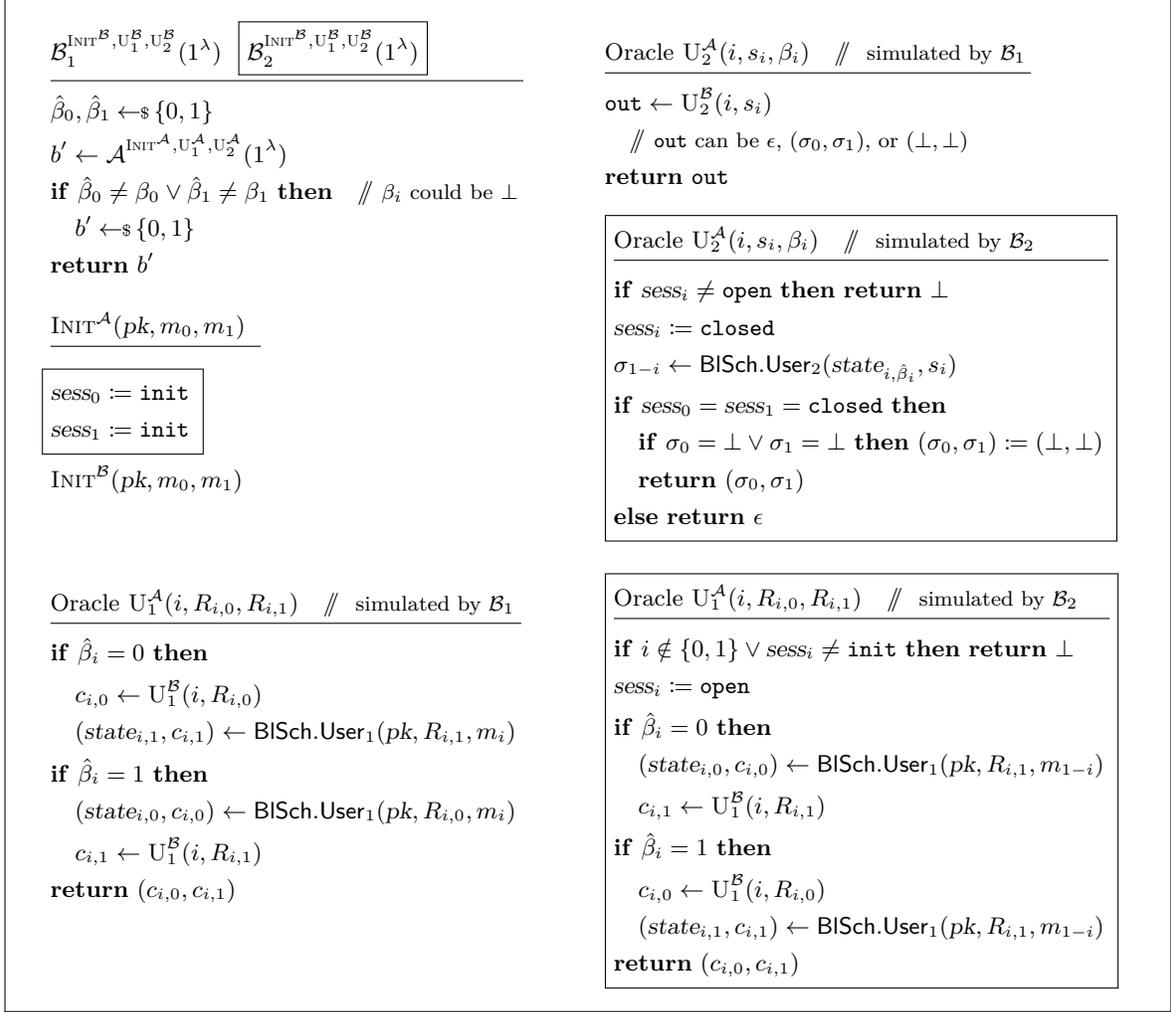
its simulation of  $U_2$ , the variable  $\text{state}_{i, \hat{\beta}_i}$  is always defined because of our syntactical change in Figure 16). We have

$$\begin{aligned}
\Pr[1 \leftarrow \text{BLIND}_{0, \text{BISch}}^{\mathcal{B}_2}(\lambda)] &= \Pr[1 \leftarrow G_*^A(\lambda)] \quad \text{and} \\
\Pr[1 \leftarrow \text{BLIND}_{1, \text{BISch}}^{\mathcal{B}_2}(\lambda)] &= \Pr[1 \leftarrow G_1^A(\lambda)].
\end{aligned} \tag{30}$$

From Eqs. (28)–(30) we get

$$\text{Adv}_{\mathcal{A}}^G(\lambda) = \text{Adv}_{\text{BISch}, \mathcal{B}_1}^{\text{blind}}(\lambda) + \text{Adv}_{\text{BISch}, \mathcal{B}_2}^{\text{blind}}(\lambda),$$

which, together with Eq. (26), concludes the proof.  $\square$



**Fig. 18.** Description of adversaries  $\mathcal{B}_1$  and  $\mathcal{B}_2$ .