Blind Schnorr Signatures and Signed ElGamal Encryption in the Algebraic Group Model

Georg Fuchsbauer[1], Antoine Plouviez[2,3], and Yannick Seurin[4]

[1] TU Wien, Austria
[2] Inria, Paris, France
[3] ENS, CNRS, PSL, Paris, France
[4] ANSSI, Paris, France
first.last@{tuwien.ac.at,ens.fr,m4x.org}

Abstract. The Schnorr blind signing protocol allows blind issuing of Schnorr signatures, one of the most widely used signatures. Despite its practical relevance, its security analysis is unsatisfactory. The only known security proof is informal and in the combination of the generic group model (GGM) and the random oracle model (ROM) assuming that the "ROS problem" is hard. The situation is similar for (Schnorr-)signed ElGamal encryption, a simple CCA2-secure variant of ElGamal.
We analyze the security of these schemes in the algebraic group model (AGM), an idealized model closer to the standard model than the GGM. We first prove tight security of Schnorr signatures from the discrete logarithm assumption (DL) in the AGM+ROM. We then give a rigorous proof for blind Schnorr signatures in the AGM+ROM assuming hardness of the one-more discrete logarithm problem and ROS.
As ROS can be solved in sub-exponential time using Wagner's algorithm, we propose a simple modification of the signing protocol, which leaves the signatures unchanged. It is therefore compatible with systems that already use Schnorr signatures, such as blockchain protocols. We show that the security of our modified scheme relies on the hardness of a problem related to ROS that appears much harder. Finally, we give tight reductions, again in the AGM+ROM, of the CCA2 security of signed ElGamal encryption to DDH and signed hashed ElGamal key encapsulation to DL.

Keywords: Schnorr signatures, blind signatures, algebraic group model, ElGamal encryption, blockchain protocols

1 Introduction

<u>Schnorr Signatures.</u> The Schnorr signature scheme [Sch90, Sch91] is one of the oldest and simplest signature schemes based on prime-order groups. Its adoption was hindered for years by a patent which expired in February 2008, but it is by now widely deployed: EdDSA [BDL+12], a specific instantiation based on twisted Edward curves, is used for example in OpenSSL, OpenSSH, GnuPG and more. Schnorr signatures are also expected to be implemented in Bitcoin

[Wui18], enabling multi-signatures supporting public key aggregation, which will result in considerable scalability and privacy enhancements [BDN18, MPSW19].

The security of the Schnorr signature scheme has been analyzed in the random oracle model (ROM) [BR93], an idealized model which replaces cryptographic hash functions by truly random functions. Pointcheval and Stern [PS96b, PS00] proved Schnorr signatures secure in the ROM under the discrete logarithm assumption (DL). The proof, based on the so-called Forking Lemma, proceeds by rewinding the adversary, which results in a loose reduction (the success probability of the DL solver is a factor q_{h} smaller than that of the adversary, where q_{h} is the number of the adversary's random oracle queries). Using the "meta reduction" technique, a series of works showed that this security loss is unavoidable when the used reductions are either algebraic [PV05, GBL08, Seu12] or generic [FJS19]. Although the security of Schnorr signatures is well understood (in the ROM), the same cannot be said for two related schemes, namely blind Schnorr signatures and Schnorr-signed ElGamal encryption.

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

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

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

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

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

has a solution, where $\mathsf{H}_{\mathrm{ros}} \colon (\mathbb{Z}_p)^\ell \to \mathbb{Z}_p$ is a random oracle. Schnorr showed that an attacker able to solve the ROS_ℓ problem can produce $\ell+1$ valid signatures while interacting (concurrently) only ℓ times with the signer. Slightly later, Wagner

[Wag02] showed that the ROS_ℓ problem can be reduced to the $(\ell+1)$-sum problem, which can solved with time and space complexity $O\big((\ell+1)2^{\lambda/(1+\lfloor \lg(\ell+1)\rfloor)}\big)$, where λ is the bit size of p. For example, for $\lambda = 256$, this attack yields 16 valid signatures after $\ell = 15$ interactions with the signer in time and space close to 2^{55}. For $\ell + 1 = 2^{\sqrt{\lambda}}$, the attack has sub-exponential time and space complexity $O(2^{2\sqrt{\lambda}})$, although the number of signing sessions becomes arguably impractical. Asymptotically, this attack can be thwarted by increasing the group order, but this would make the scheme quite inefficient.

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

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

<u>The Algebraic Group Model.</u> The *generic group model* (GGM) [Nec94, Sho97] is an idealized model for the security analysis of cryptosystems defined over cyclic groups. Instead of receiving concrete group elements, the adversary only gets "handles" for them and has access to an oracle that performs the group operation (denoted additively) on handles. This implies that if the adversary is given a list of (handles of) group elements $(X_1, \ldots, X_n)$ and later returns (a handle of) a group element Z, then by inspecting its oracle calls one can derive a "representation" $\vec{z} = (z_1, \ldots, z_n)$ such that $Z = \sum_{i=1}^{n} z_i X_i$.

Fuchsbauer, Kiltz, and Loss [FKL18] introduced the *algebraic group model* (AGM), a model that lies between the standard model and the GGM. On the one hand, the adversary has direct access to group elements; 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 it outputs, the adversary also gives a representation $\vec{z}$ of Z in terms of 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.

Our starting point is the observation that in the combination[5] AGM+ROM Schnorr signatures have a *tight* security proof under the DL assumption. This is because we can give a reduction which works *straight-line*, i.e., unlike the forking-lemma-based reduction [PS96b, PS00], which must rewind the adversary, it runs the adversary only once.[6] Motivated by this, we then turn to blind Schnorr

[5] This combination was already considered when the AGM was first defined [FKL18].

[6] 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

signatures, whose security in the ROM remains elusive, and study their security in the AGM+ROM.

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 ℓ interactions with the signer must either solve the one-more discrete logarithm (OMDL) problem or the ROS_ℓ problem. Although this is not overly surprising in view of the previous results in the GGM [SJ99, Sch01], this gives a more satisfying characterization of the security of this protocol. Moreover, all previous proofs [SJ99, Sch01] were rather informal; in particular, the reduction solving ROS was not explicitly described. In contrast, we provide precise definitions (in particular for the ROS problem, whose exact specification is central for a security proof) and work out the details of the reductions to both OMDL and ROS, which yields the first rigorous proof.

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

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

Chosen-Ciphertext-Secure ElGamal Encryption. Recall the ElGamal public-key encryption (PKE) scheme [ElG85]: given a cyclic group $(\mathbb{G}, +)$ of prime order p and a generator G, a secret/public key pair is of the form $(y, yG) \in \mathbb{Z}_p \times \mathbb{G}$. A message $M \in \mathbb{G}$ is encrypted as $(X := xG, M + xY)$ for a random $x \leftarrow_\$ \mathbb{Z}_p$. 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

simulation-sound extractable, via a straight-line extractor. Our proof is much simpler and gives a concrete security statement.

distinguish). However, ElGamal has been shown to be IND-CCA1-secure (where no decryption queries can be made after receiving the challenge ciphertext) in the AGM under a "q-type" variant of DDH [FKL18].[7]

A natural way to make ElGamal encryption IND-CCA2-secure is to add a proof of knowledge of the randomness x used to encrypt. (Intuitively, this would make the scheme *plaintext-aware* [BR95].) The reduction of IND-CCA2 security can then extract x to answer decryption queries. Since 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 (Schnorr-)signed ElGamal encryption. This scheme has a number of attractive properties: ciphertext validity can be checked without knowledge of the decryption key, and one can work homomorphically with the "core" ElGamal ciphertext (a property sometimes called "submission-security" [Wik08]), which is very useful in e-voting.

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

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

<ins>Our Results on Signed ElGamal Encryption.</ins> Our second line of contributions is twofold. First, we prove (via a tight reduction) that in the AGM+ROM, Schnorr-signed ElGamal encryption is IND-CCA2-secure under the DDH assumption. While intuitively this should follow naturally from the straight-line extractability of Schnorr proofs of knowledge for algebraic adversaries, the formal proof is technically quite delicate: since messages are group elements, the "basis" of group-element inputs in terms of which the adversary provides representations contains not only the three group elements of the challenge ciphertext but also grows as the adversary queries the decryption oracle.[8]

We finally consider the "hashed" variant of ElGamal (also known as DHIES) [ABR01], in which a key is derived as $k = \mathsf{H}(xY)$. In the ROM, the corresponding key-encapsulation mechanism (KEM) is IND-CCA2-secure under the strong Diffie-Hellman assumption (i.e, CDH is hard even when given a DDH oracle)

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

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

[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.

2 Preliminaries

General Notation. We denote the (closed) integer interval from a to b by $[a, b]$ and let $[b] := [1, b]$. A function $\mu \colon \mathbb{N} \to [0, 1]$ is *negligible* (denoted $\mu = \mathsf{negl}$) if $\forall\, c \in \mathbb{N}\ \exists\, \lambda_c \in \mathbb{N}\ \forall\, \lambda \geq \lambda_c : \mu(\lambda) \leq \lambda^{-c}$. A function ν is *overwhelming* if $1 - \nu = \mathsf{negl}$. The logarithm in base 2 is denoted lg and $x \equiv_p y$ denotes $x \equiv y \pmod p$. For a non-empty finite set S, sampling an element x from S uniformly at random is denoted $x \leftarrow_{\$} S$. All algorithms are probabilistic unless stated otherwise. By $y \leftarrow \mathcal{A}(x_1, \ldots, x_n)$ we denote running algorithm $\mathcal{A}$ on inputs $(x_1, \ldots, x_n)$ and uniformly random coins and assigning the output to y. If $\mathcal{A}$ has oracle access to some algorithm Oracle, we write $y \leftarrow \mathcal{A}^{\textsc{Oracle}}(x_1, \ldots, x_n)$. A list $\vec{z} = (z_1, \ldots, z_n)$, 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* GAME_{par} (see e.g. in Fig. 1) indexed by a set of parameters par consists of a main and oracle procedures. The main procedure has input the security parameter λ and runs an adversary $\mathcal{A}$, which interacts with the game by calling the provided oracles. When the adversary stops, the game computes its output b, which we write $b \leftarrow \mathrm{GAME}^{\mathcal{A}}_{par}(\lambda)$. For truth values we identify **false** with 0 and **true** with 1. Games are either computational or decisional. The *advantage* of $\mathcal{A}$ in GAME_{par} is defined as $\mathsf{Adv}^{\mathrm{game}}_{par,\mathcal{A}}(\lambda) := \Pr[1 \leftarrow \mathrm{GAME}^{\mathcal{A}}_{par}(\lambda)]$ if the game is computational and as $\mathsf{Adv}^{\mathrm{game}}_{par,\mathcal{A}}(\lambda) := 2 \cdot \Pr[1 \leftarrow \mathrm{GAME}^{\mathcal{A}}_{par}(\lambda)] - 1$ if it is decisional, where the probability is taken over the random coins of the game and the adversary. We say that GAME_{par} is *hard* if $\mathsf{Adv}^{\mathrm{game}}_{par,\mathcal{A}}(\lambda) = \mathsf{negl}(\lambda)$ for any probabilistic polynomial-time (p.p.t.) adversary $\mathcal{A}$.

Algebraic Algorithms. A *group description* is a tuple $\mathit{\Gamma} = (p, \mathbb{G}, G)$ where p is an odd prime, $\mathbb{G}$ is an abelian group of order p, and G is a generator of $\mathbb{G}$. We use additive notation for the group law and denote group elements with uppercase letters. We assume the existence of a p.p.t. algorithm GrGen which, on input the security parameter 1^λ in unary, outputs a group description $\mathit{\Gamma} = (p, \mathbb{G}, G)$ where p is of bit-length λ. 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.

An *algebraic security game* (w.r.t. GrGen) is a game $\mathrm{GAME}_{\mathsf{GrGen}}$ that (among other things) runs $\mathit{\Gamma} \leftarrow \mathsf{GrGen}(1^\lambda)$ and runs the adversary on input $\mathit{\Gamma} = (p, \mathbb{G}, G)$. An algorithm $\mathcal{A}_{\mathrm{alg}}$ executed in an algebraic game $\mathrm{GAME}_{\mathsf{GrGen}}$ is *algebraic* if for all group elements Z that it outputs, it also provides a representation of Z relative to all previously received group elements: if $\mathcal{A}_{\mathrm{alg}}$ has so far received $\vec{X} = (X_0, \ldots, X_n) \in \mathbb{G}^{n+1}$ (where by convention we let $X_0 = G$), then $\mathcal{A}_{\mathrm{alg}}$ must output Z together with $\vec{z} = (z_0, \ldots, z_n) \in (\mathbb{Z}_p)^{n+1}$ such that $Z = \sum_{i=0}^{n} z_i X_i$. We

```
Game $\mathrm{DL}^{\mathcal{A}}_{\mathsf{GrGen}}(\lambda)$
    $(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
    $x \leftarrow_{\$} \mathbb{Z}_p\,;\ X := xG$
    $y \leftarrow \mathcal{A}(p, \mathbb{G}, G, X)$
    return $(y = x)$
```

```
Game $\mathrm{OMDL}^{\mathcal{A}}_{\mathsf{GrGen}}(\lambda)$
    $(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
    $\vec{x} := (\,)\,;\ q := 0$
    $\vec{y} \leftarrow \mathcal{A}^{\textsc{Chal},\textsc{DLog}}(p, \mathbb{G}, G)$
    return $\big(\vec{y} = \vec{x} \ \wedge\ q < |\vec{x}|\big)$
```

```
Oracle $\textsc{Chal}()$
    $x \leftarrow_{\$} \mathbb{Z}_p\,;\ X := xG$
    $\vec{x} := \vec{x} \,\|\, (x)$
    return $X$

Oracle $\textsc{DLog}(X)$
    $q := q + 1\,;\ x := \log_G(X)$
    return $x$
```

Fig. 1. The DL and OMDL problems.

let $Z_{[\vec{z}]}$ denote such an augmented output. When writing $\vec{z}$ explicitly, we simply write $Z_{[z_0,\ldots,z_n]}$ (rather than $Z_{[(z_0,\ldots,z_n)]}$) to lighten the notation.

<u>Algebraic Algorithms in the Random Oracle Model.</u> The original paper [FKL18] considered the algebraic group model augmented by a random oracle and proved tight security of BLS signatures [BLS04] in this model. The random oracle in that work is of type $\mathsf{H}\colon \{0,1\}^* \to \mathbb{G}$, and as the outputs are group elements, the adversary's group element representations could depend on them.

In this work the RO is typically of type $\mathsf{H}\colon \mathbb{G} \times \{0,1\}^* \to \mathbb{Z}_p$. Thus, an algebraic adversary querying H on some input (Z, m) must also provide a representation $\vec{z}$ 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 $\widetilde{\mathsf{H}}$, which is used by the game itself (and thus does not take representations of group elements as input) and implements the same function as H.

<u>The One-More Discrete Logarithm Problem.</u> We recall the discrete logarithm (DL) problem in Fig. 1. The one-more discrete logarithm (OMDL) problem, also defined in Fig. 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 SIG consists of the following algorithms:

- $par \leftarrow \mathsf{SIG.Setup}(1^\lambda)$: the setup algorithm takes as input the security parameter λ in unary and outputs public parameters par;
- $(sk, pk) \leftarrow \mathsf{SIG.KeyGen}(par)$: the key generation algorithm takes parameters par and outputs a secret key sk and a public key pk;

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

Oracle $\textsc{Sign}(m)$
    $\sigma \leftarrow \mathsf{SIG.Sign}(sk, m)$
    $\mathsf{Q} \coloneqq \mathsf{Q} \parallel (m)$
    return $\sigma$
```

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

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

Correctness requires that for any λ and any message m, when running $par \leftarrow \mathsf{SIG.Setup}(1^\lambda)$, $(sk, pk) \leftarrow \mathsf{SIG.KeyGen}(par)$, $\sigma \leftarrow \mathsf{SIG.Sign}(sk, m)$, and $b \leftarrow \mathsf{SIG.Ver}(pk, m, \sigma)$, one has $b = 1$ with probability 1. The standard security notion for a signature scheme is *existential unforgeability under chosen-message attack* (EUF-CMA), formalized via game EUF-CMA, which we recall in Fig. 2. The Schnorr signature scheme [Sch91] is specified in Fig. 3.

3.2 Security of Schnorr Signatures in the AGM

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

Theorem 1. *Let* GrGen *be a group generator. Let* $\mathcal{A}_{\mathrm{alg}}$ *be an algebraic adversary against the EUF-CMA security of the Schnorr signature scheme* $\mathsf{Sch}[\mathsf{GrGen}]$ *running in time at most* τ *and making at most* q_{s} *signature queries and* q_{h} *queries*

```
$\mathsf{Sch.Setup}(1^\lambda)$
    $(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
    Select $\mathsf{H}\colon \{0,1\}^* \to \mathbb{Z}_p$
    return $par \coloneqq (p, \mathbb{G}, G, \mathsf{H})$

$\mathsf{Sch.KeyGen}(par)$
    $(p, \mathbb{G}, G, \mathsf{H}) \coloneqq par\,;\ x \leftarrow\!\!\$\ \mathbb{Z}_p\,;\ X \coloneqq xG$
    $sk \coloneqq (par, x)\,;\ pk \coloneqq (par, X)$
    return $(sk, pk)$

$\mathsf{Sch.Sign}(sk, m)$
    $(p, \mathbb{G}, G, \mathsf{H}, x) \coloneqq sk\,;\ r \leftarrow\!\!\$\ \mathbb{Z}_p\,;\ R \coloneqq rG$
    $c \coloneqq \mathsf{H}(R, m)\,;\ s \coloneqq r + cx \bmod p$
    return $\sigma \coloneqq (R, s)$

$\mathsf{Sch.Ver}(pk, m, \sigma)$
    $(p, \mathbb{G}, G, \mathsf{H}, X) \coloneqq pk\,;\ (R, s) \coloneqq \sigma$
    $c \coloneqq \mathsf{H}(R, m)$
    return $(sG = R + cX)$
```

Fig. 3. The Schnorr signature scheme $\mathsf{Sch}[\mathsf{GrGen}]$ based on a group generator GrGen.

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*

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

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

In the AGM+ROM, extraction is straight-line and the security proof thus tight: A valid forgery satisfies $R^* = s^*G - c^*X$, with $c^* := \mathsf{H}(R^*, m^*)$. On the other hand, since the adversary is algebraic, when it made its first query $\mathsf{H}(R^*, m^*)$, it provided a representation of R^* in basis (G, X), that is (γ^*, ξ^*) 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 ξ^*, 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 of Theorem 1. Let $\mathcal{A}_{\text{alg}}$ be an algebraic adversary in EUF-CMA$_{\mathsf{Sch[GrGen]}}$ and making at most q_s signature queries and q_h RO queries. We proceed by a sequence of games specified in Fig. 4.

$\underline{\mathsf{Game}_0}$. The first game is EUF-CMA (Fig. 2) for the Schnorr signature scheme (Fig. 3) with a random oracle H. The game maintains a list Q of queried messages and T of values sampled for H. To prepare the change to Game_1, we have written the finalization of the game in an equivalent way: it first checks that $m^* \notin \mathsf{Q}$ and then runs $\mathsf{Sch.Ver}(pk, m^*, (R^*, s^*))$, which we have written explicitly. Since the adversary is algebraic, it must provide a representation (γ^*, ξ^*) for its forgery $(m^*, (R^*_{[\gamma^*,\xi^*]}, s^*)$ such that $R^* = \gamma^*G + \xi^*X$, and similarly for each RO query $\mathsf{H}(R_{[\gamma,\xi]}, m)$. By definition,

$$\mathsf{Adv}^{\mathsf{game}_0}_{\mathcal{A}_{\text{alg}}}(\lambda) = \mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Sch[GrGen]},\mathcal{A}_{\text{alg}}}(\lambda) \ . \qquad (1)$$

$\underline{\mathsf{Game}_1}$. We introduce an auxiliary table U that for each query $\mathsf{H}(R_{[\gamma,\xi]}, m)$ stores the representation (γ, ξ) of R. Second, when the adversary returns its forgery $(m^*, (R^*_{[\gamma^*,\xi^*]}, s^*))$ and previously made a query $\mathsf{H}(R^*_{[\gamma',\xi']}, m^*)$ for some (γ', ξ'), then we consider this previous representation of R^*, that is, we set $(\gamma^*, \xi^*) := (\gamma', \xi')$. The only actual difference to Game_0 is that Game_1 returns 0 in case $\xi^* \equiv_p -\mathsf{T}(R^*, m^*)$ (line (I)).

We show that this happens with probability $1/p \leq 1/2^{\lambda-1}$. First note that line (I) is only executed if $m^* \notin \mathsf{Q}$, as otherwise the game would already have

```
$\mathsf{Game}_0$ $\left(\text{EUF-CMA}^{\mathcal{A}_{\text{alg}}}_{\mathsf{Sch}[\mathsf{GrGen}]}\right)$, $\boxed{\mathsf{Game}_1}$
                                   $\boxed{\mathsf{Game}_2}$
$(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$x \leftarrow\!\!\$\, \mathbb{Z}_p$ ; $X := xG$
$\mathsf{Q} := ()$ ; $\mathsf{T} := ()$ ; $\boxed{\mathsf{U} := ()}$
$\left(m^*, (R^*_{[\gamma^*,\xi^*]}, s^*)\right) \leftarrow \mathcal{A}^{\mathsf{H},\textsc{Sign}}_{\text{alg}}(p, \mathbb{G}, G, X)$
    // $R^* = \gamma^* G + \xi^* X$
if $m^* \in \mathsf{Q}$ then return 0
$c^* := \widetilde{\mathsf{H}}(R^*, m^*)$
┌──────────────────────────────────────────
│if $\mathsf{U}(R^*, m^*) \neq \perp$ then
│    $(\gamma^*, \xi^*) := \mathsf{U}(R^*, m^*)$
│if $\xi^* \equiv_p -\mathsf{T}(R^*, m^*)$ then
│    return 0                              (I)
└──────────────────────────────────────────
return $(s^* G = R^* + c^* X)$

Oracle $\widetilde{\mathsf{H}}(R, m)$
if $\mathsf{T}(R, m) = \perp$ then
    $\mathsf{T}(R, m) \leftarrow\!\!\$\, \mathbb{Z}_p$
return $\mathsf{T}(R, m)$

Oracle $\mathsf{H}(R_{[\gamma,\xi]}, m)$ // $R = \gamma G + \xi X$
if $\mathsf{T}(R, m) = \perp$ then
    $\mathsf{T}(R, m) \leftarrow\!\!\$\, \mathbb{Z}_p$
    $\boxed{\mathsf{U}(R, m) := (\gamma, \xi)}$
return $\mathsf{T}(R, m)$

Oracle $\textsc{Sign}(m)$ // in $\mathsf{Game}_0$ and $\mathsf{Game}_1$
$r \leftarrow\!\!\$\, \mathbb{Z}_p$ ; $R := rG$
$c := \widetilde{\mathsf{H}}(R, m)$ ; $s := r + cx \bmod p$
$\mathsf{Q} := \mathsf{Q} \| (m)$
return $(R, s)$

Oracle $\textsc{Sign}(m)$ // in $\mathsf{Game}_2$
$c, s \leftarrow\!\!\$\, \mathbb{Z}_p$ ; $R := sG - cX$
 if $\mathsf{T}(R, m) = \perp$ then $\mathsf{T}(R, m) := c$
else abort game and
    return 0                               (II)
$\mathsf{Q} := \mathsf{Q} \| (m)$
return $(R, s)$
```

Fig. 4. Games in the proof of Theorem 1. Game_0 is defined by ignoring all boxes; $\boxed{\text{boxes}}$ are included in Game_1 and Game_2; $\boxed{\text{Gray boxes}}$ are only included in Game_2.

returned 0. Hence $\mathsf{T}(R^*, m^*)$ can only have been defined either (1) during a call to H or (2), if it is still undefined when $\mathcal{A}_{\text{alg}}$ stops, by the game when defining c^*. In both cases the probability of returning 0 in line (I) is $1/p$:

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

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

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

$$\mathsf{Adv}^{\mathsf{game}_2}_{\mathcal{A}_\mathrm{alg}}(\lambda) \ge \mathsf{Adv}^{\mathsf{game}_1}_{\mathcal{A}_\mathrm{alg}}(\lambda) - \frac{q_\mathrm{s}(q_\mathrm{s}+q_\mathrm{h})}{2^{\lambda-1}}\ . \tag{3}$$

<u>Reduction to DL</u>. We now construct an adversary $\mathcal{B}$ solving DL with the same probability as $\mathcal{A}_\mathrm{alg}$ wins Game_2. On input $(p, \mathbb{G}, G)$ and X, the adversary runs $\mathcal{A}_\mathrm{alg}$ on input $(p, \mathbb{G}, G, X)$ and simulates Game_2, which can be done without knowledge of $\log_G(X)$. Assume that the adversary wins Game_2 by returning (m^*, R^*, s^*) and let $c^* := \mathsf{T}(R^*, m^*)$ and (γ^*, ξ^*) be defined as in the game. Thus, $\xi^* \ne -c^* \bmod p$ 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 $\log X = (s^* - \gamma^*)(\xi^* + c^*)^{-1} \bmod p$. Combining this with Eqs. (1)–(3), we have

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

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

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.

<u>Syntax.</u> A blind signature scheme BS consists of the following algorithms:

- $par \leftarrow \mathsf{BS.Setup}(1^\lambda)$ and $(sk, pk) \leftarrow \mathsf{BS.KeyGen}(par)$ and $b \leftarrow \mathsf{BS.Ver}(pk, m, \sigma)$ are defined as for regular signature schemes (Sect. 3.1).
- $(b, \sigma) \leftarrow \langle \mathsf{BS.Sign}(sk), \mathsf{BS.User}(pk, m)\rangle$: an interactive protocol is run between the signer with private input a secret key sk and the user with private input a public key pk and a message m; the signer outputs $b = 1$ if the interaction completes successfully and $b = 0$ otherwise, while the user outputs a signature σ if it terminates correctly, and $\perp$ otherwise. For a 2-round protocol the interaction can be realized by the following algorithms:

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

(Typically, $\mathsf{BS.User}_0$ just initiates the session, and thus $msg_{U,0} = (\,)$ and $state_{U,0} = (pk, m)$.)

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

```
Game $\mathrm{UNF}^{\mathcal{A}}_{\mathsf{BS}}(\lambda)$
$par \leftarrow \mathsf{BS.Setup}(1^{\lambda})$
$(sk, pk) \leftarrow \mathsf{BS.KeyGen}(par)$
$k_1 := 0\,;\; k_2 := 0\,;\; \mathcal{S} := \emptyset$
$(m_i^*, \sigma_i^*)_{i\in[n]} \leftarrow \mathcal{A}^{\textsc{Sign}_1,\textsc{Sign}_2}(pk)$
return $\big(k_2 < n$
    $\wedge\ \forall i \neq j \in [n] : (m_i^*, \sigma_i^*) \neq (m_j^*, \sigma_j^*)$
    $\wedge\ \forall i \in [n] : \mathsf{BS.Ver}(pk, m_i^*, \sigma_i^*) = 1\big)$

Oracle $\textsc{Sign}_1(msg)$
$k_1 := k_1 + 1$    // session id
$(msg', state_{k_1}) \leftarrow \mathsf{BS.Sign}_1(sk, msg)$
$\mathcal{S} := \mathcal{S} \cup \{k_1\}$    // open sessions
return $(k_1, msg')$

Oracle $\textsc{Sign}_2(j, msg)$
if $j \notin \mathcal{S}$ then return $\perp$
$(msg', b) \leftarrow \mathsf{BS.Sign}_2(state_j, msg)$
if $b = 1$ then $\mathcal{S} := \mathcal{S} \setminus \{j\}\,;\; k_2 := k_2 + 1$
return $msg'$
```

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

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 $\mathrm{UNF}^{\mathcal{A}}_{\mathsf{BS}}$ defined in Fig. 5 the adversary has access to two oracles $\textsc{Sign}_1$ and $\textsc{Sign}_2$ 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 $\textsc{Sign}_1$ 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 $\textsc{Sign}_2$ 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.

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

Blindness. Blindness requires that a signer cannot link a message/signature pair to a particular execution of the signing protocol. Formally, the adversary chooses two messages m_0 and m_1 and the experiment runs the signing protocol acting as the user with the adversary, first obtaining a signature σ_b on m_b and then σ_{1-b} on m_{1-b} for a random bit b. If both signatures are valid, the adversary is given (σ_0, σ_1) and must determine the value of b. A formal definition can be found in the full version [FPS19].

Blind Schnorr signatures. A blind signature scheme BlSch is obtained from the scheme Sch in Fig. 3 by replacing Sch.Sign with the interactive protocol

$$
\begin{array}{lcl}
\underline{\mathsf{BlSch.Sign}((p,\mathbb{G},G,\mathsf{H}),x)} & & \underline{\mathsf{BlSch.User}(((p,\mathbb{G},G,\mathsf{H}),X),m)} \\
r \leftarrow_{\$} \mathbb{Z}_p\,;\; R := rG & \xrightarrow{\quad R \quad} & \alpha,\beta \leftarrow_{\$} \mathbb{Z}_p \\
 & & R' := R + \alpha G + \beta X \\
 & & c' := \mathsf{H}(R',m) \\
 & \xleftarrow{\quad c \quad} & c := c' + \beta \bmod p \\
s := r + cx \bmod p & \xrightarrow{\quad s \quad} & \\
 & & \textbf{if } sG \neq R + cX \textbf{ then return } \perp \\
 & & s' := s + \alpha \bmod p \\
\textbf{return } 1 & & \textbf{return } \sigma := (R',s')
\end{array}
$$

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

specified in Fig. 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 $\mathsf{Sch.Ver}$:

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

Moreover, Schnorr signatures achieve perfect blindness [CP93].

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 $\mathrm{ROS}_{\mathsf{GrGen},\ell,\Omega}$ in Fig. 7, parameterized by a group generator GrGen,[9] an integer $\ell \geq 1$, and a set Ω (we omit GrGen and Ω from the notation when they are clear from context). The adversary $\mathcal{A}$ receives a prime p and has access to a random oracle $\mathsf{H}_{\mathrm{ros}}$ taking as input $(\vec{\rho}, aux)$ where $\vec{\rho} \in (\mathbb{Z}_p)^\ell$ and $aux \in \Omega$. Its goal is to find $\ell + 1$ distinct pairs $(\vec{\rho}_i, aux_i)_{i\in[\ell+1]}$ together with a solution $(c_j)_{j\in[\ell]}$ to the linear system $\sum_{j=1}^{\ell} \rho_{i,j} c_j \equiv_p \mathsf{H}_{\mathrm{ros}}(\vec{\rho}_i, aux_i)$, $i \in [\ell + 1]$.[10]

The lemma below, which refines Schnorr's observation [Sch01], shows how an algorithm $\mathcal{A}$ solving the ROS_ℓ problem can be used to break the one-more unforgeability of blind Schnorr signatures. The proof is deferred to the full version [FPS19] due to space constraints.

Lemma 1. *Let* GrGen *be a group generator. Let* $\mathcal{A}$ *be an algorithm for game* $\mathrm{ROS}_{\mathsf{GrGen},\ell,\Omega}$, *where* $\Omega = (\mathbb{Z}_p)^2 \times \{0,1\}^*$, *running in time at most* τ *and making*

[9] The group generator GrGen is only used to generate a prime p of length λ; the group $\mathbb{G}$ is not used in the game.

[10] 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.

```
Game $\mathrm{ROS}^{\mathcal{A}}_{\mathsf{GrGen},\ell,\Omega}(\lambda)$
$(p,\mathbb{G},G) \leftarrow \mathsf{GrGen}(1^\lambda)$ ; $\mathsf{T}_{\mathrm{ros}} := (\,)$
$\big((\vec{\rho}_i, aux_i)_{i\in[\ell+1]}, (c_j)_{j\in[\ell]}\big) \leftarrow \mathcal{A}^{\mathsf{H}_{\mathrm{ros}}}(p)$
// $\vec{\rho}_i = (\rho_{i,1},\dots,\rho_{i,\ell})$
return $\big(\forall\, i \neq i' \in [\ell+1] : (\vec{\rho}_i, aux_i) \neq (\vec{\rho}_{i'}, aux_{i'})$
    $\wedge\ \forall\, i \in [\ell+1] : \sum_{j=1}^{\ell} \rho_{i,j} c_j \equiv_p \mathsf{H}_{\mathrm{ros}}(\vec{\rho}_i, aux_i)\big)$

Oracle $\mathsf{H}_{\mathrm{ros}}(\vec{\rho}, aux)$
if $\mathsf{T}_{\mathrm{ros}}(\vec{\rho}, aux) = \bot$ then
    $\mathsf{T}_{\mathrm{ros}}(\vec{\rho}, aux) \leftarrow_{\$} \mathbb{Z}_p$
return $\mathsf{T}_{\mathrm{ros}}(\vec{\rho}, aux)$
```

Fig. 7. The ROS game, where $\mathsf{H}_{\mathrm{ros}} \colon (\mathbb{Z}_p)^\ell \times \Omega \to \mathbb{Z}_p$ is a random oracle.

at most q_{h} random oracle queries. Then there exists an (algebraic) adversary $\mathcal{B}$ running in time at most $\tau + O(\ell + q_{\mathrm{h}})$, making at most ℓ queries to Sign$_1$ *and* Sign$_2$ *and q_{h} random oracle queries, such that*

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

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

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

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

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

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

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

$$\textstyle\sum_{j=1}^{\ell} \rho_{i,j} c_j \equiv_p \mathsf{H}_{\mathrm{ros}}(\vec{\rho}_i, aux_i), \quad i \in [\ell+1] \tag{4}$$

with unknowns $c_1, \dots, c_\ell$ has a solution. Consider any subset of $\ell+1$ queries $(\vec{\rho}_i, aux_i)_{i\in[\ell+1]}$ 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. (4) has a solution if and only if the row vector $\vec{h} := (\mathsf{H}_{\mathrm{ros}}(\vec{\rho}_i, aux_i))^{\mathrm{T}}_{i\in[\ell+1]}$ 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,

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

which concludes the proof. □

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

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

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

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

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

We start with explaining the proof idea. Consider an adversary in the unforgeability game, let X be the public key and $R_1,\ldots,R_\ell$ be the elements returned by the oracle $\textsc{Sign}_1$ and let (R_i^*, s_i^*) be the adversary's forgeries on messages m_i^*. As $\mathcal{A}_\mathrm{alg}$ 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 $c_i^* = \mathsf{H}(R_i^*, m_i^*)$. Together, these yield

$$(c_i^* + \xi_i^*)X + \textstyle\sum_{j=1}^{\ell} \rho_{i,j}^* R_j = (s_i^* - \gamma_i^*)G \ , \tag{5}$$

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

However, the reduction also needs to simulate $\textsc{Sign}_2$ 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 win 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 $\textsc{Sign}_2$ oracle: on input (j, c_j), it simply returns $s_j \leftarrow \textsc{DLog}(R_j + c_j X)$.

But this means that in Eq. (5) 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. (5) yields

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

Thus, if for some i, $\chi_i \neq 0$, the reduction can compute $x = \log X$, and 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 (Fig. 7): A reduction to ROS would answer the adversary's queries $\mathsf{H}(R_{[\gamma,\xi,\vec{\rho}]}, m)$ by $\mathsf{H}_{\mathrm{ros}}(\vec{\rho}, (\gamma, \xi, m)) - \xi$; then $\chi_i = 0$ implies (recall that $c_i^* = \mathsf{H}(R_i^*, m^*)$)

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

meaning $\big((\vec{\rho}_i^*, (\gamma_i^*, \xi_i^*, m_i^*))_i, (c_j)_j\big)$ 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* $\mathsf{BlSch}[\mathsf{GrGen}]$ *running in time at most* τ *and making at most* q_{s} *queries to* $\textsc{Sign}_1$ *and* q_{h} *queries to the random oracle. Then there exists an adversary* $\mathcal{B}$ *that makes* exactly q_{s} *queries to* $\textsc{Sign}_1$ *and* q_{s} *queries to* $\textsc{Sign}_2$ *that do not return* $\perp$*, and returns* $q_{\mathrm{s}} + 1$ *forgeries, running in time at most* $\tau + O(q_{\mathrm{s}})$*, such that*

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

Proof. We construct the following adversary that plays game UNF (Fig. 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 $\textsc{Sign}_1$ 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 $\textsc{Sign}_2$ on some input $(j, *)$ and $\textsc{Sign}_2$ did not reply $\perp$. Let $(m_i^*, (R_i^*, s_i^*))_{i \in [n]}$ be $\mathcal{A}$'s output, for which we must have $k = |\mathcal{C}| < n$ when $\mathcal{A}$ wins.

$\mathcal{B}$ then makes $q_{\mathrm{s}} - q$ queries to $\textsc{Sign}_1$ to receive $R_{q+1}, \dots, R_{q_{\mathrm{s}}}$. Next, $\mathcal{B}$ completes all $q_{\mathrm{s}} - k$ open signing sessions for distinct messages by following the protocol in Fig. 6: for every $j \in \mathcal{S} := [1, \dots, q_{\mathrm{s}}] \setminus \mathcal{C}$, 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 $\mathsf{H}(R', m_j)$ to get c'_j, computes $c_j := c'_j + \beta_j \bmod p$ and queries (j, c_j) to $\textsc{Sign}_2$. Upon receiving s_j, $\mathcal{B}$ computes $s'_j := s_j + \alpha_j \bmod p$, which yields a signature (R'_j, s'_j) on message m_j.

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

Proof of Theorem 2. Let $\mathcal{A}_{\mathrm{alg}}$ be an algebraic adversary making at most q_s queries to $\textsc{Sign}_1$ and q_h random oracle queries. By the above lemma, we can assume that $\mathcal{A}_{\mathrm{alg}}$ makes exactly $\ell := q_s$ queries to $\textsc{Sign}_1$, closes all sessions, and returns $\ell + 1$ valid signatures. We proceed with a sequence of games defined in Fig. 8.

$\underline{\mathsf{Game}_0}$. The first game is the UNF game (Fig. 5) for scheme $\mathsf{BlSch}[\mathsf{GrGen}]$ played with $\mathcal{A}_{\mathrm{alg}}$ in the random oracle model. We have written the finalization of the game in a different but equivalent way. In particular, instead of checking that $(m_i^*, (R_i^*, s_i^*)) \neq (m_{i'}^*, (R_{i'}^*, s_{i'}^*))$ for all $i \neq i' \in [\ell + 1]$, we simply check that $(m_i^*, R_i^*) \neq (m_{i'}^*, R_{i'}^*)$. This is equivalent since for any pair (m, R), there is a single $s \in \mathbb{Z}_p$ such that (R, s) is a valid signature for m. Hence, if the adversary returns $(m_i^*, (R_i^*, s_i^*))$ and $(m_{i'}^*, (R_{i'}^*, s_{i'}^*))$ with $(m_i^*, R_i^*) = (m_{i'}^*, R_{i'}^*)$ and $s_i^* \neq s_{i'}^*$, at least one of the two forgeries is invalid. Thus,

$$\mathsf{Adv}^{\mathsf{game}_0}_{\mathcal{A}_{\mathrm{alg}}}(\lambda) = \mathsf{Adv}^{\mathrm{unf}}_{\mathsf{BlSch}[\mathsf{GrGen}], \mathcal{A}_{\mathrm{alg}}}(\lambda) \ . \tag{6}$$

$\underline{\mathsf{Game}_1}$. In Game_1, we make the following changes (which are analogous to those in the proof of Theorem 1). First, we introduce an auxiliary table U that for each query $\mathsf{H}(R_{[\gamma,\xi,\vec{\rho}]}, m)$ stores the representation $(\gamma, \xi, \vec{\rho})$ of R. 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 $\mathsf{H}(R^*_{i\,[\gamma_i,\xi_i,\vec{\rho}_i]}, m_i^*)$. Again, this does not change the output of the game, since in Game_0, the value $\mathsf{T}(R_i^*, m_i^*)$ would be randomly assigned when the game calls $\widetilde{\mathsf{H}}$ to check the signature. Finally, for each $i \in [\ell + 1]$, we retrieve $(\gamma_i^*, \xi_i^*, \vec{\rho}_i^*) := \mathsf{U}(R_i^*, m_i^*)$ (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 $\textsc{Sign}_2$ together with j and not answered by $\perp$.

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

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

```
Game_0 $\left(\mathrm{UNF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{BlSch}[\mathsf{GrGen}]}(\lambda)\right)$, $\boxed{\mathsf{Game}_1}$

$(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$x \leftarrow\!\!\$\ \mathbb{Z}_p$ ; $X := xG$ ; $k_1 := 0$ ; $k_2 := 0$ ;
$\mathcal{S} := \emptyset$ ; $\mathsf{T} := ()$ ; $\boxed{\mathsf{U} := ()}$
$\left(m_i^*, (R^*_{i\,[\gamma_i,\xi_i,\vec{\rho}_i]}, s_i^*)\right)_{i\in[\ell+1]}$
        $\leftarrow \mathcal{A}_{\mathrm{alg}}^{\mathsf{H},\textsc{Sign}_1,\textsc{Sign}_2}(p, \mathbb{G}, G, X)$
// $R_i^* = \gamma_i G + \xi_i X + \sum_{j=1}^{\ell} \rho_{i,j} R_j$
if $k_2 > \ell$ then return 0
if $\exists\, i \neq i' \in [\ell+1] : (m_i^*, R_i^*) = (m_{i'}^*, R_{i'}^*)$
    then return 0
┌
│ for $i = 1 \ldots \ell+1$ do
│     if $\mathsf{T}(R_i^*, m_i^*) = \bot$ then
│         $\mathsf{T}(R_i^*, m_i^*) \leftarrow\!\!\$\ \mathbb{Z}_p$
│         // $\mathsf{T}(R_i^*, m_i^*) := \mathsf{H}_{\mathrm{ros}}(\vec{\rho}_i, (\gamma_i, \xi_i, m_i^*)) - \xi_i$
│         $\mathsf{U}(R_i^*, m_i^*) := (\gamma_i, \xi_i, \vec{\rho}_i)$
└
for $i = 1 \ldots \ell+1$ do
    $c_i^* := \widetilde{\mathsf{H}}(R_i^*, m_i^*)$ // doesn't modify $\mathsf{T}$ in $\mathsf{Game}_1$
┌
│   $(\gamma_i^*, \xi_i^*, \vec{\rho}_i^{\,*}) := \mathsf{U}(R_i^*, m_i^*)$
│ if $\forall\, i \in [\ell+1] : \sum_{j=1}^{\ell} \rho_{i,j}^* c_j \equiv_p c_i^* + \xi_i^*$
│     then return 0                                          (I)
│   // $((\vec{\rho}_i^{\,*}, (\gamma_i^*, \xi_i^*, m_i^*))_{i\in[\ell+1]}, \vec{c})$ solves ROS
└
return $(\forall\, i \in [\ell+1] : s_i^* G = R_i^* + c_i^* X)$
```

```
Oracle $\widetilde{\mathsf{H}}(R, m)$

if $\mathsf{T}(R, m) = \bot$ then $\mathsf{T}(R, m) \leftarrow\!\!\$\ \mathbb{Z}_p$
return $\mathsf{T}(R, m)$
```

```
Oracle $\mathsf{H}(R_{[\gamma,\xi,\vec{\rho}]}, m)$

// $R = \gamma G + \xi X + \sum_{j=1}^{|\vec{\rho}|} \rho_j R_j$
if $\mathsf{T}(R, m) = \bot$ then
    $\mathsf{T}(R, m) \leftarrow\!\!\$\ \mathbb{Z}_p$
    // $\mathsf{T}(R, m) := \mathsf{H}_{\mathrm{ros}}(\vec{\rho}, (\gamma, \xi, m)) - \xi$
    $\boxed{\mathsf{U}(R, m) := (\gamma, \xi, \vec{\rho})}$
return $\mathsf{T}(R, m)$
```

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

$k_1 := k_1 + 1$ ; $r_{k_1} \leftarrow\!\!\$\ \mathbb{Z}_p$
$R_{k_1} := r_{k_1} G$    // $R_{k_1} \leftarrow \textsc{Chal}()$
$\mathcal{S} := \mathcal{S} \cup \{k_1\}$
return $(k_1, R_{k_1})$
```

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

if $j \notin \mathcal{S}$ then return $\bot$
$s_j := r_j + c_j x$    // $s_j \leftarrow \textsc{DLog}(R_j + c_j X)$
$\mathcal{S} := \mathcal{S} \setminus \{j\}$ ; $k_2 := k_2 + 1$
return $s_j$
```

Fig. 8. Games used in the proof of Theorem 2. Game_0 ignores all boxes. The light-gray comments in Game_1 and oracle H show how reduction $\mathcal{B}_{\mathrm{ros}}$ solves ROS; the comments in the $\textsc{Sign}$ oracles show how $\mathcal{B}_{\mathrm{omdl}}$ embeds its challenges and simulates Game_1.

$\mathsf{T}(R_i^*, m_i^*) = \bot$, reduction $\mathcal{B}_{\mathrm{ros}}$ assigns $\mathsf{T}(R_i^*, m_i^*) := \mathsf{H}_{\mathrm{ros}}(\vec{\rho}_i, (\gamma_i, \xi_i, m_i^*)) - \xi_i$. Since $\mathsf{H}_{\mathrm{ros}}$ returns uniformly random elements in $\mathbb{Z}_p$, the simulation is perfect.

If Game_1 aborts in line (I), $\mathcal{B}_{\mathrm{ros}}$ 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 $i \in [\ell+1]$, $c_i^* = \mathsf{H}_{\mathrm{ros}}(\vec{\rho}_i^{\,*}, (\gamma_i^*, \xi_i^*, m_i^*)) - \xi_i^*$. This is clearly the case if $\mathsf{T}(R_i^*, m_i^*) = \bot$ 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

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

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

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

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

Reduction to OMDL. In our last step, we construct an algorithm $\mathcal{B}_{\mathrm{omdl}}$ solving OMDL whenever $\mathcal{A}_{\mathrm{alg}}$ wins Game_1. Algorithm $\mathcal{B}_{\mathrm{omdl}}$, which has access to two oracles Chal and DLog (see Fig. 1) takes as input a group description $(p, \mathbb{G}, G)$, makes a first query $X \leftarrow$ Chal(), and runs $\mathcal{A}_{\mathrm{alg}}$ on input $(p, \mathbb{G}, G, X)$, simulating Game_1 as follows (cf. comments in Fig. 8). Each time $\mathcal{A}_{\mathrm{alg}}$ makes a Sign$_1$() query, $\mathcal{B}_{\mathrm{omdl}}$ queries its Chal oracle to obtain R_j. It simulates Sign$_2(j, c)$ without knowledge of x and r_j by querying $s_j \leftarrow$ DLog$(R_j + cX)$.

Assume that Game_1 returns 1, which implies that all forgeries (R_i^*, s_i^*) returned by $\mathcal{A}_{\mathrm{alg}}$ are valid. We show how $\mathcal{B}_{\mathrm{omdl}}$ solves OMDL. First, note that $\mathcal{B}_{\mathrm{omdl}}$ made exactly ℓ calls to its oracle DLog in total (since it makes exactly one call for each (valid) Sign$_2$ query made by $\mathcal{A}_{\mathrm{alg}}$).

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

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

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

$$R_i^* = \gamma_i^* G + \xi_i^* X + \textstyle\sum_{j=1}^{\ell} \rho_{i,j}^* R_j \ . \tag{9}$$

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

$$(c_i^* + \xi_i^*) X + \textstyle\sum_{j=1}^{\ell} \rho_{i,j}^* R_j = (s_i^* - \gamma_i^*) G \ . \tag{10}$$

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

$$R_j = s_j G - c_j X \ . \tag{11}$$

Injecting Eq. (11) in Eq. (10), we obtain

$$\left(c_i^* + \xi_i^* - \textstyle\sum_{j=1}^{\ell} \rho_{i,j}^* c_j\right) X = \left(s_i^* - \gamma_i^* - \textstyle\sum_{j=1}^{\ell} \rho_{i,j}^* s_j\right) G \ . \tag{12}$$

Since by Eq. (8) the coefficient in front of X is non-zero, this allows $\mathcal{B}_{\mathrm{omdl}}$ to compute $x := \log X$. Furthermore, from Eq. (11) we have $r_j := \log R_j = s_j - c_j x$

$\underline{\mathsf{CBlSch.Sign}((p, \mathbb{G}, G, \mathsf{H}), x)}$		$\underline{\mathsf{CBlSch.User}(((p, \mathbb{G}, G, \mathsf{H}), X), m)}$
$r_0, r_1 \leftarrow_{\$} \mathbb{Z}_p$		
$R_0 := r_0 G\, ; R_1 := r_1 G$	$\xrightarrow{R_0, R_1}$	$\alpha_0, \beta_0, \alpha_1, \beta_1 \leftarrow_{\$} \mathbb{Z}_p$
		$R'_0 := R_0 + \alpha_0 G + \beta_0 X$
		$R'_1 := R_1 + \alpha_1 G + \beta_1 X$
		$c'_0 := \mathsf{H}(R'_0, m)$
		$c'_1 := \mathsf{H}(R'_1, m)$
		$c_0 := c'_0 + \beta_0 \bmod p$
	$\xleftarrow{c_0, c_1}$	$c_1 := c'_1 + \beta_1 \bmod p$
$b \leftarrow_{\$} \{0, 1\}$		
$s := r_b + c_b x \bmod p$	$\xrightarrow{b, s}$	**if** $sG \neq R_b + c_b X$ **then return** $\perp$
		$s' := s + \alpha_b \bmod p$
return 1		**return** $\sigma := (R'_b, s')$

Fig. 9. The clause blind Schnorr signing protocol.

for all $j \in [\ell]$. By returning $(x, r_1, \ldots, r_\ell)$, $\mathcal{B}_{\text{omdl}}$ solves the OMDL problem whenever $\mathcal{A}_{\text{alg}}$ wins Game_1, which implies

$$\mathsf{Adv}^{\text{omdl}}_{\mathsf{GrGen}, \mathcal{B}_{\text{omdl}}}(\lambda) = \mathsf{Adv}^{\mathsf{game}_1}_{\mathcal{A}_{\text{alg}}}(\lambda) \ . \tag{13}$$

The theorem now follows from Equations (6), (7) and (13). $\square$

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 Fig. 10.

We start with explaining the modified signing protocol, formally defined in Fig. 9. In the first round the signer and the user execute two parallel runs of the blind signing protocol from Fig. 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. In the attack against the standard blind signature scheme (see Sect. 4.2), the adversary opens ℓ signing sessions, receiving $R_1, \ldots, R_\ell$, then searches a solution $\vec{c}$ to the ROS problem and closes the signing sessions by sending $c_1, \ldots, c_\ell$. Our modified signing protocol prevents this attack, as now for every opened session the adversary must *guess* which of the two challenges the signer will reply to. Only if all its guesses are correct is the attack successful. As the attack only works for large values of ℓ, this probability vanishes exponentially.

```
Game $\mathrm{MROS}^{\mathcal{A}}_{\mathsf{GrGen},\ell,\Omega}(\lambda)$
$(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$\mathsf{T}_{\mathrm{ros}} := (\,)$
$(\vec{\rho}_{i,0}, \vec{\rho}_{i,1}, aux_i)_{i\in[\ell+1]} \leftarrow \mathcal{A}^{\mathsf{H}_{\mathrm{ros}},\textsc{Select}}(p)$
  $/\!\!/\ \vec{\rho}_{i,b} = (\rho_{i,b,1}, \ldots, \rho_{i,b,\ell})$
return $\big(\forall\, i \neq i' : (\vec{\rho}_{i,0}, \vec{\rho}_{i,1}, aux_i) \neq (\vec{\rho}_{i',0}, \vec{\rho}_{i',1}, aux_{i'})$
  $\wedge\ \forall\, i \in [\ell+1] : \sum_{j=1}^{\ell} \rho_{i,b_j,j} c_j \equiv_p \mathsf{H}_{\mathrm{ros}}(\vec{\rho}_{i,0}, \vec{\rho}_{i,0}, aux_i)$
  $\wedge\ \forall\, i \in [\ell+1] \forall\, j \in [\ell] : \rho_{i,1-b_j,j} = 0\big)$

Oracle $\mathsf{H}_{\mathrm{ros}}(\vec{\rho}_0, \vec{\rho}_1, aux)$
if $\mathsf{T}_{\mathrm{ros}}(\vec{\rho}_0, \vec{\rho}_1, aux) = \bot$ then
  $\mathsf{T}_{\mathrm{ros}}(\vec{\rho}_0, \vec{\rho}_1, aux) \leftarrow\!\!\$\ \mathbb{Z}_p$
return $\mathsf{T}_{\mathrm{ros}}(\vec{\rho}_0, \vec{\rho}_1, aux)$

Oracle $\textsc{Select}(j, c'_0, c'_1)$
  $/\!\!/$ must be queried $\forall\, j \in [\ell]$
$b_j \leftarrow\!\!\$\ \{0,1\}\,;\ c_j := c'_{b_j}$
return $b_j$
```

Fig. 10. The modified ROS problem.

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 $(\mathrm{Run}_1 \vee \ldots \vee \mathrm{Run}_t)$. This decreases the probability that the user correctly guesses which challenges will be answered by the signer in ℓ concurrent sessions.

<u>The Modified ROS Problem.</u> Consider Fig. 10. The difference to the original ROS problem (Fig. 7) is that the queries to the $\mathsf{H}_{\mathrm{ros}}$ 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 $(\vec{\rho}_{i,0}, \vec{\rho}_{i,1}, aux_i)$, except that the ROS solution $c_1^*, \ldots, c_\ell^*$ is selected as follows: for every index $j \in [\ell]$ the adversary must query an additional oracle $\textsc{Select}(j, c_{j,0}, c_{j,1})$, which flips a random bit b_j and sets the j-th coordinate of the solution to $c_j^* := c_{j,b_j}$.

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 ρ values that correspond to the complement of the selected bit must be zero, that is, for all $i \in [\ell+1]$ and all $j \in [\ell]$: $\rho_{i,1-b_j,j} = 0$. The adversary thus must commit to the solution coordinate c_j^* before it learns b_j, which then restricts the format of its ρ values.

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

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

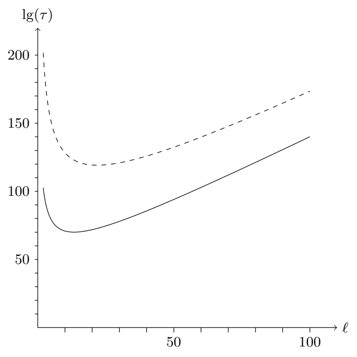

Fig. 11. Estimated complexity τ of conjectured best attack against the modified ROS problem as a function of parameter ℓ for $\lambda = 256$ (solid line) and $\lambda = 512$ (dashed line).

This estimated complexity is plotted for $\lambda \in \{256, 512\}$ in Fig. 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.

<u>Unforgeability of Clause Blind Schnorr Signatures.</u> We now prove that the Schnorr signature scheme from Fig. 3, with the signing algorithm replaced by the protocol in Fig. 9 is secure under the OMDL assumption for the underlying group and hardness of the modified ROS problem.

Theorem 3. *Let* GrGen *be a group generator. Let* $\mathcal{A}_{\mathrm{alg}}$ *be an algebraic adversary against the UNF security of the clause blind Schnorr signature scheme* $\mathsf{CBlSch}[\mathsf{GrGen}]$ *running in time at most* τ *and making at most* q_{s} *queries to* $\textsc{Sign}_1$ *and* q_{h} *queries to the random oracle. Then there exist an algorithm* $\mathcal{B}_{\mathrm{mros}}$ *for the MROS*$_{q_{\mathrm{s}}}$ *problem making at most* $q_{\mathrm{h}} + q_{\mathrm{s}} + 1$ *random oracle queries and an algorithm* $\mathcal{B}_{\mathrm{omdl}}$ *for the OMDL problem w.r.t.* GrGen *making at most* q_{s} *queries to its oracle* $\textsc{DLog}$*, both running in time at most* $\tau + O(q_{\mathrm{s}} + q_{\mathrm{h}})$*, such that*

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

The theorem follows by adapting the proof of Theorem 2; we therefore discuss the changes and refer to Fig. 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 $\textsc{Sign}_2$, the reduction $\mathcal{B}_{\mathrm{mros}}$ calls $\textsc{Select}(j, c_{j,0}, c_{j,1})$ to obtain bit b instead of choosing it itself. By definition, Game_1 aborts in line (I) if and only if $\mathcal{B}_{\mathrm{mros}}$ has found a solution for MROS.

The difference in the reduction to OMDL of the modified game is that the adversary can fail to solve MROS in two ways: (1) its values $((\rho_{i,b_j,j})_{i,j}, (c_j)_j)$ are not a ROS solution; in this case the reduction can solve OMDL as in the

```
$\mathsf{Game}_0\ \big(\mathrm{UNF}^{\mathcal{A}_{\mathrm{alg}}}_{\mathsf{CBlSch[GrGen]}}(\lambda)\big)$, $\boxed{\mathsf{Game}_1}$

$(p, \mathbb{G}, G) \leftarrow \mathsf{GrGen}(1^\lambda)$
$x \leftarrow_\$ \mathbb{Z}_p\,;\ X := xG$
$k_1 := 0\,;\ k_2 := 0\,;\ \mathcal{S} := \emptyset\,;\ \mathsf{T} := (\,)\,;\ \boxed{\mathsf{U} := (\,)}$
$(m_i^*, (R^*_{i\,[\gamma_i, \xi_i, \vec{\rho}_{i,0}, \vec{\rho}_{i,1}]}, s_i^*))_{i \in [\ell+1]}$
                    $\leftarrow \mathcal{A}_{\mathrm{alg}}^{\mathsf{H}, \mathrm{SIGN}_1, \mathrm{SIGN}_2}(p, \mathbb{G}, G, X)$
$/\!/\ R_i^* = \gamma_i G + \xi_i X + \sum \rho_{i,0,j} R_{j,0} + \sum \rho_{i,1,j} R_{j,1}$
if $k_2 > \ell$ then return $0$
if $\exists\, i \neq i' \in [\ell+1] : (m_i^*, R_i^*) = (m_{i'}^*, R_{i'}^*)$
    then return $0$

  for $i = 1 \ldots \ell+1$ do
    if $\mathsf{T}(R_i^*, m_i^*) = \bot$ then
      $\mathsf{T}(R_i^*, m_i^*) \leftarrow_\$ \mathbb{Z}_p$
    $/\!/\ \mathsf{T}(R_i^*, m_i^*) := \mathsf{H}_{\mathrm{ros}}(\vec{\rho}_{i,0}, \vec{\rho}_{i,1}, (\gamma_i, \xi_i, m_i^*)) - \xi_i$
      $\mathsf{U}(R_i^*, m_i^*) := (\gamma_i, \xi_i, \vec{\rho}_{i,0}, \vec{\rho}_{i,1})$
for $i = 1 \ldots \ell+1$ do
  $c_i^* := \widetilde{\mathsf{H}}(R_i^*, m_i^*)$ $/\!/$ does not modify $\mathsf{T}$ in $\mathsf{Game}_1$

    $(\gamma_i^*, \xi_i^*, \vec{\rho}^*_{i,0}, \vec{\rho}^*_{i,1}) := \mathsf{U}(R_i^*, m_i^*)$
  if $\forall\, i \in [\ell+1] : \sum_{j=1}^{\ell} \rho^*_{i,b_j,j} c_j \equiv_p c_i^* + \xi_i^*$
    $\wedge\, \forall\, i \in [\ell+1], \forall\, j \in [\ell] : \rho^*_{i,1-b_j,j} = 0$
     then return $0$                                        (I)
    $/\!/\ ((\vec{\rho}^*_{i,0}, \vec{\rho}^*_{i,1}, (\gamma_i^*, \xi_i^*, m_i^*))_{i \in [\ell+1]})$ solves MROS
return $(\forall\, i \in [\ell+1] : s_i^* G = R_i^* + c_i^* X)$
$/\!/$ {
  $\varphi_i := s_i^* - \gamma_i^* - \sum_{j=1}^{\ell} \rho^*_{i,b_j,j}\, s_j$
  if $\chi_i := c_i^* + \xi_i^* - \sum_{j=1}^{\ell} \rho^*_{i,b_j,j}\, c_j \not\equiv_p 0$
    $x := \chi_i^{-1} \varphi_i \bmod p$
    for $j \in [\ell] : r_{j,1-b_j} \leftarrow \mathrm{DLOG}(R_{j,1-b_j})$
  else if $\psi := \rho^*_{i,1-b_{\hat{\jmath}},\hat{\jmath}} \neq 0$ for some $i, \hat{\jmath}$
    for $j \neq \hat{\jmath} : r_{j,1-b_j} \leftarrow \mathrm{DLOG}(R_{j,1-b_j})$
    $r_{\hat{\jmath},1-b_{\hat{\jmath}}} := \psi^{-1}(\varphi_i - \sum_{j \neq \hat{\jmath}} \rho^*_{i,1-b_j,j}\, r_{j,1-b_j})$
    $x \leftarrow \mathrm{DLOG}(X)$
  for $j \in [\ell] : r_{j,b_j} := s_j - c_j x$
  $(x, r_{1,0}, \ldots, r_{\ell,0}, r_{1,1}, \ldots, r_{\ell,1})$ solves OMDL
```

```
Oracle $\widetilde{\mathsf{H}}(R, m)$

if $\mathsf{T}(R, m) = \bot$ then
  $\mathsf{T}(R, m) \leftarrow_\$ \mathbb{Z}_p$
return $\mathsf{T}(R, m)$
```

```
Oracle $\mathsf{H}(R_{[\gamma, \xi, \vec{\rho}_0, \vec{\rho}_1]}, m)$

$/\!/\ R = \gamma G + \xi X + \sum \rho_{0,j} R_{j,0}$
$/\!/\qquad\qquad\qquad + \sum \rho_{1,j} R_{j,1}$
if $\mathsf{T}(R, m) = \bot$ then
  $\mathsf{T}(R, m) \leftarrow_\$ \mathbb{Z}_p$
  $/\!/\ \mathsf{T}(R, m) :=$
  $/\!/\quad \mathsf{H}_{\mathrm{ros}}(\vec{\rho}_0, \vec{\rho}_1, (\gamma, \xi, m)) - \xi$
  $\boxed{\mathsf{U}(R, m) := (\gamma, \xi, \vec{\rho}_0, \vec{\rho}_1)}$
return $\mathsf{T}(R, m)$
```

```
Oracle $\mathrm{SIGN}_1()$

$k_1 := k_1 + 1$
$r_{k_1,0}, r_{k_1,1} \leftarrow_\$ \mathbb{Z}_p$
$R_{k_1,0} := r_{k_1,0} G$ $/\!/\ R_{k_1,0} \leftarrow \mathrm{CHAL}()$
$R_{k_1,1} := r_{k_1,1} G$ $/\!/\ R_{k_1,1} \leftarrow \mathrm{CHAL}()$
$\mathcal{S} := \mathcal{S} \cup \{k_1\}$
return $(k_1, R_{k_1,0}, R_{k_1,1})$
```

```
Oracle $\mathrm{SIGN}_2(j, c_{j,0}, c_{j,1})$

if $j \notin \mathcal{S}$ then return $\bot$
$b_j \leftarrow_\$ \{0, 1\}$
  $/\!/\ b_j \leftarrow \mathrm{SELECT}(j, c_{j,0}, c_{j,1})$
$c_j := c_{j,b_j}$
$s_j := r_{j,b_j} + c_j x$
  $/\!/\ s_j \leftarrow \mathrm{DLOG}(R_{j,b_j} + c_j X)$
$\mathcal{S} := \mathcal{S} \setminus \{j\}$
$k_2 := k_2 + 1$
return $(b_j, s_j)$
```

Fig. 12. Games used in the proof of Theorem 3. The comments in light gray show how $\mathcal{B}_{\mathrm{mros}}$ solves MROS; the dark comments show how $\mathcal{B}_{\mathrm{omdl}}$ solves OMDL.

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

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

(this corresponds to Eq. (9) in the proof of Theorem 2). Validity of the forgery implies $R_i^* = s_i^* G - c_i^* X$, which together with the above yields

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

(cf. Eq. (10)). 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

$$\begin{aligned}\big(c_i^* + \xi_i^* - \textstyle\sum_{j=1}^{\ell} \rho_{i,b_j,j}^* c_j\big)X \\ = \big(s_i^* - \gamma_i^* - \textstyle\sum_{j=1}^{\ell} \rho_{i,b_j,j}^* s_j\big)G - \sum_{j=1}^{\ell} \rho_{i,1-b_j,j}^* R_{j,1-b_j} \qquad (14)\end{aligned}$$

(which corresponds to Eq. (12) 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 case (2) defined above). In this case Game_1 does not abort and the OMDL reduction $\mathcal{B}_{\text{omdl}}$ must succeed. Since in this case the left-hand side of Eq. (14) is then 0, $\mathcal{B}_{\text{omdl}}$ can, after querying $\text{DLog}(R_{j,1-b_j})$ for all $j \neq j^*$, compute $\text{DLog}(R_{j^*,1-b_{j^*}})$, which breaks OMDL.

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

Fig. 12 presents the unforgeability game and Game_1, which aborts if the adversary solved MROS. The gray and dark gray comments also precisely define how a reduction $\mathcal{B}_{\text{mros}}$ solves MROS whenever Game_1 aborts in line (I), and how a reduction $\mathcal{B}_{\text{omdl}}$ solves OMDL whenever $\mathcal{A}_{\text{alg}}$ wins Game_1.

Blindness of the Clause Blind Schnorr Signature Scheme. Blindness of the "clause" variant in Fig. 9 follows via a hybrid argument from blindness of the standard scheme (Fig. 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

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

Fig. 13. The DDH problem.

both sessions terminate, the adversary is given the resulting signatures and must determine b.

In the blindness game for scheme CBlSch, the challenger runs *two* instances of the issuing protocol from BlSch for m_b of which the signer finishes one, as determined by its message (β_b, s_b) in the third round (β_b corresponds to b in Fig. 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 BlSch.

A technical detail is that the above argument only works when $\beta_0 = \beta_1$, as otherwise both reductions (between each original game and the hybrid game) abort one session and do not get any signatures from its challenger. The reductions thus guess the values β_0 and β_1 (and return a random bit if the guess turns out wrong). The hybrid game then replaces the β_0-th message of the first two and the β_1-th of the last two (as opposed to the ones underlined as above). Following this argument, in the full version [FPS19] we prove the following:

Theorem 4. *Let $\mathcal{A}$ be a p.p.t. adversary against blindness of the scheme* CBlSch. *Then there exist two p.p.t. algorithms $\mathcal{B}_1$ and $\mathcal{B}_2$ against blindness of* BlSch *such that*

$$\mathsf{Adv}^{\mathrm{blind}}_{\mathsf{CBlSch},\mathcal{A}}(\lambda) \le 4 \cdot \left(\mathsf{Adv}^{\mathrm{blind}}_{\mathsf{BlSch},\mathcal{B}_1}(\lambda) + \mathsf{Adv}^{\mathrm{blind}}_{\mathsf{BlSch},\mathcal{B}_2}(\lambda)\right) .$$

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

6 Schnorr-Signed ElGamal Encryption

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

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

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

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

SEG.Dec($sk, (X, C, R, s)$)
    $(p, \mathbb{G}, G, \mathsf{H}, y) := sk$
    if $sG \neq R + \mathsf{H}(X, C, R) \cdot X$ then
        return $\perp$
    return $M := C - yX$
```

Fig. 14. The Schnorr-Signed ElGamal PKE scheme $\mathsf{SEG}[\mathsf{GrGen}]$.

Under the decisional Diffie-Hellman (DDH) assumption (see Fig. 13), ciphertexts of different messages are computationally indistinguishable: replacing K by a random value K' makes the ciphertext C perfectly hide the message. In the AGM, ElGamal, viewed as a key-encapsulation mechanism (KEM) was shown to satisfy CCA1-security (where the adversary can only make decryption queries before seeing the challenge key) under a parametrized variant of DDH [FKL18].

The idea of *Schnorr-signed* ElGamal is to accompany the ciphertext by a proof of knowledge of the randomness $x = \log X$ used to encrypt, in particular, a Schnorr signature on the pair (X, C) under the public key X. The scheme is detailed in Fig. 14. (Note that we changed the argument order in the hash function call compared to Sect. 3 so that it is the same as in ciphertexts.)

The strongest security notion for PKE is indistinguishability of ciphertexts under adaptive chosen-ciphertext attack (IND-CCA2), where the adversary can query decryptions of ciphertexts of its choice even after receiving the challenge. The (decisional) game IND-CCA2 is defined in Fig. 15.

When ephemeral keys are hashed (that is, defined as $k := \mathsf{H}'(xY)$) and the scheme is viewed as a KEM, then CCA2-security can be reduced to the *strong* Diffie-Hellman (SDH) assumption[11] [ABR01, CS03] in the ROM. In the full version [FPS19] we show that when key hashing is applied to the Schnorr-signed ElGamal scheme from Fig. 14, then in the AGM+ROM we can directly reduce CCA2-security of the corresponding KEM to the DL assumption (Fig. 1); in particular, we do so using a *tight* security proof (note that SDH is equivalent to DL in the AGM [FKL18] but the reduction from DL to SDH is non-tight). Here we prove that the Schnorr-signed ElGamal PKE is IND-CCA2-secure in the AGM+ROM under the DDH assumption.

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

```
Game IND-CCA2$^{\mathcal{A}}_{\mathsf{PKE}}(\lambda)$
    $par \leftarrow \mathsf{PKE.Setup}(\lambda)$
    $(pk, sk) \leftarrow \mathsf{PKE.KeyGen}(par)$
    $b \leftarrow_\$ \{0,1\}$
    $b' \leftarrow \mathcal{A}^{\textsc{Enc},\textsc{Dec}}(pk)$
    return $(b = b')$

Oracle Enc$(m_0, m_1)$    // one time
    $c^* \leftarrow \mathsf{PKE.Enc}(pk, m_b)$ ; return $c^*$

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

Fig. 15. The IND-CCA2 security game for a PKE scheme PKE.

Theorem 5. *Let* GrGen *be a group generator. Let* $\mathcal{A}_{\mathrm{alg}}$ *be an algebraic adversary against the IND-CCA2 security of the Schnorr-signed ElGamal PKE scheme* $\mathsf{SEG}[\mathsf{GrGen}]$ *making at most* q_{d} *decryption queries and* q_{h} *queries to the random oracle. Then there exist two algorithms* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *solving respectively the DL problem and the DDH problem w.r.t.* GrGen*, such that*

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

We start with the proof idea. The full proof can be found in the full version [FPS19]. Let Y be the public key, let P_0 and P_1 denote the challenge plaintexts, and let $(X^* = x^*G, C^* = x^*Y + P_b, R^*, s^*)$ be the challenge ciphertext. Under the DDH assumption, given Y and X^*, the value x^*Y looks random. We can thus replace x^*Y by a random group element Z^*, which perfectly hides P_b and leads to a game where the adversary gains no information about the challenge bit b.

It remains to show how the reduction can simulate the game without knowledge of $\log X^*$ (needed to sign the challenge ciphertext) and $\log Y$ (needed to answer decryption queries). The Schnorr signature under X^* contained in the challenge ciphertext can be simulated by programming the random oracle H as for Theorem 1.

Decryption queries leverage the fact that the Schnorr signature contained in a queried ciphertext (X, C, R, s) proves knowledge of x with $X = xG$. Thus, intuitively, the reduction should be able to answer a query by extracting x and returning $M = C - xY$. However, this extraction is a lot trickier than in the proof of Theorem 1: During the game the adversary obtains group elements Y, X^*, C^*, and R^*, as well as the answers $M_1, \ldots, M_{q_{\mathrm{d}}}$ to its queries to Dec. The adversary's representations of group elements can thus depend on all these elements. In particular, since Dec on input $(X, C, \ldots)$ computes $M \coloneqq C - yX$, by successive calls to Dec, the adversary can obtain arbitrary powers of y.

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

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

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

$$X = \gamma_x G + \xi_x X^* + \textstyle\sum_{i=1}^{q_d+1} \upsilon_x^{(i)} y^i G + \sum_{i=1}^{q_d+1} \zeta_x^{(i)} x^* y^i G \ , \qquad R = \gamma_r G + \dots \quad (15)$$

We show that each query falls into one of three categories:
(1) The choice of $c = \mathsf{H}(X, C, R)$ was unlucky, which only happens with negligible probability.
(2) The representation of X is independent of Y, that is, $X = \gamma_x G + \xi_x X^*$. Then xY (and hence the answer $M = C - xY$ to the query) can be computed as $xY \coloneqq \gamma_x Y + \xi_x Z^*$ (where $Z^* := x^* Y$ is known by the reduction).
(3) Otherwise we show that the adversary has computed $\log Y$: If the DEC query was valid then $sG = R + cX$, which, by plugging in the representations (15) yields

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

If $\beta^{(i)} \equiv_p 0$ for all i, we are in case (2). If $\beta^{(j)} \not\equiv_p 0$ for some j and $\alpha^{(i)} \equiv_p 0$ for all i, then $c \equiv_p -(\upsilon_r^{(j)} + x^* \zeta_r^{(j)}) \cdot (\beta^{(j)})^{-1}$ was an unlucky choice (made *after* the adversary chose its representations from (15)) (case (1)). Otherwise $\alpha^{(j)} \equiv_p 0$ for some j and

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

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

ACKNOWLEDGEMENTS. The first author is supported by the Vienna Science and Technology Fund (WWTF) through project VRG18-002. Parts of this work were done while he was visiting the Simons Institute for the Theory of Computing. This is work is funded in part by the *MSR–Inria Joint Centre*.

References

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

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

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

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

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

[BDL+12] D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, and B.-Y. Yang. High-speed high-security signatures. *Journal of Cryptographic Engineering*, 2(2):77–89, September 2012.

[BDN18] D. Boneh, M. Drijvers, and G. Neven. Compact multi-signatures for smaller blockchains. In *ASIACRYPT 2018, Part II*, pages 435–464.

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

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

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

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

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

[BNPS03] M. Bellare, C. Namprempre, D. Pointcheval, and M. 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.

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

[Bol03] A. Boldyreva. Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In *PKC 2003*, pages 31–46.

[BP02] M. Bellare and A. Palacio. GQ and Schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *CRYPTO 2002*, pages 162–177.

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

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

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

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

[Cha82] D. Chaum. Blind signatures for untraceable payments. In *CRYPTO'82*, pages 199–203.

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

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

[CP93] D. Chaum and T. P. Pedersen. Wallet databases with observers. In *CRYPTO'92*, pages 89–105.

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

[ElG85] T. ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, 31(4):469–472, 1985.

[FHS15] G. Fuchsbauer, C. Hanser, and D. Slamanig. Practical round-optimal blind signatures in the standard model. In *CRYPTO 2015, Part II*, pages 233–253.

[FJS19] N. Fleischhacker, T. Jager, and D. Schröder. On tight security proofs for Schnorr signatures. *Journal of Cryptology*, 32(2):566–599, April 2019.

[FKL18] G. Fuchsbauer, E. Kiltz, and J. Loss. The algebraic group model and its applications. In *CRYPTO 2018, Part II*, pages 33–62.

[FOO93] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *AUSCRYPT'92*, pages 244–251.

[FPS19] G. Fuchsbauer, A. Plouviez, and Y. Seurin. Blind schnorr signatures and signed elgamal encryption in the algebraic group model. Cryptology ePrint Archive, Report 2019/877, 2019. https://eprint.iacr.org/2019/877.

[FPV09] G. Fuchsbauer, D. Pointcheval, and D. Vergnaud. Transferable constant-size fair e-cash. In *CANS 09*, pages 226–247.

[FS10] M. Fischlin and D. Schröder. On the impossibility of three-move blind signature schemes. In *EUROCRYPT 2010*, pages 197–215.

[Fuc11] G. Fuchsbauer. Commuting signatures and verifiable encryption. In *EUROCRYPT 2011*, pages 224–245.

[GBL08] S. Garg, R. Bhaskar, and S. V. Lokam. Improved bounds on security reductions for discrete log based signatures. In *CRYPTO 2008*, pages 93–107.

[GG14] S. Garg and D. Gupta. Efficient round optimal blind signatures. In *EUROCRYPT 2014*, pages 477–495.

[GRS$^+$11] S. Garg, V. Rao, A. Sahai, D. Schröder, and D. Unruh. Round optimal blind signatures. In *CRYPTO 2011*, pages 630–648.

[HHK10] J. Herranz, D. Hofheinz, and E. Kiltz. Some (in)sufficient conditions for secure hybrid encryption. *Inf. Comput.*, 208(11):1243–1257, 2010.

[HKL19] E. Hauck, E. Kiltz, and J. Loss. A modular treatment of blind signatures from identification schemes. In *EUROCRYPT 2019, Part III*, pages 345–375.

[Jak98] M. Jakobsson. A practical mix. In *EUROCRYPT'98*, pages 448–461.

[MPSW19] G. Maxwell, A. Poelstra, Y. Seurin, and P. Wuille. Simple Schnorr multi-signatures with applications to Bitcoin. *Des. Codes Cryptogr.*, 87(9):2139–2164, 2019.

[MS12] L. Minder and A. 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] J. 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] I. Nikolic and Y. Sasaki. Refinements of the k-tree algorithm for the generalized birthday problem. In *ASIACRYPT 2015, Part II*, pages 683–703.

[OO92] T. Okamoto and K. Ohta. Universal electronic cash. In *CRYPTO'91*, pages 324–337.

[Pas11] R. Pass. Limits of provable security from standard assumptions. In *43rd ACM STOC*, pages 109–118.

[PS96a] D. Pointcheval and J. Stern. Provably secure blind signature schemes. In *ASIACRYPT'96*, pages 252–265.

[PS96b] D. Pointcheval and J. Stern. Security proofs for signature schemes. In *EUROCRYPT'96*, pages 387–398.

[PS00] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *Journal of Cryptology*, 13(3):361–396, June 2000.

[PV05] P. Paillier and D. Vergnaud. Discrete-log-based signatures may not be equivalent to discrete log. In *ASIACRYPT 2005*, pages 1–20.

[Sch90] C.-P. Schnorr. Efficient identification and signatures for smart cards. In *CRYPTO'89*, pages 239–252.

[Sch91] C.-P. Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4(3):161–174, January 1991.

[Sch01] C.-P. Schnorr. Security of blind discrete log signatures against interactive attacks. In *ICICS 01*, pages 1–12.

[Seu12] Y. Seurin. On the exact security of Schnorr-type signatures in the random oracle model. In *EUROCRYPT 2012*, pages 554–571.

[SG02] V. Shoup and R. Gennaro. Securing threshold cryptosystems against chosen ciphertext attack. *Journal of Cryptology*, 15(2):75–96, March 2002.

[Sho97] V. Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT'97*, pages 256–266.

[SJ99] C.-P. Schnorr and M. Jakobsson. Security of discrete log cryptosystems in the random oracle and the generic model, 1999. Available at https://core.ac.uk/download/pdf/14504220.pdf.

[SJ00] C.-P. Schnorr and M. Jakobsson. Security of signed ElGamal encryption. In *ASIACRYPT 2000*, pages 73–89.

[ST13] Y. Seurin and J. Treger. A robust and plaintext-aware variant of signed ElGamal encryption. In *CT-RSA 2013*, pages 68–83.

[TY98] Y. Tsiounis and M. Yung. On the security of ElGamal based encryption. In *PKC'98*, pages 117–134.

[Wag02] D. Wagner. A generalized birthday problem. In *CRYPTO 2002*, pages 288–303.

[Wik08] D. Wikström. Simplified submission of inputs to protocols. In *SCN 08*, pages 293–308.

[Wui18] P. Wuille. Schnorr signatures for secp256k1. Bitcoin Improvement Proposal, 2018. See https://github.com/sipa/bips/blob/bip-schnorr/bip-schnorr.mediawiki.