Designated Verifier Signature Schemes: Attacks, New Security Notions and A New Construction

Helger Lipmaa[1], Guilin Wang[2], Feng Bao[2]

[1] Cybernetica AS and University of Tartu, Estonia
[2] Institute for Infocomm Research (I^2R), Singapore

Abstract. We show that the signer can abuse the disavowal protocol in the Jakobsson-Sako-Impagliazzo designated-verifier signature scheme. In addition, we identify a new security property—non-delegatability—that is essential for designated-verifier signatures, and show that several previously proposed designated-verifier schemes are delegatable. We give a rigorous formalisation of the security for designated-verifier signature schemes, and propose a new and efficient designated-verifier signature scheme that is provably unforgeable under a tight reduction to the Decisional Diffie-Hellman problem in the non-programmable random oracle model, and non-delegatable under a loose reduction in the programmable random oracle model. As a direct corollary, we also get a new efficient conventional signature scheme that is provably unforgeable under a tight reduction to the Decisional Diffie-Hellman problem in the non-programmable random oracle plus common reference string model.
Keywords. Designated verifier signature scheme, non-delegatability, non-programmable random oracle model, signature scheme.

1 Introduction

In 1996, Jakobsson, Sako and Impagliazzo introduced the concept of designated-verifier signature (DVS) schemes [JSI96]. A DVS scheme makes it possible for a prover Signy to convince a designated verifier Desmond that she has signed a statement so that Desmond cannot transfer the signature to a third party Trevor. This is achieved since Desmond himself can efficiently simulate signatures that are indistinguishable from Signy's signatures. Moreover, in a disavowable DVS scheme, Signy can prove to Trevor that a simulated signature was not created by Desmond, while she can not disavow her own signatures. This is possible only when Signy's and Desmond's signatures are computationally but not perfectly indistinguishable.

We point out weaknesses in the designated-verifier signature schemes of [JSI96,SKM03,SBWP03,SWP04,LV04]. Of these schemes, the JSI scheme from [JSI96] is the only disavowable DVS scheme. However, we show that in the JSI scheme, a malicious Signy can generate signatures exactly from the same distribution as Desmond and thus the JSI scheme is perfectly non-transferable and thus also not disavowable. Our attack against the DVS schemes from [SKM03,SBWP03,SWP04,LV04] is not an attack according to the definitions of the designated verifier signatures in these papers, although it is an attack according to the original informal definition of [JSI96]: namely, we show that Signy can delegate her signing ability—with respect to a fixed

designated verifier Desmond—to a third party Trevor, without revealing her secret key or making it possible for Trevor to sign with respect to other designated verifiers. This delegation property, while desirable in some settings (e.g., proxy DVS schemes), is extremely undesirable in many other settings and must therefore be considered as a serious weakness of a DVS scheme.

By pointing out the described flaws in these designated verifier signature scheme, we arrive to a stronger security notion for DVS that includes two novel requirements: (a) most importantly, *non-delegatability*: there exists an efficient knowledge extractor that can extract either Signy's secret key or Desmond's secret key, when given oracle access to an adversary who can create valid signatures with a high probability (this property is not shared by the DVS schemes from [SKM03,SBWP03,SWP04,LV04]), and (b) secure disavowability: if the DVS scheme has a disavowal protocol, it must be the case that Signy cannot disavow signatures, given by herself (this property is not shared by the DVS scheme from [JSI96]).

Non-delegatability of a DVS means that a valid designated-verifier signature constitutes a proof of knowledge of either Signy's or Desmond's secret key. Now, for conventional signatures, ability to sign is conceptually equal to the knowledge of the secret key. Therefore, a valid signature does not be a proof of knowledge. Now, as it is known from [KW03], one can construct conventional signature schemes whose unforgeability is proven by giving a tight reduction to an underlying cryptographic problem; this is achieved by specially avoiding the use of proofs of knowledge. However, in the case of a DVS scheme, we can also avoid proofs of knowledge in the proof of unforgeability, but not in the proof of non-delegatability. Therefore, even if we have a proof that a DVS scheme is unforgeable (w.r.t. any verifier), we cannot directly derive from that that this scheme is also non-delegatable. Therefore, in some sense, a (non-delegatable) DVS is a more "complex" notion than a conventional signature scheme.

It is not difficult to show that the DVS scheme from [JSI96] is secure—more precisely, unforgeable, non-delegatable, computationally non-transferable and securely disavowable—after a trivial fix of just adding some additional variables under the used hash value, by following the usual proof of knowledge methodology. We do not present these proofs in this paper: while it is straightforward to prove these results, the corresponding proofs do not really give an insight to the just pointed out difference between unforgeability and non-delegatability.

Instead, we propose a new DVS scheme, DVS-KW, based on the provably secure signature scheme of Katz and Wang [KW03], where the signer presents a designated-verifier proof that his public key is a Decisional Diffie-Hellman (DDH) tuple. We prove that DVS-KW is unforgeable by providing a tight reduction to the underlying cryptographic problem (DDH) in the non-programmable random oracle (NPRO) model. The NPRO model is is known to be strictly weaker than the random oracle (RO) model [Nie02] and thus the unforgeability of DVS-KW in the NPRO model is interesting by itself, especially since the unforgeability proof of the original Katz-Wang signature scheme relies heavily on the programmability of the random oracle. We also prove non-delegatability of DVS-KW, though this proof is in the programmable random oracle model and has a larger security degradation due to the involved proof-of-knowledge property. More precisely, we show that if some forger can create valid signatures with

probability $\varepsilon > \kappa$ where κ is the knowledge error, then there exists a knowledge extractor that extracts one of the two secret keys in time, dominated by $56/\kappa$ oracle queries to the forger.

DVS-KW can be seen as a proof of concept, showing how to design DVS schemes that have a tight reduction in the unforgeability proof and are still non-delegatable. Moreover, DVS-KW is more efficient than the JSI scheme from [JSI96], and DVS-KW does not allow the signer to disavow simulated signatures; the latter property makes DVS-KW attractive in many applications. (Recall also that we broke the disavowability of the JSI scheme.) At this moment, the most efficient secure disavowable designated-verifier signature scheme seems to be the corrected JSI DVS scheme, while the most efficient secure designated-verifier signature scheme seems to be the DVS-KW scheme.

We also show the existence of an efficient conventional signature scheme that is unforgeable under a tight reduction to the Decisional Diffie-Hellman problem in the NPRO+CRS (common reference string) model. In this model, all parties will additionally have access to the common reference string that corresponds to Desmond's public key in DVS-KW. The importance of this result is that signature schemes, secure in the plain model, are considerably slower than this scheme and/or are secure under incompatible and often less studied assumptions. Therefore, if one wants to avoid the programmable random oracle model—where one can construct very efficient signature schemes—one might want to use the new scheme.

2 Preliminaries

Let $\mathcal{G}_q$ be a finite, cyclic group of prime order q in which the group operation is represented multiplicatively; furthermore, let g be a generator of $\mathcal{G}$. The most common setting is as follows: let p, q be two large primes such that $q|(p-1)$, then $\mathcal{G}_q$ a multiplicative subgroup of $\mathbb{Z}_p^*$ of order q, and g a generator of $\mathcal{G}_q$. Other settings (e.g., using elliptic curves) are possible. A distinguishing algorithm $\mathcal{A}$ is said to (τ, ε)-break DDH (Decisional Diffie-Hellman) in group $\mathcal{G}_q$ if $\mathcal{A}$ runs in time at most τ and furthermore $\mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{G}}(\mathcal{A}) := |\Pr[x, y, z \leftarrow_r \mathbb{Z}_q : \mathcal{A}(g, g^x, g^y, g^z) = 1] - \Pr[x, y \leftarrow_r \mathbb{Z}_q : \mathcal{A}(g, g^x, g^y, g^{xy}) = 1]| \geq \varepsilon$, where the probability is taken over the choice of random variables and the coin tosses of $\mathcal{A}$. We say that $\mathcal{G}$ is a (τ, ε)-DDH group if no algorithm (τ, ε)-breaks DDH in $\mathcal{G}$.

A *designated-verifier signature scheme* [JSI96] is a tuple of probabilistic algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Simul}, \mathsf{Vrfy})$ over a message space $\mathcal{M}$, such that: (a) The key-generation algorithm Gen outputs a public key pk and a secret key sk; (b) The signing algorithm Sign takes as input signer's secret key sk_S, designated verifier's public key pk_D and a message $m \in \mathcal{M}$ and returns signature σ; (c) The simulation algorithm Simul takes as input signer's public key pk_S, designated verifier's secret key sk_D and a message $m \in \mathcal{M}$ and returns signature σ; (d) Verification algorithm Vrfy takes as input signer's public key pk_S, designated verifier's public key pk_D, a message $m \in \mathcal{M}$, and a signature σ and returns accept or reject. In some of the existing designated verifier schemes, the verification algorithm must have access to the designated verifier's secret key sk_D. We call such a designated-verifier signature scheme *privately verifiable*. We make the standard correctness requirement: for all $(\mathsf{sk}_S, \mathsf{pk}_S)$ and $(\mathsf{sk}_D, \mathsf{pk}_D)$ output by Gen and for

all $m \in \mathcal{M}$ we have $\mathsf{Vrfy}_{\mathsf{pk}_S,\mathsf{pk}_D}(\mathsf{Sign}_{\mathsf{sk}_S,\mathsf{pk}_D}(m)) = \mathsf{Vrfy}_{\mathsf{pk}_S,\mathsf{pk}_D}(\mathsf{Sign}_{\mathsf{sk}_D,\mathsf{pk}_S}(m)) = \mathsf{accept}$. We say that a signature σ is *valid* if $\mathsf{Vrfy}_{\mathsf{pk}_S,\mathsf{pk}_D}(\sigma) = \mathsf{accept}$.

3 Previous DVS Schemes and Their Security

Jakobsson-Sako-Impagliazzo Disavowable DVS Scheme [JSI96]. Let p, q and $\mathcal{G}_q$ be as described in Sect. 2. Assume that Signy and Desmond have the Diffie-Hellman key pairs $(x_S, y_S = g^{x_S} \mod p)$ and $(x_D, y_D = g^{x_D} \mod p)$, respectively. Assume that H_q is a random oracle mapping to $\mathbb{Z}_q$. (Note: If $m \notin \mathbb{Z}_p$ then m must be hashed by using a full-domain hash, modelled by a random oracle. We will ignore this issue throughout this paper.) In $\mathsf{Sign}_{\mathsf{sk}_S,\mathsf{pk}_D}(m)$, Signy sets $s \leftarrow m^{x_S} \mod p$, selects three random numbers $w, t, r \leftarrow_r \mathbb{Z}_q$, and computes $G \leftarrow g^r \mod p$, $M \leftarrow m^r \mod p$, $h \leftarrow H_q(g^w y_D^t \mod p, G, M)$ and $z \leftarrow r + (h + w)x_S \mod q$. Then, Signy sends the signature $\sigma := (s, P)$, where $P = (w, t, G, M, z)$, to the designated verifier, Desmond. In $\mathsf{Simul}_{\mathsf{sk}_D,\mathsf{pk}_S}(m, s)$, by selecting three random numbers $z, \alpha, \beta \leftarrow_r \mathbb{Z}_q$, Desmond creates $P = (w, t, G, M, z)$, for any message m and any $s \leftarrow \mathcal{G}_q$, as follows: $(G, M) \leftarrow (g^z y_S^{-\beta} \mod p, m^z s^{-\beta} \mod p)$, $h \leftarrow H_q(g^\alpha \mod p, G, M)$, $w \leftarrow \beta - h \mod q$, $t \leftarrow (\alpha - w)x_D^{-1} \mod q$. He sets $\sigma \leftarrow (s, P)$. In $\mathsf{Vrfy}_{\mathsf{pk}_S,\mathsf{pk}_D}(m; s, w, t, G, M, z)$, the verifier computes $h \leftarrow H_q(g^w y_D^t \mod p, G, M)$ and checks whether $G = g^z y_S^{-(h+w)} \mod p$ and $M = m^z s^{-(h+w)} \mod p$. The JSI scheme can be made more communication-efficient by transferring h (instead of G and M) to Desmond. Then the verifier must check that $h = H_q(g^w y_D^t \mod p, g^z \cdot y_S^{-(h+w)} \mod p, m^z \cdot s^{-(h+w)} \mod p)$. This version is security-wise equivalent to the original scheme but somewhat more efficient.

Our Attack: First, a honest Signy generates valid signatures only for $s = m^{x_S}$ while Desmond can generate valid signatures for any $s \in \mathbb{Z}_p^*$. That is, knowing x_S, a honest Signy generates valid designated-verifier proof P only for $s = m^{x_S}$, while knowing x_D, Desmond generates valid designated-verifier proofs P for any $s \in \mathbb{Z}_p^*$. Thus it suffices to have a disavowal where Signy proves in non-interactive zero-knowledge that $s \neq m^{x_S}$. Next, we show that Signy can also compute valid signatures for any $\bar{s} \in \mathbb{Z}_p^*$, therefore, Signy can create signatures from the same distribution as Desmond and thus, the JSI scheme is perfectly non-transferable. This means that there exists no disavowal protocol for the JSI scheme at all.

Here is how Signy does it. Signy computes a signature $(\bar{s}; w, t, G, \bar{M}, z)$ for a message m, with $\bar{s} \neq m^{x_S}$, as follows. She selects four random numbers $w, t, r, \bar{r} \leftarrow_r \mathbb{Z}_q$ and then sets $c \leftarrow g^w y_D^t \mod p$, $G \leftarrow g^r \mod p$, $\bar{M} \leftarrow m^{\bar{r}} \mod p$, $h \leftarrow H_q(c, G, \bar{M})$, $z \leftarrow r + (h + w)x_S \mod q$ and $\bar{s} \leftarrow m^{x_S} \cdot m^{(r-\bar{r})/(h+w) \mod q} \mod p$. After that, Signy sends the message-signature pair $(m, \bar{s})$ with $\bar{\sigma} = (\bar{s}, P = (w, t, G, \bar{M}, z))$, to Desmond. Clearly, $\mathsf{Vrfy}_{\mathsf{pk}_S,\mathsf{pk}_D}(m, \bar{\sigma}) = \mathsf{accept}$ so Desmond will believe that $\bar{s}$ is Signy's signature for message m. In later disputes, however, Signy can convince a third party (e.g., a judge) that $\bar{s}$ was simulated by Desmond, by using a standard disavowal protocol to show that $\log_g y_S \neq \log_m \bar{s}$. This attack does not result in a signature forgery, it just shows that the JSI scheme is not disavowable.

There are two intuitive countermeasures to avoid this attack. First, Signy provides an additional proof of knowledge that $\log_m M = \log_g G$. However, this increases the signature length. Second, include s (together with pk_S and pk_D) to the input of the hash function. This turns out to be is sufficient, related discussion can be found in Sect. 6.

Saeednia-Kremer-Markowitch privately verifiable DVS Scheme [SKM03]. Let p, q and $\mathcal{G}_q$ be as defined in Sect. 2. In the SKM scheme, Signy and Desmond have the Diffie-Hellman key pairs $(x_S, y_S = g^{x_S} \mod p)$ and $(x_D, y_D = g^{x_D} \mod p)$, respectively. Assume that $H_q(\cdot)$ is a random oracle mapping to $\mathbb{Z}_q$. In $\mathsf{Sign}_{\mathsf{sk}_S, VK_D}(m)$, Signy selects two random numbers $k \leftarrow_r \mathbb{Z}_q$, $t \leftarrow_r \mathbb{Z}_q^*$, and then computes the signature $\sigma = (h, d, t)$ by setting $c \leftarrow y_D^k \mod p$, $h \leftarrow H_q(m, c)$ and $d \leftarrow kt^{-1} - h \cdot x_S \mod q$. In $\mathsf{Simul}_{\mathsf{sk}_D, \mathsf{pk}_S}(m)$, Desmond picks two random numbers $\bar{d} \leftarrow_r \mathbb{Z}_q$, $\bar{r} \leftarrow_r \mathbb{Z}_q^*$, and then computes $\sigma = (h, d, t)$ as follows: for $c \leftarrow g^{\bar{d}} y_S^{\bar{r}} \mod p$, set $h \leftarrow H_q(m, c)$, $d \leftarrow h \cdot \bar{d} \cdot \bar{r}^{-1} \mod q$ and $t \leftarrow \bar{r} \cdot (x_D \cdot h)^{-1} \mod q$. In $\mathsf{Vrfy}_{\mathsf{pk}_S, \mathsf{sk}_D}(m; h, d, t)$, the verifier accepts iff $h = H_q(m, (g^d y_S^h)^{t x_D} \mod p)$.

Our attack: We show that the knowledge of $y_{SD} := g^{x_S \cdot x_D} \mod p$ is sufficient to generate both a valid signature and to verify it, and therefore, this scheme is delegatable (both in the sense of signing and verifying). On the one hand, given y_{SD}, one can verify whether a message-signature pair (m, h, d, t) is valid for the designated verifier Desmond by checking whether $h = H_q(m, (y_D^d y_{SD}^h)^t \mod p)$. On the other hand, given y_{SD}, anybody can produce a valid designated-verifier signature $\sigma = (h, d, t)$ for any message m by selecting two random numbers $\bar{d} \leftarrow_r \mathbb{Z}_q$, $\bar{r} \leftarrow_r \mathbb{Z}_q^*$, and then setting $c \leftarrow y_D^{\bar{d}} y_{SD}^{\bar{r}} \mod p$, $h \leftarrow H_q(m, c)$, $d \leftarrow h \cdot \bar{d} \cdot \bar{r}^{-1} \mod q$ and $t \leftarrow \bar{r} h^{-1} \mod q$. The resulting signature $\sigma = (h, d, t)$ is accepted by the verifier, since $(g^d y_S^h)^{t x_D} \mod p = (g^{h \cdot \bar{d} \cdot \bar{r}^{-1}} y_S^h)^{\bar{r} \cdot h^{-1} x_D} = y_D^{\bar{d}} y_{SD}^{\bar{r}} \mod p = c$. In particular, this means that this scheme is perfectly non-transferable. (The authors of [SKM03] get the same result by a complicated analysis of probability theory.) It might again seem that one can remedy this situation by including t under the hash. However, in the simulation, Desmond needs to choose t after fixing h, and therefore this fix does not work.

Steinfeld-Wang-Pieprzyk DVS. Assume again that p, q and $\mathcal{G}_q$ are as defined in Sect. 2. At least the first privately-verifiable DVS scheme, $\mathsf{SchUDVS}_1$, from [SWP04] is also delegatable. Recall that there, a designated-verifier signature of message m is equal to (r, u, K), for $r \leftarrow H_q(m, u)$, and the verifier accepts only when $K = (u \cdot y_S^r)^{x_D} \mod p$. But if Signy publishes $y_{SD} \leftarrow y_D^{x_S}$, anybody can produce valid signatures by setting $u \leftarrow g^k \mod p$ and $K \leftarrow y_D^k \cdot y_{SD}^r \mod p$ for random $k \leftarrow_r \mathbb{Z}_q$. Thus, also this scheme is delegatable.

Steinfeld-Bull-Wang-Pieprzyk and Laguillaumie-Vergnaud DVS. The SBWP designated-verifier signature scheme from [SBWP03] is based on a group pair $(\mathcal{G}_1, \mathcal{G}_2)$ with bilinear pairing $\langle \cdot, \cdot \rangle$ from $\mathcal{G}_1^2$, with $\mathrm{ord}\, \mathcal{G}1 = q$, to $\mathcal{G}_2$. (See [SBWP03] for more details.) Let m be the message to be signed, $H_{\mathcal{G}_1}$—a random oracle with outputs from $\mathcal{G}_1$, and let g be a generator of $\mathcal{G}_1$. Assume that Signy's key-pair is $(x_S, y_S = g^{x_S})$ and that Desmond's key-pair is $(x_D, y_D = g^{x_D})$ for randomly chosen $x_S, x_D \leftarrow \mathbb{Z}_q$. Signy's designated-verifier signature on a message m is $\sigma \leftarrow \langle H_{\mathcal{G}_1}(m)^{x_S}, y_D \rangle$. Desmond simulates the signature by setting $\sigma \leftarrow \langle H_{\mathcal{G}_1}(m)^{x_D}, y_S \rangle$. The signature

is verified by checking that $\sigma = \langle H_{\mathcal{G}_1}(m)^{x_D}, y_S \rangle$. Again, if the value of $y_{SD} = g^{x_S x_D \mod q}$ is compromised, any entity who gets to know y_{SD} can also produce a valid σ by computing $\sigma \leftarrow \langle H_{\mathcal{G}_1}(m), y_{SD} \rangle$. Therefore, this scheme is also delegatable. The Laguillaumie-Vernaud [LV04] DVS scheme is delegatable for the same reason.

4 DVS Security Definitions

In the original paper [JSI96], a designated-verifier signature was required to convince the designated verifier Desmond that the signer Signy has signed the message, in a way that Desmond is able to simulate signature that is indistinguishable from the real signatures, even by a verifier who has access to the designated verifier's private key. A more formal definition was given in say [SBWP03]. We will first repeat formal definitions of unforgeability and non-transferability and then give a definition of the new notion, non-delegatability.

A designated-verifier signature scheme must satisfy at least the next two conditions: (a) Unforgeability: signatures are verifiable by the designated verifier Desmond who rejects it when the signature was not signed by himself or Signy, and (b) Non-transferability: given a message-signature pair (m, σ), that is accepted by the designated-verifier, and without access to the secret key of the signer, it is computationally infeasible to determine whether the message was signed by the signer, or the signature was simulated by the designated verifier.

In the following, Ω denotes the space from which the random oracle H is selected; definition without a random oracle is analogous. Depending on the situation, we will have either $\Omega = \Omega_{\mathsf{npro}}$ to be the set of all non-programmable random oracles [Nie02] or $\Omega = \Omega_{\mathsf{ro}}$ to be the set of all random oracles with proper input and output domains.

Let $\Delta = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Simul}, \mathsf{Vrfy})$ be a designated-verifier signature scheme with the message space $\mathcal{M}$. We say that Δ is *perfectly non-transferable* if $\mathsf{Sign}_{\mathsf{sk}_S, \mathsf{pk}_D}(m) = \mathsf{Simul}_{\mathsf{sk}_D, \mathsf{pk}_S}(m)$ as distributions for every $(\mathsf{pk}_S, \mathsf{sk}_S) \leftarrow \mathsf{Gen}$, $(\mathsf{pk}_D, \mathsf{sk}_D) \leftarrow \mathsf{Gen}$, $H_q \leftarrow \Omega$ and $m \leftarrow \mathcal{M}$. Analogously, one can define statistically non-transferable and computationally non-transferable schemes. An adversial forging algorithm $\mathcal{F}$ is said to $(\tau, q_h, q_s, \varepsilon)$-*forge* Δ if $\mathcal{F}$ runs in time at most τ, makes at most q_h hash queries and in total at most q_s signing and simulation queries, and furthermore

$$\mathsf{Adv}^{\mathsf{forge}}_{\Delta}(\mathcal{F}) := \Pr\left[\begin{array}{l}(\mathsf{pk}_S, \mathsf{sk}_S) \leftarrow \mathsf{Gen}; (\mathsf{pk}_D, \mathsf{sk}_D) \leftarrow \mathsf{Gen}; H \leftarrow \Omega; \\ (m, \sigma) \leftarrow \mathcal{F}^{\mathsf{Sign}_{\mathsf{sk}_S, \mathsf{pk}_D}(\cdot), \mathsf{Simul}_{\mathsf{sk}_D, \mathsf{pk}_S}(\cdot), H(\cdot)}(\mathsf{pk}_S, \mathsf{pk}_D) : \\ \sigma \notin \Sigma(m) \wedge \mathsf{Vrfy}_{\mathsf{pk}_S, \mathsf{pk}_D}(m, \sigma) \neq \mathsf{reject}\end{array}\right] \geq \varepsilon \ ,$$

where $\Sigma(m)$ is the set of signatures received either from $\mathsf{Sign}_{\mathsf{sk}_S, \mathsf{pk}_D}(m)$ or from $\mathsf{Simul}_{\mathsf{sk}_D, \mathsf{pk}_S}(m)$. A designated-verifier signature scheme is $(\tau, q_h, q_s, \varepsilon)$-*unforgeable* if no forger can $(\tau, q_h, q_s, \varepsilon)$-forge it. In the case a DVS scheme is only computationally non-transferable, it is important that $\Sigma(m)$ also includes signatures received from $\mathsf{Simul}_{\mathsf{sk}_D, \mathsf{pk}_S}(m)$. If a scheme is perfectly non-transferable then an access to the Simul oracle will not help the forger.

Delegatability. The definition of unforgeability does not cover the case when the signer, without disclosing her secret key sk_S, delegates her signing rights (w.r.t. to

a concrete designated verifier Desmond) to $\mathcal{F}$ by disclosing some side information $y_{SD} := f_S(\mathsf{sk}_S, \mathsf{pk}_D)$, that helps the latter to produce valid signatures. Analogously, the designated verifier might delegate his signature simulating capability by disclosing some—potentially different—side information $y'_{SD} := f_D(\mathsf{sk}_D, \mathsf{pk}_S)$. This implies that when Desmond sees a valid signature σ that is not generated by himself, he can only conclude that σ is generated by somebody who knows either y_{SD} or y'_{SD}. In some scenarios, Signy may reveal or be forced to reveal the value of y_{SD} to a third party Trevor so that Trevor can generate valid Signy's signatures for Desmond. Such a delegation is essentially different from the situation where Signy reveals her secret key x_S to Trevor, since knowledge of y_{SD} allows Trevor to sign messages designated only to Desmond, and not to anybody else. Therefore, Signy might be more willing to give out the value y_{SD} than her secret key sk_S. This is not an issue in the case of conventional signature schemes where non-delegatability follows from unforgeability.

We will next give a longer explanation why delegatability is bad. First of all, in the original definition of designated-verifier proofs [JSI96], it was said that a proof of the truth of some statement Φ is a designated-verifier proof if it is a proof that either Φ is true or the signer knows Desmond's secret key. This intuitive requirement is clearly not satisfied by delegatable DVS schemes, where a signer proves that either Φ is true or she knows y_{SD} or she knows y'_{SD}.

Moreover, delegatability is undesirable in many applications of the designated-verifier signature scheme. We will give two examples. First, in an hypothetical e-voting protocol where voters sign messages by using a designated-verifier signature scheme (with the tallier being the designated verifier Desmond), knowing that this information can only be used to vote in this concrete election, a voter Signy could be motivated to give a copy of y_{SD} to the coercer for a moderate sum of money. On the other hand, since Signy might use sk_S in many other applications, she might not be willing to send sk_S to the coercer for any imaginable "moderate" amount of money. Second, assume that Signy is a subscriber to an e-library, maintained by Desmond, and that she identifies herself by using a designated-verifier signature scheme so that Desmond could not sell Signy's access history to third parties. If the DVS scheme is delegatable, Signy could however send y_{SD} to a non-subscriber who could then also start enjoying Desmond's service. Since y_{SD} is used only in this application, Signy could be happily willing to do that. On the other hand, if the DVS were not delegatable, Signy would have to reveal her secret key. Finally, it may happen that Signy and Desmond also participate in some other protocols where y_{SD} or y'_{SD} is revealed legitimately and thus the attacker can gain access to either of these values. Note also that a DVS scheme with delegatability is somewhat similar to the proxy signatures, except that in the case of proxy signatures, (a) the verifier can distinguish between messages, signed by Signy and a proxy, and (b) the signatures are universally verifiable.

The preceding discussion motivates the next definition. It basically says that a non-delegatable DVS scheme is a non-interactive system of proofs of knowledge of either sk_S or sk_D. Here, $\mathcal{F}_m$ denotes $\mathcal{F}$ with m as its input, and oracle calls are counted as one step. More precisely, let $\kappa \in [0, 1]$ be the knowledge error. We say that Δ is (τ, κ)-*non-delegatable* if there exists a black-box knowledge extractor $\mathcal{K}$ that, for every algorithm $\mathcal{F}$ and for every valid signature σ, satisfies the following condition: For every

$(\mathsf{pk}_S, \mathsf{sk}_S) \leftarrow \mathsf{Gen}$, $(\mathsf{pk}_D, \mathsf{sk}_D) \leftarrow \mathsf{Gen}$ and message m, if $\mathcal{F}$ produces a valid signature on m with probability $\varepsilon > \kappa$ then, on input m and on access to the oracle $\mathcal{F}_m$, $\mathcal{K}$ produces either sk_S or sk_D in expected time $\frac{\tau}{\varepsilon - \kappa}$ (without counting the time to make the oracle queries). (Here, $\mathcal{F}$'s probability is taken over the choice of her random coins and over the choice of $H_q \leftarrow \Omega$.)

5 New DVS Scheme with Tight Reduction to DDH in NPRO

Let p, q and $\mathcal{G}_q$ be as defined in Sect. 2. Let $g_1, g_2 \in \mathcal{G}_q$ be two elements such that nobody knows the mutual discrete logarithms of g_1 and g_2. Following the ideas from [KW03], in the next DVS-KW DVS scheme, we let Signy to prove Desmond that $(g_1, g_2, y_{1S}, y_{2S})$ is a Decisional Diffie-Hellman tuple, where $x_i \leftarrow_r \mathbb{Z}_q$ is i's private key and $\mathsf{pk}_i := (g_1, g_2, y_{1i}, y_{2i})$ is i's public key with $y_{1i} = g_1^{x_i}$ and $y_{2i} = g_2^{x_i}$. This proof is made designated-verifier by using the same trick as in the JSI scheme [JSI96], and non-interactive by using a non-programmable random oracle [Nie02] H_q with outputs from $\mathbb{Z}_q$. In particular, the random oracle H_q can be chosen at the same stage as other system parameters, g_1 and g_2. This is an interesting result by itself since in [KW03], H_q had to be a *programmable* random oracle for their security proof to go through. The description of the full DVS-KW scheme follows:

- $\mathsf{Sign}_{\mathsf{sk}_S, \mathsf{pk}_D}(m)$: Signy generates random $r, w, t \leftarrow \mathbb{Z}_q$, and sets $a_1 \leftarrow g_1^r \mod p$, $a_2 \leftarrow g_2^r \mod p$, $c \leftarrow g_1^w y_{1D}^t \mod p$, $h \leftarrow H_q(\mathsf{pk}_S, \mathsf{pk}_D, a_1, a_2, c, m)$ and $z \leftarrow r + (h + w)x_S \mod q$. She outputs the signature $\sigma \leftarrow (w, t, h, z)$.
- $\mathsf{Simul}_{\mathsf{sk}_D, \mathsf{pk}_S}(m)$: By selecting three random numbers $z, \alpha, \beta \leftarrow_r \mathbb{Z}_q$, Desmond creates $\sigma = (w, t, h, z)$ for any message m as follows: $(a_1, a_2) \leftarrow (g_1^z y_{1S}^{-\beta} \mod p, g_2^z y_{2S}^{-\beta} \mod p)$, $h \leftarrow H_q(\mathsf{pk}_S, \mathsf{pk}_D, a_1, a_2, g_1^\alpha \mod p, m)$, $w \leftarrow \beta - h \mod q$, $t \leftarrow (\alpha - w)x_D^{-1} \mod q$.
- $\mathsf{Vrfy}_{\mathsf{pk}_S, \mathsf{pk}_D}(m; w, t, h, z)$: The verifier checks whether $h = H_q(\mathsf{pk}_S, \mathsf{pk}_D, g_1^z y_{1S}^{-(h+w)} \mod p, g_2^z y_{2S}^{-(h+w)} \mod p, g_1^w y_{1D}^t \mod p, m)$.

Security: First, clearly this scheme is correct and perfectly non-transferable. Second, we can prove the next result that is a generalisation of a proof from [KW03]:

Theorem 1. *Let $\mathcal{G}$ be a (τ', ε')-DDH group with $|\mathcal{G}| = q$ such that exponentiation in $\mathcal{G}$ takes time $t_{\exp}$. Then the above designated-verifier signature scheme is $(\tau, q_h, q_s, \varepsilon)$-unforgeable in the non-programmable random oracle model, where $\tau \leq \tau' - (3.2q_s + 5.6)t_{\exp}$ and $\varepsilon \geq \varepsilon' + q_s q_h q^{-2} + q^{-1} + q_h q^{-2}$.*

Proof. Assume that we have an algorithm $\mathcal{F}$, running in time at most τ and making at most q_h hash queries and in total at most q_s signing and simulation queries, which forges a new message/signature or a new message/simulated signature pair with probability at least ε. We use $\mathcal{F}$ to construct an algorithm $\mathcal{A}$ running in time τ' which solves the DDH problem with probability ε'. The stated result follows.

Algorithm $\mathcal{A}$ is given as input a tuple $(g_1, g_2, y_{1D}, y_{2D})$; its goal is to determine whether this represents a "random tuple" or a "DDH tuple". To this end, it sets $\mathsf{pk}_D =$

$(g_1, g_2, y_{1D}, y_{2D})$, sets Signy's public key pk_S to be equal to a random DDH tuple $\mathsf{pk}_S = (g_1, g_2, y_{1S} = g_1^{x_S}, y_{2S} = g_2^{x_S})$ for which she chooses the corresponding secret key $x_S \leftarrow \mathbb{Z}_q$, and runs $\mathcal{F}$ on input $(\mathsf{pk}_S, \mathsf{pk}_D)$. Algorithm $\mathcal{A}$ simulates the signing oracle and the random oracle for $\mathcal{F}$ as follows:

Hash queries. In response to a query $H_q(\mathsf{pk}_S, \mathsf{pk}_D, a_1, a_2, c, m)$, algorithm $\mathcal{A}$ responds with $h \leftarrow H_q(\mathsf{pk}_S, \mathsf{pk}_D, a_1, a_2, c, m)$. Additionally, if this query was not made before, $\mathcal{A}$ stores the tuple $(\mathsf{pk}_S, \mathsf{pk}_D, a_1, a_2, c, m)$.

Signing and simulation queries. If $\mathcal{F}$ asks either for a signature or a simulation on message m, then $\mathcal{A}$ attempts to sign pk_S. That is, $\mathcal{A}$ chooses random $r, w, t \leftarrow \mathbb{Z}_q$, sets $a_1 \leftarrow g_1^r \mod p$, $a_2 \leftarrow g_2^r \mod p$ and $c \leftarrow g_1^w y_{1D}^t \mod p$. If H_q had previously been queried on input $H_q(\mathsf{pk}_S, \mathsf{pk}_D, a_1, a_2, c, m)$ then $\mathcal{A}$ aborts (with output 0); otherwise, $\mathcal{A}$ sets $h \leftarrow H_q(\mathsf{pk}_S, \mathsf{pk}_D, a_1, a_2, c, m)$ and outputs the signature (w, t, h, z), where $z \leftarrow r + (h + w)x_S \mod q$.

At some point, $\mathcal{F}$ outputs its forgery $(\bar{m}, \bar{\sigma} = (\bar{w}, \bar{t}, \bar{h}, \bar{z}))$, where we assume that $\bar{\sigma}$ was not previously the response to a query $\mathsf{Sign}_{\mathsf{sk}_S}(\bar{m})$ or $\mathsf{Simul}_{\mathsf{sk}_D}(\bar{m})$. Set $\bar{a_1} \leftarrow g_1^{\bar{z}} y_{1S}^{-(\bar{h}+\bar{w})} \mod p$, $\bar{a_2} \leftarrow g_2^{\bar{z}} y_{2S}^{-(\bar{h}+\bar{w})} \mod p$, $\bar{c} \leftarrow g_1^{\bar{w}} y_{1D}^{\bar{t}} \mod p$. Now, if $H_q(\mathsf{pk}_S, \mathsf{pk}_D, \bar{a_1}, \bar{a_2}, \bar{c}, \bar{m}) = \bar{h}$ (i.e., $\mathsf{Vrfy}_{\mathsf{pk}}(\bar{m}, \bar{\sigma}) = 1$), then $\mathcal{A}$ outputs 1; otherwise, $\mathcal{A}$ outputs 0.

We now analyse the probability that $\mathcal{A}$ outputs 1. If $(g_1, g_2, y_{1D}, y_{2D})$ is a Diffie-Hellman tuple, then $\mathcal{A}$ provides a perfect simulation for $\mathcal{F}$ except for the possibility of an abort. An abort can occur during $\mathcal{A}$'s simulation of any of the signing queries; in the simulation of any particular signing query, it is not hard to see that the probability of abort is at most q_h/q^2. Thus the overall probability that $\mathcal{A}$ aborts is at most $q_s q_h/q^2$. This means that $\mathcal{A}$ outputs a forgery (and hence $\mathcal{A}$ outputs 1) with probability at least $\varepsilon - q_s q_h/q^2$. On the other hand, if $(g_1, g_2, y_{1D}, y_{2D})$ is a random tuple then with probability $1 - q^{-1}$ it is not a Diffie-Hellman tuple. In this case, for any query $H_q(\mathsf{pk}_S, \mathsf{pk}_D, a_1, a_2, c, m)$ made by $\mathcal{F}$ there is at most q possible values of (w, t, z) such that the verification succeeds. Thus, $\mathcal{F}$ outputs a forgery (and hence $\mathcal{A}$ outputs 1) with probability at most $q^{-1} + q_h q^{-2}$. Putting everything together, we see that $\mathsf{Adv}^{\mathsf{ddh}}_{\mathcal{G}}(\mathcal{A}) \geq \varepsilon - q_s q_h q^{-2} - q^{-1} - q_h q^{-2} = \varepsilon'$. The running time of $\mathcal{A}$ includes the running time of $\mathcal{F}$ and is otherwise dominated by two exponentiations and one multi-exponentiation that are performed for each query to the signing oracle plus those done in generating Signy's key and verifying the output of $\mathcal{F}$. Assuming as in [KW03] that a two-exponent multi-exponentiation takes time $1.2t_{\mathrm{exp}}$ gives the result of the theorem. □

On NPRO versus RO model. The non-programmable random oracle model is known to be strictly weaker than the random oracle model [Nie02]. This is not surprising looking at the corresponding security proofs, e.g., at the security proof in [KW03] and at the proof of Thm. 1. In the former, the adversary does not know the secret key, and therefore is forced to program the random oracle to be able to answer successfully to the signature queries. In the latter, the adversary knows Signy's secret key, and knowing this, can answer successfully to the signature and simulation queries without a need to program the random oracle. A conceptual difference between the two models is that proofs in the RO model work for the "best case" (showing that for every forger, there

exists a function H_q such that the signature scheme is unforgeable), while proofs in the NPRO model work for the "average case" (showing that the signature scheme is unforgeable for a randomly chosen function $H_q \leftarrow \Omega_{\mathsf{npro}}$, independent of the forger). Given such arguments, we think that unforgeability in the NPRO model is an important property of DVS-KW.

Conventional signature scheme with tight reduction to DDH in the NPRO+CRS model. From DVS-KW, one can build a conventional signature scheme with the same properties (tight reduction to the DDH problem in the non-programmable random oracle model) if one additionally assumes the common reference string (CRS) model. More precisely, in this case, the CRS is equal to a pair (y_{1D}, y_{2D}), drawn randomly from the set $\{(g_1^{x_D}, g_2^{x_D}) : x_D \in \mathbb{Z}_q\}$. It is assumed that nobody will use the corresponding secret key to simulate a DVS; in our case, this is easily achieved since CRS can be a random DDH tuple. On the other hand, for the proof to go through, $\mathcal{A}$ is allowed to generate the CRS herself. This scheme is about twice less efficient than the Katz-Wang scheme, but its security does not rely on programmable random oracles. On the other hand, this scheme is more efficient than the currently known signature schemes that are unforgeable in the plain model, and/or has a more standard underlying assumption.

Non-delegatability of DVS-KW. Next, we will give a proof of the non-delegatability of DVS-KW. Because of the structure of our proof, we expect that an arbitrary DVS scheme would have a similar time bound in its security proof.

Theorem 2. *Assume that for some $m \in \mathcal{M}$, $\mathcal{F}$ can produce valid signatures in time τ and with probability ε. Then* DVS-KW *is $(56\tau/\varepsilon, \frac{1}{q})$-non-delegatable in the programmable random oracle model.*

Proof. Assume that $\varepsilon > \kappa = 1/q$. We must show there exists a knowledge extractor $\mathcal{K}$ that on input σ and on black-box oracle access to $\mathcal{F}$ can produce either sk_S or sk_D in expected time $\tau' \leq 56\tau/\varepsilon$ and with probability 1.

Let $\mathcal{F}_m$ be a forger with input m. Consider two executions of $\mathcal{F}_m$ by $\mathcal{K}$ on the same random input of $\mathcal{F}_m$. In both cases, $\mathcal{K}$ executes $\mathcal{F}_m$ step-by-step, except that $\mathcal{K}$ returns different random values (h versus h') as the answer to the hash query $H_q(\mathsf{pk}_S, \mathsf{pk}_D, a_1, a_2, c, m)$. Since (a_1, a_2, c) are under the hash, their values must be equal in both runs. If both signatures are valid, one must have $z - (h + w)x_S \equiv z' - (h' + w')x_S \mod q$ and $w + tx_D \equiv w' + t'x_D \mod q$, where (w, t, h, z) is the first signature and (w', t', h', z'), $h \neq h'$, is the second signature. Now we have two cases. If $w \not\equiv w' \mod q$ then one can find $x_D \leftarrow (w - w')/(t' - t) \mod q$. If $w \equiv w' \mod q$ then $h + w \not\equiv h' + w' \mod q$ and thus one can find $x_S \leftarrow (z - z')/(h - h' + w - w') \mod q$. Now, assume Rewind is an algorithm that, given an oracle access to $\mathcal{F}_m$, in time τ_R, for some τ_R, produces two different valid signatures (w, t, h, z) and (w', t', h', z') on m, such that $h \neq h'$ but $(a_1, a_2, c) = (a_1', a_2', c')$. Then one can compute either x_S or x_D with probability 1. Thus, given that algorithm Rewind runs in expected time $56/\varepsilon$ (counting every access to $\mathcal{F}_m$ as one step), we have proven the theorem.

Next, let us describe the algorithm Rewind. We are given a forger $\mathcal{F}_m$ who returns a signature (w, t, h, z) that may or may not be correct. We are given that the probability of a correct answer taken over $\mathcal{F}_m$ coins and the choice of h is at least ε. We want to find correct answers to two different h-values for a given (a_1, a_2, c, m) as efficiently as

possible. The idea is to run the prover, and use rewinding to try to make him answer two different challenges correctly. But to run him, we need to supply random coins. Although we know that the average success probability is ε, we do not know that $\mathcal{F}_m$ is equally successful with any random input. To get a better view of this, let H be a matrix with a row for each possible set of random coins for $\mathcal{F}_m$, and one column for each possible challenge value. Write 1 in an entry if $\mathcal{F}_m$ answers correctly with the corresponding random choices and challenge, and 0 otherwise. Using $\mathcal{F}_m$ as black-box, we can probe any entry we want in H, and our goal can be rephrased to: find two 1's in the same row. What we know is that ε equals the fraction of 1-entries in H. Now, Rewind uses an algorithm from [DF02] to find such 1 entries in time $56/\varepsilon$. □

6 On Disavowability

In a few proposed designated-verifier signature schemes [JSI96], given a pair (m, σ), Signy and Desmond cannot prove to a third party, even when they join their forces, which one of them generated σ. In some other schemes—that we call *disavowable*—Signy can prove that (a) she signed messages that she really signed, and (b) she has not signed signatures, simulated by Desmond. This is achieved by decomposing the signature σ of a message m in two non-empty parts, s ("undeniable signature") and P ("designated-verifier proof"), where for every message m and for every possibly incorrect $\bar{s}$, the designated verifier can produce a simulated proof $\bar{P}$ such that the distribution of $(\bar{s}, \bar{P})$ is computationally indistinguishable from the distribution of the real signature (s, P). Therefore, in this case, Simul takes an additional argument $\bar{s}$, and one requires also the existence of four additional protocols, Confirm, ConfirmVrfy, Disavowal and DisavowalVrfy. To model our attack on the JSI scheme we must also define "secure disavowability". The idea behind such definition is to prove that if $\mathsf{Vrfy}_{\mathsf{pk}_S,\mathsf{pk}_D}(m, s) = \mathsf{accept}$ and on input (m, s), an oracle machine $\mathcal{F}$ can, with high probability, produce a δ such that $\mathsf{DisavowalVrfy}_{\mathsf{pk}_S,\mathsf{pk}_D}(m; \sigma, \delta) = \mathsf{success}$ then there exists a knowledge extractor that can use $\mathcal{F}$ to recover sk_D.

The JSI scheme [JSI96] was claimed to be disavowable—although, as shown earlier in this paper, it is not—while most of the subsequent DVS schemes provide perfect non-transferability and thus are not disavowable. (If a DVS scheme is perfectly non-transferable then there is no trapdoor information that Signy might use to prove that a message was or was not signed by her. Thus, to make it able for Signy to disavow a signature, a DVS scheme must be computationally but not perfectly non-transferable.) However, even in [JSI96], disavowability was not seen as a feature of the DVS schemes and disavowability was never formally defined; their scheme, being based on Chaum's undeniable signature, just happens to have this property.

Corrected JSI Scheme: JSI+. As mentioned before, the original JSI scheme, is secure including pk_S, pk_D and $s = m^{x_S}$ under the hash. Since we want this scheme to be disavowable, $\mathsf{Simul}_{\mathsf{sk}_D,\mathsf{pk}_S}$ also gets an additional input $s \in \mathbb{Z}_p$. Clearly, JSI+ is computationally but not perfectly non-transferable. Unforgeability, non-delegatability and secure disavowability of this scheme can be proven by using standard cryptographic tools, although most probably either in the RO model or with looser reductions. One of the reasons is that since DVS-KW is *perfectly* non-transferable, in its unforgeability

proof $\mathcal{A}$ could answer signing and simulation queries in the same manner. Therefore, it was sufficient for $\mathcal{A}$ to know only one of sk_S and sk_D to simulate both kind of queries without using a programmable random oracle. It is not a priori clear how to achieve the same in the case of JSI+.

Further work and acknowledgments. We feel that this paper raises interesting questions about the relationship between different non-standard security models; it is an interesting open question to study the necessity of these models in any concrete case. We would like to thank anonymous referees for useful comments. The first author was partially supported by the Estonian Science Foundation.

References

[DF02] Ivan Damgård and Eiichiro Fujisaki. An Integer Commitment Scheme Based on Groups with Hidden Order. In Yuliang Zheng, editor, *Advances on Cryptology — ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 125–142, Queenstown, New Zealand, December 1–5, 2002. Springer-Verlag.

[JSI96] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated Verifier Proofs and Their Applications. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154, Saragossa, Spain, May 12–16, 1996. Springer-Verlag.

[KW03] Jonathan Katz and Nan Wang. Efficiency Improvements for Signature Schemes with Tight Security Reductions. In *10th ACM Conference on Computer and Communications Security*, pages 155–164, Washington, D.C., USA, October 27–31, 2003. ACM Press.

[LV04] Fabien Laguillaumie and Damien Vergnaud. Designated Verifier Signatures: Anonymity and Efficient Construction from Any Bilinear Map. In Carlo Blundo and Stelvio Cimato, editors, *Security in Communication Networks, 4th International Conference, SCN 2004*, volume 3352 of *Lecture Notes in Computer Science*, pages 105–119, Amalfi, Italy, September 8–10, 2004. Springer Verlag.

[Nie02] Jesper Buus Nielsen. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. In Moti Yung, editor, *Advances in Cryptology — CRYPTO 2002, 22nd Annual International Cryptology Conference*, volume 2442 of *Lecture Notes in Computer Science*, pages 111–126, Santa Barbara, USA, August 18–22, 2002. Springer-Verlag.

[SBWP03] Ron Steinfeld, Laurence Bull, Huaxiong Wang, and Josef Pieprzyk. Universal Designated-Verifier Signatures. In Chi Sung Laih, editor, *Advances on Cryptology — ASIACRYPT 2003*, volume 2894 of *Lecture Notes in Computer Science*, pages 523–542, Taipei, Taiwan, November 30–December 4, 2003. Springer-Verlag.

[SKM03] Shahrokh Saeednia, Steve Kremer, and Olivier Markowitch. An Efficient Strong Designated Verifier Signature Scheme. In Jong In Lim and Dong Hoon Lee, editors, *Information Security and Cryptology - ICISC 2003*, volume 2971 of *Lecture Notes in Computer Science*, pages 40–54, Seoul, Korea, November 27–28, 2003. Springer-Verlag.

[SWP04] Ron Steinfeld, Huaxiong Wang, and Josef Pieprzyk. Efficient Extension of Standard Schnorr/RSA Signatures into Universal Designated-Verifier Signatures. In Feng Bao, Robert H. Deng, and Jianying Zhou, editors, *Public Key Cryptography 2004*, volume 2947 of *Lecture Notes in Computer Science*, pages 86–100, Singapore, March 1–4, 2004. Springer-Verlag.