

Revisiting Lattice-based Blind Signatures Again

Yi-Fu Lai and Yu Yu

Shanghai Jiao Tong University

Abstract. We show an attack on a lattice-based blind signature proposed in Crypto’20. Formally it is a linear hash function based framework with a lattice-based instantiation. We first notice a bug in their security proof of blindness, which is overlooked these years. Then, we develop an attack under the honest key and the honest-but-curious signer setting against the blindness notion of the schemes by exploiting the correlations between the leaves of the tree. We also discuss how to repair the scheme.

1 Introduction

Blind signatures, introduced by Chaum [Cha82], are a fundamental cryptographic primitive that allows a signer to issue signatures on user-chosen messages without learning the message contents. Through the decades of development they have found applications in privacy-preserving systems ranging from electronic cash and electronic voting [Cha82] to anonymous authorization and payment systems such as Privacy Pass [DGS⁺18] and TumbleBit [HAB⁺17].

The security of a blind signature is commonly captured by two complementary requirements. *Blindness* prevents the signer from linking a resulting message–signature pair to the signing interaction in which it was issued, whereas *one-more unforgeability* prevents a malicious user from producing more signatures on distinct messages than the number of completed signing interactions. These notions have been extensively studied and refined as the security models for blind signatures evolved, in particular with respect to malicious signers, aborts, and concurrent signing sessions [Abe01,AO00,Sch01,HKLN20,BLL⁺21,KLX22,KLR24].

While many lattice-based blind signatures have been proposed in recent years [AKSY22,dPK22,BLNS23,JS25], the history of their security proofs is less straightforward. The first lattice-based blind signature was proposed by Rückert [Rüc10], followed by several extensions and optimizations. However, it was not until the work of [HKLN20] finds that a flaw in Rückert’s proof of one-more unforgeability, inherited by several subsequent constructions, was identified. The rewinding argument did not establish that the extracted solution to the underlying lattice problem was nontrivial. [HKLN20] discussed attacks on other earlier constructions, concluding that the previous direct lattice-based proposals either had flawed security proofs or were insecure. They proposed a new construction with claimed one-more unforgeability and perfect blindness. The work has been subsequently viewed as the first provably secure lattice-based blind signature [dPK22,KLLQ23,JS25,HLM⁺25,KLR25,HLPP26].

In this work, we revisit [HKLN20] again. Instead of the one-more unforgeability, we identify a bug in the blindness proof. Basing on the gap, we develop a polynomial-time distinguisher against the blindness experiment with a constant advantage. The distinguisher is effective under the honest-but-curious signer setting. That is, the signer follows the scheme honestly, and is able to distinguish. We also discuss how to repair the scheme.

2 Preliminaries

Notations.

We denote the security parameter by κ and the hash-length parameter by λ ; the hash function G below has 2λ -bit outputs. PPT stands for probabilistic polynomial time, with running time polynomial in κ . We write $\text{negl}(\kappa)$ for a nonnegative negligible function: for every positive polynomial p , it holds that $\text{negl}(\kappa) < 1/p(\kappa)$ for all sufficiently large κ . For a finite nonempty set S , $x \xleftarrow{\$} S$ denotes uniform sampling from S ; for a randomized algorithm A , $y \xleftarrow{\$} A(x)$ denotes its output on input x using fresh random coins. For a distribution D , the notation $x \xleftarrow{\$} D$ means the sampling following its own distribution. For a positive integer d , let $\mathbb{Z}_d = \mathbb{Z}/d\mathbb{Z}$, represented by $\{0, \dots, d-1\}$, and let $[d] = \{1, \dots, d\}$. For a proposition E , the Boolean value $[E]$ is 1 if E holds and 0 otherwise.

Game $\text{Blind}_{\text{BS}, \text{par}}^{\mathcal{A}}$ 01: $b \xleftarrow{\\$} \{0, 1\}$ 02: $b_1 \leftarrow b; b_2 \leftarrow 1 - b$ 03: $(\text{sk}, \text{pk}) \xleftarrow{\\$} \text{KG}(\text{par})$ 04: $b' \xleftarrow{\\$} \mathcal{A}^{\text{Init}, \text{O}_{\text{U}_1}, \text{O}_{\text{U}_2}}(\text{pk}, \text{sk})$ 05: return $[b = b']$ Oracle $\text{Init}(\widetilde{\text{msg}}_0, \widetilde{\text{msg}}_1)$ 06: $\text{msg}_0 \leftarrow \widetilde{\text{msg}}_0; \text{msg}_1 \leftarrow \widetilde{\text{msg}}_1$ 07: $\text{sess}_1 \leftarrow \text{sess}_2 \leftarrow \text{init}$	Oracle $\text{O}_{\text{U}_1}(\text{sid}, R)$ 08: if $\text{sid} \notin \{1, 2\}$ or $\text{sess}_{\text{sid}} \neq \text{init}$ then 09: return $\perp$ 10: $\text{sess}_{\text{sid}} \leftarrow \text{open}$ 11: $R_{\text{sid}} \leftarrow R$ 12: $(c_{\text{sid}}, \text{st}_{\text{sid}}) \xleftarrow{\\$} \text{U}_1(\text{pk}, R_{\text{sid}}, \text{msg}_{b_{\text{sid}}})$ 13: return $(\text{sid}, c_{\text{sid}})$ Oracle $\text{O}_{\text{U}_2}(\text{sid}, s)$ 14: if $\text{sess}_{\text{sid}} \neq \text{open}$ then return $\perp$ 15: $\text{sess}_{\text{sid}} \leftarrow \text{closed}$ 16: $s_{\text{sid}} \leftarrow s$ 17: $\sigma_{b_{\text{sid}}} \leftarrow$ $\text{U}_2(\text{pk}, R_{\text{sid}}, c_{\text{sid}}, s_{\text{sid}}, \text{msg}_{b_{\text{sid}}}, \text{st}_{\text{sid}})$ 18: if $\text{sess}_1 = \text{sess}_2 = \text{closed}$ then 19: if $\sigma_0 = \perp$ or $\sigma_1 = \perp$ then 20: return $(\perp, \perp)$ 21: return (σ_0, σ_1) 22: return $(\text{sid}, \text{closed})$
---	--

Fig. 1. The blindness experiment. The single call to Init precedes all session queries. Oracle names are distinguished from the local user algorithms. The message argument of U_2 is written explicitly. An aborted user state propagates $\perp$ to finalization.

2.1 Blind Signatures and Blindness

This subsection recalls the basic definitions of a three-move blind signature and the standard blindness notion. Following [HKLN20, Section 4.1], a canonical three-move blind signature scheme is defined as follows.

Definition 1 (Three-move blind signature scheme). A canonical three-move blind signature scheme is a tuple $\text{BS} = (\text{PGen}, \text{KG}, \text{S}, \text{U}, \text{BSVer})$, where $\text{S} = (\text{S}_1, \text{S}_2)$ and $\text{U} = (\text{U}_1, \text{U}_2)$. The randomized algorithm $\text{PGen}(1^\kappa)$ generates public parameters par , which are implicit inputs to all subsequent algorithms. The randomized algorithm $\text{KG}(\text{par})$ returns a key pair (sk, pk) . An execution on a message $\text{msg} \in \{0, 1\}^*$ proceeds as follows:

- (1) The signer computes $(\text{st}_S, R) \xleftarrow{\$} \text{S}_1(\text{sk})$ and sends R .
- (2) The user computes $(c, \text{st}_U) \xleftarrow{\$} \text{U}_1(\text{pk}, R, \text{msg})$ and sends c .
- (3) The signer computes $s \leftarrow \text{S}_2(\text{sk}, R, c, \text{st}_S)$ and sends s .
- (4) Locally, the user computes $\sigma \leftarrow \text{U}_2(\text{pk}, R, c, s, \text{msg}, \text{st}_U)$ and outputs σ .

The last step is local, so only three messages are exchanged. The user may output $\perp$. The deterministic algorithm $\text{BSVer}(\text{pk}, \sigma, \text{msg})$ returns a bit; $\text{BSVer}(\text{pk}, \perp, \text{msg}) = 0$ by convention. The second signer and user algorithms are deterministic once their respective states are fixed. The issuance transcript is $T = (R, c, s)$; the message, user randomness, and local state are not part of T .

Definition 2 (Honest-Key Blindness). For fixed public parameters par , consider the honest-key blindness experiment $\text{Blind}_{\text{BS}, \text{par}}$ in Figure 1. The blindness advantage of an adversary $\mathcal{A}$ is

$$\text{Adv}_{\text{BS}, \text{par}}^{\text{Blind}}(\mathcal{A}) = \left| \Pr[\text{Blind}_{\text{BS}, \text{par}}^{\mathcal{A}} = 1] - \frac{1}{2} \right|.$$

Blindness requires this advantage to be negligible for every PPT adversary; perfect blindness requires it to be zero for every adversary.

2.2 Full Description of HKLN20

Linear hash functions and filter sets. The construction uses a linear hash function family $\text{LHF} = (\text{PGen}, F)$. Public parameters specify a scalar ring S , two S -modules D and R , and a map $F : D \rightarrow R$ satisfying

$$F(a \cdot x + y) = a \cdot F(x) + F(y) \quad (a \in S, x, y \in D).$$

Set	Role
D_{sk}	Secret-key sampling
D_r	Signer randomness sampling
D_s	Signer-response acceptance
D_α	User response-mask sampling
$D_{s'}$	Final-response acceptance
S_β	User challenge-mask sampling
$S_{c'}$	Hash-challenge range
S_c	Transmitted-challenge acceptance

Table 1. The samplings used in [HKLN20].

They also specify finite sampling and acceptance sets $S_\beta, S_c, S_{c'} \subseteq S$ and $D_{\text{sk}}, D_r, D_s, D_{s'}, D_\alpha \subseteq D$. Their roles are given in Table 1. The parameter argument of F is suppressed, as in [HKLN20].

Lattice Instantiation. In [HKLN20, Section 6], the scalar ring is $R_q = \mathbb{Z}_q[X]/(X^n + 1)$, where n is a power of two and q is an odd prime. The modules are $S = R_q$, $D = R_q^m$, and $R = R_q$. In this paragraph, m denotes the module dimension rather than a message. For a public uniformly sampled vector $\hat{a} = (a_1, \dots, a_m) \in R_q^m$, the map is $F(\hat{z}) = \sum_{i=1}^m a_i z_i \bmod q$. Write $B_q(d) = \{x \in R_q : \|x\|_\infty \leq d\}$, with the norm taken over centered coefficient representatives. Scalar filter sets are $S_x = B_q(d_x)$ for $x \in \{\beta, c, c'\}$; vector filter sets are $D_y = B_q(d_y)^m$ for $y \in \{\text{sk}, r, s, s', \alpha\}$. Figure 9 of [HKLN20] show the relations as follows

$$\begin{aligned}
d_\beta &= u n d_{c'}, & d_c &= d_\beta - d_{c'}, \\
d_r &\geq v m n^2 d_{\text{sk}} d_c, & d_s &= d_r - n d_{\text{sk}} d_c, \\
d_\alpha &= w m n d_s, & d_{s'} &= d_\alpha - d_s,
\end{aligned}$$

where u, v, w are public positive parameters. These parameters specify the samplers, linear operations, and membership tests used below.

Repetition parameters and hashes. Let η, ν, μ be positive integers, and let $H : \{0, 1\}^* \rightarrow S_{c'}$ and $G : \{0, 1\}^* \rightarrow \{0, 1\}^{2\lambda}$ be public hash functions, modeled as random oracles. The signer prepares η commitments. The user samples μ challenge masks β_j and ν response masks α_k . These repetitions provide multiple candidates for the respective range tests; they are not repetitions in which every response must be accepted. The purpose is to avoid the abort and the related tricky issues. The candidate commitment list contains $\eta\mu\nu$ leaves, but the final signature opens only one of them.

Hash trees. The hash-tree subroutine $\text{HT}[G] = (\text{HashTree}, \text{BuildAuth}, \text{RootCalc})$ is a standard one. On a list $\mathbf{v}$, HashTree returns $(\text{root}, \text{tree})$; BuildAuth returns an authentication path for one position; RootCalc reconstructs a root from a leaf value and its path. For a complete tree with $\ell = 2^h$ leaves $v_0, \dots, v_{\ell-1}$, the bottom nodes are $t_{0,j} = G(v_j)$, and an internal node is $t_{i,j} = G(t_{i-1,2j}, t_{i-1,2j+1})$. A path is $\text{auth} = (m, a_0, \dots, a_{h-1})$, where m indicates the opened leaf's position and a_i is the sibling digest at level i . In particular, a_0 is the hash of the sibling *leaf value*, not the leaf value itself. Figure 2 makes the level indices explicit.

Indexing for Leaves. [HKLN20] uses the map

$$2\text{Int}_{\eta,\nu,\mu}(i, j, k) = i + \eta(j - 1) + \eta\nu(k - 1), \quad (1)$$

with domain $[\eta] \times [\nu] \times [\mu]$, where $[d] = \{1, \dots, d\}$. Figure 5 of [HKLN20], however, uses $(i, j, k) \in [\eta] \times [\mu] \times [\nu]$ and opens the position $2\text{Int}_{\eta,\nu,\mu}(i \oplus \gamma, j, k)$. We retain that notation in Figure 3.

Issuance and final signature. Figure 3 follows Figure 5 of [HKLN20]. The signer samples $r_i \in D_r$, publishes $R_i = F(r_i)$, and later returns the first $s_i = c \cdot \text{sk} + r_i$ lying in D_s . The user prepares every candidate $R'_{i \oplus \gamma, j, k} = R_i + F(\alpha_k) + \beta_j \cdot \text{pk}$, hashes their root together with the message as a standard Fiat-Shamir hash, and sends the first $c_j = c' + \beta_j$ lying in $S_{c'}$. After receiving s , the user first checks its range and its public relation to some R_i . It then takes the first mask with $s'_k = s + \alpha_k \in D_{s'}$ and opens the corresponding leaf. The resulting signature is (c', s'_k, auth) .

HashTree(v) 01: $\ell \leftarrow v $; $(v_0, \dots, v_{\ell-1}) \leftarrow v$ 02: $h \leftarrow \lceil \log_2 \ell \rceil$ 03: for $j \in \{0, \dots, \ell - 1\}$ do 04: $t_{0,j} \leftarrow G(v_j)$ 05: for $i \in \{1, \dots, h\}$ do 06: for $j \in \{0, \dots, 2^{h-i} - 1\}$ do 07: $t_{i,j} \leftarrow G(t_{i-1,2j}, t_{i-1,2j+1})$ 08: root $\leftarrow t_{h,0}$ 09: tree $\leftarrow (h, (t_{i,j})_{i,j})$ 10: return (root , tree)	RootCalc(v, auth) 21: $(m, a_0, \dots, a_{h-1}) \leftarrow \text{auth}$ 22: $b_0 \leftarrow G(v)$ 23: for $i \in \{1, \dots, h\}$ do 24: $s \leftarrow \lfloor m/2^{i-1} \rfloor$ 25: $b \leftarrow s \bmod 2$ 26: if $b = 0$ then 27: $b_i \leftarrow G(b_{i-1}, a_{i-1})$ 28: else 29: $b_i \leftarrow G(a_{i-1}, b_{i-1})$ 30: return root $\leftarrow b_h$
BuildAuth(m, tree) 11: Read h and $(t_{i,j})_{i,j}$ from tree 12: for $i \in \{0, \dots, h - 1\}$ do 13: $s \leftarrow \lfloor m/2^i \rfloor$ 14: $b \leftarrow s \bmod 2$ 15: if $b = 0$ then 16: $a_i \leftarrow t_{i,s+1}$ 17: else 18: $a_i \leftarrow t_{i,s-1}$ 19: auth $\leftarrow (m, a_0, \dots, a_{h-1})$ 20: return auth	

Fig. 2. The required subroutines used in [HKLN20, Figure 4], for a complete binary tree. The stored tree explicitly includes level 0, which is needed by BuildAuth.

KG(par) 01: $\text{sk} \xleftarrow{\$} D_{\text{sk}}; \text{pk} \leftarrow F(\text{sk})$ 02: return (sk , pk) $S_1(\text{sk})$ 03: for $i \in [\eta]$: $r_i \xleftarrow{\$} D_r; R_i \leftarrow F(r_i)$ 04: $\text{st}_S \leftarrow (r_1, \dots, r_\eta);$ $R \leftarrow (R_1, \dots, R_\eta)$ 05: return (st_S , R) $S_2(\text{sk}, R, c, \text{st}_S)$ 06: $(r_1, \dots, r_\eta) \leftarrow \text{st}_S$ 07: if $c \notin S_c$ then 08: return $\perp$ 09: for $i \in [\eta]$ do 10: $s_i \leftarrow c \cdot \text{sk} + r_i$ 11: if $s_i \in D_s$ then 12: return $s \leftarrow s_i$ 13: return $\perp$ BSVer($\text{pk}, \sigma, \text{msg}$) 14: $(c', s', \text{auth}) \leftarrow \sigma$ 15: $R' \leftarrow F(s') - c' \cdot \text{pk}$ 16: root $\leftarrow \text{RootCalc}(R', \text{auth})$ 17: if $c' = H(\text{root}, \text{msg})$ and $s' \in D_{s'}$ then 18: return 1 19: return 0	$U_1(\text{pk}, R, \text{msg})$ 20: $(R_1, \dots, R_\eta) \leftarrow R$ 21: $\alpha_1, \dots, \alpha_\nu \xleftarrow{\$} D_\alpha;$ $\beta_1, \dots, \beta_\mu \xleftarrow{\$} S_\beta$ 22: $\gamma \xleftarrow{\$} \mathbb{Z}_\eta$ 23: for $(i, j, k) \in [\eta] \times [\mu] \times [\nu]$ do 24: $R'_{i \oplus \gamma, j, k} \leftarrow R_i + F(\alpha_k) + \beta_j \text{pk}$ 25: (root , tree) $\leftarrow$ HashTree($R'_{1,1,1}, \dots, R'_{\eta,\mu,\nu}$) 26: $c' \leftarrow H(\text{root}, \text{msg})$ 27: for $j \in [\mu]$ do 28: $c_j \leftarrow c' + \beta_j$ 29: if $c_j \in S_c$ then 30: return $(c \leftarrow c_j, \text{st}_U)$, where $\text{st}_U = (\alpha_1, \dots, \alpha_\nu, c', j, \gamma, \text{tree})$ 31: return $\perp$ $U_2(\text{pk}, R, c, s, \text{msg}, \text{st}_U)$ 32: $(R_1, \dots, R_\eta) \leftarrow R$ 33: $(\alpha_1, \dots, \alpha_\nu, c', j, \gamma, \text{tree}) \leftarrow \text{st}_U$ 34: if $s \notin D_s$ then 35: return $\perp$ 36: Find $i \in [\eta]$: $F(s) = c \cdot \text{pk} + R_i$ 37: return $\perp$ if no such i exists 38: for $k \in [\nu]$ do 39: $s'_k \leftarrow s + \alpha_k$ 40: if $s'_k \in D_{s'}$ then 41: auth $\leftarrow$ BuildAuth($2\text{Int}_{\eta,\nu,\mu}(i \oplus \gamma, j, k), \text{tree}$) 42: return $\sigma \leftarrow (c', s'_k, \text{auth})$ 43: return $\perp$
--	--

Fig. 3. The blind signature scheme of [HKLN20, Figure 5].

3 Insecurity of Blindness

We describe our blindness distinguisher against the scheme. We first remark that there is a minor issue in indexing in the work. Precisely [HKLN20] lets the user to use the standard XOR operation $\oplus$ to mask the indices of the signer's first message of an η -tuple $(R_i)_i$ by $R_{i \oplus \gamma}$ for some $\gamma \in [\eta]$. However, in their instantiation the parameter is taken to be 60, which makes the values at some positions undefined.

To avoid the issue, we may assume $\eta \geq 2$ and $\mu = \nu$, all powers of two, with indices starting from 0. We then continue the notation of shifting $i \oplus \gamma$, and position (i, j, k) is naturally flattened to $i + \eta j + \eta \mu k$. Thus two bottom-level siblings lie in the same (j, k) block.

Bug in the Proof. In the proof of Theorem 2, the authors argue perfect blindness through two claims, denoted by (B1) and (B2). In particular, statement (B1) claims that, for each of the four combinations $(T_{\text{sid}}, \sigma_i)$, where $(\text{sid}, i) \in \{1, 2\} \times \{0, 1\}$, there exists user randomness

$$\text{rnd}_{U_{\text{sid}, i}} = (\alpha_{\text{sid}, i, 1}, \dots, \alpha_{\text{sid}, i, \nu}, \beta_{\text{sid}, i, 1}, \dots, \beta_{\text{sid}, i, \mu}, \gamma_{\text{sid}, i})$$

that results in the tuple $(T_{\text{sid}}, \sigma_i)$.

To establish this claim, let $\sigma_i = (c'_i, s'_i, \text{auth}_i)$ and let $(i_i, j_i, k_i) = 2\text{Int}^{-1}(n_i)$, where $\text{auth}_i = (n_i, a_{i,1}, \dots, a_{i,h})$. The proof constructs the selected masks so that the opened leaf satisfies

$$R'_{\text{sid}, i, i_i, j_i, k_i} = F(s'_i) - c'_i \cdot \text{pk},$$

and consequently $c'_i = H(\text{RootCalc}(R'_{\text{sid}, i, i_i, j_i, k_i}, \text{auth}_i), \text{msg}_i)$. This is sufficient to show that the *opened leaf* is consistent with the fixed authentication path auth_i .

However, this does NOT show that the hash tree generated from the constructed randomness is consistent with auth_i . This is because there are $\eta \cdot \mu \cdot \nu$ leaves in total and the user's randomness has only $\mu \cdot \nu$ freedom. In particular, if $(\text{root}_{\text{sid}, i}, \text{tree}_{\text{sid}, i})$ denotes the output of `HashTree` on the complete list of leaves generated from $\text{rnd}_{U_{\text{sid}, i}}$, statement (B1) additionally requires

$$\text{BuildAuth}(2\text{Int}_{\eta, \nu, \mu}(i_{\text{sid}} \oplus \gamma_{\text{sid}, i}, j_i, k_i), \text{tree}_{\text{sid}, i}) = \text{auth}_i.$$

Concrete Attack. Our key observation is as follows. For fixed (j, k) , all signer commitments receive the same additive mask. Consequently,

$$R'_{i_1 \oplus \gamma, j, k} - R'_{i_2 \oplus \gamma, j, k} = R_{i_1} - R_{i_2}.$$

Moreover, for a returned signature (c', s', auth) , its opened leaf is publicly recoverable as $V = F(s') - c' \cdot \text{pk}$. Hence *the differences between leaves, the single opened leaf in the signature, and the authentication path will serve as our distinguisher.*

Writing $\text{auth} = (p, a_0, \dots, a_{h-1})$, the value a_0 is not the opened leaf V itself; it is the hash of the sibling leaf of V . Our distinguisher $\mathcal{A}$ proceeds as follows.

1. **Initialize the blindness experiment.** Receive the honestly generated key pair (sk, pk) and choose two distinct messages $\text{msg}_0, \text{msg}_1$.
2. **Run the signer honestly.** For each session $t \in \{1, 2\}$, the signer honestly sample $r_i^{(t)} \xleftarrow{\$} D_r$ and set $R_i^{(t)} = F(r_i^{(t)})$ for every $i \in \{0, \dots, \eta - 1\}$. Send $R^{(t)} = (R_0^{(t)}, \dots, R_{\eta-1}^{(t)})$ to the user. After receiving c_t , compute $s_i^{(t)} = c_t \cdot \text{sk} + r_i^{(t)}$ in the prescribed order. Return the first value in D_s , or return $\perp$ if no such value exists. If either session returns $\perp$, output an independent random bit; otherwise, let i_t be the index of the returned response in session t .
3. **Record the session differences.** For each $t \in \{1, 2\}$, compute

$$\Delta_t := R_{i_t \oplus 1}^{(t)} - R_{i_t}^{(t)} = F(r_{i_t \oplus 1}^{(t)} - r_{i_t}^{(t)}).$$

4. **Test the returned signature.** If the experiment returns $(\perp, \perp)$, output an independent random bit. Otherwise, parse $\sigma_0 = (c'_0, s'_0, \text{auth}_0)$ and $\text{auth}_0 = (p_0, a_{0,0}, \dots, a_{0,h-1})$, and compute $V_0 = F(s'_0) - c'_0 \cdot \text{pk}$, which is the starting leaf for the authentication path. For each $t \in \{1, 2\}$, evaluate

$$T_t := [a_{0,0} = G(V_0 + \Delta_t)].$$

5. **Guess the session-to-message assignment.** If $(T_1, T_2) = (1, 0)$, output $b' = 0$. If $(T_1, T_2) = (0, 1)$, output $b' = 1$. Otherwise, output an independent random bit.

Theorem 1. Let p_{succ} be the probability that both honest signing sessions output signatures, and assume that G is collision resistant. Conditioned on both sessions succeeding, let **Good** be the event that $\Delta_1 \neq \Delta_2$. Then the distinguisher $\mathcal{A}$ has advantage

$$\text{Adv}_{\text{BS,par}}^{\text{Blind}}(\mathcal{A}) = \frac{1}{2} p_{\text{succ}} \cdot \Pr[\text{Good}].$$

In particular, if both factors are non-negligible, then $\mathcal{A}$ has non-negligible advantage.

Proof. Suppose that σ_0 was produced in session t , and let (j, k, γ) be the user indices selected in that session. The recovered opened leaf is

$$V_0 = F(s'_0) - c'_0 \cdot \text{pk} = R_{i_t}^{(t)} + F(\alpha_k) + \beta_j \cdot \text{pk}.$$

The second equality follows from $s'_0 = s_{i_t}^{(t)} + \alpha_k$ (by the user), $s_{i_t}^{(t)} = c_t \cdot \text{sk} + r_{i_t}^{(t)}$, $c_t = c'_0 + \beta_j$, $R_{i_t}^{(t)} = F(r_{i_t}^{(t)})$, $\text{pk} = F(\text{sk})$ (by the signer), and the linearity of F . At the leaf level, the sibling of position $(i_t \oplus \gamma, j, k)$ has position

$$((i_t \oplus \gamma) \oplus 1, j, k) = ((i_t \oplus 1) \oplus \gamma, j, k).$$

The position therefore contains the sibling leaf value

$$\tilde{V}_0 = R_{i_t \oplus 1}^{(t)} + F(\alpha_k) + \beta_j \cdot \text{pk} = V_0 + \Delta_t.$$

By the definition of **BuildAuth**, the first authentication-path value is $a_{0,0} = G(\tilde{V}_0) = G(V_0 + \Delta_t)$. Hence the test corresponding to the actual session always accepts.

On **Good**, the two candidates $V_0 + \Delta_1$ and $V_0 + \Delta_2$ are distinct and have distinct hashes by assuming the collision-resistance. The test for the actual session therefore accepts, whereas the test for the other session rejects. Thus, $\mathcal{A}$ identifies the correct bit b on **Good**. In all other cases, $\mathcal{A}$ outputs an independent random bit. Since both sessions output signatures with probability p_{succ} , we have

$$\Pr[\text{Blind}_{\text{BS,par}}^{\mathcal{A}} = 1] = \frac{1}{2} + \frac{1}{2} p_{\text{succ}} \cdot \Pr[\text{Good}],$$

and consequently

$$\text{Adv}_{\text{BS,par}}^{\text{Blind}}(\mathcal{A}) = \left| \Pr[\text{Blind}_{\text{BS,par}}^{\mathcal{A}} = 1] - \frac{1}{2} \right| = \frac{1}{2} p_{\text{succ}} \cdot \Pr[\text{Good}].$$

4 Countermeasure

Repairing the scheme does not seem to be difficult. The modification is to randomize the leaf hashes in **HashTree**(v), denoted as **HashTree**^{*}(v). For each leaf v_j , **HashTree**^{*} independently samples a fresh nonce $\rho_j \xleftarrow{\$} \{0, 1\}^\kappa$ and computes $t_{0,j} \leftarrow G(v_j, \rho_j)$. The internal nodes and the root are then remained the same as before. The nonces are stored in **tree**, and the modified **BuildAuth**^{*} includes the one and only one nonce ρ_m of the opened leaf in the authentication path, so that the modified **RootCalc**^{*} starts from $G(v, \rho_m)$ when reconstructing the root. By appending nonces to each leaf, the statement (B1) mentioned above will hold under the random oracle model as the existence is guaranteed. Then, the (B2) statement [HKLN20], on the identical distributions, has to be relaxed to the statistical argument.

Regarding the OMUF of the modified scheme, we can reuse the OMUF of the original scheme to imply it. Building a tree **HashTree**^{*} to tree **HashTree** mapping through the standard projective mapping under ROM is the first step. It is inevitable that the extraction will fail when the modified version multiple new roots will be mapped to the same root in **HashTree**.

To handle the case, we append the root and the message to the original message into the Fiat-Shamir hash in the original scheme in the reduction. In this case, while the root mapping is not surjective, it will lead to the distinct message during the extraction. We will give a more formal sketch in the update version.

Acknowledgement

We thank Julian Loss for the discussion of the draft of this work.

References

- Abe01. Masayuki Abe. A secure three-move blind signature scheme for polynomially many signatures. In Birgit Pfitzmann, editor, *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 136–151, Innsbruck, Austria, May 6–10, 2001. Springer Berlin Heidelberg, Germany.
- AKSY22. Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav. Practical, round-optimal lattice-based blind signatures. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 39–53, Los Angeles, CA, USA, November 7–11, 2022. ACM Press.
- AO00. Masayuki Abe and Tatsuaki Okamoto. Provably secure partially blind signatures. In Mihir Bellare, editor, *CRYPTO 2000*, volume 1880 of *LNCS*, pages 271–286, Santa Barbara, CA, USA, August 20–24, 2000. Springer Berlin Heidelberg, Germany.
- BLL⁺21. Fabrice Benhamouda, Tancrede Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 33–53, Zagreb, Croatia, October 17–21, 2021. Springer, Cham, Switzerland.
- BLNS23. Ward Beullens, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Lattice-based blind signatures: Short, efficient, and round-optimal. In Weizhi Meng, Christian Damsgaard Jensen, Cas Cremers, and Engin Kirda, editors, *ACM CCS 2023*, pages 16–29, Copenhagen, Denmark, November 26–30, 2023. ACM Press.
- Cha82. David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *CRYPTO’82*, pages 199–203, Santa Barbara, CA, USA, 1982. Plenum Press, New York, USA.
- DGS⁺18. Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *PoPETs*, 2018(3):164–180, July 2018.
- dPK22. Rafaël del Pino and Shuichi Katsumata. A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 306–336, Santa Barbara, CA, USA, August 15–18, 2022. Springer, Cham, Switzerland.
- HAB⁺17. Ethan Heilman, Leen Alshenibr, Foteini Baldimtsi, Alessandra Scafuro, and Sharon Goldberg. TumbleBit: An untrusted Bitcoin-compatible anonymous payment hub. In *NDSS 2017*, San Diego, CA, USA, February 26 – March 1, 2017. The Internet Society.
- HKLN20. Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen. Lattice-based blind signatures, revisited. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part II*, volume 12171 of *LNCS*, pages 500–529, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland.
- HLM⁺25. Lucjan Hanzlik, Yi-Fu Lai, Marzio Mula, Eugenio Paracucchi, Daniel Slamanig, and Gang Tang. Tanuki: New frameworks for (concurrently secure) blind signatures from post-quantum group actions. In Goichiro Hanaoka and Bo-Yin Yang, editors, *ASIACRYPT 2025, Part IV*, volume 16248 of *LNCS*, pages 35–69, Melbourne, VIC, Australia, December 8–12, 2025. Springer, Singapore, Singapore.
- HLPP26. Lucjan Hanzlik, Yi-Fu Lai, Eugenio Paracucchi, and Edoardo Persichetti. Post-quantum blind signature from standard group action assumptions and more. In Joan Daemen and Emmanuel Thomé, editors, *EUROCRYPT 2026, Part I*, volume 16541 of *LNCS*, pages 277–307, Rome, Italy, May 10–14, 2026. Springer, Cham, Switzerland.
- JS25. Corentin Jeudy and Olivier Sanders. Improved lattice blind signatures from recycled entropy. In Yael Tauman Kalai and Seny F. Kamara, editors, *CRYPTO 2025, Part I*, volume 16000 of *LNCS*, pages 477–513, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland.
- KLLQ23. Shuichi Katsumata, Yi-Fu Lai, Jason T. LeGrow, and Ling Qin. CSI-Otter: Isogeny-based (partially) blind signatures from the class group action with a twist. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part III*, volume 14083 of *LNCS*, pages 729–761, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.
- KLR24. Shuichi Katsumata, Yi-Fu Lai, and Michael Reichle. Breaking parallel ROS: Implication for isogeny and lattice-based blind signatures. In Qiang Tang and Vanessa Teague, editors, *PKC 2024, Part I*, volume 14601 of *LNCS*, pages 319–351, Sydney, NSW, Australia, April 15–17, 2024. Springer, Cham, Switzerland.
- KLR25. Michael Kloof, Russell W. F. Lai, and Michael Reichle. Blind signatures from arguments of inequality. Cryptology ePrint Archive, Report 2025/1886, 2025.
- KLX22. Julia Kastner, Julian Loss, and Jiayu Xu. The Abe-Okamoto partially blind signature scheme revisited. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 279–309, Taipei, Taiwan, December 5–9, 2022. Springer, Cham, Switzerland.
- Rüc10. Markus Rückert. Lattice-based blind signatures. In Masayuki Abe, editor, *ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 413–430, Singapore, December 5–9, 2010. Springer Berlin Heidelberg, Germany.
- Sch01. Claus-Peter Schnorr. Security of blind discrete log signatures against interactive attacks. In Sihon Qing, Tatsuaki Okamoto, and Jianying Zhou, editors, *ICICS 01*, volume 2229 of *LNCS*, pages 1–12, Xian, China, November 13–16, 2001. Springer Berlin Heidelberg, Germany.