A Note on the Blindness of the Scheme from ePrint 2025/397

Lucjan Hanzlik

CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
hanzlik@cispa.de

Abstract. This note demonstrates that the blind signature scheme based on cryptographic group actions, as proposed in ePrint paper 2025/397, fails to ensure blindness. Specifically, we construct an adversary that achieves a 1/8 advantage in the blindness experiment. The attack leverages selective abort techniques (also known as selective failure attacks), a well-known strategy in the MPC literature.

1 Introduction

Blind signatures, introduced by Chaum [1], enable a signer to issue signatures on user-chosen messages without learning their content, making them a crucial tool for privacy-preserving applications such as e-cash, e-voting, and anonymous credentials. The primary privacy guarantee of blind signatures is *blindness*, which ensures that a signer cannot later link a given signature to the message it was issued for. This property is formally captured through a security experiment: an adversary first submits two messages, M_0 and M_1, to a challenger. The challenger then initializes two signing sessions, one for M_{coin} and another for $M_{1-\mathsf{coin}}$, where coin is a randomly chosen bit. After interacting with the signing oracles, the adversary receives the corresponding signatures and attempts to determine coin. Note that the adversary only receives the signatures if the challenger does not abort any of the two sessions with the adversary (e.g., because of receiving invalid signatures). If the adversary cannot do so with a significant advantage, the scheme is considered blind. This ensures that even a malicious signer cannot extract meaningful information about the user's chosen message during the signing process, preserving user privacy.

The ePrint Paper 2025/397 [2] proposes a new blind signature scheme from cryptographic group actions. While the scheme builds on top of the framework introduced by CSI-Otter [3], a wider range of cryptographic group actions can instantiate it. In particular, the authors of [2] try to solve the following research question:

Can we construct a blind signature from non-commutative group actions?

Contribution. In this short note, we show that despite the claims, *the scheme proposed in [2] is not blind*. Moreover, the reason behind the attack is how the authors solve the missing commutativity of the used cryptographic group action, which makes the attack and the vulnerability inherent and not fixable for the given research question.

2 Preliminaries

Before we shortly describe the scheme, we recall the notions and the blindness definition used in [2]. For a positive integer k, we denote $[k]$ as the set $\{1, \ldots, k\}$. We also use bold characters, e.g., $\mathbf{h}$, to represent vectors. For a finite set S, we write $x \leftarrow\$ S$ to indicate that x is sampled randomly from S. We use $\odot$ to denote component-wise multiplication of vectors in $\mathbb{R}$. Specifically, for $c \in \mathbb{R}$ and vectors $\mathbf{a} = (a_1, \ldots, a_k)$, $\mathbf{b} = (b_1, \ldots, b_k)$, we write $c \odot \mathbf{a} = (c \cdot a_1, \ldots, c \cdot a_k)$ and $\mathbf{a} \odot \mathbf{b} = (a_1 \cdot b_1, \ldots, a_k \cdot b_k)$. For group actions, if a vector $\mathbf{a}$ acts on an element $\mathbf{s}$, we write this as $\mathbf{a} * \mathbf{s} = (a_1 * s_1, \ldots, a_k * s_k)$, where $*$ denotes the action operation. In this note, we will only provide a high-level overview of the scheme for which the above notion is sufficient. For more detail, we refer the reader to the original paper [2].

Definition 1 (Blind Signature). *A three-move blind signature BS with efficient decidable public key space* PK *consists of the following PPT algorithms:*

– $\mathsf{BS.KGen}(1^n) \to (\mathsf{pk}, \mathsf{sk})$*: On input the security parameter* 1^n*, the key generation algorithm outputs a pair of public and secret keys* $(\mathsf{pk}, \mathsf{sk})$*.*
– $\mathsf{BS.S} = (\mathsf{BS.S1}, \mathsf{BS.S2})$*: The signer consists of two phases:*
  - $\mathsf{BS.S1}(\mathsf{sk}) \to (\mathsf{state_S}, \rho_{S,1})$*: On input the secret key, outputs an internal signer state* $\mathsf{state_S}$ *and the first-sender message* $\rho_{S,1}$*.*
  - $\mathsf{BS.S2}(\mathsf{state_S}, \rho_U) \to \rho_{S,2}$*: On input the signer state* $\mathsf{state_S}$ *and a user message* ρ_U*, outputs a second-sender message* $\rho_{S,2}$*.*
– $\mathsf{BS.U} = (\mathsf{BS.U1}, \mathsf{BS.U2})$*: The user consists of two phases:*
  - $\mathsf{BS.U1}(\mathsf{pk}, M, \rho_{S,1}) \to (\mathsf{state_U}, \rho_U)$*: On input the public key* pk*, a message* M*, and the first-sender message* $\rho_{S,1}$*, outputs an internal user state* $\mathsf{state_U}$ *and a user message* ρ_U*.*
  - $\mathsf{BS.U2}(\mathsf{state_U}, \rho_{S,2}) \to \sigma$*: On input a user state* $\mathsf{state_U}$ *and a second-signer message* $\rho_{S,2}$*, outputs a signature* σ*.*
– $\mathsf{BS.Verify}(\mathsf{pk}, M, \sigma)$*: On input the public key* pk*, a message* M*, and a signature* σ*, it outputs* 1 *to indicate the signature is valid, and* 0 *otherwise.*

Definition 2 (Blindness under Chosen Keys). *For a blind signature BS, define the blindness game* Blind_{BS} *with an adversary* A *as follows:*

– ***Setup:*** *The challenger samples a bit* $\mathsf{coin} \leftarrow\$ \{0, 1\}$ *and runs* A *on input* 1^n*.*
– ***Online Phase:*** A *outputs two messages* M_0^* *and* M_1^* *and a public key* $\mathsf{pk} \in \mathsf{PK}$*. The game checks if* pk *is valid; otherwise, it aborts. If valid, it sets* $(M_0, M_1) = (M^*_{\mathsf{coin}}, M^*_{1-\mathsf{coin}})$*.* A *is given access to the following oracles:*

- ***Oracle*** U_1***:*** *On input* $b \in \{0,1\}$ *and a first-signer message* $\rho_{S,1,b}$*, if session* b *is not yet open, it marks session* b *as open and generates* $(\mathsf{state}_{U,b}, \rho_{U,b}) \leftarrow \mathsf{BS.U1}(\mathsf{pk}, M_b, \rho_{S,1})$*. It returns* $\rho_{U,b}$ *to* A*.*
- ***Oracle*** U_2***:*** *On input* $b \in \{0,1\}$ *and a second-signer message* $\rho_{S,2,b}$*, if session* b *is open, it computes* $\sigma_b \leftarrow \mathsf{BS.U2}(\mathsf{state}_{U,b}, \rho_{S,2,b})$ *and marks session* b *as closed.*

– ***Output:*** *If both sessions are closed and* $\mathsf{BS.Verify}(\mathsf{pk}, M_b, \sigma_b) = 1$ *for both* $b = 0, 1$*, the game returns* $(\sigma_{\mathsf{coin}}, \sigma_{1-\mathsf{coin}})$ *to* A*.* A *then outputs a guess* coin^**. We say that* A *wins if* $\mathsf{coin}^* = \mathsf{coin}$*.*

We say that BS *is blind under chosen keys if the probability that* A *wins is negligible.*

3 The Scheme and Attack

The blind signature scheme from [2] implements a similar framework to the CSI-Otter scheme proposed at Crypto'23. The construction uses a base OR Sigma protocol for a relation on group actions. To elevate this protocol to a blind signature, the approach is first to repeat the protocol n-times to reduce the soundness error and later use randomization techniques to achieve blindness. The main difference between CSI-Otter [3] and the proposed scheme is that it relies on non-commutative group actions, which require the signer to provide more values than otherwise the user would be able to compute using the commutative property of the group action. The scheme itself involves complex details. Thus, in the description below, we extract the essential parts of the scheme to highlight the attack. See figure 4 in [2] for a full scheme description.

We begin with the public key of the signer $\mathsf{pk} = (A_0^{(1)}, A_0^{(-1)}, A_1^{(1)}, A_1^{(-1)})$ where $A_i^{(1)} = g_i * E$ and $A_i^{(-1)} = g_i^{-1} * E$ for a base set element E. The blind signature protocol starts with a commitment $(\mathbf{Y}_0^{(1)}, \mathbf{Y}_0^{(-1)}, \mathbf{Y}_1^{(1)}, \mathbf{Y}_1^{(-1)})$ from the signer (i.e., algorithm BS.S1), where the $\mathbf{Y}$ elements are actually vectors of commitments of size n (recall the number we repeat the base protocol).

In the next step (BS.U1), the user picks two strings of size n, $\mathbf{d}_0, \mathbf{d}_1 \leftarrow\$ \{-1,1\}^n$. The purpose of $\mathbf{d}_0$ and $\mathbf{d}_1$ is to determine which of the $\mathbf{Y}$ elements will be used by the user in the final signature. In particular, the user computes $\mathbf{Z}_b \leftarrow \mathbf{z}_b * \mathbf{Y}_b^{\mathbf{d}_b}$ for a vector of random group elements $\mathbf{z}_b$ and the hash challenge $\mathbf{c} \leftarrow H(\mathbf{Z}_0, \mathbf{Z}_1, M)$ for the message M. Note that $\mathbf{c}$ will be the hash challenge used in the final signature. Therefore, instead of sending $\mathbf{c}$ directly to the signer, the user "blinds" it using $\mathbf{d}_0$ and $\mathbf{d}_1$, i.e., the challenge sent to the signer is $\mathbf{c}^* \leftarrow \mathbf{c} \odot \mathbf{d}_0 \odot \mathbf{d}_1$, where $\odot$ is component-wise multiplication. Note that the hash challenges and the values $\mathbf{d}$ are vectors of size n with elements from set $\{-1,1\}$.

Once the signer obtains $\mathbf{c}^*$ in BS.S2, it prepares responses $(\mathbf{r}_0^{(1)}, \mathbf{r}_0^{(-1)}, \mathbf{r}_1^{(1)}, \mathbf{r}_1^{(-1)})$ based on it. Here, it is worth noting that due to the non-commutive property, the signer provides $4 \cdot n$ different responses, while we will later see the user only uses $2 \cdot n$ out of those values. The signer's response also includes a mask to the challenge $\mathbf{c}^*$ that the signer adds as part of the OR protocol. However, this part is unimportant for the attack, so we omit it.

To finalize the signature in algorithm $\mathsf{BS.U2}$, the user computes $\mathbf{r}_0 = \mathbf{z}_0 * (\mathbf{r}^{(\mathbf{d}_0)})$ and $\mathbf{r}_1 = \mathbf{z}_1 * (\mathbf{r}^{(\mathbf{d}_1)})$ and checks that those responses lead to valid signatures. Otherwise, the algorithm aborts by outputting $\perp$ instead of a valid signature. We now recall the statement we made above. While the signer provides $4 \cdot n$ values in the 4 vectors $(\mathbf{r}_0^{(1)}, \mathbf{r}_0^{(-1)}, \mathbf{r}_1^{(1)}, \mathbf{r}_1^{(-1)})$, the user only uses $2 \cdot n$ of those values. What is more, which values are used depends on the vectors $\mathbf{d}_0$ and $\mathbf{d}_1$, which hide the final challenge $\mathbf{c}$.

The attack we employ is known in the MPC literature as selective failure or selective aborts. The purpose of the adversary is to guess $\mathbf{d}_0$ and $\mathbf{d}_1$ by exploiting the fact that the user does not abort the protocol if the provided $(\mathbf{r}_0^{(1)}, \mathbf{r}_0^{(-1)}, \mathbf{r}_1^{(1)}, \mathbf{r}_1^{(-1)})$ by the signer lead to a valid signature. However, since not all of those values are used by the user, the signer can use dummy (i.e., randomly selected) values instead and check whether the response is accepted by the user.

In more detail, the adversary can first guess $\mathbf{d}_0$, $\mathbf{d}_1$ and use the signing protocol to verify its guess. To do so, instead of sending correct values for $\mathbf{r}_0^{-(\mathbf{d}_0)}$ and $\mathbf{r}_1^{-(\mathbf{d}_1)}$ it sends random values. The adversary's guess was correct if the user produced a valid signature. Note that the user only uses $\mathbf{r}_0^{(\mathbf{d}_0)}$ and $\mathbf{r}_1^{(\mathbf{d}_1)}$ in the protocol and does not verify the validity of $\mathbf{r}_0^{-(\mathbf{d}_0)}$ and $\mathbf{r}_1^{-(\mathbf{d}_1)}$, which cannot be done due to the lack of commutativity in the group action. We will later show that it is actually not necessary to guess the *full* vector $\mathbf{d}_0$, $\mathbf{d}_1$, and an adversary can break blindness with the i-th elements of each of the vectors.

Once the adversary's guess is correct and it knows the correct $\mathbf{d}_0$ and $\mathbf{d}_1$, it can break blindness as defined in definition 2. At the end of the blindness experiment, the adversary receives $(\sigma_{\mathsf{coin}}, \sigma_{1-\mathsf{coin}})$, where one of the two signatures will "hash" to the challenge $\mathbf{c}$. We assume here that the above attack is performed against one of the two sessions between the adversary and the challenger while the other session is executed according to the protocol. Since the adversary knows $\mathbf{c}^*$, $\mathbf{d}_0$ and $\mathbf{d}_1$ it can check if $\mathbf{c}^* \odot \mathbf{d}_0 \odot \mathbf{d}_1$ correspond to the hash challenge for signature σ_{coin} or $\sigma_{1-\mathsf{coin}}$ easily distinguishing the coin picked by the challenger. Thus breaking the blindness experiment.

Interestingly, the above attack is also acknowledged by the authors [2]. In the proof of theorem 2 (page 22), they state the exact same attack as above. See the exact citation below.

In fact, an adversary can guess $\mathbf{d}_0, \mathbf{d}_1$ picked by the user as follows. First, it randomly chooses $\mathbf{d}_0, \mathbf{d}_1$ and changes the way it generates a signature in line 406 by returning random $\mathbf{r}_0^{-(\mathbf{d}_0)}$ and $\mathbf{r}_1^{-(\mathbf{d}_1)}$, hence a different choice of $\rho_{S,2}$. Note that these values are never used by the user in $\mathsf{BS.U2}$*. Thus, the user will return a valid signature in line 508 if the guess is correct. Otherwise, the check in line 507 will fail and the user will return $\perp$ in line 509. It occurs with probability $1/4^n$, which is negligible for n at the security level. This completes the proof.*

While the above statement is true, the adversary does not need to actually guess the whole vector $\mathbf{d}_0$ and $\mathbf{d}_1$. Assume that $\mathbf{d}_0 = (d_{0,1}, \ldots, d_{0,n})$ and $\mathbf{d}_1 = (d_{1,1}, \ldots, d_{1,n})$. It is sufficient that the adversary learns only $d_{0,i}$ and $d_{1,i}$ for a random $i \in [n]$. In other words, we do not need to reveal the real challenge across all n repetitions of the base OR Sigma protocol, but it is enough to reveal it for just one of the instances. Using the above estimation, by randomly guessing the elements $d_{0,i}$ and $d_{1,i}$, which are from $\{-1, 1\}$, the user will not abort the protocol with probability 1/4. Thus, now the adversary knows the correct $d_{0,i}$ and $d_{1,i}$ for one of the two final signatures $(\sigma_{\mathsf{coin}}, \sigma_{1-\mathsf{coin}})$.

Let us now assume that $\mathbf{c}_{\mathsf{coin}}$ and $\mathbf{c}_{1-\mathsf{coin}}$ are the two hash challenges received from the signatures $(\sigma_{\mathsf{coin}}, \sigma_{1-\mathsf{coin}})$. Moreover, assume that $c_{\mathsf{coin}} \in \{-1, 1\}$ and $c_{1-\mathsf{coin}} \in \{-1, 1\}$ correspond to the i-th element of those vectors. Moreover, let $c^* \in \{-1, 1\}$ correspond to the i-th element of the vector $\mathbf{c}^*$ received from the challenger in the session the adversary tries to attack using the abort technique. Without loss of generality, let's assume we target the first session. To output coin the adversary checks if

$$c_{\mathsf{coin}} \stackrel{?}{=} c^* \cdot d_{0,i} \cdot d_{1,i}$$

and

$$c_{1-\mathsf{coin}} \stackrel{?}{=} c^* \cdot d_{0,i} \cdot d_{1,i}.$$

In case both equations are correct, the adversary aborts. Otherwise, it outputs 0 if the first equation holds and 1 if the second one holds. It is worth noting that the adversary aborts if both equations are correct, which only happens in case $c_{\mathsf{coin}} = c_{1-\mathsf{coin}}$. However, since the adversary randomly picks the position i, the probability of this happening is 1/2. Thus, the total advantage of the adversary is $1/4 \cdot 1/2 = 1/8$.

Consequently, the scheme proposed in [2] is not blind despite the claims.

References

1. Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) Advances in Cryptology – CRYPTO'82. pp. 199–203. Plenum Press, New York, USA, Santa Barbara, CA, USA (1982). https://doi.org/10.1007/978-1-4757-0602-4_18
2. Duong, D.H., Khuc, X.T., Qiao, Y., Susilo, W., Zhang, C.: Blind signatures from cryptographic group actions. Cryptology ePrint Archive, Paper 2025/397 (2025), https://eprint.iacr.org/2025/397
3. Katsumata, S., Lai, Y.F., LeGrow, J.T., Qin, L.: CSI-Otter: Isogeny-based (partially) blind signatures from the class group action with a twist. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023, Part III. Lecture Notes in Computer Science, vol. 14083, pp. 729–761. Springer, Cham, Switzerland, Santa Barbara, CA, USA (Aug 20–24, 2023). https://doi.org/10.1007/978-3-031-38548-3_24