

Failure Is Not Silent: Attacks on Blind Signatures

Maksymilian Gorski^{1,2}[0009–0003–5221–8829],
Lucjan Hanzlik¹[0009–0006–1941–693X]

¹ CISP Helmholz Center for Information Security, Saarbrücken, Germany
`{firstname.surname}@cispa.de`

² Saarland University, Saarbrücken, Germany

Abstract. Blind signatures and OPRFs are widely used to build anonymous tokens and privacy-preserving protocols. This paper shows that several prior claims of blindness and request-privacy for such primitives are false under the standard security definitions. In particular, we give explicit malicious-signer/evaluator adversaries showing that the theorems claimed for several existing constructions, including the blind ECDSA scheme of Qin et al. and the verifiable Dark Matter OPRF, do not hold. The attacks exploit a simple but previously overlooked channel: a malicious signer can craft its response so that the honest user’s final verification aborts if and only if the user’s hidden message satisfies a chosen predicate. The abort/no-abort outcome, therefore, becomes a one-bit testing oracle on the hidden message. Importantly, this is not a failure of the standard blindness definition: our attacks win the standard blindness/request-privacy games. Rather, the flaw is that prior proofs implicitly used encryption hiding or commitment hiding in settings where the adversary also learns an input-dependent abort event.

Our main attack applies to a general pattern of homomorphic-encryption-based blind signatures and OPRFs, in which the user encrypts its input, then decrypts and verifies the signer’s homomorphic response. We also present two additional case studies, based on derandomization and selective use of signer components, which illustrate the same proof pitfall in other constructions. We identify a positive design criterion: schemes with publicly checkable signature derivation, in the sense of Fischlin-Schröder, do not leak extra information through such selective aborts. Finally, we discuss mitigations. Generic protection requires publicly verifiable honest evaluation, e.g., via zero-knowledge proofs. For random-token use cases of some HE-based blind signatures, we give a partial mitigation based on random unknown-message blindness (Rumbl) and a random-oracle transformation to standard blindness.

Keywords: Blind Signatures · OPRFs · Selective-Aborts · Blindness

1 Introduction

Blind signatures allow a user to obtain a signature on a message while hiding the message from the signer. The standard blindness game formalizes this privacy guarantee as follows. A malicious signer chooses two messages, interacts with two

honest users who sign them in a hidden order, and must guess the order. Analogously, game-based request privacy for OPRFs asks that a malicious evaluator cannot distinguish which of two client inputs is being evaluated.

This paper shows that several previously claimed results are false in these standard games. We do not propose a new attack model, nor do we argue that the standard definitions are too weak. Instead, we show that a malicious signer/evaluator can use the honest user’s abort behavior as an input-dependent testing oracle, and that several prior proofs failed to account for this oracle. Concretely, our attacks refute the claimed malicious blindness or request privacy guarantees for multiple published or proposed schemes, including the blind ECDSA scheme of Qin et al. [22] and the verifiable Dark Matter OPRF [2].

Blind signatures are a central tool for anonymous tokens, e-cash, anonymous credentials, and privacy-preserving authentication systems. OPRFs provide a closely related primitive in which a client obtains $F_k(x)$ without revealing x to the server. While OPRFs are typically modeled using the Universal Composability (UC) framework to capture all relevant properties, game-based definitions for request privacy also exist. In this paper, we focus strictly on game-based blindness for blind signatures and game-based request privacy for OPRFs, as these two notions are conceptually similar. Our results should not be read as attacks on UC-secure OPRF instantiations unless the specific OPRF construction matches one of the vulnerable patterns analyzed here. The formal way to model the signer’s inability to learn the client’s message is through a game-based indistinguishability notion known as blindness.

The experiment works as follows. After receiving the public key from the challenger, the adversary selects and submits two messages, $\mathbf{m}_0$ and $\mathbf{m}_1$. The challenger then initiates two client oracles: one with $\mathbf{m}_b$ and the other with $\mathbf{m}_{1-b}$, where b is a randomly chosen bit. The adversary is given access to these oracles. After interacting with the oracles, the challenger receives two signatures, (σ_0, σ_1) . If, during any of the interactions, a signature results in $\perp$ (indicating that the user aborted the protocol), the challenger returns $(\perp, \perp)$ to the adversary. This step prevents trivial attacks where the adversary could force an abort in one session, thereby winning the experiment for any scheme. If both signatures are valid, the challenger sends (σ_0, σ_1) to the adversary. The challenge for the adversary is to determine the bit b . It must distinguish whether the first oracle was initialized with $\mathbf{m}_0$ or $\mathbf{m}_1$, and similarly for the second oracle.

The abort rule in the blindness game is meant to prevent trivial wins: if one session aborts, the challenger returns $(\perp, \perp)$ rather than revealing which session failed. Nevertheless, this rule does not hide the fact that some sessions were aborted. A malicious signer can sometimes arrange for the challenged user to abort exactly when the hidden message is not equal to the signer’s guess. The adversary then runs one session honestly and one session maliciously. If the challenger returns valid signatures, the guess was correct; if it returns $(\perp, \perp)$, the guess was wrong. Thus, the honest user’s final verification turns into a one-bit equality test on the hidden message.

This is a standard-game attack. The adversary does not require any information beyond what the blindness/request-privacy experiment already gives it. The lesson is therefore not that the standard definition omits aborts, but that security proofs must show that the probability of abort is independent of the hidden message, or else make aborts publicly predictable from the transcript.

1.1 Our Motivation and Contributions

Our starting point is a proof gap that appears in several prior works. These works prove blindness or request privacy by appealing to the hiding of a ciphertext or commitment. This reasoning is sound only if the adversary’s view consists solely of the ciphertext or commitment and the protocol messages, independent of the hidden message. In the schemes we attack, however, the adversary additionally learns whether the honest user aborted after processing the adversarial response. This abort event may depend on the hidden message. Consequently, the reductions in the prior works do not simulate the adversary’s view, and the corresponding security theorems are false.

Main attack: HE-based blind signatures and OPRFs. Our main contribution is a selective-abort attack against a broad pattern of homomorphic-encryption-based blind signatures and OPRFs. In this pattern, the user encrypts its hidden input, the signer/evaluator homomorphically computes a response, and the user decrypts and verifies the result. A malicious signer/evaluator can modify the homomorphic response so that verification succeeds exactly when the encrypted input satisfies a chosen test. This produces a zero/equality-testing oracle for the hidden input and invalidates proofs that rely solely on the IND-CPA security of the encryption scheme.

Concrete false security claims. We instantiate the attack against existing constructions. For the blind ECDSA scheme of Qin et al. [22], we give an adversary with blindness advantage $1/2$, showing that the claimed malicious blindness theorem is false. We also attack an HE-based blind Schnorr proposal [1] and the verifiable Dark Matter OPRF from Eurocrypt 2024 [2], showing that its claimed game-based request privacy against malicious evaluators does not hold. In each case, we also explain where the original proof strategy breaks down.

Additional case studies. We also describe two further ways in which the same proof pitfall can appear. First, in a derandomizable blind signature, a malicious signer can craft a presignature that derandomizes correctly only for a guessed committed message. Second, in a selective-component construction, the signer can corrupt a component that the user touches only for some messages. These examples should be read as case studies rather than as a claim that all schemes of these types are insecure: the common issue is an input-dependent abort event that is not publicly predictable from the transcript.

A positive criterion. We give a simple sufficient condition that rules out this class of attacks: publicly checkable signature derivation [14]. If the public transcript already determines whether the honest user accepts or aborts, then the user’s abort behavior reveals no additional information to the signer. This con-

Sect.	Construction	Claim	Failure mode / lesson
§3.1	Blind ECDSA [22]	blindness	Final ECDSA verification gives the signer an abort-induced zero/equality test.
§3.2	HE blind Schnorr [1]	blindness	Ciphertext replacement makes the user’s abort event message-dependent.
§3.3	Dark Matter OPRF [2]	request privacy	Verifiability of accepted outputs does not imply input-independent aborts.
§4.1	Derandomizable CL BS [16]	perfect blindness	Derandomization leaks whether the commitment opens to the guessed message.
§4.2	Group-action BS [12]	statistical blindness	Unchecked components leak which message-dependent component was selected.
§5	Generic Rumb1 transform	generic mitigation	A Rumb1-secure blind signature can be upgraded to standard blindness in the ROM.
§6	Rumb1 analysis of blind ECDSA	weaker Rumb1	Random-token use cases survive under the stronger HE zero-testing assumption.
§7	Sign-derivation check [14]	positive criterion	Publicly predictable aborts do not leak additional hidden-message information.

Table 1. Constructions considered in the paper, the privacy claim at issue, and the corresponding abort channel or lesson.

dition is sufficient but not necessary; schemes may also avoid our attacks by proving honest evaluation in zero-knowledge or by providing a scheme-specific proof that aborts are input-independent.

Mitigations and scope. The fully generic mitigation is to force the signer/evaluator to prove correct execution, for example, with zero-knowledge proofs. This may be expensive or unavailable for some constructions. For random-token applications of blind signatures, we also give a partial mitigation: we define random unknown-message blindness (Rumb1), prove that a Rumb1-secure blind signature can be transformed into a standard blindness-secure scheme in the random-oracle model, and apply this methodology to the blind ECDSA scheme. This mitigation does not apply to OPRFs, where the client needs an evaluation of its actual input rather than a random commitment to that input.

1.2 Prior and Related Work

Selective-failure blindness. Camenisch, Neven, and shelat [8] introduced selective-failure blindness, and Fischlin and Schröder [13] later showed how to transform standard blind signatures into selective-failure blind ones using commitments. These notions are often useful when blind signatures are used as components of larger protocols. They do not resolve the issue studied here. Our attacks target the claimed standard blindness/request-privacy of the underlying primitive itself: the schemes we attack were claimed to satisfy the relevant base privacy notion, but their proofs failed to account for input-dependent aborts.

Comparison with Fischlin–Schröder. Fischlin and Schröder study several related but distinct issues. Their work [13] introduces selective-failure blindness, where the adversary learns not only that an abort occurred, but also which of the two user sessions aborted. This is a stronger output convention for the blindness experiment, and it is useful when blind signatures are used as subroutines in larger protocols. However, selective-failure blindness is orthogonal to the issue studied here. A correct proof of standard blindness would already rule out our attacks, since our adversaries win the standard blindness games. The problem is therefore not that standard blindness is too weak, nor that selective-failure blindness is the missing definition. Rather, the schemes we attack were claimed to satisfy the ordinary base privacy notion. Still, their proofs appeal to hiding of a ciphertext or commitment while failing to simulate the user’s input-dependent accept/abort event. Their work [14] introduces signature-derivation checks for a different purpose: to establish impossibility results for three-move blind signatures. We use this notion positively. If a transcript publicly determines whether the user obtains a valid signature, then observing the user’s abort behavior provides the signer with no additional information.

Finally, our Rumbl-to-blind transformation is superficially similar to the Fischlin–Schröder commitment transformation, but the proof requirements are different. Their transformation starts from a blind scheme and upgrades it to selective-failure blindness using ordinary commitments. Our transformation starts from the weaker Rumbl notion and must equivocate commitments to messages chosen later by the blindness adversary. This is why we use the random oracle: ordinary Pedersen-style commitments do not provide the reduction with the opening/trapdoor information it needs.

Attacks on Blind Signatures. Most prior attacks on blind signatures target unforgeability, for example, the well-known concurrent attacks on Schnorr-type blind signatures [4] and later attacks on Schnorr-type blind signatures with parallel repetition used for post-quantum secure schemes [11, 19]. Attacks on blindness have received less systematic attention. Our work contributes to this direction by identifying a recurring proof pitfall: hiding of an encrypted or committed message is insufficient when the adversary also learns a verification-dependent abort bit.

Weaker blindness for random tokens. Lysyanskaya [21] studied a weaker blind-token notion in which messages are sampled from a distribution rather than chosen directly by the adversary. Our Rumbl notion can be viewed as the special case relevant for random-token applications: the challenge messages are uniformly random and not adversarially chosen. This restriction is too weak for general-purpose blind signatures. Still, it enables a random-oracle transformation from Rumbl security to standard blindness by signing a commitment to the real message under a random oracle.

Selective-failure attacks in other primitives. Input-dependent aborts are a classical issue in secure computation and have also appeared in settings such as PIR. Existing mitigations in those areas usually make aborts independent of the

private input or add validation mechanisms to detect malicious behavior. The same high-level lesson applies here, but the technical fixes do not transfer directly. In blind signatures and OPRFs, the message is often algebraically embedded in a signing/evaluation computation that cannot be simply encoded or checked without changing the primitive.

2 Preliminaries

For a set X , $|X|$ denotes the cardinality of X , and for a positive integer k we write $[k] := \{1, \dots, k\}$. We denote the security parameter by λ . For convenience, vectors are written in bold, e.g., $\mathbf{v} = (v_1, \dots, v_k)$, and we use $\|$ for string concatenation. We use uppercase calligraphic or sans-serif letters, such as $\mathcal{A}$ or $\mathbf{A}$, to denote algorithms. We write $y \leftarrow \mathcal{A}(x)$ for the output of a deterministic algorithm and $y \leftarrow_{\$} \mathcal{A}(x)$ when $\mathcal{A}$ is randomized. For a finite set X , the notation $x \leftarrow_{\$} X$ means that x is sampled uniformly from X . We use st for the internal state of an algorithm and omit it when it is clear from context. A function is negligible, denoted $\text{negl}(\lambda)$, if it vanishes faster than the inverse of any polynomial in λ .

We give a high-level overview of the cryptographic primitives and models used in the paper; complete definitions appear in the supplementary material. Necessary notation for the concrete attacks is introduced inline. We will also use the algebraic group model (AGM) [15]. In the AGM, an adversary may manipulate group elements only algebraically: whenever it outputs a group element Z , it must also provide a representation $\mathbf{z}$ expressing Z as a known combination of the group elements it has previously received. This representation requirement makes the adversary’s group operations traceable and allows reductions to extract the algebraic relations used by the adversary.

Blind Signatures. This work considers blind signature schemes that enable an issuer to sign a user’s message (e.g., a document) without learning its contents, while also ensuring that a presented valid signature cannot be linked back to the signing interaction. Formally, a blind signature scheme BS over message space $\mathcal{M}$ is a tuple of PPT algorithms $(\text{KeyGen}, \langle \text{Sign}, \text{User} \rangle, \text{Ver})$. The key-generation algorithm $\text{BS.KeyGen}(1^\lambda)$ outputs (sk, pk) ; the interactive protocol $\langle \text{BS.Sign}(\text{sk}), \text{BS.User}(\text{pk}, \mathbf{m}) \rangle$ runs between signer and user and yields the user’s output σ (or $\perp$ on failure); and $\text{BS.Ver}(\text{pk}, \mathbf{m}, \sigma)$ checks signature validity. The interaction may be multi-round; unless specified, we adopt a round-independent definition of blindness. As a compact example, a two-move instantiation can be written as $(\text{bsm}_0, \text{st}) \leftarrow \text{BS.User}_0(\text{pk}, \mathbf{m})$, $(\text{bsm}_1, b) \leftarrow \text{BS.Sign}_0(\text{sk}, \text{bsm}_0)$, $\sigma \leftarrow \text{BS.User}_1(\text{st}, \text{bsm}_1)$, where bsm_i denotes the messages exchanged and st the user state. If the signer initiates the protocol we omit BS.User_0 ; otherwise the same notation captures more rounds by indexing additional oracle steps.

Blindness is defined via the standard challenge experiment. A scheme BS is blind if every PPT adversary $\mathcal{A}$ has negligible advantage $\text{Adv}_{\text{Blind}}(\mathcal{A}) = |\Pr[\text{Blind}_{\mathcal{A}, \text{BS}}(\lambda) = 1] - \frac{1}{2}|$. In $\text{Blind}_{\mathcal{A}, \text{BS}}(\lambda)$, the adversary acts as a malicious signer and may generate pk itself. It first outputs $(\text{st}_0, \text{pk}, \mathbf{m}_0, \mathbf{m}_1)$, after which the challenger samples $b \leftarrow_{\$} \{0, 1\}$ and gives $\mathcal{A}$ oracle access to two honest users

initialized with $\mathbf{m}_b$ and $\mathbf{m}_{1-b}$:

$$\text{st}_1 \leftarrow \mathcal{A}^{(\cdot, \text{BS.User}(\text{pk}, \mathbf{m}_b))^1, (\cdot, \text{BS.User}(\text{pk}, \mathbf{m}_{1-b}))^1}(\text{st}_0).$$

If either user session outputs $\perp$, the challenger gives $\mathcal{A}$ only $(\perp, \perp)$; otherwise it gives $\mathcal{A}$ the signatures (σ_0, σ_1) in message order. The adversary then outputs $\hat{b}$, and the experiment returns 1 iff $\hat{b} = b$. For completeness, we give the fully formal experiment in the supplementary material.

Oblivious Pseudorandom Functions (OPRFs). An OPRF is an interactive protocol between a client and a server in which the client obtains $z = F(\text{sk}, x)$ for a pseudorandom function F on an input x of its choice while the server learns nothing about x or z , and the client learns nothing about the server's key sk . A partial OPRF (POPRF) additionally exposes a cleartext tag t to the server. Formally, an OPRF over input space $\mathcal{X}$ is a tuple of PPT algorithms (KeyGen, Request, BlindEval, Eval, Finalise) where $\text{OPRF.KeyGen}(1^\lambda)$ produces (sk, pk) ; $\text{OPRF.Request}(\text{pk}, t, x)$ produces a request req ; $\text{OPRF.BlindEval}(\text{sk}, t, \text{req})$ produces a response rep ; $\text{OPRF.Eval}(\text{sk}, t, x)$ outputs the direct evaluation z ; and $\text{OPRF.Finalise}(\text{rep})$ recovers z (or $\perp$ on failure). The primary privacy property we use is request-privacy (POPRIV) against malicious servers: informally, no PPT adversary acting as a malicious server (and allowed oracle access to Request and Finalise, and to maliciously choose pk) can distinguish which of two challenge inputs the honest client used. This is captured by the experiment $\text{POPRIV}_{\mathcal{A}, \text{OPRF}}^2(1^\lambda)$, in which a hidden bit b selects which request oracle output the adversary receives; the adversary's advantage $\text{Adv}_{\text{POPRIV}^2}(\mathcal{A}) = |\Pr[\text{POPRIV}_{\mathcal{A}, \text{OPRF}, 0}^2(1^\lambda) = 1] - \Pr[\text{POPRIV}_{\mathcal{A}, \text{OPRF}, 1}^2(1^\lambda) = 1]|$ must be negligible for security. For completeness, we provide the formal definition of the POPRIV^2 in the supplementary material (see Fig. 7).

Homomorphic Encryption. A homomorphic encryption scheme HE is a tuple of PPT algorithms (KeyGen, Enc, Dec, Eval) with message space a ring $\mathcal{R}_{\mathcal{M}}$ and a family $\mathcal{F}$ of arithmetic circuits over $\mathcal{R}_{\mathcal{M}}$. Concretely, $\text{HE.KeyGen}(1^\lambda)$ outputs (sk, pk) ; $\text{HE.Enc}(\text{pk}, \mathbf{m})$ produces a ciphertext ct for $\mathbf{m} \in \mathcal{R}_{\mathcal{M}}$; $\text{HE.Dec}(\text{sk}, \text{ct})$ returns a message or $\perp$; and $\text{HE.Eval}(\text{pk}, f, \{\text{ct}_i\}_{i=1}^\ell)$ takes $f \in \mathcal{F}$ and ciphertexts $\text{ct}_1, \dots, \text{ct}_\ell$ and outputs a ciphertext ct_f intended to encrypt $f(m_1, \dots, m_\ell)$. Correctness requires that for all $f \in \mathcal{F}$ and $m_1, \dots, m_\ell \in \mathcal{R}_{\mathcal{M}}$, if each $\text{ct}_i \leftarrow \text{HE.Enc}(\text{pk}, m_i)$ then $\text{HE.Dec}(\text{sk}, \text{ct}_f) = f(m_1, \dots, m_\ell)$ except with negligible probability; the scheme is *compact* if decryption cost depends only on 1^λ (not on the circuit). Security is standard IND-CPA: no PPT adversary $\mathcal{A}$ has non-negligible advantage $\text{Adv}_{\text{CPA}, \text{HE}}(\mathcal{A}) = |\Pr[\text{CPA}_{\mathcal{A}, \text{HE}}(1^\lambda) = 1] - \frac{1}{2}|$ in distinguishing encryptions.

Non-Interactive Zero-Knowledge Proof Systems. We recall the syntax of non-interactive zero-knowledge proofs for $\mathcal{L}_{\mathcal{R}}$, which consists of algorithms: $\text{crs} \leftarrow \text{Setup}(1^\lambda)$, $\pi/\perp \leftarrow \text{Prove}(\text{crs}, \text{stmt}, \text{wit})$ and $\{0, 1\} \leftarrow \text{Verify}(\text{crs}, \text{stmt}, \pi)$. In the supplementary material, we also provide the well-known definitions of completeness, soundness, and zero-knowledge for completeness. We will use $\text{Adv}_{\text{ZK}}(\mathcal{A})$ to denote the advantage of an adversary in distinguishing simulated proof in the zero-knowledge property. NIZKs can be instantiated in the random

BS.User ₀ (pk, m)	BS.User ₁ (pk, bsm ₁ , st)
$(sk_{HE}, pk_{HE}) \leftarrow HE.KeyGen(1^\lambda)$	$(sk_{HE}, m) \leftarrow st$
$r' \leftarrow \mathcal{R}$	$ct_f \leftarrow bsm_1$
$ct_m \leftarrow HE.Enc(pk_{HE}, m)$	$\sigma \leftarrow HE.Dec(sk_{HE}, ct_f)$
$ct_{r'} \leftarrow HE.Enc(pk_{HE}, r')$	if BS.Ver(pk, m, σ) = 0 then
$bsm_0 \leftarrow (pk_{HE}, ct_m, ct_{r'})$	return $\perp$
$st \leftarrow (sk_{HE}, m)$	return σ
return (bsm ₀ , st)	
BS.Sign ₀ (sk, bsm ₀)	
$(pk_{HE}, ct_m, ct_{r'}) \leftarrow bsm_0$	
return $\perp$	
$f(sk, \cdot) \leftarrow$ circuit evaluating signature using sk	
$ct_f \leftarrow HE.Eval(pk_{HE}, f, (ct_m, ct_{r'}))$	
$bsm_1 \leftarrow ct_f$	
return (bsm ₁ , $\perp$)	

Fig. 1. Generic prototype for a Blind Signature scheme based on Homomorphic Encryption.

oracle model without a **Setup** algorithm. To simplify the description of the schemes in the paper, we will implicitly assume that the **crs** is provided to all the parties, and we will omit it as an explicit parameter.

3 Homomorphic Encryption based Blind Signatures

Before detailing the selective-abort attack and its applicability to the attacked schemes, we give a generic prototype for blind signatures constructed from homomorphic encryption (HE). This blueprint, illustrated in Fig. 1, captures the core interactive elements common to such schemes. The user generates a fresh keypair for the HE scheme and encrypts their message m , yielding ct_m . Depending on the underlying signature structure, the user may also sample and encrypt client-side randomness r' into a ciphertext $ct_{r'}$, and provide the zero-knowledge proof of the ciphertext(-s) well-formedness (for clearance, omitted in the scheme in Fig. 1). These ciphertexts are then transmitted to the signer. The signer homomorphically evaluates a circuit f (representing the signing operation) over the received ciphertexts and the signing key sk to compute the response ciphertext ct_f . The user then decrypts ct_f to recover a signature σ ; if the signature fails the final verification step, the user aborts. As we will demonstrate - this final abort condition, in specific cases, may act as the oracle that allows to break the underlying scheme's blindness.

We first analyze the main vulnerable pattern: blind signatures and OPRFs in which the user encrypts its hidden input and the signer/evaluator homomorphically computes a response. Many proofs for such schemes argue that blindness follows from the encryption scheme's IND-CPA security. This argument misses the fact that the adversary does not merely see ciphertexts. It also learns whether the honest user accepts after decrypting and verifying the adversarially chosen response. The final verification step can therefore serve as a zero-test on an encrypted value. This section formalizes that attack pattern and then instantiates it against concrete schemes.

Unforgeability of this construction is based on the security of the underlying signature scheme, while blindness is typically argued from the IND-CPA security of the homomorphic encryption scheme. Unfortunately, the latter intuition is wrong and is the source of our attack. In a malicious setting, the client might abort if the adversary behaves dishonestly. In those cases, the blindness experiment returns $\perp$ as a sign that the client aborted both challenge sessions independent of the results of the second session. Note that the adversary's goal is to distinguish the order in which two client sessions are initialized with messages it picked, i.e., the order is decided by the challenger based on a bit b ; session one uses message m_b while the second one uses message m_{1-b} , where the adversary picks m_0, m_1 .

The idea behind the attack is as follows. The adversary picks a random bit $\hat{b}$ and behaves honestly for the second session. For the first session (initialized with m_b), it constructs the responses maliciously in a way that if the guess was correct, i.e., $b = \hat{b}$, then this session does not abort and the adversary receives valid signatures at the end. Otherwise, if $b \neq \hat{b}$, then the session should abort, and the adversary will receive $(\perp, \perp)$ instead. It remains to show how to generate such a malicious response. Recall that adversary receives the ciphertexts $\text{ct}_{m_b} = \text{HE.Enc}(\text{pk}, m_b)$ and $\text{ct}_{r'} = \text{HE.Enc}(\text{pk}, r')$ it uses to compute the response ciphertext ct_f . If the adversary behaves honestly and ct_f encrypts the signature, the client will not abort the protocol. In our attack, the adversary uses ct_{m_b} and computes the ciphertext ct_{zero} for plaintext $m_b - m_{\hat{b}}$. This can be done using homomorphic addition without the need for multiplication. The main observation is now that the ciphertext ct_{zero} is encrypting 0 in the case $b = \hat{b}$ and a non-zero value otherwise. Instead of sending out ct_f to the client, the adversary sends $\text{ct}_{f'}$, where $\text{ct}_{f'}$ is constructed by homomorphically adding ciphertexts ct_f and ct_{zero} . In case the adversary guesses bit b correctly ($b = \hat{b}$), then $\text{ct}_{f'}$ and ct_f contain the same plaintext. Otherwise, $\text{ct}_{f'}$ will encrypt a signature that will fail the verification of the client, leading to the client aborting the protocol.

Interestingly, the attack becomes more straightforward in the case of blind signature built from unique signatures. In those scenarios, there is no need for client randomness (i.e., ciphertext $\text{ct}_{r'}$), and consequently, the adversary's attack can be performed as follows. Given the guessed bit $\hat{b}$, the adversary omits the ciphertext ct_{m_b} and computes its own ciphertext $\text{ct}_{\text{guess}} = \text{HE.Enc}(\text{pk}, m_{\hat{b}})$. It then computes the response ct_f using ct_{guess} in the evaluation instead of the ciphertext provided by the client. It is easy to see that if $b = \hat{b}$, then ct_f will encrypt the correct unique signature and an invalid one otherwise.

The attack works generically against any scheme that uses the above-described folklore construction as a blueprint, assuming there are no additional steps taken to ensure the honest behavior of the signer, e.g., a zero-knowledge proof of correct computation of ct_f , and the client decrypts the ciphertext it received from the signer without further checks. We will now show how to attack concrete schemes constructed from homomorphic encryption using selective aborts. We start with a blind ECDSA construction published at Esorics 2021 [22]. Next, we focus on an unpublished post-quantum construction [1] that was withdrawn from ePrint and dropped following our responsible disclosure of this attack to the authors. Lastly, we show that this type of selective-abort attack can also be used against oblivious PRF constructions relying on homomorphic encryption. In particular, we show that the verifiable version of the Dark Matter shallow OPRF published at Eurocrypt 2024 [2] does not provide the claimed privacy guarantees.

3.1 Blind ECDSA [22]

This subsection will provide more details behind the generic attack and apply it directly to the blind ECDSA scheme from [22]. The attacked scheme uses additive homomorphism to compute the ECDSA signature jointly. Recall that an ECDSA signature consists of $r = f_q(g^k)$ and $s = k^{-1} \cdot (\text{H}(\text{m}) + r \cdot \text{sk})$. First, observe that any blind ECDSA signature scheme needs to ensure that the user and the signer jointly generate the value k . If the signer picks this value, then there is no blindness since the value r can be linked back to the signature's issuing process. On the other hand, if the user picks k , then the scheme is not unforgeable since the sk can be computed using k and s .

Therefore, in the scheme from [22], the signer first commits to a value $R_a \leftarrow g^{k_a}$ and shares it with the user. The user then computes the final r as $f_q(R_a^{k_b}) = f_q(g^{k_a \cdot k_b})$ using its share k_b , making the value k of the final signature a multiplicative share between parties. The user then generates a homomorphic encryption scheme HE and constructs two ciphertexts ct_1 and ct_2 , encrypting r and $h = \text{H}(\text{m})$ respectively. The user also provides proof of correctness π for those ciphertexts, i.e., using a proof system for the relation

$$\mathcal{R} = \{(\text{m}_1, \text{m}_2, r_1, r_2) : \text{ct}_1 \leftarrow \text{HE}.\text{Enc}(\text{pk}_{\text{HE}}, \text{m}_1; r_1) \wedge \text{ct}_2 \leftarrow \text{HE}.\text{Enc}(\text{pk}_{\text{HE}}, \text{m}_2; r_2) \wedge \text{m}_1, \text{m}_2 \in \mathbb{Z}_q\}$$

It then sends $\text{ct}_1, \text{ct}_2, \pi$ to the signer. The signer uses the homomorphic properties of HE to compute a ciphertext ct for the plaintext $k_a^{-1}(h + r \cdot \text{sk})$, which is sent to the user. Note that this can be done only using an additively homomorphic HE . If the message space $|\mathcal{M}|$ of the encryption scheme is not $\mathbb{Z}_q$, the scheme requires that $|\mathcal{M}|$ is larger than $q^3 + q^2$ to ensure that the result $k_a^{-1}(h + r \cdot \text{sk})$ is over integers. To ensure that the decrypted value does not leak additional information due to not being an element of $\mathbb{Z}_q$, the signer computes a factor $t \leftarrow_{\$} [|\mathcal{M}|/q - q^2 - q]$ and adds the factor $t \cdot q$ to the plaintext, i.e., ct encrypts $k_a^{-1}(h + r \cdot \text{sk}) + t \cdot q$. Finally, the user decrypts ct , receiving s' , computes

$s \leftarrow s' \cdot k_b^{-1}$ and aborts in case $s = 0$ or (r, s) is an invalid ECDSA signature. We give more details about the scheme in the supplementary material (see Fig. 9).

We attack this scheme by observing that the user aborts the protocol in case (r, s) is an invalid ECDSA or, in other words, the decrypted value modulo q is not equal to $k_a^{-1}(h + r \cdot sk)$. A malicious signer can use this fact as a guessing oracle. Let $m_{\bar{b}}$ be one of the two messages m_0, m_1 that the adversary specifies in the blindness experiment, and $\bar{b}$ is the adversary's guessed bit. The malicious signer computes the ciphertext ct_{zero} for the plaintext $m - m_{\bar{b}}$, which is 0 in case the adversary guesses the correct message m (i.e., either m_0 or m_1) and some random offset otherwise. After computing the response ciphertext ct , the malicious signer can add the plaintext from ct_{zero} to the plaintext of ct . After the decryption and computing taking the result modulo q , the user will end up with $k_a^{-1}(h + r \cdot sk) + 0$ or $k_a^{-1}(h + r \cdot sk) + (m - m_{\bar{b}})$. The former is a valid response, and the user will not abort. However, for the latter value, the final signature (r, s) will be invalid due to the non-zero factor $(m - m_{\bar{b}})$.

To break the blindness of the scheme, an adversary performs the above attack for one of the sessions while behaving honestly for the second one. If, at the end of the experiment, the adversary receives $(\perp, \perp)$ instead of valid ECDSA signatures, it knows that one of the sessions aborted and can infer that the bit $\bar{b}$ was a wrong guess and return $b = 1 - \bar{b}$. Otherwise, the adversary knows that both sessions did not abort; therefore, the bit $b = \bar{b}$. More formally, we show that $\text{Adv}_{\text{Blind}}(\mathcal{A}) = 1/2$ for the blind ECDSA scheme, which is non-negligible.

Theorem 1. *There exists an adversary $\mathcal{A}$ against the blind ECDSA scheme from [22] (Fig. 9) for which $\text{Adv}_{\text{Blind}}(\mathcal{A}) = 1/2$.*

Proof. The adversary $\mathcal{A}$ honestly generates the signing keys by running $(sk, pk) \leftarrow \text{BS.KeyGen}(1^\lambda)$ and picks two messages m_0, m_1 , which it returns to the challenger. The adversary then samples a bit $\bar{b}$. It then initiates both user sessions and finishes one according to protocol. For the second session, the adversary $\mathcal{A}$ executes the $\text{BS.SignAttack}_1(sk, bsm_1, st, m_{\bar{b}})$ (Fig. 2) instead of the honest algorithm $\text{BS.Sign}_1(sk, bsm_1, st)$ to receive the response bsm_2 . At the end of both interactions, the challenger provides (σ_0, σ_1) to the adversary. The adversary outputs $1 - \bar{b}$ if $(\sigma_0, \sigma_1) = (\perp, \perp)$ and otherwise the bit $\bar{b}$. $\square$

Where the proof of Qin et al. breaks. The blindness proof relies on the intuition that the ciphertexts sent by the user hide $h = H(m)$ and r by the IND-CPA security of the homomorphic encryption scheme. This intuition would be sufficient if the adversary's view ended with these ciphertexts. In the malicious-signer experiment, however, the adversary also chooses the ciphertext returned to the user and then learns whether the user's final ECDSA verification accepts or aborts. Our attack uses this extra information as follows. The signer returns a ciphertext whose decryption is a valid ECDSA response exactly when the encrypted hash equals the signer's guess. Thus, the abort event is a zero/equality test on the encrypted plaintext. Standard IND-CPA security gives no guarantee in the presence of such a test, and a reduction that replaces encryptions of m_0

HEZeroAttack ($\text{pk}_{\text{HE}}, \text{ct}_{m_b}, \text{ct}_f, m_{\hat{b}}$)	
$f_{\text{zero}}(X) \leftarrow X - m_{\hat{b}} ; f_{\text{add}}(Y, Z) \leftarrow Y + Z$ $\text{ct}_{\text{zero}} \leftarrow \text{HE.Eval}(\text{pk}_{\text{HE}}, f_{\text{zero}}, \text{ct}_{m_b}) \quad // \text{Dec}(\text{ct}_{\text{zero}}) = 0 \text{ iff } m_b = m_{\hat{b}}$ $\text{ct}'_f \leftarrow \text{HE.Eval}(\text{pk}_{\text{HE}}, f_{\text{add}}, (\text{ct}_f, \text{ct}_{\text{zero}}))$ return ct'_f	
OPRF.BlindEvalAttack ($\text{sk}, \text{req}, x_{\hat{b}}$)	BS.SignAttack₁ ($\text{sk}, \text{bsm}_1, \text{st}, m_{\hat{b}}$)
$(\text{pk}_{\text{HE}}, \text{ct} = (\text{ct}_1, \dots, \text{ct}_{v \cdot (\beta+1)}), \pi) \leftarrow \text{req}$ if π does not verify then abort $j \leftarrow \$ \{1, \dots, v \cdot (\beta+1)\}$ $g(X) \leftarrow F_{\mathbf{A}}(\mathbf{t}, X)$ $\text{ct}_j \leftarrow \text{HE.Enc}(\text{pk}_{\text{HE}}, x_{\hat{b}})$ $\text{ct}' \leftarrow \text{HE.Eval}(\text{pk}_{\text{HE}}, g, \text{ct})$ $\text{rep} \leftarrow \text{ct}'$ return rep	$(\text{pk}_{\text{HE}}, \text{ct}_1, \text{ct}_2, \pi) \leftarrow \text{bsm}_1$ if π does not verify then abort $h_{\hat{b}} \leftarrow H(m_{\hat{b}}) ; k \leftarrow k_a^{-1}$ $t \leftarrow \$ [\mathcal{M} /q - q^2 - q]$ $g(X) \leftarrow k \cdot (h_{\hat{b}} + X \cdot \text{sk}) + t \cdot q$ $\text{ct} \leftarrow \text{HE.Eval}(\text{pk}_{\text{HE}}, g, \text{ct}_1)$ $\text{bsm}_2 \leftarrow \text{ct}$ return $(\text{bsm}_2, \cdot)$

Fig. 2. Generic and concrete HE selective-abort attacks. The top procedure uses the notation from Section 3: starting from the user’s challenge ciphertext ct_{m_b} and an honestly computed response ciphertext ct_f , the adversary computes ct_{zero} encrypting $m_b - m_{\hat{b}}$ and returns ct'_f , obtained by homomorphically adding ct_{zero} to ct_f . Thus, ct'_f decrypts to the honest response exactly when $m_b = m_{\hat{b}}$, and to a malformed response otherwise. The two lower procedures are concrete replacement shortcuts for the Dark Matter verifiable OPRF scheme [2] and the blind ECDSA scheme [22].

with encryptions of m_1 cannot preserve the distribution of the abort event. This is the precise point at which the claimed blindness proof fails.

3.2 HE-based Round-Optimal Blind Schnorr Signature from [1]

We now discuss another example of a blind signature scheme that uses homomorphic encryption and is vulnerable to our attack. While the scheme appeared only on ePrint, it was withdrawn after we notified the authors of the attack. Using homomorphic encryption, the authors attempted to construct a two-move blind signature in which the final signature is a standard Schnorr signature.

The scheme begins with the user generating a keypair $(\text{pk}_{\text{HE}}, \text{sk}_{\text{HE}})$ for a fully homomorphic encryption scheme. It then computes ciphertexts encrypting the message $\text{ct}_m \leftarrow \text{HE.Enc}(\text{pk}_{\text{HE}}, m)$ and $\text{ct}_{r_u} \leftarrow \text{HE.Enc}(\text{pk}_{\text{HE}}, r_u)$, where $r_u = g^{k_u}$ for a randomly chosen $k_u \leftarrow \$ \mathbb{Z}_q^*$. The user then creates a proof π_m and π_{r_u} showing the ciphertexts’ well-formedness. The scheme defines additional steps for the user, which allow the reduction in the unforgeability experiment to online-extract the secret key sk_{HE} . Hence, we omit those steps as they are unnecessary when describing the attack. After receiving the public key pk_{HE} , ciphertexts ct_m and

ct_{r_u} , the signer verifies all the proofs and aborts in case they are invalid. It then computes its share to the randomness $k_s \leftarrow \mathbb{Z}_q^*$ and the group element $r_s = g^{k_s}$. The signer also generates a standard proof of discrete logarithm knowledge π_{k_s} of k_s . It then homomorphically computes the ciphertext ct_s using $\text{pk}_{\text{HE}}, \text{ct}_m, \text{ct}_{r_u}$ in a way that the plaintext in ct_s is equivalent to $k_s + \text{sk} \cdot \text{H}(m, r_u \cdot g^{k_s})$ for the signer's secret key sk . The signer then sends $r_s, \pi_{r_s}, \text{ct}_s$ to the user. The user first verifies the proof π_{r_s} , computes $r = r_s \cdot r_u$, and decrypts ciphertext ct_s receiving s' . The user then computes $s = s' + k_u$ and checks if (r, s) is a valid Schnorr signature under m . The user aborts the protocol if (r, s) is invalid. Otherwise, it returns (r, s) . Note that $s = s' + k_u = k_s + \text{sk} \cdot \text{H}(m, r_u \cdot g^{k_s}) + k_u = k + \text{sk} \cdot \text{H}(m, r)$ is a valid signature with commitment $r = r_s \cdot r_u = g^{k_s} \cdot g^{k_u} = g^k$ and message m .

Theorem 2. *There exists an adversary $\mathcal{A}$ against the Blind Schnorr Signature scheme from [1] (Fig.3 in [1]) for which $\text{Adv}_{\text{HBlind}}(\mathcal{A}) = 1/2$.*

Proof. Follows similarly to the proof of Theorem 1 and the concrete attack described in Fig. 2. The main differences are as follows. Instead of generating the signing keys (sk, pk) honestly, the adversary $\mathcal{A}$ receives them from the challenger. In the next step, it picks two messages m_0, m_1 , which it returns to the challenger. The adversary then samples a bit $\bar{b}$. It then initiates both user sessions and finishes one according to protocol. For the second session, the adversary $\mathcal{A}$ proceeds as follows. Instead of using the ciphertext ct_m received from the user for the computation, the adversary computes a fresh ciphertext $\text{ct}'_m \leftarrow \text{HE.Enc}(\text{pk}_{\text{HE}}, m_{\bar{b}})$ and proceeds according to the protocol for the remaining part. At the end of both interactions, the challenger provides (σ_0, σ_1) to the adversary. The adversary outputs $1 - \bar{b}$ if $(\sigma_0, \sigma_1) = (\perp, \perp)$ and otherwise the bit $\bar{b}$. $\square$

Where the proof strategy fails. As in the blind ECDSA attack, the proof cannot rely solely on the semantic security of the ciphertext that encrypts the user's message. A malicious signer can replace the message ciphertext with an encryption of its guess and then observe whether the user's final Schnorr verification accepts. The abort event therefore distinguishes whether the hidden message equals the guess. Any simulation that changes the encrypted message must also preserve this accept/abort behavior, which the IND-CPA argument alone does not provide.

3.3 Dark Matter OPRF [2]

We will now apply our attack directly to the OPRF scheme from [2]. The attacked scheme uses the dark matter PRF [6] as the building block PRF since it provides a shallow (in the number of multiplications) evaluation circuit, which can efficiently be implemented using leveled homomorphic schemes. We will define it as the function $F_{\mathbf{A}}(\mathbf{t}, \mathbf{x})$, which takes as input tag $\mathbf{t}$ from space $\mathbb{Z}_p^n$ and point $\mathbf{x}$ from space $\mathbb{Z}_p^n$ and outputs element $\mathbf{z}$ from $\mathbb{Z}_q^m$ with $p = 2$ and $q = 3$.

The idea behind the scheme is to let the client generate a homomorphic encryption scheme with keys $(\text{sk}_{\text{HE}}, \text{pk}_{\text{HE}})$, encrypt its input point $\mathbf{x}$, and generate a proof π of well-formedness of the public key pk_{HE} and ciphertext ct . Before

homomorphically evaluating the function $F_{\mathbf{A}}$ on ct , the evaluator checks the proof π . Unfortunately, the above version is only a recipient-private scheme against semi-honest evaluators. This simple scheme also misses the verifiability property, i.e., a property that ensures consistency of the evaluation with respect to the values committed in the public key. In other words, a verifiable OPRF ensures that the output of the OPRF is consistent with the evaluation of the underlying PRF for the key committed in the public key pk .

To ensure verifiability, the authors use a cut-and-choose technique. The evaluator commits to β points $\hat{\mathbf{x}}_i$ for $i \in [\beta]$ and their evaluations $\hat{\mathbf{z}}_i$. Additionally, it provides a proof π that those points were evaluated using the same function $F_{\mathbf{A}}$ for a secret $\text{sk} = \mathbf{A}$. Given a point $\mathbf{x}$, the client runs the base protocol above, but instead of encrypting just $\mathbf{x}$, it does the following. It prepares a vector of inputs by replicating $\mathbf{x}$ and all $\hat{\mathbf{x}}_i$ v -times, so there are $v \cdot (\beta + 1)$ elements in the vector and then applies a permutation on this vector. The client then runs the protocol for the whole vector instead of one input. This can be done by independently running $v \cdot (\beta + 1)$ the semi-honest protocol or reusing the same HE and sending $v \cdot (\beta + 1)$ ciphertext in the protocol. The scheme in [2] uses the second approach. The evaluator runs the homomorphic evaluation on each ciphertext separately and sends the response to the user. The client decrypts all ciphertexts and first checks that for every $i \in [\beta]$ there are exactly v occurrences of $\hat{\mathbf{z}}_i$ and the remaining v values are the same, which is the evaluation of $\mathbf{x}$. If any of the tests fail, the client aborts.

This technique ensures the verifiability of the scheme since the adversary would have to guess the positions of ciphertexts encrypting $\mathbf{x}$ to break verifiability. To decrease the adversary's chances, the authors propose to have κ fixed points evaluated in the public key pk , while the client only picks β of those for the protocol execution. The authors also introduce a batch evaluation of α points at once. Still, for simplicity and consistency with the recipient privacy definition, we stick to the case of $\alpha = 1$, i.e., a single point is evaluated. Our attack applies to the batch evaluation scenario with the same advantage. We will now discuss our attack in detail and more formally.

Theorem 3. *There exists an adversary $\mathcal{A}$ against the Dark Matter verifiable OPRF scheme from [2] (Fig. 10) for which $\text{Adv}_{\text{POPRIV}^2}(\mathcal{A}) = 1/2$.*

Proof. We will prove the theorem by constructing adversary $\mathcal{A}$. The adversary begins the experiment by honestly generating the public key pk and the secret key sk used for evaluation. It then is provided access to two oracles. The adversary starts the attack by picking two random inputs $(\mathbf{x}_0, \mathbf{x}_1)$ and querying the $\text{Request}(\text{pk}, t, x_0, x_1)$ oracle (for an arbitrary tag t). The oracle returns $(\text{req}_0, \text{req}_1)$ (where req_0 corresponds to x_b for some bit b). The adversary honestly parses the first request by running $\text{rep}_0 \leftarrow \text{OPRF}.\text{BlindEval}(\text{sk}, t, \text{req}_0)$. It picks a random bit $\bar{b}$ and then computes the second response as $\text{rep}_1 \leftarrow \text{OPRF}.\text{BlindEvalAttack}(\text{sk}, t, \text{req}_1, x_{\bar{b}})$ (Fig. 2). The idea behind the BlindEvalAttack algorithm is to let the adversary pick one of the $v \cdot (\beta + 1)$ ciphertext, and replace it with an independent ciphertext for the value $\mathbf{x}_{\bar{b}}$. With probability $1/(\beta + 1)$, the

adversary picks the ciphertext corresponding to the point requested by the client, and with probability $1/2$, the bit choice $\bar{b}$ was correct. Thus, with probability $1/(2 \cdot (\beta + 1))$ the challenger/client will not return $(\perp, \perp)$ as a response to a $\text{Finalise}(j = 1, \text{rep}_0, \text{rep}_1)$ query. In such a case, the adversary is convinced that the bit choice $\bar{b}$ is correct and returns it as the answer to the challenger. Otherwise, the adversary repeats the above steps for the next session j while keeping the same $(\mathbf{x}_0, \mathbf{x}_1)$, but sampling a fresh bit $\bar{b}$. The probability of a successful attack for the next session is again $1/(2 \cdot (\beta + 1))$. Thus, the adversary needs to, on average, run $2 \cdot (\beta + 1)$ sessions in expectation before the challenger does not abort in the Finalise oracle. β needs to be polynomial in the security parameter for efficiency reasons, so $\mathcal{A}$ runs in polynomial time. $\square$

Where the Dark Matter OPRF proof breaks. The malicious-security argument for the verifiable Dark Matter OPRF treats the cut-and-choose mechanism as ensuring that a cheating evaluator cannot change the client's output without being detected. This is enough to reason about the verifiability of successful outputs, but it is not enough for request privacy. Request privacy also requires that the probability of detection, and hence of client abort, be independent of the hidden input. Our adversary violates exactly this condition. It modifies one encrypted coordinate so that the client's finalization checks succeed if the hidden input agrees with the adversary's guess and fail otherwise. Therefore, the Finalise oracle leaks a selective-abort bit. The proof would need either a simulation of this abort event without knowing the hidden input or an argument that the abort probability is input-independent. The existing proof provides neither, which is why the claimed request-privacy theorem does not hold.

Lemma 1. *The adversary $\mathcal{A}$ from Theorem 3 is not required to pick the points $(\mathbf{x}_0, \mathbf{x}_1)$, i.e., the attack works even if the challenger chooses the inputs.*

Sketch. The attack can be adapted to challenger-sampled inputs because the adversary does not need to know the whole input. It suffices to learn one input coordinate on which the two challenge inputs differ. The key observation is that the input space of the PRF is $\mathbb{Z}_p^n$, and hence, the scheme encrypts each of the n elements separately, while the plaintext space is bits (i.e., $p = 2$). The attacker from Theorem 3 picks one of the inputs $(\mathbf{x}_0, \mathbf{x}_1)$ as a guess and runs the protocol. In our case, $\mathcal{A}$ does not have access to either of the messages since those are picked by the challenger. The idea for $\mathcal{A}$ is to pick a bit at random $\bar{b}$, encrypt it, creating ciphertext $\text{ct}_{\bar{b}}$. It then tries to guess the ciphertext ct^* from req_1 which encrypts $\mathbf{x}_{\mathbf{b}}$. As already mentioned, this happens with probability $1/(\beta + 1)$. The adversary then replaces the last ciphertext in ct^* with $\text{ct}_{\bar{b}}$ and uses such a modified request req_1 according to the protocol. It is easy to see that if the bit $\bar{b}$ corresponds to the last bit of the value $\mathbf{x}_{\mathbf{b}}$, then the protocol will not abort. In such a case, the adversary can later check and compare $\bar{b}$ with the last bits of $\mathbf{x}_0$ and $\mathbf{x}_1$. When the last bits of $\mathbf{x}_0$ and $\mathbf{x}_1$ differ, which happens with probability $1/2$, the adversary $\mathcal{A}$ can distinguish the challenge bit and break recipient privacy. $\square$

4 Beyond Homomorphic Encryption: Two Case Studies

The HE-based attack of Section 3 is the main technical contribution. This section presents two shorter case studies that show that the same proof pitfall can arise outside the HE setting. These examples are not intended as a taxonomy and do not imply that all derandomizable or selective-component constructions are insecure. Their purpose is to isolate the same missing component: the malicious signer learns an accept/abort bit whose dependence on the hidden message was not simulated correctly in the proof.

4.1 Derandomizable Blind Signatures

This section presents a first non-HE case study that illustrates the same proof pitfall: a hiding argument is used, while the malicious signer additionally learns an input-dependent abort bit. The construction we analyze achieves blindness by having the user present a randomized encoding of the message to the signer (for example, a Pedersen commitment). Concretely, the user computes a commitment com to the desired message $\mathbf{m}$ (e.g., $\text{com} = g_1^{\mathbf{m}} Z^r$ for fresh randomness r) and submits com to the signer in the clear. The signer returns a presignature σ' on com , and the user uses the trapdoor randomness r to derandomize σ' into a final signature σ on $\mathbf{m}$. The intended blindness intuition is that the hiding property of com hides $\mathbf{m}$ from the signer. The attack below shows that this intuition is incomplete. The signer can craft a presignature whose derandomization succeeds only when com opens to a guessed message, thereby turning the user's abort behavior into a test on the committed value.

The Scheme. A representative instantiation of this design is the scheme presented in [16], which adapts Camenisch–Lysyanskaya (CL) signatures [7] to the blind setting. We briefly recall the (non-blind) CL signature form and then describe the blind variant. The CL scheme is defined over groups $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ endowed with a bilinear pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$. Let $g_1 \in \mathbb{G}_1$ and $g_2 \in \mathbb{G}_2$ be generators, and let q be the common group order. The signer's keypair is $\text{sk}_{CL} = (x, y)$ and $\text{pk}_{CL} = (X, Y)$ with $X = g_2^x$ and $Y = g_2^y$. For a message $\mathbf{m}$ and a fresh random $a \leftarrow \mathbb{Z}_q^*$ chosen by the signer, a CL signature has the form (A, B, C) with $A = g_1^a$, $B = g_1^{a \cdot y}$ and $C = g_1^{a \cdot (x + x \cdot y \cdot \mathbf{m})}$. Verification checks the pairing equation $e(C, g_2) = e(A, X) \cdot e(B, X)^{\mathbf{m}}$. The blind variant of [16] introduces a common reference string element $Z = g_1^z$ (for random $z \leftarrow \mathbb{Z}_q^*$), which serves as the commitment key for a Pedersen commitment. The protocol proceeds as follows. The user computes the Pedersen commitment $\text{com} = g_1^{\mathbf{m}} Z^r$ for fresh $r \leftarrow \mathbb{Z}_q^*$ and sends com to the signer. The signer samples $a \leftarrow \mathbb{Z}_q^*$ and returns the 4-tuple (A, B, C, D) defined by $A = g_1^a$, $B = g_1^{a \cdot y}$, $C = g_1^{a \cdot x} \cdot \text{com}^{a \cdot x \cdot y}$, and $D = Z^{a \cdot x \cdot y}$. Observe that the triple (A, B, C) computed in this way is a valid CL signature on the shifted message $\mathbf{m}' = \mathbf{m} + z \cdot r$. The extra component D enables the user to “unshift” the signature: upon receiving (A, B, C, D) , the user computes $C' = C \cdot D^{-r}$ so that (A, B, C') is a CL signature on $\mathbf{m}$. The user

verifies this by checking $e(C', g_2) = e(A, X) \cdot e(B, X)^m$; if the check fails, the user aborts. Otherwise the user applies self-randomization by picking $t \leftarrow \$ \mathbb{Z}_q^*$ and outputting the final blind signature $(A^t, B^t, (C')^t)$. Verification of the blind signature is identical to verification of a standard CL signature. The authors of [16] argue that blindness follows from the Pedersen commitment's hiding property and thus claim perfect blindness. We now show that, regrettably, this claim does not hold.

The Attack. We observe that the user never checks the tuple (A, B, C, D) for internal consistency, and in particular does not validate the value D . This omission enables an adversary to turn the user's abort decision into an oracle for whether the user's committed message equals a guessed value m . The attack proceeds as follows. The adversary receives a commitment com (which may or may not be a commitment to m). It computes a standard CL signature for the guessed message m , $(A_m, B_m, C_m) = (g_1^a, g_1^{a \cdot y}, g_1^{a \cdot (x + x \cdot y \cdot m)})$, for some a , picks $u \leftarrow \$ \mathbb{Z}_q^*$, and forges the signer's reply by setting

$$A = A_m \quad B = B_m \quad C = C_m \cdot \text{com}^u \cdot g_1^{-u \cdot m} \quad D = Z^u$$

which it sends to the user. If the user's commitment indeed corresponds to m , i.e. $\text{com} = g_1^m Z^r$, then it is easy to see that $C' = C_m$ and so $(A, B, C') = (A_m, B_m, C_m)$ is a valid CL signature on m ; the user's verification succeeds and no abort occurs. Conversely, if it is to some other message m' , i.e. $\text{com} = g_1^{m'} Z^r$, then

$$\begin{aligned} C' &= C \cdot D^{-r} = C_m \cdot \text{com}^u \cdot g_1^{-u \cdot m} \cdot Z^{-u \cdot r} \\ &= C_m \cdot (g_1^{m'} Z^r)^u \cdot g_1^{-u \cdot m} \cdot Z^{-u \cdot r} \\ &= C_m \cdot (g_1^{m' - m})^u. \end{aligned}$$

In general $(g_1^{m' - m})^u \neq 1_{\mathbb{G}_1}$. Hence (A, B, C') is not a valid CL signature on m' ; the user's verification fails and the protocol aborts. Therefore, the adversary learns whether com commits to the guessed message m .

Theorem 4. *There exists an adversary $\mathcal{A}$ against the blind CL signature scheme from [16] for which $\text{Adv}_{\text{Blind}}(\mathcal{A}) = 1/2$.*

Proof. We will prove this theorem by constructing the adversary $\mathcal{A}$. In the first step, the adversary honestly generates the signing keys by running $(\text{sk}, \text{pk}) \leftarrow \text{BS.KeyGen}(1^\lambda)$ and picks two messages m_0, m_1 , which it returns to the challenger. The adversary then samples a bit $\bar{b}$. It then initiates both user sessions and finishes one according to protocol. For the second session, the adversary $\mathcal{A}$ executes the attack described above using the message $m_{\bar{b}}$ for its guess. At the end of both interactions, the challenger provides (σ_0, σ_1) to the adversary. The adversary outputs $1 - \bar{b}$ if $(\sigma_0, \sigma_1) = (\perp, \perp)$ and otherwise the bit $\bar{b}$. $\square$

Where the proof of [16] breaks. The proof of Theorem 1 in [16] conditions on the event E that the blindness challenger returns $(\perp, \perp)$ and then argues separately

about the experiments conditioned on E and on $\neg E$. The conditional probabilities are mathematically well-defined, but the claimed reductions to the hiding of the Pedersen commitment are not implementable.

Indeed, a reduction to commitment hiding must simulate the blindness experiment for the adversary and eventually output a bit in the commitment-hiding game. To use the conditional argument, the reduction must know whether it is simulating the branch E or the branch $\neg E$: if E occurs, it should analyze the adversary’s output in one way, while if $\neg E$ occurs, it should analyze it in another. But deciding whether E occurs requires knowing whether the honest user aborts after derandomizing the adversarially crafted tuple (A, B, C, D) . In our attack, this abort decision is exactly the hidden predicate “does the Pedersen commitment open to the guessed message?” A commitment hiding reduction does not know the opening of the challenge commitment, and therefore cannot decide whether E occurs. Consequently, the proof does not produce a valid reduction to commitment hiding. This is the same structural flaw as in the HE-based attacks: a hiding argument is used as if the adversary saw only the commitment. At the same time, in the real blindness experiment, the adversary additionally learns an input-dependent abort bit.

Possible fix. The attack would be prevented if the user could publicly check that the value D is consistent with the rest of the signer’s response. In the current construction, we do not see such a check from the pairing equations alone. A generic fix is to require the signer to include a zero-knowledge proof that (A, B, C, D) was computed honestly from a single signing randomness and the public key. By soundness, a malicious signer could then no longer choose D to make derandomization succeed only for one guessed opening of the commitment.

Takeaway. The attack does not contradict the hiding of the Pedersen commitment, which still hides the message. The problem is that the malicious signer also learns whether its crafted presignature is successfully derandomized. A reduction to commitment hiding must therefore simulate this abort event, which requires knowing whether the challenge commitment opens to the guessed message.

4.2 Selective-Component Blind Signatures

This section presents a second case study that illustrates the same selective-abort pitfall outside the HE setting. We do not claim that every construction in this syntactic category is insecure. The vulnerable pattern is narrower: the signer sends several components, the user uses only a message-dependent subset of them, and the unused components are neither checked nor otherwise tied to the final signature. If the malicious signer can corrupt one component and observe whether the user aborts, then the abort event may reveal which component the user used, and hence information about the hidden message.

This limitation is important. The attack is not purely syntactic. It requires a correlation between the user’s component choice and the message being signed. If the components are merely two random encodings of the same value, or if the

component choice is independent of the message, then corrupting one component does not reveal useful information. Thus, the lesson of this section is not that selective-component designs are inherently flawed, but that any such design must prove that omitted or unchecked components cannot make the abort probability depend on the hidden message.

As a case study, we demonstrate that the group-action-based blind signature scheme of [12] falls squarely into this category and is vulnerable to the attack described above. After responsibly disclosing our findings, the authors issued a corrected version. We cite only the original (vulnerable) ePrint report for anonymization purposes.

4.3 The Scheme and Attack against [12]

Notions. Before briefly describing the scheme, we recall the notions from [12]. We 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 $\mathbf{a} = (a_1, \dots, a_k)$, $\mathbf{b} = (b_1, \dots, b_k)$, we write $c \odot \mathbf{a} = (c \cdot a_1, \dots, c \cdot a_k)$ and $\mathbf{a} \odot \mathbf{b} = (a_1 \cdot b_1, \dots, a_k \cdot b_k)$. For group actions, if a vector $\mathbf{a}$ acts on an element $\mathbf{s}$, we write this as $\mathbf{a} * \mathbf{s} = (a_1 * s_1, \dots, a_k * s_k)$, where $*$ denotes the action operation. We 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 [12].

Scheme. The blind signature scheme from [12] 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 overall three-move exchange of the underlying Sigma protocol remains the same (i.e., commitment from signer, challenge from user, response from signer). The main difference between CSI-Otter [18] 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 [12] for a full scheme description.

We begin with the public key of the signer $\mathbf{pk} = (A_0^{(1)}, A_0^{(-1)}, A_1^{(1)}, A_1^{(-1)})$ 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, 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, 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. The hash challenges and the values $\mathbf{d}$ are vectors of size n with elements from set $\{-1, 1\}$. Once the signer obtains $\mathbf{c}^*$, 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-commutative 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, the user computes $\mathbf{r}_0 = \mathbf{z}_0 * (\mathbf{r}_0^{(\mathbf{d}_0)})$ and $\mathbf{r}_1 = \mathbf{z}_1 * (\mathbf{r}_1^{(\mathbf{d}_1)})$ 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 $\mathbf{c}$.

Attack. 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 test 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. At the end of the blindness experiment, the adversary receives $(\sigma_{\text{coin}}, \sigma_{1-\text{coin}})$, where one of the two signatures will correspond 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 whether $\mathbf{c}^* \odot \mathbf{d}_0 \odot \mathbf{d}_1$ correspond to the hash challenge for signature σ_{coin} or $\sigma_{1-\text{coin}}$ easily distinguishing the coin picked by the challenger. Thus, it breaks the blindness experiment.

Interestingly, the authors of [12] already discuss a closely related attack in the proof of their Theorem 2 (page 22). They dismiss this attack because guessing the full vectors succeeds with probability only $1/4^n$. See the exact citation below. Our observation is that guessing a single coordinate is enough to obtain a non-negligible distinguishing advantage.

Citation from [12] (page 22, Theorem 2): *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 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.*

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 $\mathbf{d}_0 = (d_{0,1}, \dots, d_{0,n})$ and $\mathbf{d}_1 = (d_{1,1}, \dots, d_{1,n})$. 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 $(\sigma_{\text{coin}}, \sigma_{1-\text{coin}})$.

Let us now assume that $\mathbf{c}_{\text{coin}}$ and $\mathbf{c}_{1-\text{coin}}$ are the two hash challenges received from the signatures $(\sigma_{\text{coin}}, \sigma_{1-\text{coin}})$. Moreover, assume that $c_{\text{coin}} \in \{-1, 1\}$ and $c_{1-\text{coin}} \in \{-1, 1\}$ 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 us assume we target the first session. To output coin the adversary checks if $c_{\text{coin}} \stackrel{?}{=} c^* \cdot d_{0,i} \cdot d_{1,i}$ and $c_{1-\text{coin}} \stackrel{?}{=} c^* \cdot d_{0,i} \cdot d_{1,i}$. 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 $c_{\text{coin}} = c_{1-\text{coin}}$. 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 $1/4 \cdot 1/2 = 1/8$. This gives the following theorem.

Theorem 5. *There exists an adversary $\mathcal{A}$ against the blind signature scheme from [12] for which $\text{Adv}_{\text{Blind}}(\mathcal{A}) = 1/8$.*

Takeaway. This example shows that the same issue can occur even without homomorphic encryption or commitments. If the signer sends components that are not all checked, and the user's choice of components is correlated with the hidden message, then corrupting unchecked components can turn the user's abort behavior into information about that choice.

5 Rumbl and Generic Transformation

The attacks above are independent of the mitigation in this section. We include the mitigation to show that some random-token applications of HE-based blind

$\text{Rumbl}_{\mathcal{A}, \text{BS}}(\lambda)$
$(\text{st}_0, \text{pk}) \leftarrow \mathcal{A}(1^\lambda)$ $(\text{m}_0, \text{m}_1) \leftarrow \mathcal{M} ; b \leftarrow \{0, 1\}$ $\text{st}_1 \leftarrow \mathcal{A}^{\langle \cdot, \text{User}(\text{pk}, \text{m}_b) \rangle^1, \langle \cdot, \text{User}(\text{pk}, \text{m}_{1-b}) \rangle^1}(\text{st}_0)$ If $\sigma_b = \perp$ or $\sigma_{1-b} = \perp$, then $\mathcal{A}$ learns $(\perp, \perp, \perp, \perp)$ and $(\sigma_0, \text{m}_0, \sigma_1, \text{m}_1)$ otherwise. $\hat{b} \leftarrow \mathcal{A}(\text{st}_1, \sigma_0, \text{m}_0, \sigma_1, \text{m}_1)$ if $\hat{b} == b$ then return 1 else return 0

Fig. 3. Random unknown-message blindness experiment (Rumbl)

signatures can be salvaged without a full ZK proof of honest signing. It is not a generic repair for all schemes attacked in Section 3, and it does not apply to OPRFs such as the Dark Matter OPRF [2]. The reason is conceptual: in an OPRF, the client needs $F_k(x)$ for its actual input x , whereas the transformation below has the signer sign a random-oracle commitment to the message. The transformation is therefore useful for blind signatures, especially anonymous-token settings, where the signed value can be a fresh random string or a hash commitment to the real message. We formalize the property required for this approach as random-unknown-message blindness (Rumbl). In Rumbl, the adversary does not choose the two challenge messages; they are sampled uniformly by the challenger. This removes the exact equality test exploited by our attacks, while still capturing the privacy needed when the signed value is random. We then show how to compile Rumbl security into standard blindness in the random-oracle model by signing $H(M, \rho)$ instead of M .

Definition 1 (Random Unknown-Message Blindness (Rumbl)). *We say that the blind signature scheme $\text{BS} = (\text{KeyGen}, \langle \text{Sign}, \text{User} \rangle, \text{Ver})$ is blind under random unknown messages if it holds that every PPT adversary $\mathcal{A}$ has at most negligible advantage in the $\text{Rumbl}_{\mathcal{A}, \text{BS}}(1^\lambda)$ experiment defined in Fig. 3. We define the advantage of adversary $\mathcal{A}$ in $\text{Rumbl}_{\mathcal{A}, \text{BS}}(1^\lambda)$ experiment as $\text{Adv}_{\text{Rumbl}}(\mathcal{A}) = |\Pr[\text{Rumbl}_{\mathcal{A}, \text{BS}}(\lambda) = 1] - \frac{1}{2}|$.*

In the Rumbl experiment for blindness under random unknown messages the adversary $\mathcal{A}$ takes the role of the malicious signer against challenger $\mathcal{C}$ that controls the user oracles and verifies the signatures. Our definition for $\text{Rumbl}_{\mathcal{A}, \text{BS}}$ includes malicious key generation by $\mathcal{A}$ as well. The challenger $\mathcal{C}$ samples messages m_0, m_1 from the message space $\mathcal{M}$ and a random bit $b \leftarrow \{0, 1\}$. $\mathcal{C}$ also instantiates two User oracles, that take messages m_b and m_{1-b} as their inputs. Adversary $\mathcal{A}$ is granted one-time access to these User oracles and interacts with them during the signing protocol. If any signature $\sigma_i = \perp$ or fails verification, the adversary is provided with an empty tuple $(\perp, \perp, \perp, \perp)$. Otherwise, the adversary $\mathcal{A}$ learns both signatures and messages sampled by $\mathcal{C}$ in the beginning of the experiment

BS'.KeyGen (1^λ)	BS'.User₀ (pk, m)
$(sk, pk) \leftarrow \$ BS.KeyGen(1^\lambda)$	$r \leftarrow \$ R ; st = (m, r)$
return (sk, pk)	$com \leftarrow \$ RO.Eval(m, r)$
BS'.Sign₀ (sk, bsm ₀)	$bsm_0 \leftarrow BS.User_0(pk, com)$
$bsm_1 \leftarrow BS.Sign_0(sk, bsm_0)$	return (bsm ₀ , st)
return bsm ₁	BS'.User₁ (st, bsm ₁)
BS'.Ver (pk, m, σ')	$\sigma \leftarrow BS.User_1(bsm_1)$
$(\sigma, r) \leftarrow \sigma'$	$(m, r) \leftarrow st ; \sigma' \leftarrow (\sigma, r)$
$com \leftarrow \$ RO.Eval(m, r)$	if BS'.Ver(pk, m, σ') = 0 then
return BS.Ver(pk, com, σ)	return $\perp$
	return σ'

Fig. 4. Generic transformation of two-move blind signature scheme BS blind under Rumbl, into a scheme BS' secure under the standard blindness Blind definition.

as a tuple $(\sigma_0, m_0, \sigma_1, m_1)$. $\mathcal{A}$ then makes a guess of bit $\hat{b} \in \{0, 1\}$, if $\hat{b} == b$ the experiment outputs 1 and 0 otherwise.

Generic Transformation from Rumbl to Blind. We now show that given a generic blind signature scheme secure in Rumbl, one can modify it in a way that it will be secure in the standard blindness Blind notion. For this purpose, we use a random oracle RO. The user commits to the message m using some randomness r and computes $H(m, r)$, now treated as the message in the signature issuance process. For simplicity, we describe our construction for a two-move blind signature scheme. Still, it can be easily extended to a n -round construction since the presented modification affects only the first and the last step of user algorithm BS.User₀ and BS.User _{n} .

Let BS be a digital signature scheme blind under Rumbl. Then, we define a blind signature scheme BS' (Fig. 4) that is secure under the standard blindness Blind definition. We define random oracle evaluation as $RO.Eval : \mathcal{M} \times R \rightarrow \mathcal{M}$, where $\mathcal{M}$ is a message space for BS and R is a randomness space. The differences between BS and BS' lie in implementation of BS'.User and BS'.Ver algorithms. For BS'.User, for every message m to be issued to the signer party, the user randomly picks a new randomness value $r \leftarrow \$ R$ used along with m to obtain commitment $com \leftarrow RO.Eval(m, r)$. Then, the created commitment com is input to BS.User instead of m . Since the signature σ yielded by BS.User₁ was issued under commitment com , BS'.User₁ outputs the signature along with randomness r , i.e., $\sigma' = (\sigma, r)$. In BS'.Ver, we run the underlying verification algorithm BS.Ver with commitment $com \leftarrow RO.Eval(m, r)$ as its input.

Remark 1. While the construction might seem similar to that in [13], we require more advanced machinery and a different type of commitment. The problem is that our reduction receives "commitments" from the Rumb1 challenger and must equivocate those to messages picked by the adversary of the standard blindness experiment. This is a simple task for an RO, which we use, but does not work with Pedersen commitments, which is sufficient for the construction in [13]. The problem is that to equivocate Pedersen's commitment, we would need to know at least one valid opening (and the trapdoor commitment key), which we do not have here. A potential direction for using commitments instead of the ROM is to leverage the commitments from [17], where one can equivocate only with a trapdoor. However, that would introduce new problems with trapdoor extraction, which can be easily omitted with our RO approach.

Remark 2 (Scope of the mitigation). Rumb1 should not be interpreted as a replacement for standard blindness in general. It is useful when the application builds on the message that is blindly signed to be random, or when the real message can be encoded as a random-oracle commitment $H(M, \rho)$, with ρ later revealed along with the signature. For applications where the signed message is inherently a fresh, random value (e.g., random-token applications such as Privacy Pass [10]), the transformation of Theorem 6 may be unnecessary. In such settings, a Rumb1-secure scheme - such as the blind ECDSA scheme analyzed in Section 6 - can be deployed directly to produce standard signatures without requiring an additional salt or a modified verification procedure. It does not however apply to OPRFs, because replacing x by $H(x, \rho)$ changes the functionality from $F_k(x)$ to $F_k(H(x, \rho))$. Nor does it remove the need for a proof of honest evaluation in HE-based schemes whose applications require adversarially chosen signed messages without the random-oracle commitment transformation.

Theorem 6. *In the random oracle model and for exponentially large randomness space $|R| \geq 2^\lambda$, the blind signature scheme BS' from Fig. 4 is blind under the Blind definition if BS is blind under the Rumb1 definition. More formally, we will show for any adversary $\mathcal{A}$ against blindness of BS' , making at most q_h oracle queries, we have: $\text{Adv}_{\text{Blind}, BS'}(\mathcal{A}) \leq \text{Adv}_{\text{Rumb1}, BS}(\mathcal{R}) + \frac{2 \cdot q_h}{2^\lambda}$.*

Proof. We will use the game-based approach to prove this theorem. We first exclude certain collisions in the random oracle and later show a reduction $\mathcal{R}$ that uses simple programming to reduce the security of BS' to the blindness of BS . We will use the following to denote the advantage of adversary $\mathcal{A}$ in G_i for $i \in \{0, 1\}$ as $\text{Adv}_{G_i}(\mathcal{A}) = \left| \Pr[G_{i, \mathcal{A}, BS'} = 1] - \frac{1}{2} \right|$.

Game G_0 : Let us denote G_0 as the Blind experiment for the blind signature scheme BS' constructed according to the transformation in Fig. 4.

Thus, we have $\text{Adv}_{G_0}(\mathcal{A}) = \text{Adv}_{\text{Blind}, BS'}(\mathcal{A})$.

Game G_1 : Now, let us denote G_1 as game G_0 with a change that whenever $\mathcal{A}$ queries (m_0, r_0) or (m_1, r_1) to RO.Eval , the experiment is aborted.

Claim. For games G_0 and G_1 it holds that: $|\text{Adv}_{G_0}(\mathcal{A}) - \text{Adv}_{G_1}(\mathcal{A})| \leq \frac{2 \cdot q_h}{2^\lambda}$.

Proof. In the setting of G_1 , the experiment is aborted if, at any point of the interaction, adversary $\mathcal{A}$ queries to RO.Eval message m with corresponding randomness r used to evaluate commitments $\text{com}_0, \text{com}_1$. In other words, if $\mathcal{A}$ queries for any of the pairs (m_0, r_0) or (m_1, r_1) . Given that $\mathcal{A}$ queries RO.Eval at most q_h times, we observe that $\mathcal{A}$ can guess $r^* \in R$, such that $r^* = r_0$ or $r^* = r_1$, with probability of at most $\frac{q_h}{2^\lambda}$. Since the difference between G_0 and G_1 is the abort condition, using the union bound, the advantage of $\mathcal{A}$ in G_1 is as claimed. $\square$

Let $\mathcal{A}$ be a PPT adversary with non-negligible advantage in G_1 . We construct a reduction $\mathcal{R}$ that breaks the Rumbl blindness of BS. The reduction $\mathcal{R}$ runs $\mathcal{A}$ internally and simulates the G_1 challenger, while itself interacting with the Rumbl challenger $\mathcal{C}$ for BS. It also implements a programmable random oracle RO by lazy sampling, storing its table in st_{RO} . Conceptually, $\mathcal{R}$ acts as an interface: messages from $\mathcal{C}$ are forwarded to $\mathcal{A}$, and messages from $\mathcal{A}$ are translated back into queries to $\mathcal{C}$. Below we provide a step-by-step description of the experiment run between $\mathcal{R}$ and $\mathcal{C}$ in Rumbl setting for blind signature scheme BS:

1. $\mathcal{R}$ receives a public key pk from $\mathcal{A}$ and forwards it to $\mathcal{C}$.
2. $\mathcal{R}$ receives messages $(m_0, m_1) \leftarrow \mathcal{M}$ from $\mathcal{A}$. Simultaneously $\mathcal{C}$ samples random bit $b \leftarrow \{0, 1\}$ and randomly picks two messages $(\text{com}_0, \text{com}_1) \leftarrow \mathcal{M}$.
3. At some point $\mathcal{A}$ initiates the sessions with users. The reduction $\mathcal{R}$ initiates the protocol with the challenger $\mathcal{C}$ to answer those sessions.
 - For one of the sessions $\mathcal{C}$ runs $\text{bsm}_{0,b} \leftarrow \text{BS.User}_0(\text{pk}, \text{com}_b)$ and shares $\text{bsm}_{0,b}$ with $\mathcal{R}$. $\mathcal{R}$ forwards $\text{bsm}_{0,b}$ to $\mathcal{A}$ and $\mathcal{A}$ responds with $\text{bsm}_{1,b}$. $\mathcal{R}$ forwards $\text{bsm}_{1,b}$ to $\mathcal{C}$ and $\mathcal{C}$ obtains signature $\sigma_b \leftarrow \text{BS.User}_1(\text{bsm}_{1,b})$.
 - For the other session $\mathcal{C}$ repeats the protocol from the previous step yielding σ_{1-b} , by using com_{1-b} as BS.User_0 input.
4. If $\sigma_0 = \perp$ or $\sigma_1 = \perp$, $\mathcal{C}$ sends $(\perp, \perp, \perp, \perp)$ to $\mathcal{R}$ and $(\sigma_0, \text{com}_0, \sigma_1, \text{com}_1)$ otherwise.
5. $\mathcal{R}$ randomly picks $(r_0, r_1) \leftarrow R$, such that pairs $(m_0, r_0), (m_1, r_1)$ are not in st_{RO} . Then, $\mathcal{R}$ programs random oracle for values $\text{com}_0 \leftarrow \text{RO.Eval}(m_0, r_0)$, $\text{com}_1 \leftarrow \text{RO.Eval}(m_1, r_1)$ and provides $\mathcal{A}$ with (σ'_0, σ'_1) , where $\sigma'_i = (\sigma_i, r_i)$.
6. $\mathcal{A}$ outputs bit $\hat{b}$ to $\mathcal{R}$, $\mathcal{R}$ outputs bit $\bar{b} = \hat{b}$ to $\mathcal{C}$.

The reduction $\mathcal{R}$ perfectly simulates G_1 for $\mathcal{A}$, except with the same abort behavior as in the real experiment. Hence $\mathcal{A}$ cannot distinguish the simulation from an honest G_1 execution, and $\text{Adv}_{\text{Rumbl,BS}}(\mathcal{R}) = \text{Adv}_{G_1}(\mathcal{A})$. A non-negligible advantage for $\mathcal{A}$ would therefore contradict the Rumbl blindness of BS. Thus, no PPT adversary can break the blindness of BS' with non-negligible advantage. $\square$

Finally, the generic transformation also preserves one-more unforgeability. Since this paper focuses on blindness, we present only an informal argument. Any forgery against the transformed scheme either yields a forgery against the underlying blind signature scheme or reuses a signature via a collision in H . In the random-oracle model, the latter event occurs only with negligible probability.

Lemma 2. *In the random oracle model, the blind signature scheme BS' from Fig. 4 preserves one more unforgeability.*

Sketch. Assume an adversary $\mathcal{A}$ can output $\ell + 1$ message/signature pairs for distinct messages while only running the signing oracle ℓ -times. Such an adversary can be successful by either breaking the unforgeability of the BS scheme or by finding a collision in the hash function (i.e., there exists a pair of messages in $\mathcal{A}$'s output that share a signature). In both cases, the advantage is negligible. $\square$

6 Rumbl Security of Blind ECDSA [22]

We instantiate this methodology for the blind ECDSA scheme from [22]. The scheme is not blind in the standard sense by Theorem 1, but the attack relies on the adversary choosing the challenge messages and testing equality with one of them. When the signed value is uniformly random, the corresponding attack becomes ineffective except with negligible probability. To capture the remaining power of the malicious signer, we introduce IND-RPA-ZT for the underlying homomorphic encryption scheme: the adversary receives a limited zero-testing capability, but the challenge plaintexts are random and unknown. Under this assumption, the original proof strategy can be repaired for Rumbl security, and the random-oracle transformation then yields a standard blind signature for arbitrary messages.

Random Plaintext Attacks with Zero-Testing (IND-RPA-ZT) It is known that homomorphic encryption can support the known IND-CPA security definition for public key encryption but fail if the challenger provides the adversary with a decryption oracle after the challenged ciphertext is given to the adversary, i.e., the notion of indistinguishability under chosen ciphertext attacks (IND-CCA). To see this, consider ct^* to be the challenged ciphertext, which cannot be queried to the decryption oracle. An adversary can efficiently compute (using the homomorphic properties) a ciphertext $\text{ct}' \neq \text{ct}^*$ which encrypts the same messages as ct^* , i.e., by adding ct^* to a ciphertext of 0. This textbook attack on encryption schemes with homomorphic properties is the main reason why they cannot achieve the IND-CCA or, in other words, do not support a decryption oracle after seeing the challenge ciphertext ct^* .

Replacing the decryption oracle in IND-CCA with a zero-testing oracle (i.e., an oracle that returns 1 if the plaintext is a zero and 0 otherwise) also does not work since the above attack can be adapted if such an oracle is provided instead. Since the adversary is allowed to specify two messages m_0, m_1 , where the challenged ciphertext ct^* encrypts one of the two messages m_b . The adversary can use the homomorphic property to compute a ciphertext for plaintext $m_b - m_0$ and ask the zero-testing oracle if the result is a zero. If true, it can break IND-CCA with a zero-testing oracle by returning the bit 0 or bit 1 if the zero-test fails. In other words, homomorphic encryption schemes cannot achieve CPA/CCA indistinguishability definitions if the adversary has a zero-testing oracle.

The question is whether we can have a meaningful indistinguishability definition that gives the adversary access to a zero-testing oracle. The attack described above implicitly uses the knowledge of a candidate plaintext (in CPA/CCA, the

adversary specifies m_0, m_1) to compute $m_b - m_0$ and use the zero-testing oracle to check this value. However, what if the adversary is challenged against random messages that it will only learn *after* it makes all queries to the zero-testing oracle? To make valuable use of the zero-testing oracle, the adversary would have to guess the message m_b or evaluate a circuit f on the challenged ciphertext ct^* that would give it meaningful information about bit b .

In this section, we introduce the notion of indistinguishability under random plaintext attacks with zero-testing (IND-RPA-ZT). This notion is similar to plaintext checking attacks [23] with a slight change since those attacks are defined for classical IND-CPA experiments. At the same time, in the case of IND-RPA-ZT, we allow the challenger to pick the messages randomly. Our main observation is that IND-CPA secure homomorphic encryption schemes that only support a limited family of arithmetic circuits $\mathcal{F}$ (i.e., polynomials of small degree) are secure in this new notion if we only allow for one-time zero-testing and the message space is exponential. In other words, we conjecture that IND-CPA security implies 1-RPA-ZT. The intuition here is that since the adversary learns the challenge messages only after it is allowed to use the zero-testing oracle, the chances of the oracle outputting anything other than 0 (for non-trivial queries, i.e., adversary can always ask the oracle with a ciphertext of zero) are negligible assuming an exponential message space. The supplementary material provides an algebraic analysis of the Paillier instantiation used in [22], which we view as evidence for the corresponding one-query zero-testing assumption.

Definition 2 (IND-RPA-ZT). A HE scheme is secure against random plaintext attacks with zero-testing if it holds that every PPT adversary $\mathcal{A}$ has at most negligible advantage in the $\text{RPA-ZT}_{\mathcal{A}, \text{HE}}(\lambda)$ experiment defined below. We define the adversary $\mathcal{A}$ advantage by $\text{Adv}_{\text{RPA-ZT}, \text{HE}}(\mathcal{A}) = |\Pr[\text{RPA-ZT}_{\mathcal{A}, \text{HE}}(\lambda) = 1] - \frac{1}{2}|$.

$\text{RPA-ZT}_{\mathcal{A}, \text{HE}}(\lambda)$	$\text{ZT}(\text{sk}, \text{ct})$
$(\text{sk}, \text{pk}) \leftarrow \text{HE.KeyGen}(1^\lambda)$	$\text{m} \leftarrow \text{HE.Dec}(\text{sk}, \text{ct})$
$(\text{m}_0, \text{m}_1) \leftarrow \mathcal{R}_{\mathcal{M}} ; b \leftarrow \mathcal{S} \{0, 1\}$	if $\text{m} = 0$ return 1
$\text{ct} \leftarrow \text{HE.Enc}(\text{pk}, \text{m}_b)$	return 0
$\text{st} \leftarrow \mathcal{A}^{\text{ZT}(\text{sk}, \cdot)}(\text{pk}, \text{ct})$	
$\hat{b} \leftarrow \mathcal{A}(\text{st}, \text{m}_0, \text{m}_1)$	
return $b = \hat{b}$	

Remark 3. We will use $k\text{-RPA-ZT}$ to define the above experiment, where the number of queries to the ZT oracle is limited to k times.

Rumbl Security of Blind ECDSA

Theorem 7. The blind ECDSA scheme [22] (Fig. 9) is Rumbl secure in the algebraic group model [15] and the random oracle model if HE is 1-RPA-ZT secure

and *NIZK* is zero-knowledge. Let $\mathcal{A}$ be an adversary against the *Rumbl* security of the scheme, making at most q_h random oracle queries. Then we have:

$$\text{Adv}_{\text{Rumbl}}(\mathcal{A}) \leq 4 \cdot \text{Adv}_{1\text{-RPA-ZT,HE}}(\mathcal{R}_{\text{HE}}) + 2 \cdot \text{Adv}_{\text{ZK}}(\mathcal{R}_{\text{ZK}}) + 2 \cdot q_h/2^\lambda.$$

Sketch. The proof follows the same steps as in the wrong proof of blindness from [22], but we replace the classical blindness experiment with our *Rumbl* experiment and IND-CPA security of the HE scheme with 1-RPA-ZT security. The idea is that thanks to IND-RPA-ZT, the reduction can decide if the adversary is supposed to receive signatures or $(\perp, \perp)$ even if it does not know the decryption keys while playing the indistinguishability game for the homomorphic encryption scheme. To this end, it uses the zero-testing oracle. For completeness, we provide the full proof in the supplementary material. $\square$

Theorem 8. Let BS be the blind ECDSA scheme from [22] and let BS' be a blind signature scheme constructed using our generic construction as in Fig. 4. Let $\mathcal{A}$ be an adversary against the blindness of scheme BS' . We have

$$\text{Adv}_{\text{Blind,BS}'}(\mathcal{A}) \leq 4 \cdot \text{Adv}_{1\text{-RPA-ZT,HE}}(\mathcal{R}_{\text{HE}}) + 2 \cdot \text{Adv}_{\text{ZK}}(\mathcal{R}_{\text{ZK}}) + \frac{4 \cdot q_h}{2^\lambda}.$$

7 Discussions

Circumventing Selective-Failure Attacks The easiest way to circumvent the attack is to ensure the blind signature scheme provides a signature derivation check property [14]. This property allows anyone to publicly decide if a given transcript of communication between the signer and the user will end with the user outputting a valid signature or aborting. Many signature schemes like blind RSA [3, 9] and blind BLS [5] provide this property.

The attacks in the previous sections all exploit the same missing property: after seeing the transcript, the malicious signer cannot itself determine whether the honest user will accept. The signer, therefore, learns new information by observing whether the user aborts. Signature-derivation checks rule out exactly this situation. They make acceptance or abort publicly decidable from the transcript, so the abort event no longer carries hidden-message information.

Definition 3 (Signature-Derivation Check). A blind signature scheme BS allows computational (resp. statistical) signature-derivation checks if there exists an efficient algorithm SDCh such that for any efficient (resp. unbounded) algorithm $\mathcal{A}$ the probability that the experiment $\text{SigDerCheck}_{\mathcal{A}, \text{SDCh}}$ evaluates to 1 is negligible, where the experiment is defined in Fig. 5.

The intuition is simple: if a malicious signer can determine from the public transcript whether the honest user will abort (via the signature-derivation check), then observing an abort produces no information beyond what the signer could already compute. In other words, selective-abort behavior does not increase the adversary's distinguishing power because the adversary can already predict the outcome from the transcript alone. We formalize this intuition in the following theorem. For completeness, we provide the proof in the supplementary material.

$\text{SigDerCheck}_{\mathcal{A}, \text{SDCh}}(\lambda)$
$(\text{st}_0, \text{pk}, \text{m}) \leftarrow \mathcal{A}(1^\lambda)$ $(\sigma, \text{trans}) \leftarrow \langle \mathcal{A}(\text{st}_0), \text{User}(\text{pk}, \text{m}) \rangle$ where trans denotes the communication transcript between $\mathcal{A}$ and User and is not protocol's direct output $c \leftarrow \text{SDCh}(\text{pk}, \text{trans})$ return 1 if $(\sigma \neq \perp \text{ and } c = 0) \text{ or } (\sigma = \perp \text{ and } c = 1)$.

Fig. 5. Signature-derivation Check Experiment (SigDerCheck)

Theorem 9. *Let BS be a blind signature scheme with signature-derivation check SDCh and let $\mathcal{R}$ be an adversary in the $\text{SigDerCheck}_{\mathcal{R}, \text{SDCh}}$ experiment. Let $\text{Blind}_{\text{SDCh}}$ be a variant of the *Blind* experiment, with the change that the challenger aborts the experiment (i.e., returns 0) if given the public key pk , transcripts $\text{trans}_0, \text{trans}_1$ and signatures σ_0, σ_1 from the two interactions with the adversary for any $i \in \{0, 1\}$ we have: $\sigma_i \neq \perp$ while $\text{SDCh}(\text{pk}, \text{trans}_i) = 0$ or $\sigma_i = \perp$ while $\text{SDCh}(\text{pk}, \text{trans}_i) = 1$. For any PPT adversary $\mathcal{A}$ against the blindness of BS , we have: $|\text{Adv}_{\text{Blind}}(\mathcal{A}) - \text{Adv}_{\text{Blind}_{\text{SDCh}}}(\mathcal{A})| \leq 2 \cdot \text{SigDerCheck}_{\mathcal{R}, \text{SDCh}}$. Moreover, there exists no selective-abort attacks for experiment $\text{Blind}_{\text{SDCh}}$.*

The theorem should be interpreted as a design guideline. A proof of blindness must account not only for the distribution of protocol messages, but also for the distribution of the user's accept/abort decision. A signature-derivation check is one clean way to do so: it lets the simulator determine the abort event from public information. HE-based schemes generally lack this property because the transcript contains ciphertexts whose decryptions determine whether the final signature verifies. A generic way to recover the property is to add a proof that the signer computed its response honestly. This is why NIZKs prevent the attacks, while cut-and-choose verifiability that ignores aborts may not.

Attack Practical Applicability and Scope

Selective-abort attacks are most relevant when the malicious signer/evaluator has a small or structured set of candidate inputs. This occurs, for example, when blind signatures are used on user attributes, policy identifiers, or other values drawn from a polynomial-size dictionary. In such cases, repeated protocol failures can leak information even though each individual execution leaks only one bit.

Our attacks should not be read as attacks on every protocol that uses blind signatures or OPRFs. In particular, they do not apply to UC-secure OPRF instantiations merely because they are used within larger protocols. Applying our results to a deployed protocol requires checking the specific primitive and proof model used in that protocol. The attacks in this paper apply to constructions with an input-dependent abort channel of the kind analyzed above; they do not

by themselves refute security theorems for unrelated UC-secure OPRFs. For example, the attack becomes practically relevant only if a deployed system uses one of the vulnerable HE-based OPRFs or blind signatures and treats aborts as recoverable failures that can be retried without user-visible consequences.

References

1. Agrawal, S., Cheon, J.H., Choe, H., Stehlé, D., Yadav, A.: Round optimal blind signatures: Short signatures with post-quantum blindness. Cryptology ePrint Archive, Report 2022/910 (2022), <https://eprint.iacr.org/2022/910>
2. Albrecht, M.R., Davidson, A., Deo, A., Gardham, D.: Crypto dark matter on the torus - oblivious PRFs from shallow PRFs and TFHE. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part VI. LNCS, vol. 14656, pp. 447–476. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58751-1_16
3. Bellare, M., Namprempre, C., Pointcheval, D., Semanko, M.: The power of RSA inversion oracles and the security of Chaum’s RSA-based blind signature scheme. In: Syverson, P.F. (ed.) FC 2001. LNCS, vol. 2339, pp. 319–338. Springer, Berlin, Heidelberg (Feb 2002). https://doi.org/10.1007/3-540-46088-8_25
4. Benhamouda, F., Lepoint, T., Loss, J., Orrù, M., Raykova, M.: On the (in)security of ROS. In: Canteaut, A., Standaert, F.X. (eds.) EUROCRYPT 2021, Part I. LNCS, vol. 12696, pp. 33–53. Springer, Cham (Oct 2021). https://doi.org/10.1007/978-3-030-77870-5_2
5. Boldyreva, A.: Threshold signatures, multisignatures and blind signatures based on the gap-Diffie-Hellman-group signature scheme. In: Desmedt, Y. (ed.) PKC 2003. LNCS, vol. 2567, pp. 31–46. Springer, Berlin, Heidelberg (Jan 2003). https://doi.org/10.1007/3-540-36288-6_3
6. Boneh, D., Ishai, Y., Passelègue, A., Sahai, A., Wu, D.J.: Exploring crypto dark matter: New simple PRF candidates and their applications. In: Beimel, A., Dziembowski, S. (eds.) TCC 2018, Part II. LNCS, vol. 11240, pp. 699–729. Springer, Cham (Nov 2018). https://doi.org/10.1007/978-3-030-03810-6_25
7. Camenisch, J., Lysyanskaya, A.: Signature schemes and anonymous credentials from bilinear maps. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 56–72. Springer, Berlin, Heidelberg (Aug 2004). https://doi.org/10.1007/978-3-540-28628-8_4
8. Camenisch, J., Neven, G., shelat, a.: Simulatable adaptive oblivious transfer. In: Naor, M. (ed.) EUROCRYPT 2007. LNCS, vol. 4515, pp. 573–590. Springer, Berlin, Heidelberg (May 2007). https://doi.org/10.1007/978-3-540-72540-4_33
9. Chaum, D.: Blind signatures for untraceable payments. In: Chaum, D., Rivest, R.L., Sherman, A.T. (eds.) CRYPTO’82. pp. 199–203. Plenum Press, New York, USA (1982). https://doi.org/10.1007/978-1-4757-0602-4_18
10. Davidson, A., Goldberg, I., Sullivan, N., Tankersley, G., Valsorda, F.: Privacy Pass: Bypassing Internet Challenges Anonymously. Proceedings on Privacy Enhancing Technologies **2018**(3), 164–180 (Jun 2018). <https://doi.org/10.1515/popets-2018-0026>, <https://petsymposium.org/popets/2018/popets-2018-0026.php>
11. Do, K., Hanzlik, L., Paracucchi, E.: M&M’S: Mix and match attacks on Schnorr-type blind signatures with repetition. In: Joye, M., Leander, G. (eds.) EUROCRYPT 2024, Part VI. LNCS, vol. 14656, pp. 363–387. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58751-1_13

12. 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/archive/2025/397/20250303:050457>
13. Fischlin, M., Schröder, D.: Security of blind signatures under aborts. In: Jarecki, S., Tsudik, G. (eds.) PKC 2009. LNCS, vol. 5443, pp. 297–316. Springer, Berlin, Heidelberg (Mar 2009). https://doi.org/10.1007/978-3-642-00468-1_17
14. Fischlin, M., Schröder, D.: On the impossibility of three-move blind signature schemes. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 197–215. Springer, Berlin, Heidelberg (May / Jun 2010). https://doi.org/10.1007/978-3-642-13190-5_10
15. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62. Springer, Cham (Aug 2018). https://doi.org/10.1007/978-3-319-96881-0_2
16. Ghadafi, E., Smart, N.P.: Efficient two-move blind signatures in the common reference string model. In: Gollmann, D., Freiling, F.C. (eds.) ISC 2012. LNCS, vol. 7483, pp. 274–289. Springer, Berlin, Heidelberg (Sep 2012). https://doi.org/10.1007/978-3-642-33383-5_17
17. Groth, J.: Homomorphic trapdoor commitments to group elements. *Cryptology ePrint Archive*, Report 2009/007 (2009), <https://eprint.iacr.org/2009/007>
18. 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.) CRYPTO 2023, Part III. LNCS, vol. 14083, pp. 729–761. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38548-3_24
19. Katsumata, S., Lai, Y.F., Reichle, M.: Breaking parallel ROS: Implication for isogeny and lattice-based blind signatures. In: Tang, Q., Teague, V. (eds.) PKC 2024, Part I. LNCS, vol. 14601, pp. 319–351. Springer, Cham (Apr 2024). https://doi.org/10.1007/978-3-031-57718-5_11
20. Katz, J., Loss, J., Xu, J.: On the security of time-lock puzzles and timed commitments. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part III. LNCS, vol. 12552, pp. 390–413. Springer, Cham (Nov 2020). https://doi.org/10.1007/978-3-030-64381-2_14
21. Lysyanskaya, A.: Security analysis of RSA-BSSA. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part I. LNCS, vol. 13940, pp. 251–280. Springer, Cham (May 2023). https://doi.org/10.1007/978-3-031-31368-4_10
22. Qin, X., Cai, C., Yuen, T.H.: One-more unforgeability of blind ECDSA. In: Bertino, E., Shulman, H., Waidner, M. (eds.) ESORICS 2021, Part II. LNCS, vol. 12973, pp. 313–331. Springer, Cham (Oct 2021). https://doi.org/10.1007/978-3-030-88428-4_16
23. Tanaka, Y., Ueno, R., Xagawa, K., Ito, A., Takahashi, J., Homma, N.: Multiple-valued plaintext-checking side-channel attacks on post-quantum KEMs. *IACR TCHES* **2023**(3), 473–503 (2023). <https://doi.org/10.46586/tches.v2023.i3.473-503>

A Omitted Preliminaries

A.1 Blind Signatures

Blind signature schemes allow the issuer to sign client's messages (e.g., documents, tokens), without learning anything about the contents of the message. Blind signatures should also be blind, meaning that when the issuer is presented a valid signature, they are unable to tell in which interaction this signature was issued.

Definition 4 (Blind Signatures). *A blind signature scheme BS over the message space $\mathcal{M}$ consists of tuple of PPT algorithms $(\text{KeyGen}, \langle \text{Sign}, \text{User} \rangle, \text{Ver})$ with the following syntax.*

$\text{BS.KeyGen}(1^\lambda)$: On input a security parameter 1^λ , outputs key pair (sk, pk) .
 $\langle \text{BS.Sign}(\text{sk}), \text{BS.User}(\text{pk}, \text{m}) \rangle$: Interactive protocol run between two PPT algorithms - the signer Sign with a secret key sk as their input and the user User with a public key pk and a message $\text{m} \in \mathcal{M}$ as their input. The user User outputs a signature σ if the interaction was successful and $\perp$ otherwise.
 $\text{BS.Ver}(\text{pk}, \text{m}, \sigma)$: On input a public key pk , a message m and a signature σ , outputs $b' = 1$ if the signature is valid and $b' = 0$ otherwise.

Remark 4. We note that the interaction between Sign and User may vary depending on the number of rounds that a specific blind signature scheme requires. In this work, we focus on using the generic definition of blindness independent of the number of rounds, unless it is explicitly specified. Below we give a step-by-step example of protocol $\langle \text{BS.Sign}(\text{sk}), \text{BS.User}(\text{pk}, \text{m}) \rangle$ for a two-move scheme: $(\text{bsm}_0, \text{st}) \leftarrow \text{BS.User}_0(\text{pk}, \text{m})$, $(\text{bsm}_1, b) \leftarrow \text{BS.Sign}_0(\text{sk}, \text{bsm}_0)$, $\sigma \leftarrow \text{BS.User}_1(\text{st}, \text{bsm}_1)$, where bsm_i denotes the outputs of each execution of algorithms BS.User and BS.Sign , that are exchanged between the parties in the protocol. If the signer initiates the protocol, we will omit BS.User_0 .

Definition 5 (Blindness). *We say that the blind signature scheme BS is blind if it holds that every PPT adversary $\mathcal{A}$ has at most negligible advantage in the $\text{Blind}_{\mathcal{A}, \text{BS}}(\lambda)$ experiment. We define the advantage of adversary $\mathcal{A}$ in $\text{Blind}_{\mathcal{A}, \text{BS}}(\lambda)$ as $\text{Adv}_{\text{Blind}}(\mathcal{A}) = |\Pr[\text{Blind}_{\mathcal{A}, \text{BS}}(\lambda) = 1] - \frac{1}{2}|$. Similarly, we will use $\text{HBlind}_{\mathcal{A}, \text{BS}}$ to denote the honest-key blindness experiment, where the signing keys are generated honestly rather than allowing the adversary to generate them.*

In this blindness experiment the adversary $\mathcal{A}$ takes the role of the malicious signer against challenger $\mathcal{C}$ that controls the user oracles and verifies the signatures. The definition of blindness includes malicious key generation, where $\mathcal{A}$ outputs public key pk that can (but does not have to) result from running BS.KeyGen algorithm. In the same way, $\mathcal{A}$ does not have to hold the secret key sk that corresponds with pk . $\mathcal{A}$ is granted access to two user oracles, where each is instantiated with messages m_b and m_{1-b} . Since the adversary $\mathcal{A}$ has no knowledge of the value of the bit b , it does not know the order of messages held by the user oracles. The verification algorithm BS.Ver is run for both signatures by the challenger $\mathcal{C}$, and if both of them verify correctly, the adversary $\mathcal{A}$ learns tuple (σ_0, σ_1) and $(\perp, \perp)$ otherwise. The experiment is provided in Fig. 6.

$\text{Blind}_{\mathcal{A}, \text{BS}}(\lambda)$ $(\text{st}_0, \text{pk}, \text{m}_0, \text{m}_1) \leftarrow \mathcal{A}(1^\lambda) ; b \leftarrow_{\$} \{0, 1\}$ $\text{st}_1 \leftarrow \mathcal{A}^{(\langle \cdot, \text{User}(\text{pk}, \text{m}_b) \rangle)^1, (\langle \cdot, \text{User}(\text{pk}, \text{m}_{1-b}) \rangle)^1}(\text{st}_0)$ If $\sigma_b = \perp$ or $\sigma_{1-b} = \perp$, then $\mathcal{A}$ learns $(\perp, \perp)$ and (σ_0, σ_1) otherwise. $\hat{b} \leftarrow \mathcal{A}(\text{st}_1, \sigma_0, \sigma_1)$ if $\hat{b} == b$ then return 1 else return 0

Fig. 6. Blindness experiment for Blind Signatures

A.2 Oblivious Pseudorandom Functions (OPRFs)

Informally, the OPRF primitive is a protocol run between a client and a server parties, where the client outputs evaluation $z = F(\text{sk}, x)$ for a pseudorandom function $F(\cdot, \cdot)$ under input x of their choice and server outputs nothing. Obliviousness comes from the fact, that the client party learns nothing about the evaluation key sk held by the server and server learns nothing about client's input x and evaluation $z = F(\text{sk}, x)$. We say that an OPRF is partial, if part of the client's input is provided to the server party as cleartext, usually denoted as a tag t .

Definition 6 ((Partial) OPRF). A (partial) oblivious pseudorandom function scheme OPRF over the input space $\mathcal{X}$ consists of tuple of PPT algorithms (KeyGen, Request, BlindEval, Eval, Finalise) with the following syntax.

OPRF.KeyGen(1^λ): On input security parameter 1^λ , outputs key pair (sk, pk) .
 OPRF.Request(pk, t, x): On input pk , tag t and input x , outputs request req .
 OPRF.BlindEval(sk, t, req): On input sk , tag t and req , outputs response rep .
 OPRF.Eval(sk, t, x): On input secret key sk , (optional) tag t and client's input x , directly outputs OPRF evaluation z .
 OPRF.Finalise(rep): On input takes response rep , outputs client's input OPRF evaluation z or $\perp$ on failure.

Remark 5. We note that when the tag value is fixed beforehand, e.g. $t = \perp$, then the POPRF behaves as generic OPRF scheme.

Definition 7 (Correctness of POPRF evaluation). We say that a partial OPRF scheme $\text{OPRF} = (\text{KeyGen}, \text{Request}, \text{BlindEval}, \text{Eval}, \text{Finalise})$ is correct if for all security parameters $n \in \mathbb{N}$, all key pairs $(\text{sk}, \text{pk}) \leftarrow \text{OPRF.KeyGen}(1^\lambda)$ and any client input $x \in \mathcal{X}$ we have:

POPRIV_{$\mathcal{A}, \text{OPRF}, b$}¹($\lambda$) (sk, pk) $\leftarrow$ OPRF.KeyGen(1^λ) $\hat{b} \leftarrow \mathcal{A}^{\text{Run}^\infty}(1^\lambda, \text{sk}, \text{pk})$ return $\hat{b}$ Oracle Run(t, x_0, x_1) for $j \in \{0, 1\}$ do (st_j, req_j) $\leftarrow$ OPRF.Request(pk, t, x_j) rep_j $\leftarrow$ OPRF.BlindEval(sk, t, req_j) z_j $\leftarrow$ OPRF.Finalise(rep_j, st_j) $\tau_0 \leftarrow (\text{req}_b, \text{rep}_b, z_0)$ $\tau_1 \leftarrow (\text{req}_{1-b}, \text{rep}_{1-b}, z_1)$ return (τ_0, τ_1)	POPRIV_{$\mathcal{A}, \text{OPRF}, b$}²($\lambda$) (st₀, pk) $\leftarrow \mathcal{A}(1^\lambda)$ $i \leftarrow 0$ $\hat{b} \leftarrow \mathcal{A}^{\text{Request}^\infty, \text{Finalise}^\infty}(\text{st}_0)$ return $\hat{b}$ Oracle Request(pk, t, x₀, x₁) $i \leftarrow i + 1$ (st_{i,0}, req₀) $\leftarrow$ OPRF.Request(pk, t, x₀) (st_{i,1}, req₁) $\leftarrow$ OPRF.Request(pk, t, x₁) return (req_b, req_{1-b}) Oracle Finalise(j, rep₀, rep₁) if $j > i$ then return $\perp$ $z_b \leftarrow$ OPRF.Finalise(rep₀, st_{j,b}) $z_{1-b} \leftarrow$ OPRF.Finalise(rep₁, st_{j,1-b}) if $z_0 = \perp$ or $z_1 = \perp$ then return $\perp$ return (z_0, z_1)
--	--

Fig. 7. Request privacy against honest-but-curious servers (left) and against malicious servers (right).

$$\Pr \left[z = \text{OPRF.Eval}(\text{sk}, t, x) : \begin{array}{l} (\text{sk}, \text{pk}) \leftarrow \text{OPRF.KeyGen}(1^\lambda); \\ \text{req} \leftarrow \text{OPRF.Request}(\text{pk}, t, x); \\ \text{rep} \leftarrow \text{OPRF.BlindEval}(\text{sk}, t, \text{req}); \\ z \leftarrow \text{OPRF.Finalise}(\text{rep}); \end{array} \right] = 1.$$

Definition 8 (Request Privacy for OPRFs (POPRIV)). We say that a (Partial) OPRF scheme $\text{OPRF} = (\text{KeyGen}, \text{Request}, \text{BlindEval}, \text{Eval}, \text{Finalise})$ has request privacy against honest-but-curious and malicious servers respectively if it holds that every PPT adversary $\mathcal{A}$ has at most negligible advantage in the $\text{POPRIV}_{\mathcal{A}, \text{OPRF}}^k$ experiments defined in Fig. 7, where $k \in \{1, 2\}$. We define the advantage of adversary $\mathcal{A}$ in $\text{POPRIV}_{\mathcal{A}, \text{OPRF}, b}^k(\lambda)$ experiments as

$$\text{Adv}_{\text{POPRIV}^k}(\mathcal{A}) = \left| \Pr[\text{POPRIV}_{\mathcal{A}, \text{OPRF}, 0}^k(\lambda) = 1] - \Pr[\text{POPRIV}_{\mathcal{A}, \text{OPRF}, 1}^k(\lambda) = 1] \right|.$$

Our POPRIV definition is based on the one presented in [2]. In the experiments depicted in Fig. 7, the adversary $\mathcal{A}$ takes the role of honest-but-curious server

(for POPRIV¹) or malicious server (for POPRIV²). The experiment is initialized with the bit $b \leftarrow \{0, 1\}$, sampled at the beginning of each experiment instance. In the honest-but-curious setting the adversary $\mathcal{A}$ is granted unlimited number of queries to Run oracle, which returns two triples consisting of evaluations z_j for $j \in \{0, 1\}$ along with req and rep values which order depends on the value of the bit b . The goal of the adversary $\mathcal{A}$ is to output a guess $\hat{b}$. In the malicious server setting, the adversary $\mathcal{A}$ is granted unlimited queries to Request and Finalise oracles. In this setting, we also assume malicious generation of public key pk by $\mathcal{A}$. The adversary $\mathcal{A}$ aims to output a guess $\hat{b}$. We want to note that the output of both POPRIV experiments is not dependent on usually expected condition $\hat{b} == b$, but rather represents the bit $\hat{b}$ guessed by $\mathcal{A}$.

A.3 Homomorphic Encryption

We will now recall the notion of public key encryption with homomorphic properties, which is called homomorphic encryption. The main difference is the evaluation function, which allows anyone to compute the homomorphic operations of the ciphertexts. The security property we define for homomorphic encryption is the well-known IND-CPA notion.

Definition 9. *A homomorphic encryption scheme HE consists of PPT algorithms (KeyGen, Enc, Dec, Eval). We assume in this work that the message space of the scheme will always be some ring $\mathcal{R}_{\mathcal{M}}$. Homomorphic operations will be defined as arithmetic circuits from a family $\mathcal{F}$ defined over this ring*

HE.KeyGen(1^λ) : *On input a security parameter 1^λ , outputs keypair (sk, pk) .*
 HE.Enc(pk, m) : *On input a public key pk and message m , outputs ciphertext ct .*
 HE.Dec(sk, ct) : *On input a secret key sk and ciphertext ct , outputs a message m or $\perp$ if decryption fails.*
 HE.Eval($\text{pk}, f, \{\text{ct}_i\}_{i=1}^\ell$) : *On input public key pk , an arithmetic circuit f over $\mathcal{R}_{\mathcal{M}}$, and ciphertexts $\text{ct}_1, \dots, \text{ct}_\ell$, where ℓ is the number of inputs to f , outputs ciphertext ct_f .*

Definition 10 (Correctness of Evaluation). *A homomorphic encryption correctly evaluates a circuit family $\mathcal{F}$ if for all $f \in \mathcal{F}$, for all $m_1, \dots, m_\ell \in \mathcal{R}_{\mathcal{M}}$, for $\text{ct}_f \leftarrow \text{HE.Eval}(\text{pk}, f, \{\text{ct}_i\}_{i=1}^\ell)$, where for all $i \in [\ell]$ $\text{ct}_i \leftarrow \text{HE.Enc}(\text{pk}, m_i)$, we have that $\Pr[\text{HE.Dec}(\text{sk}, \text{ct}_f) \neq f(m_1, \dots, m_\ell)] = \text{negl}(\lambda)$, where the probability is taken over all the randomness in the experiment and $(\text{sk}, \text{pk}) \leftarrow \text{HE.KeyGen}(1^\lambda)$. We say that the scheme compactly evaluates the family if in addition the running time of the decryption circuit only depends on λ and not on its input.*

Definition 11 (Indistinguishability under Chosen Plaintext Attacks). *A homomorphic encryption scheme HE is secure against the well-known IND-CPA attacks (or CPA secure) if it holds that every PPT adversary $\mathcal{A}$ has at most negligible advantage in the $\text{CPA}_{\mathcal{A}, \text{HE}}(\lambda)$ experiment. We define the adversary $\mathcal{A}$ advantage by $\text{Adv}_{\text{CPA}, \text{HE}}(\mathcal{A}) = |\Pr[\text{CPA}_{\mathcal{A}, \text{HE}}(\lambda) = 1] - \frac{1}{2}|$.*

A.4 Non-Interactive Zero-Knowledge (ZK)

Definition 12 (Non-Interactive Zero-Knowledge (ZK)).

Let $\mathcal{R}$ be an NP-relation and $\mathcal{L}_{\mathcal{R}}$ be the language defined by $\mathcal{R}$. A non-interactive proof system for $\mathcal{L}_{\mathcal{R}}$ consists of algorithms (Setup, Prove, Verify) with the following syntax.

Setup(1^λ): Takes as input a security parameter 1^λ , and outputs a common reference string crs .

Prove($\text{crs}, \text{stmt}, \text{wit}$): Takes as input a common reference string crs , a statement stmt and a witness wit , and outputs either a proof π or $\perp$.

Verify($\text{crs}, \text{stmt}, \pi$): Takes as input the common reference string crs , a statement stmt , a proof π , and outputs either 0 or 1.

Perfect Completeness: It holds for all security parameters $\lambda \in \mathbb{N}$, all statements $\text{stmt} \in \mathcal{L}_{\mathcal{R}}$ and all witnesses wit that if $\mathcal{R}(\text{stmt}, \text{wit}) = 1$, $\text{crs} \leftarrow \text{Setup}(1^\lambda)$, and $\pi \leftarrow \text{Prove}(\text{crs}, \text{stmt}, \text{wit})$, then $\text{Verify}(\text{crs}, \text{stmt}, \pi) = 1$.

Computational Soundness: We define the advantage $\text{Adv}_{\text{Sound}}(\mathcal{A})$ of an adversary $\mathcal{A}$ as follows.

$$\Pr \left[\begin{array}{l} \text{Verify}(\text{crs}, \text{stmt}, \arg) = 1 : (\pi, \text{stmt}) \leftarrow \mathcal{A}^{\text{Verify}(\text{crs}, \cdot, \cdot)}(\text{crs}); \\ \text{stmt} \notin \mathcal{L}_{\mathcal{R}} \end{array} \right]$$

where the probability is taken over the random coins of Verify. We say that an argument is sound if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\text{negl}(\lambda)$ such that $\text{Adv}_{\text{Sound}}(\mathcal{A}) \leq \text{negl}(\lambda)$.

Zero-Knowledge: Let λ be a security parameter and $\text{Simul} = (\text{Simul}_1, \text{Simul}_2)$ PPT simulator algorithm. We define the advantage of $\mathcal{A}$ against zero-knowledge as follows:

$$\text{Adv}_{\text{ZK}}(\mathcal{A}) = \left| \Pr \left[\begin{array}{l} \mathcal{A}(\arg^*) = 1 : \begin{array}{l} \text{crs} \leftarrow \text{Setup}(1^\lambda); \\ (\text{stmt}, \text{wit}) \leftarrow \mathcal{A}(\text{crs}); \\ \mathcal{R}(\text{stmt}, \text{wit}) = 1; \\ \pi^* \leftarrow \text{Prove}(\text{crs}, \text{stmt}, \text{wit}_b) \end{array} \right] - \right. \\ \left. \Pr \left[\begin{array}{l} \mathcal{A}(\arg^*) = 1 : \begin{array}{l} (\text{crs}, \text{td}) \leftarrow \text{Simul}_1(1^\lambda); \\ (\text{stmt}, \text{wit}) \leftarrow \mathcal{A}(\text{crs}); \\ \mathcal{R}(\text{stmt}, \text{wit}) = 1; \\ \pi^* \leftarrow \text{Simul}_2(\text{crs}, \text{stmt}, \text{td}) \end{array} \right] \right|,$$

where the probability is taken over random coins of Prove. We say that the argument system is zero-knowledge if for all PPT adversaries $\mathcal{A}$ there exists a negligible function $\text{negl}(\lambda)$ such that $\text{Adv}_{\text{ZK}}(\mathcal{A}) \leq \text{negl}(\lambda)$.

A.5 ECDSA Signature Scheme

ECDSA.KeyGen(1^λ)	ECDSA.Sign(sk, m)	ECDSA.Ver(pk, m, σ)
$\text{sk} \leftarrow \mathbb{Z}_q$	$k \leftarrow \mathbb{Z}_q$	$(r, s) = \sigma$
$\text{pk} \leftarrow g^{\text{sk}}$	$r \leftarrow f_q(g^k)$	$v \leftarrow (g^{\text{H}(m)} \cdot \text{pk}^r)^{s^{-1}}$
return (sk, pk)	$s \leftarrow k^{-1}(\text{H}(m) + r \cdot \text{sk}) \pmod q$	return $f_q(v) == r$
	$\sigma = (r, s)$	
	return σ	

Fig. 8. Elliptic Curve Digital Signature Scheme (ECDSA)

A.6 Algebraic Group Model [15]

In the AGM, the adversary is granted direct access to the group elements. However, manipulating these elements is restricted: it can only produce new group elements by applying the group operation to those it has already obtained. Specifically, each time the adversary outputs a new group element Z , it must also provide a corresponding algebraic representation z , which expresses Z in terms of the group elements that have been previously received. This ensures that the adversary's operations are algebraically traceable within the model. As in the standard model, security proofs in the AGM rely on reductions to computationally hard problems, balancing practical realism and mathematical rigor. This blend of characteristics makes the AGM a valuable tool for analyzing the security of cryptographic protocols.

B Omitted Proofs

B.1 Proof of Theorem 7

Proof. Let $\mathcal{A}$ be an adversary playing the Rumbl experiment. After its execution, $\mathcal{A}$ holds $(\mathbf{m}_0, \sigma_0), (\mathbf{m}_1, \sigma_1)$ where $\sigma_0 = (r_0, s_0)$ is a signature on $\mathbf{m}_0$ and $\sigma_1 = (r_1, s_1)$ is a signature on $\mathbf{m}_1$ (or $\perp$ in case one of the sessions aborts). The adversary furthermore learns two transcripts $T_1 = (R_{1,a}, \text{ct}_{1,1}, \text{ct}_{1,2}, \pi_1, \text{ct}_1)$ and $T_2 = (R_{2,a}, \text{ct}_{2,1}, \text{ct}_{2,2}, \pi_2, \text{ct}_2)$ from its interaction with the first and the second session, respectively. Furthermore, we will use the game-based approach and denote the advantage of adversary $\mathcal{A}$ in G_i (i.e., the $G_0 := \text{Rumbl}$ experiment with small changes) as $\text{Adv}_{G_i}(\mathcal{A}) = |\Pr[G_i = 1] - \frac{1}{2}|$.

Game G_0 : The original Rumbl experiment.

Game G_1 : Same as G_0 , except we abort the experiment in case the adversary queries the random oracle for $\mathbf{m}_0$ or $\mathbf{m}_1$.

Game G_2 : Same as G_1 , except we replace the proof π_1 with a simulated proof using the zero-knowledge property.

Game G_3 : Same as G_2 , except we replace the proof π_2 with a simulated proof using the zero-knowledge property.

Game G_4 : Same as G_3 , except we change how the ciphertext $\text{ct}_{1,2}$ is generated in the first session. Instead of computing it as $\text{ct}_{1,2} \leftarrow \text{HE.Enc}(\text{pk}_{\text{HE}}, m_b)$ it is computed as $\text{ct}_{1,2} \leftarrow \text{HE.Enc}(\text{pk}_{\text{HE}}, m')$ for some random $m' \leftarrow_{\$} \mathbb{Z}_q$.

Game G_5 : We do a similar change for $\text{ct}_{2,2}$ in the second session.

Game G_6 : Same as G_5 , except we change how the ciphertext $\text{ct}_{1,1}$ is generated in the first session. Instead of computing it as $\text{ct}_{1,1} \leftarrow \text{HE.Enc}(\text{pk}_{\text{HE}}, r_b)$ it is computed as $\text{ct}_{1,1} \leftarrow \text{HE.Enc}(\text{pk}_{\text{HE}}, r')$ for some random $r' \leftarrow_{\$} \mathbb{Z}_q$.

Game G_7 : We do a similar change for $\text{ct}_{2,1}$ in the second session.

Claim. We have $\text{Adv}_{G_7}(\mathcal{A}) = 0$.

Proof. The transcripts T_1 and T_2 the adversary receives from interacting with the user oracles are independent of the final message-signature pairs. Therefore, the only valid strategy of the adversary in G_7 is to guess the challenged bit b . $\square$

Claim. For games G_0 and G_1 it holds that: $|\text{Adv}_{G_0}(\mathcal{A}) - \text{Adv}_{G_1}(\mathcal{A})| \leq 2 \cdot q_h/2^\lambda$.

Proof. Since the protocol executions are independent of the messages, i.e., the Paillier ciphertexts contain $H(\mathbf{m})$ and not the actual message, we conclude that the probability of the adversary querying $\mathbf{m}_0$ is $q_h/2^\lambda$. The above statement follows using the union bound for the case of $\mathbf{m}_1$. $\square$

Claim. For games G_1 and G_2 it holds that: $|\text{Adv}_{G_1}(\mathcal{A}) - \text{Adv}_{G_2}(\mathcal{A})| \leq \text{Adv}_{\text{ZK}}(\mathcal{R}_{\text{ZK}})$. For games G_2 and G_3 it holds that: $|\text{Adv}_{G_2}(\mathcal{A}) - \text{Adv}_{G_3}(\mathcal{A})| \leq \text{Adv}_{\text{ZK}}(\mathcal{R}_{\text{ZK}})$.

Sketch. The proof follows directly from the zero-knowledge property. $\square$

Claim. It holds that: $|\text{Adv}_{G_3}(\mathcal{A}) - \text{Adv}_{G_4}(\mathcal{A})| \leq \text{Adv}_{1\text{-RPA-ZT,HE}}(\mathcal{R}_{\text{HE}})$.

Proof. We will show this proof by constructing a reduction $\mathcal{R}_{\text{HE}}$ that will use an adversary $\mathcal{A}$ able to distinguish between G_2 and G_3 to break the 1-RPA-ZT security of the homomorphic encryption scheme HE. In the beginning, the reduction $\mathcal{R}_{\text{HE}}$ receives the public key pk_{DS} for the ECDSA signature scheme. Since we assumed that $\mathcal{A}$ is algebraic, it also outputs how it constructed $\text{pk}_{\text{DS}} = g^{\text{sk}_{\text{DS}}}$. Before the interaction, the only information $\mathcal{A}$ learns are the group parameters and the group generator g . Therefore, the adversary "explains" the construction of pk_{DS} with respect to the generator g and will provide the secret key sk_{DS} to $\mathcal{R}_{\text{HE}}$. Note that this step can be omitted by adding proof of knowledge or making the blindness notion honest-key generation (i.e., the adversary is provided the signer's key by the challenger).

In the next step, the adversary $\mathcal{A}$ initializes both sessions independently by sending $R_{1,a}$ and $R_{2,a}$. Note that $\mathcal{A}$ does not learn any other group elements in the ECDSA group during the protocol exchange. Therefore, since $\mathcal{A}$ is assumed to be algebraic it provides $k_{1,a}$ and $k_{2,a}$ to the reduction $\mathcal{R}$, where $R_{1,a} = g^{k_{1,a}}$ and $R_{2,a} = g^{k_{2,a}}$. The reduction then queries the IND-CPA-ZT challenger, receiving a public key pk_{HE} and a ciphertext ct^* . In the first session, it uses the received public key pk_{HE} and sets $\text{ct}_{1,2} = \text{ct}^*$ while still encrypting the correct message $\text{ct}_{1,1} \leftarrow \text{HE.Enc}(\text{pk}_{\text{HE}}, r)$, where $r = f_q(R_{1,a}^{k_{1,b}})$. The reduction also uses the zero-knowledge simulator to compute π_1 and sends $(\text{pk}_{\text{HE}}, \text{ct}_{1,1}, \text{ct}_{1,2}, \pi_1)$ to the adversary in this session. The other session is simulated according to the protocol, so we omit a detailed description here. At some point, the adversary will reply with the ciphertext ct_1 , which should encrypt the correct value $s_1 = k_{1,a}^{-1}(\text{H}(\mathbf{m}_b) + r \cdot \text{sk})$. Unfortunately, the reduction $\mathcal{R}_{\text{HE}}$ cannot compute the plaintext for ct_1 since the secret key for pk_{HE} is only known to the IND-RPA-ZT challenger.

Instead, the reduction can perform the same operations as an honest signer since it knows sk and $k_{1,a}$. Consequently, the reduction can compute a ciphertext ct'_1 for a plaintext s'_1 such that $s_1 - s'_1 = 0 \pmod q$, i.e., both messages are equivalent modulo q . The idea is that $\mathcal{R}$ can compute a ciphertext $\text{ct}_{\text{zero}} \leftarrow \text{HE.Eval}(\text{pk}_{\text{HE}}, g_{\text{zero}}, \{\text{ct}_1, \text{ct}'_1\})$ for function $g_{\text{zero}}(X, Y) = X - Y$ and ask its zero-testing oracle if ct_{zero} encrypts a zero. Depending on the result, the reduction decides to abort this session if ct_{zero} is not encrypting a zero. Otherwise, it waits for the messages $\mathbf{m}_{0,\text{HE}}, \mathbf{m}_{1,\text{HE}}$ provided by the IND-CPA-ZT challenger. Those messages will correspond to the hash values encrypted using the Paillier cryptosystem. Therefore, the reduction picks one of those messages and programs the random oracle so that $\text{H}(\mathbf{m}_b) = \mathbf{m}_{0,\text{HE}}$. For all other queries, the reduction proceeds using the lazy sampling technique. It then computes a valid signature for $\mathbf{m}_b$ as $\sigma_b = (r, s)$, where $s = (k_{1,a} \cdot k_{1,b})^{-1}(\text{H}(\mathbf{m}_{0,\text{HE}}) + r \cdot \text{sk})$. Depending on the other session it provides $(\sigma_0, \mathbf{m}_0, \sigma_1, \mathbf{m}_1)$ to the adversary.

If the adversary decides that we are in G_2 , that means that $H(m_{0,HE})$ was the plaintext in the challenged ciphertext ct^* and therefore the reduction outputs 0 to its IND-CPA-ZT challenger. Otherwise, if the adversary decides that we are in G_3 , then the message in ct^* corresponds to $H(m_{1,HE})$, an independent random value. Therefore, for this case, the reduction returns 1. This proves the claim. $\square$

Claim. For games G_i and G_j and pairs $(i, j) \in \{(4, 5), (5, 6), (6, 7)\}$ it holds that:

$$|\text{Adv}_{G_i}(\mathcal{A}) - \text{Adv}_{G_j}(\mathcal{A})| \leq \text{Adv}_{1\text{-RPA-ZT,HE}}(\mathcal{R}_{HE}).$$

$\square$

B.2 Proof of Theorem 9

Proof. We will first show that the advantage of any adversary against the blindness experiment **Blind** is only lowered by a negligible factor by introducing the abort condition in the modified experiment **Blind_{SDCh}**.

Let us define the event E , which corresponds to the exact abort condition that differentiates experiments **Blind** and **Blind_{SDCh}**, i.e., the event is triggered if given the public key pk , transcripts $\text{tran}_0, \text{tran}_1$ and signatures σ_0, σ_1 from the two interactions with the adversary for any $i \in \{0, 1\}$ we have: $\sigma_i = \perp$ while $\text{SDCh}(pk, \text{tran}_i) = 1$ or $\sigma_i \neq \perp$ while $\text{SDCh}(pk, \text{tran}_i) = 0$. It is easy to see that

$$|\text{Adv}_{\text{Blind}}(\mathcal{A}) - \text{Adv}_{\text{Blind}_{\text{SDCh}}}(\mathcal{A})| \leq \Pr[E].$$

We now claim that.

Claim. $\Pr[E] \leq 2 \cdot \text{SigDerCheck}_{\mathcal{R}, \text{SDCh}}$.

Proof of Claim. We prove the claim by constructing an adversary $\mathcal{R}$ against the **SigDerCheck** experiment, while playing the challenger role in the blindness experiment. Let $\mathcal{A}$ be an adversary that triggers event E . In the first step, $\mathcal{A}$ sends pk, m_0, m_1 to $\mathcal{R}$. The reduction adversary $\mathcal{R}$ picks a random bit b as per the definition of blindness and sends (pk, m_b) to the challenger of the **SigDerCheck**. All queries of $\mathcal{A}$ for the user oracle instantiated with m_b are forwarded to the **SigDerCheck** experiment challenger instead of running them by $\mathcal{R}$. For the message m_{1-b} , the reduction $\mathcal{R}$ behaves according to the blindness experiment. At the end of the interaction, the adversary triggers the event E . The reduction $\mathcal{R}$ wins the **SigDerCheck** experiment if the event E is triggered for the session corresponding to message m_b . In other words, assume that σ_b is the output for this interaction and trans_b is the transcript of the interaction between $\mathcal{A}$ and the challenger of the **SigDerCheck** experiment. The reduction wins the **SigDerCheck** experiment if for $i = b$ we have $\sigma_i = \perp$ while $\text{SDCh}(pk, \text{tran}_i) = 1$ or $\sigma_i \neq \perp$ while $\text{SDCh}(pk, \text{tran}_i) = 0$. Note that since the event E is triggered if this holds for at least one $i \in \{0, 1\}$. Thus, the probability of $\mathcal{R}$ winning is at least $1/2$, which ends the proofs of the claim. $\square$

Finally, to show the theorem, we have to show no selective-aborts attack exists for experiment **Blind_{SDCh}**. We show this using the following claim.

Claim. Let trans_0 and trans_1 be communication transcripts between the adversary $\mathcal{A}$ and the user oracles, where the adversary outputs public key pk . Let us denote the event F which corresponds to the case when $\text{SDCh}(\text{pk}, \text{trans}_0) = 1$ and $\text{SDCh}(\text{pk}, \text{trans}_1) = 1$. In other words, event F corresponds to the case where the adversary is expected to receive valid signatures, and $\neg F$ is the event where the adversary is expected to receive $(\perp, \perp)$. Thus, we have:

$$\Pr[\text{Blind}_{\mathcal{A}, \text{BS}}(\lambda) = 1] \leq \Pr[F] \cdot \Pr[\text{Blind}_{\mathcal{A}, \text{BS}} \mid F] + \Pr[\neg F] \cdot \Pr[\text{Blind}_{\mathcal{A}, \text{BS}} \mid \neg F],$$

where

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

Proof of Claim. The clue for the claim is that anyone can check if event F happened or not using the public signature-derivation check SDCh . Therefore, inequality is true by simple probability. $\square$

The above claim shows that the adversary's chances of winning the blindness experiment can be upper-bounded by the winning probability conditioned on event F . Given those conditioned probabilities, there is no additional information for the adversary to obtain, and hence, there cannot exist any selective-aborts attacks. $\square$

C Algebraic Evidence for 1-RPA-ZT Security

Scope of this appendix. The main text uses 1-RPA-ZT security as an explicit assumption for the Ruml analysis of the blind ECDSA construction. The generic Ruml-to-blind transformation does not rely on the results in this appendix. The purpose of this section is only to give algebraic evidence that the assumption is plausible for standard homomorphic encryption schemes in the one-query zero-testing setting. The statements below are model-dependent and should not be read as generic implications from IND-CPA security alone.

Lemma 3 (Algebraic one-query zero-testing evidence for Paillier). *In the AGM (unknown order groups [20]), the Paillier encryption scheme is 1-RPA-ZT secure, assuming it is IND-CPA secure. More formally, let $\mathcal{M}$ be the message space of the encryption scheme, then we have: $\text{Adv}_{1\text{-RPA-ZT,HE}}(\mathcal{A}) \leq \text{Adv}_{\text{CPA,HE}}(\mathcal{R}) + 1/|\mathcal{M}|$.*

Proof. We will briefly recall the format of the Paillier encryption scheme. Given a semiprime modulus $n = p \cdot q$, $k = \text{lcm}(q-1, p-1)$ and a generator g for which the inverse $\mu = (L(g^k \bmod n^2))^{-1} \bmod n$ exists, where $L(x) = \frac{x-1}{n}$, the public key is $\text{pk}_{\text{HE}} = (n, g)$ and the secret key is (k, μ) . To encrypt a message $0 \leq m < n$, one computes $\text{ct} \leftarrow g^m \cdot r^n \bmod n^2$ for some $0 < r < n$.

We now observe the following. Assume we have an adversary $\mathcal{A}$ against the 1-RPA-ZT security of the Paillier encryption scheme. The challenger picks bit b , two messages $\mathbf{m}_0, \mathbf{m}_1$ and provides $\text{ct} \leftarrow \text{HE.Enc}(\text{pk}_{\text{HE}}, m_b)$ and pk_{HE} to the adversary. At some point, the adversary will query the zero-testing ZT oracle with input ct^* . Since we assume that $\mathcal{A}$ is algebraic, at the same time, it provides how ct^* was constructed based on prior group elements, so only g and ct . In other words, $\mathcal{A}$ provides (α, β) such that $\text{ct}^* = g^\alpha \cdot \text{ct}^\beta \bmod n^2$. Based on the homomorphic properties of Paillier encryption, we can conclude that ct^* is a ciphertext for the plaintext $m_b \cdot \beta + \alpha \bmod n$. In other words, without the secret key sk_{HE} , we can identify the message that the adversary asks to the ZT oracle.

Our second observation is that we can distinguish four types of zero-testing queries based on (α, β) :

1. $\alpha = 0$ and $\beta = 0$, i.e., the adversary encrypts a zero.
2. $m_b \cdot \beta + \alpha = 0$, i.e., the adversary guessed the correct plaintext of ct ,
3. $m_{1-b} \cdot \beta + \alpha = 0$, i.e., the adversary guesses the second message m_{1-b} ,
4. $t \cdot \beta + \alpha = 0$ for some message $t \notin \{m_0, m_1\}$, i.e., the adversary's ciphertext ct^* is not a zero independent of bit b .

Let us assume for a second that we do not know the bit b , i.e., if ct encrypts m_0 or m_1 . In such a scenario, the last case can be easily distinguished, and in such a case, the ZT oracle outputs 0. This happens when $m_0 \cdot \beta + \alpha \neq 0$ and $m_1 \cdot \beta + \alpha \neq 0$. The first case can also be easily distinguished, leading to the zero-testing oracle answering 1. On the other hand, without bit b , we cannot distinguish between cases 2 and 3. However, we observe that for the second type of query, the ciphertext ct is independent of m_{1-b} , and the probability of this kind

of query happening is $1/|\mathcal{M}|$, i.e., the adversary somehow created a ciphertext ct^* that is zero if ct encrypts $\mathbf{m}_{1-b}$ while having no information about $\mathbf{m}_{1-b}$.

We will construct a reduction $\mathcal{R}$ that uses an adversary $\mathcal{A}$ against the 1-RPA-ZT property of the Paillier cryptosystem to break its IND-CPA security. In the first step, the reduction receives the public key pk_{HE} from the IND-CPA challenger, and it also picks two messages $\mathbf{m}_0$ and $\mathbf{m}_1$. It queries $\mathbf{m}_0, \mathbf{m}_1$ to the IND-CPA challenger and receives the challenge ciphertext ct . It then initializes the adversary $\mathcal{A}$ with $(\text{pk}_{\text{HE}}, \text{ct})$. At some point in the execution, the adversary will ask the zero-testing oracle ZT for ciphertext ct^* . Since the adversary $\mathcal{A}$ is algebraic, it will provide the reduction with (α, β) . The reduction checks if $\mathbf{m}_0 \cdot \beta + \alpha \neq 0$ and $\mathbf{m}_1 \cdot \beta + \alpha \neq 0$. If this is true, the reduction answers the query with 0. If $\alpha = 0$ and $\beta = 0$ the reduction answers with 1. Otherwise, the reduction answers with 1 (note that this corresponds to the cases 2 and 3). At this point, the reduction perfectly simulates the zero-testing oracle except for the case 3, which should be answered with 0, but the reduction answers the adversary with 1. However, since the message space is exponential, the case 3 happens with probability $1/|\mathcal{M}|$, which is considered negligible.

After the one-time zero-testing oracle, the reduction provides $\mathbf{m}_0$ and $\mathbf{m}_1$ to the adversary $\mathcal{A}$ that, in return, outputs the bit b . The reduction outputs the bit b as its solution to the challenger of the IND-CPA experiment. Note that except for the negligible probability of the case 3 happening, the reduction perfectly simulates the 1-RPA-ZT experiment to $\mathcal{A}$. Thus, the advantage of the reduction winning the IND-CPA experiment is the same as that of $\mathcal{A}$ winning the 1-RPA-ZT experiment, assuming case 3 is not happening. $\square$

Lemma 4 (Algebraic one-query zero-testing evidence for ElGamal).

In the algebraic group model, the same simulation strategy gives the analogous one-query 1-RPA-ZT bound for ElGamal encryption, up to the corresponding bad-event probability.

Sketch. We follow the same strategy as in the proof above. We show this proof by constructing a reduction $\mathcal{R}$ that plays the role of the adversary in the IND-CPA experiment and internally uses an adversary $\mathcal{A}$ against the 1-RPA-ZT experiment. The reduction receives the public key pk for the IND-CPA experiment, picks two random messages $\mathbf{m}_0, \mathbf{m}_1$, and provides them to its challenger. The challenger responds with a challenge ciphertext ct . (pk, ct) are then provided to the adversary $\mathcal{A}$ as the 1-RPA-ZT public key and challenge ciphertext.

At some point, the adversary queries the zero testing oracle with a ciphertext ct^* . Since we assumed that $\mathcal{A}$ is algebraic, we have that $\text{ct}^* = (\text{ct}_1^*, \text{ct}_2^*)$, where

$$\text{ct}_1^* = g^{a_1} \cdot \text{pk}^{b_1} \cdot \text{ct}_1^{c_1} \cdot \text{ct}_2^{d_1},$$

and

$$\text{ct}_2^* = g^{a_2} \cdot \text{pk}^{b_2} \cdot \text{ct}_1^{c_2} \cdot \text{ct}_2^{d_2},$$

for some $(a_1, a_2, b_1, b_2, c_1, c_2, d_1, d_2)$ provided by the adversary. $\square$

D Blind ECDSA Scheme from [22]

BS.KeyGen (1^λ) $(\text{sk}, \text{pk}) \leftarrow \text{ECDSA.KeyGen}(1^\lambda)$ return (sk, pk)	BS.User₁ ($\text{pk}, \text{bsm}_0, \text{m}$) $k_b \leftarrow \mathbb{Z}_q^*$ $h \leftarrow H(\text{m})$ $r \leftarrow f_q(R_a^{k_b})$
BS.Ver ($\text{pk}, \text{m}, \sigma$) return $\text{ECDSA.Verify}(\text{pk}, \text{m}, \sigma)$	$(\text{sk}_{\text{HE}}, \text{pk}_{\text{HE}}) \leftarrow \text{HE.KeyGen}(1^\lambda)$ $\text{ct}_1 \leftarrow \text{HE.Enc}(\text{pk}_{\text{HE}}, r; t_1)$ $\text{ct}_2 \leftarrow \text{HE.Enc}(\text{pk}_{\text{HE}}, h; t_2)$ $\pi \leftarrow \text{ZK.P}((\text{ct}_1 \text{ct}_2, \text{pk}_{\text{HE}}), (r, h, t_1, t_2))$ $\text{bsm}_1 = (\text{pk}_{\text{HE}}, \text{ct}_1, \text{ct}_2, \pi)$ $\text{st} = (\text{sk}_{\text{HE}}, h, k_b, r)$ return $(\text{bsm}_1, \text{st})$
BS.Sign₀ ($\perp$) $k_a \leftarrow \mathbb{Z}_q^*$ $R_a \leftarrow g^{k_a}$ return $(\text{bsm}_0 = R_a, \text{st} = k_a)$	BS.User₂ ($\text{pk}, \text{bsm}_2 = \text{ct}, \text{st}$) $(\text{sk}_{\text{HE}}, h, k_b, r) = \text{st}$ $s \leftarrow (\text{HE.Dec}(\text{sk}_{\text{HE}}, \text{ct})) \cdot k_b^{-1} \pmod q$ if $s = 0$ then return $\perp$ $\sigma = (r, s)$ if $\text{ECDSA.Ver}(\text{pk}, \text{m}, \sigma) = 0$ then return $\perp$ return σ
BS.Sign₁ ($\text{sk}, \text{bsm}_1, \text{st}$) if $\text{ZK.V}((\text{ct}_1, \text{ct}_2, \text{pk}_{\text{HE}}), \pi) = 0$ then return $\perp$ $t \leftarrow \mathbb{Z} \cap [\mathcal{M} /q - q^2 - q]$ $k \leftarrow k_a^{-1}$ $g(X, Y) = ((k \cdot \text{sk}) \cdot X + k \cdot Y) + t \cdot q$ $\text{ct} \leftarrow \text{HE.Eval}(\text{pk}_{\text{HE}}, g, \{\text{ct}_1, \text{ct}_2\})$ $\text{bsm}_2 = \text{ct}$ return $(\text{bsm}_2, \cdot)$	

Fig. 9. Blind ECDSA Signature Scheme from [22]. If the message space of HE is $\mathbb{Z}_q$, then the parameter t is not required. The original construction in [22] was instantiated with Paillier cryptosystem with big enough message space $\mathbb{Z}_N$ to ensure the plaintext is not reduced modulo N . We assume that a modulo q computation follows the decryption of the Paillier scheme to ensure that the message space is $\mathbb{Z}_q$.

E VOPRF Scheme from [2]

OPRF.Setup (1^λ) $\mathbf{A}_{\text{pp}} \leftarrow \mathbb{Z}_Q^{N \times N}$ $\text{pp} \leftarrow \mathbf{A}$ return pp	OPRF.KeyGen (1^λ) $\text{pk} \leftarrow \perp$ $\text{sk} \leftarrow \mathbb{Z}_p^{m_p \times n_p}$ $\mathbf{x}_i \leftarrow \mathcal{X} \quad \forall i \in [\beta]$ $\mathbf{z} \leftarrow \text{OPRF.Eval}(\text{sk}, t = \mathbf{t}, x = \mathbf{x})$ $\pi \leftarrow \text{ZK.Prove}((\mathbf{t}, \mathbf{z}, \mathbf{x}), (\mathbf{A}))$ return (pk = (x, z, π), sk)	OPRF.Eval (sk = $\mathbf{A}$, t = $\perp$, x = $\mathbf{x}$) $\mathbf{z}' \leftarrow F_{\mathbf{A}}(\mathbf{t}, \mathbf{x})$ $\mathbf{z} \leftarrow \text{H}(\mathbf{t}, \mathbf{x}, \mathbf{z}')$ return z
OPRF.Request (pk, t = $\perp$, x = $\mathbf{x}$, pp) $\mathbf{A}_{\text{pp}} \leftarrow \text{pp}$ $\hat{\mathbf{x}}, \hat{\mathbf{z}}, \hat{\pi} \leftarrow \text{pk}$ if $\hat{\pi}$ does not verify then abort $\text{pk}_{\text{HE}}, \text{sk}_{\text{HE}} \leftarrow \text{HE.KeyGen}^{(\text{pp})}(1^\lambda)$ $\rho \leftarrow S_m$ $\mathbf{x}' \leftarrow (\mathbf{x}, \hat{\mathbf{x}})$ $\mathbf{x}' \leftarrow \mathbf{x}'_{\rho(i)} \quad \forall i \in [\gamma]$ $\mathbf{y} \leftarrow \text{decomp}(\mathbf{G}_{\text{inp}} \cdot \mathbf{x}' \mod p)$ $\text{ct} \leftarrow \text{HE.Enc}(\text{sk}_{\text{HE}}, \mathbf{y})$ $\pi \leftarrow \text{ZK.Prove}((\text{pk}_{\text{HE}}, \text{ct}), (\text{sk}_{\text{HE}}, \mathbf{y}))$ $\text{req} \leftarrow (\text{pk}_{\text{HE}}, \text{ct}, \pi)$ return req, ρ	OPRF.BlindEval (sk = $\mathbf{A}$, t = $\perp$, req, pp) $(\text{pk}_{\text{HE}}, \text{ct}, \pi) \leftarrow \text{req}$ $g(X) = F_{\mathbf{A}}(\mathbf{t}, X)$ $\text{ct}' \leftarrow \text{HE.Eval}(\text{pk}_{\text{HE}}, g, \text{ct})$ if π does not verify then abort $\text{rep} \leftarrow \text{ct}'$ return rep OPRF.Finalise (sk _{HE} , rep = ct, ρ) if ct not a ctxt, then return $\perp$ $\mathbf{z}^* \leftarrow \text{HE.Dec}(\text{sk}_{\text{HE}}, \text{ct})$ if OPRF.Ver(z, z*, ρ) = 0 then return $\perp$ $\mathbf{z}'_i \leftarrow \mathbf{z}^*_{\rho(i)} \quad \forall i \in [\alpha]$ $\mathbf{z} \leftarrow \text{H}(\mathbf{t}, \mathbf{x}, \mathbf{z}')$; return z	
OPRF.Ver (z, z*, ρ) if $\mathbf{z}_{j, \rho(k+\alpha)}^{(\beta)} \neq \mathbf{z}^*_{\rho(k+\alpha)}$ for any $k \in [\beta]$, return 0 if $\mathbf{z}_{\rho(i_0+r_\alpha l)}^{(\alpha)} \neq \mathbf{z}_{\rho(i_1+r_\alpha l)}^{(\alpha)}$ for $i_0 \neq i_1 \in [r_\alpha]$, return 0 if $\mathbf{z}_{\rho(i_0+r_\alpha l_0)}^{(\alpha)} = \mathbf{z}_{\rho(i_1+r_\alpha l_1)}^{(\alpha)}$ for $i_0, i_1 \in [r_\alpha]$ and $l_0 \neq l_1 \in [r_\alpha]$, return 0 return 1		

Fig. 10. Verifiable oblivious pseudorandom function from [2]. This figure is a high-level transcription of the single-input case of the construction, included only to fix notation for the attack in Section 3.3