Privacy Pass is Anamorphic: Practical Consequences and Attacks in the Black-box Model

Mirosław Kutyłowski
NASK National Research Institute
Warsaw, Poland
miroslaw.kutylowski@nask.pl

Oliwer Sobolewski
NASK National Research Institute
Warsaw, Poland
oliwer.sobolewski@nask.pl

Abstract

Privacy Pass is a cryptographic scheme for issuing one-time anonymous authorization tokens, first designed as an anti-DDoS tool and an alternative to CAPTCHAs. It has attracted a lot of attention from the industry and is currently used and supported by the technological giants such as Cloudflare, Google and Apple. At the same time, Privacy Enhancing Technologies became the focus of European and non-European lawmakers (for example in the eIDAS 2.0 regulation and GDPR). Security and privacy by-design is now quite frequently a formal requirement. Privacy Pass could be used in that context as well as a lightweight solution for many application areas, e.g., for age verification. It is therefore imperative that Privacy Pass is analyzed in all possible aspects and adversarial models that are realistic, yet have not been considered during the design process.

In this work, we first prove that the three most prominent variants of Privacy Pass are anamorphic. Then, we show that anamorphism of Privacy Pass makes it insecure in a model where user's device or client application is working against them (as it can be supplied by a malicious third party, the OS might be subverted or the device could be subverted). Due to anamorphism, the attacks on unlinkability remain undetectable even if an auditor is given all private keys used in the protocol, including the signer/issuer's private key. On the positive side, anamorphism can also be used to achieve a private metadata-like functionality and utilized, for example, for lawful deanonymization of malicious users, without reshaping Privacy Pass.

Keywords

Anamorphic Cryptography, Privacy Pass, Blind Signatures, Kleptography, Algorithm Substitution Attacks

Acknowledgments

This research was funded by the National Science Centre, Poland under the OPUS call in the Weave programme [2023/51/I/ST6/02770]. For the purpose of Open Access, the author has applied a CC-BY public copyright licence to any Author Accepted Manuscript (AAM) version arising from this submission.

1 Introduction

Protecting user's privacy in the context of Internet usage has been a hot topic for more than two decades. On one hand, users' right to privacy is justifiably considered a fundamental human right, and on the other hand, malicious actors (such as botnets) are rampaging through the Internet landscape, causing havoc with Denial of Service attacks, requiring the use of anti-DDoS tools like CAPTCHAs. Unfortunately, it seems that CAPTCHAs are sometimes detrimental to users' privacy. A study in 2023 [44] identified a number of weaknesses of Google's reCAPTCHA system [46] (the most widely adopted anti-DDoS system), one of them being that privacy-invasive tracking cookies are used for risk analysis.

This is why Privacy Pass [22] — a protocol for issuance of one-time anonymous authorization tokens — has been proposed. Human users prove that they are human only once, and then they receive a number of anonymous tokens that they can redeem at any moment to skip CAPTCHA verification. This way, a more secure (and possibly more time-consuming) CAPTCHA system can be used for the initial user validation, and later the tokens can be issued to them at their request, with sporadic re-validations. This system has already been adapted by Cloudflare, one of the biggest content delivery network providers [33]. Furthermore, a standardization effort has been initiated and a set of RFCs [16, 21, 23, 24, 37] and drafts [26, 41, 51, 52] has been developed to promote the use of the Privacy Pass protocol in the industry. As a result, another technological giant, Apple, has introduced private access tokens (*PAT*s) [7] to help eliminate the need for CAPTCHAs on iPhone and Mac devices.

Recently, an extension for Privacy Pass has been proposed by the cryptographic community. This extension allows for including a single bit of *private metadata* in the token [8, 17, 28]. It should be noted that the need for private metadata comes from the industry itself,[1] as it allows for transmitting information regarding a particular user in a covert way, while still respecting their anonymity. This private metadata can only be retrieved by the party in possession of a metadata-retrieval key (e.g. the issuer) and even just a single bit of information could be used to flag high-risk users. Importantly, the private metadata is *not* visible to the users. Otherwise, they could attempt to reverse engineer the risk assessment mechanism and try to bypass it, as is commonly done for CAPTCHAs. Of course, the private metadata could be more than just one bit, however there exists an obvious inverse relation between the number of metadata bits and anonymity of users — the set of all users could be partitioned into a number of buckets proportional to the number of metadata bits, thus reducing their *anonymity set* to just that bucket.

However, in a possible extension of the private metadata functionality, we can motivate the use-case of malicious user deanonymization, since it might be beneficial to identify the troublemaker in

Proceedings on Privacy Enhancing Technologies 2026(4), 561–586

https://doi.org/10.56553/popets-2026-0135

[1] See, e.g. [53], https://datatracker.ietf.org/wg/privacypass/about/ and https://github.com/WICG/trust-token-api?tab=readme-ov-file#extension-metadata.

addition to having them marked. To maintain honest user privacy, we could either give the metadata-retrieval key to a trusted party or distribute it among a coalition of independent, semi-trusted parties, e.g. the issuer and some non-profit organization, as has been proposed to bootstrap a PRIO system [20]. Such a system would allow for deanonymization *only if* a satisfactory number of independent, trusted or semi-trusted parties consented.

Besides being an anti-DDOS tool, Privacy Pass, could find use in other privacy-friendly applications. In 2024, the eIDAS 2.0 regulation [45] has been introduced in the European Union, extending the eIDAS regulation [36] from 2015. The most important aspect of this extension is the definition and requirements concerning the European Digital Identity Wallet (*EDIW*). The EDIW is said to be an application that can be used by European citizens to acquire, store and present Person Identification Data (*PID*) and Electronic Attestations of Attributes (*EAA*s), for, respectively, electronic means of identification and for proving some attribute of an individual (e.g. legal age, possession of a driver's license). According to the eIDAS 2.0 regulation, the functionalities provided by the EDIW should be implemented in a privacy-preserving manner: no disclosure of personal data is allowed unless it is the goal of protocol execution. This leads to the conclusion that schemes with high privacy guarantees (such as Privacy Pass) could be used in practical implementations. It also seems to be in line with the Privacy Pass Working Group objectives, since the new, multi-use token Privacy Pass variant has clearly been designed to be used as a Anonymous Credentials-like scheme, and is adequately named Anonymous Rate-limited Credentials ARC [51, 52].

An example application would be to use Privacy Pass for age verification (a similar idea using blind signatures has been proposed, for example, in [43]), however Privacy Pass could be used for proving in zero-knowledge any binary credential. Specifically, a user could prove their age to a trusted party (e.g., some public sector body), who would then regularly provide the user with batches of one-time authorization tokens that confirm the user's legal age. Although an anonymous credentials scheme could be used instead, as proposed by a team of researchers [11], there are opinions that while anonymous credentials are superior for general-purpose applications, they are too complicated and impossible to deploy in the required time.[2] The problem is that anonymous credentials make use of historically hard to standardize primitives (i.e., pairing-friendly curves[3]) or heavyweight cryptography unfit for weak, mobile devices (i.e., zk-SNARKs), with both having limited support in existing cryptographic hardware. It may therefore take years until proper, standardized solutions appear, while a privacy-preserving age verification system is required right *now*. Insecure, privacy-violating systems (e.g., AI-based age estimation [5]) are being rolled out in many countries due to legislation, e.g. in the UK [4] and in the USA [1, 3]. On the other hand, Privacy Pass is *battle-tested* and already standardized. An age-verification system based on anonymous tokens could be then implemented very rapidly, potentially saving a lot of users from privacy-infringing schemes invented by decision makers.

Of course, a token can be misused by selling it to a third person. A buyer can redeem such a token, as there is no way to check that the person redeeming the token is the same person who obtained it from the issuer. (Note that private metadata-like solution could be used to mark suspicious users' tokens and potentially allow deanonymization in case a user is discovered to be selling tokens or participating in any other malicious activity.) Thus, the tokens and the client-side protocol implementation need to be bound to the user's hardware in some way (this is also required from the legal point of view for EDIW, cf. Section 2). Even for non-EDIW applications, in a use-case like age verification, the necessity of user binding will lead, with high probability, to a situation where there exists a number of approved solutions in which all cryptographic operations are carried out by a secure hardware component, claiming device/user binding as the reason.

If user-side operations are executed inside a secure hardware component, the whole application becomes a *black-box* to the user — they can only verify the inputs and outputs, while the inner operations remain hidden. This leads to the conclusion that a standard security model in which the user controls all operations on their side of the protocol might not be diligent enough for this type of cryptographic scheme.

1.1 Paper Outline

We start by motivating the Black-box User Device model in Section 2. Then, we introduce notation and cryptographic definitions for Privacy Pass in Section 3. In Section 4, we propose formal definitions for anamorphic Privacy Pass. Next, we conduct the analysis by iterating through the three protocols: (1) Privacy Pass with Publicly Verifiable tokens in Section 5; (2) Privacy Pass with Privately Verifiable tokens in Section 6; and (3) Anonymous Rate-limited Credentials in Section 7. The analysis of a singular protocol starts with public-key anamorphic model discussion (if applicable), then we define concrete anamorphic constructions, present attacks on unlinkability and finally discuss countermeasures to these attacks.

1.2 Our Contribution

We formally define Fully-Asymmetric [14] and Public-Key [39] Anamorphic Triplets for *anonymous token* schemes (i.e. Privacy Pass in two standard variants defined in the RFC standards [21, 23, 23, 24]) and in the extended variant called Anonymous Rate-limited Credentials ARC defined in RFC drafts [51, 52]), and thus extend the range of cryptographic schemes covered by anamorphic cryptography. Additionally, we define the *robustness* property [10] and present a novel *full decryptability* property, which makes sure that the anamorphic construction functions as intended even in the presence of an active adversary/Dictator. With some assumptions regarding trust,[4] this anamorphic channel could be used to transmit private metadata [8, 28, 53] and to lawfully deanonymize malicious users, without changing the design of Privacy Pass schemes. It should be noted that private metadata bit functionality as introduced in the standard [53] is not meant to be compatible with

[2]Note that the current EDIW implementation deadline is set by the end of 2026!
[3]However, there is currently some active standardization effort [31].

[4]The assumptions are that either the anamorphic decryption sk_d is given to a trusted third party that will not misuse it, or that sk_d is distributed among a semi-trusted committee, who agree to use the key only when necessary. Additionally, our constructions for publicly verifiable and privately verifiable Privacy Pass allow for threshold decryption, due to ElGamal-like ciphertexts.

standard Privacy Pass schemes (i.e. ones with no private metadata functionality) and is only defined for a separate, special scheme. Conversely, using anamorphism for private metadata is *backwards compatible* with current implementations, which could potentially be beneficial as it allows for deploying lawful deanonymization without deprecating old servers/clients and without introducing a brand new scheme.

We also present a security analysis of the Privacy Pass schemes according to a novel security model in which the user has access to a client device that runs the protocol only as a black-box. This particular security model is relevant for a number of reasons: (1) the client application might be provided as a heavily obfuscated binary, e.g., for the user's phone; (2) the application might be provided inside a secure client device (e.g., a smart card) issued by the protocol provider (e.g., a government if said device is to be used for authorization at a national level), which is a black-box on its own; (3) the application itself might be run on the user's personal device (a phone), but the cryptographic operations might be carried out by an integrated secure hardware component (e.g., a TPM, ESIM), and the exact implementation might not be verifiable; and (4) the application and cryptographic operations might be run on the user's device, even with publicly auditable open-source code, but the OS itself might be subverted by the OS vendor or by malware (the OS could then inject malicious code into the application).

At the same time, a state-level auditor can demand protocol keys from the issuer to audit the protocol (in case malicious implementation is suspected). This scenario is then inherently similar to the case of a Dictator from [38] defined for anamorphic encryption, who can request all encryption keys in order to censor unapproved messages.

As a result of our analysis, we show that for an anamorphic version of Privacy Pass, unlinkability is broken in the black-box device model. Due to the properties of anamorphic protocols, these modified implementations are undetectable *even when an auditor is given access to the issuer's private key.* Thus, a user may trust these techniques implemented in a black box (from the user's point of view) if and only if the provider of the black-box can be fully trusted. As it is rarely the case, the application of Privacy Pass in privacy-sensitive cases can create risks that should be carefully considered.

As an additional contribution, we define a sound (and transparent) countermeasure for the Publicly Verifiable Privacy Pass in the form of an external watchdog client application, which re-randomizes random values in the protocol and thereby destroys anamorphic channels, effectively stopping the most dangerous attacks. Notably, this solution works because the anamorphic model only considers a passive adversary and our construction is not *fully decryptable*. Full decryptability (achieved by our other constructions), assures that no modifications made by a watchdog can close the anamorphic channel without invalidating the tokens. However, note that a watchdog can always be stopped by simply requiring a secure channel between the device and the issuer, or by signing all messages. A secure channel (or other methods of protecting integrity) must be then assumed for constructions using anamorphic channel for a private metadata-like/lawful deanonymization use-cases if the anamorphic scheme does not adhere to full decryptability.

1.3 Previous Work

The security of Privacy Pass [16, 21, 22] in the regular (non black-box) model has been analyzed in [22]. Standardized Blind RSA [24] has been analyzed in the regular (non black-box) model in [32].

Anonymous tokens with private metadata bit have been introduced in [28], further studied in [17] and adapted to a non-interactive version in [8].

Anamorphic cryptography model has been first defined in [38] for encryption schemes, then subsequently defined for signature schemes in [29] and had been extended for new security properties in [9, 10, 14, 39].

Kleptographic analysis of RSA signatures has been carried out in [47–50]. A preliminary analysis of privacy of Schnorr-based blind signature schemes in the black-box User Device model has been presented in [30]. A formalization of kleptographic attacks for symmetric encryption schemes, introduced as algorithm substitution attacks, is presented in [12]. Algorithm substitution attacks consider scenarios in which a user utilizes a black-box device that allegedly implements a symmetric encryption scheme to perform encryption. The attacks are realized by subverting the black-box device, thus allowing the adversary (and only the adversary) to decrypt the user's sensitive communications. This subversion should be undetectable by the user, even if the secret key of the original symmetric encryption scheme is revealed for audit purposes.

Note that our attacks on unlinkability are embedded in the same threat model as algorithm substitution attacks, yet we consider different security requirements based on the anamorphic cryptography model. Importantly, algorithm substitution attacks differ from the anamorphic model in that they require perfect correctness of the output. Specifically, for encryption schemes, *all* ciphertexts generated by the subverted encryption algorithm must be decryptable using the standard decryption algorithm. In anamorphic cryptography, we only require computational indistinguishability of the ciphertexts, without perfect correctness. Additionally, anamorphic cryptography admits a number of useful extensions (e.g., robustness and asymmetric keys) that are absent in algorithm substitution attacks.

2 Black-box Model Motivation

In a standard cryptographic setting, we model the user as the party that executes all steps of the protocol on the user's side. Thus, if we assume that the user is honest, we also assume that all protocol steps on the user's side are executed exactly as defined in the protocol specification. In reality, in most applications, user does not execute the protocol but merely starts the execution. In fact, it is safe to assume that on the user's side, the protocol is executed by some device on behalf of the user. (Note that we use the word *device* in an abstract way — it could be a tamper-proof cryptographic device, a software solution with obfuscated code, or an open-source application subverted by a malicious operating system.) We could then see the device as a *black box* — the user can only observe the inputs and the outputs and do consistency checks. This is a good reason why, in certain cases, the threat model should treat the device as a potentially malicious party. If the user cannot verify what code the device is running, they cannot automatically entrust the claimed security and privacy properties.

The European Digital Identity Wallet implementing regulations currently state that *"Wallet secure cryptographic applications [...] are necessary not only for the protection of critical assets, [...] but also for the provision of crucial functionalities, such as the presentation of electronic attestations of attributes"* [2, Preamble (6)]. We can then assume, that in the case of EDIWs, all cryptographic operations are carried out by a secure cryptographic application/device. In both the standard sense and in the context of the eIDAS regulation, a secure cryptographic device *should be* a black-box, as one of the requirements for it is tamper-resistance. In practice, tamper-resistance means restricting unauthorized physical access to the device, which primarily means protection against device modifications, but also protection against unauthorized read access (compare [34]). Such an access could enable creating a device clone with arbitrary modifications. This indicates that the Black-box User Device model is applicable to EDIWs.

Note that for the private metadata-like/lawful deanonymization use-case, it is only required that the user is not able to modify the user-side code, while read access is not as important — the fact that an anamorphic implementation is utilized can be known to the user.

3 Notation and Prerequisites

Parameters. We define the following parameters used in cryptographic constructions:

- $\lambda \in \mathbb{Z}^+$**:** a security parameter representing the number of bits determining the security level of a cryptographic primitive,
- $N = p \cdot q$**:** an RSA modulus of bit-length $n = \lceil \log_2 N \rceil$, which is determined by the security parameter λ; a product of two primes p, q of the same bit-length $\frac{n}{2}$,
- e**:** a prime; public RSA exponent,
- $g \in \langle g \rangle$**:** a generator of a cyclic group of prime order q with standard security assumptions (hardness of DLP, CDH, DDH) holding for security parameter λ, where $l_g(\lambda)$ is the bit-length of group elements. We do not specify any particular group, but we use multiplicative notation. All computations are implicitly made in $\langle g \rangle$, unless stated otherwise.

Hash Functions. We define the following secure cryptographic hash functions (modeled as Random Oracles):

- $\underline{\mathsf{H} : \{0,1\}^* \to \{0,1\}^\ell}$**:** a cryptographically secure hash function, returning ℓ-bit strings. In practice, H is a standard hash function such as SHA-3.
- $\underline{\mathsf{H}_{\langle g \rangle} : \{0,1\}^* \to \langle g \rangle}$**:** a secure hash function that maps a bit string of any length to an element of the cyclic group generated by g.
- $\underline{\mathsf{H}_q : \{0,1\}^* \to \mathbb{Z}_q^*}$**:** a secure hash function that maps a bit string of any length to an element of $\mathbb{Z}_q^*$.

We define the probabilistic encoding EMSA-PSS $\mathsf{encode}_{\mathsf{PSS}}(\mathsf{t}, n) : \{0,1\}^* \times \mathbb{Z} \to \{0,1\}^{n-1}$ for a message t and bit length n. The corresponding validation function is $\mathsf{valid}_{\mathsf{PSS}}(\mathsf{t}, n, h) : \{0,1\}^* \times \mathbb{Z} \times \{0,1\}^{n-1} \to \{0,1\}$ for a message t, the bit length n and encoding h. For correctness, it is required that for all valid t, $\mathsf{valid}_{\mathsf{PSS}}(\mathsf{t}, n, \mathsf{encode}_{\mathsf{PSS}}(\mathsf{t}, n)) = 1$, and for soundness, for any h' that is not the result of $\mathsf{encode}_{\mathsf{PSS}}(\mathsf{t}, n)$, $\mathsf{valid}_{\mathsf{PSS}}(\mathsf{t}, n, h') = 0$ with overwhelming probability.

Notation. We let $a_{\{i,\ldots,j\}}$ denote the substring of a starting at position i and terminating at position j. Let $|m|$ denote the bit length of the bit string m or the cardinality of the set m (according to the context). By $x \leftarrow \mathsf{PRNG}(\ell, seed)$, we mean taking an ℓ-bit value x from a pseudorandom number generator PRNG seeded with the value *seed*. In turn, $y \leftarrow \mathsf{PRNG}(\ell)$ denotes taking an ℓ-bit value y from a PRNG seeded with a truly random seed of adequate length. Let $[n] := \{1, \ldots, n\}$, and $[x_n]$ represent a tuple $(x_1, \ldots, x_n)$. By $x \leftarrow\!\!\$\ \mathcal{S}$ we mean sampling a value x uniformly at random from the set S, or sampling it from a distribution S or deriving x via a probabilistic algorithm $\mathcal{S}$. By $\mathcal{X}(\cdot)$ we mean that the algorithm $\mathcal{X}$ is executed with arbitrary arguments. If some argument is undefined, we denote it as $\perp$.

Additionally, we use the following syntax for interactive, possibly multi-move, algorithms. Let $\mathcal{X}$ and $\mathcal{Y}$ be two interactive algorithms executed jointly. Then, by $(a, b); \mathcal{T} \leftarrow \langle \mathcal{X}(x), \mathcal{Y}(y) \rangle$ we denote joint execution where x is the private input and a is the private output of $\mathcal{X}$, while y is the private input and b is the private output of $\mathcal{Y}$. We assume that the joint execution also produces a public transcript $\mathcal{T}$ of all messages exchanged between $\mathcal{X}$ and $\mathcal{Y}$.

If three interactive algorithms $\mathcal{X}, \mathcal{Y}, \mathcal{Z}$ are executed jointly in such a way that $\mathcal{Y}$ interacts with both $\mathcal{X}$ and $\mathcal{Z}$, while at the same time $\mathcal{X}, \mathcal{Z}$ do not interact with each other (that is, $\mathcal{Y}$ is a proxy), we denote it as $(a, b, c) \leftarrow \langle \mathcal{X}(x), \mathcal{Y}(y), \mathcal{Z}(z) \rangle$, which is a shorthand expression for $y := (y_1, y_2), \mathcal{Y}(y) := (\mathcal{Y}_1(y_1), \mathcal{Y}_2(y_2)), b := (b_1, b_2)$ and $(a, b_1) \leftarrow \langle \mathcal{X}(x), \mathcal{Y}_1(y_1) \rangle, (b_2, c) \leftarrow \langle \mathcal{Y}_2(y_2), \mathcal{Z}(z) \rangle$. It is worth noting that in that case there are two transcripts $\mathcal{T}_{\mathcal{X}-\mathcal{Y}}$ and $\mathcal{T}_{\mathcal{Y}-\mathcal{Z}}$.

We assume that the Adversary $\mathcal{A}$ has an implicit state and can be invoked in different contexts, depending on the stage of the protocol. By $\mathcal{A}_{\mathcal{X}}(\cdot)$ we denote invoking the Adversary $\mathcal{A}$ with context and state at the stage of the protocol where the algorithm $\mathcal{X}(\cdot)$ would be invoked according to the definition of the protocol.

We use the standard notion of a *negligible function*: a function $\mu(x)$ is negligible if for every positive polynomial $\mathrm{poly}(\cdot)$ there exists an integer $n \in \mathbb{Z} > 0$ such that $|\mu(x)| < \frac{1}{\mathrm{poly}(x)}$ for $x > n$.

Encoding of Group Elements. For covert channels, it is necessary to encode group elements as strings indistinguishable from random. We abstract how this encoding is implemented (as we do not set the group) and instead define a pair of abstract functions:

- $\underline{\mathsf{encode} : \langle g \rangle \to \{0,1\}^\ell}$**:** takes as an input any (random) value from $\langle g \rangle$ (e.g. a point of $E[\mathbb{F}_p]$) and returns an ℓ-bit string (the output distribution must be close to uniform on the set of all ℓ-bit strings).
- $\underline{\mathsf{decode} : \{0,1\}^\ell \to \langle g \rangle}$**:** takes as input any ℓ-bit string and returns an element of $\langle g \rangle$ (the output distribution must be close to uniform on the set of all group elements).

In Section B, we present instantiations for $\langle g \rangle \subseteq \mathbb{Z}_p$ and $\langle g \rangle \subseteq E[\mathbb{F}_p]$. For correctness, we require $\mathsf{decode}(\mathsf{encode}(x)) = x$ for all $x \in \langle g \rangle$. In general, we assume that ℓ is not much greater than the bit length of the order of the group q. We require that decode, when executed on a uniformly random string of length ℓ, returns an element of $\langle g \rangle$ indistinguishable from uniformly random (with negligible security loss). We also require that encode, when executed on a uniformly random group element, returns a string indistinguishable with a ℓ-bit uniformly random string (also with some

negligible security loss). Both of the implementations proposed by us in Section B have these properties.

Proofs of Knowledge. For ARC, we need to define a notation for generic non-interactive proofs of knowledge. Let

$$\pi \leftarrow \mathsf{PoK.Prove}\left\{\begin{pmatrix} S_1 = \dots \\ (w_1, w_2, \dots) : S_2 = \dots \\ \dots \end{pmatrix}\right.$$

be a proving algorithm for statements $S_1, S_2, \dots$ concerning the witnesses $w_1, w_2, \dots$, that returns a proof π and then let $b \leftarrow \mathsf{PoK.Verify}(\pi, S_1, S_2, \dots)$ be a verification algorithm that returns a bit $b \in \{0,1\}$ such that $b = 1$ if the proof is valid.

3.1 Cryptographic Definition of Privacy Pass

Blind Signature Schemes. Since Privacy Pass in Publicly Verifiable mode makes use of blind RSA, we provide a definition for blind signatures for the convenience of the reader in Section A.

Privacy Pass has been introduced by Davidson et al. in [22]. The original concept was to use a Verifiable Oblivious Pseudorandom Function (VOPRF) as the main cryptographic building block. However, that construction allows only the Issuer to validate tokens, as it requires knowledge of Issuer's secret key. Due to this, an alternative version allowing for public token validation, has been introduced in the Privacy Pass RFC standard [15]. An RFC-compliant Privacy Pass can be then instantiated in two modes: publicly verifiable, leveraging a blind RSA signature, and privately verifiable, based on a VOPRF function. As we will argue later, the ARC construction from [51, 52] can also be seen as a Privacy Pass protocol as defined below, but we only state so informally (while showing that the syntax matches).

According to [22], Privacy Pass should have two key properties: (1) *one-more token security* and (2) *unlinkability*, corresponding to analogous properties of blind signatures. These properties can be trivially derived from the corresponding security properties of blind RSA in case of RSA-based version of Privacy Pass.

Because we work in the Black-box User Device mode, we deal with three parties: the User $\mathcal{U}$, who asks for a token for later redemption, the Device $\mathcal{D}$ running the protocol on behalf of the User, and the Issuer $\mathcal{I}$ issuing the tokens. Note that $\mathcal{U}$ is basically an external observer. Let us recall (slightly abstracted) syntax of Privacy Pass [22]:

Definition 1 (Privacy Pass). *A Privacy Pass scheme* Π_{PP} *is a tuple of algorithms* $\Pi_{\mathrm{PP}} = (\mathsf{Gen}, \langle \mathsf{IssueS}, \mathsf{IssueU}\rangle, \mathsf{Redeem})$*, where:*

- $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$**:** *is the key generation procedure that outputs a pair of issuance keys* $(\mathsf{sk}, \mathsf{pk})$ *for* $\mathcal{I}$*.*
- $(\bot, (\mathbf{t}, \boldsymbol{\tau})); \mathcal{T} \leftarrow \langle \mathsf{IssueS}(\mathsf{sk}), \mathsf{IssueU}(\mathsf{pk})\rangle$**:** *is an interactive issuance protocol where* $\mathcal{I}$ *runs* $\mathsf{IssueS}(\mathsf{sk})$*, and* $\mathcal{D}$ *runs* $\mathsf{IssueU}(\mathsf{pk})$*. On the side of* $\mathcal{D}$*, the joint execution outputs a collection of tokens* $\mathbf{t} = (t_1, \dots, t_k)$ *for*[5] $k \in \mathbb{Z}^+$*, and their corresponding tags* $\boldsymbol{\tau} = (\tau_1, \dots, \tau_k)$*. A transcript* $\mathcal{T}$ *of the interaction is visible to all parties.*
- $b \leftarrow \mathsf{Redeem}(\mathsf{sk}\ \mathbf{or}\ \mathsf{pk}, t_i, \tau_i)$**:** *is the redeeming procedure that takes as an input the private key* sk *(in the Privately Verifiable mode) or the public key* pk *(in the Publicly Verifiable mode), a token* t_i *and a tag* τ_i*. It returns a bit* $b \in \{0,1\}$*, where* 1 *stands for 'the token is valid'.*

For ***correctness****, we require that for any keypair* $(\mathsf{sk}, \mathsf{pk})$ *generated by* Gen*,*

$$\textbf{if } \left(\ \bot, (\mathbf{t}, \boldsymbol{\tau})\right) \leftarrow \langle \mathsf{IssueS}(\mathsf{sk}), \mathsf{IssueU}(\mathsf{pk})\rangle \textbf{ then}$$
$$\mathsf{Redeem}(\mathsf{sk}\ \mathbf{or}\ \mathsf{pk}, t_i, \tau_i) = 1 \textbf{ for all } i \in [k], t_i \in \boldsymbol{t}, \tau_i \in \boldsymbol{\tau}.$$

The signing phase consists of joint execution of $\langle \mathsf{IssueS}, \mathsf{IssueU}\rangle$, while the redemption phase consists of the Redeem procedure.

We do not provide a formal definition for One-More Token Security, as we do not use it in the work. For unlinkability, [22] only sketches the formal game, so we define it in Section A. Intuitively, an unlinkable scheme should guarantee that when a token t and its tag τ are revealed to $\mathcal{I}$, there is no substantially better strategy to link (t, τ) with a signing session than random guessing.

4 Anamorphic Definitions for Privacy Pass

Informally, an anamorphic scheme [9, 14, 29, 30, 38, 39] is a cryptographic scheme Π for which it is possible to create a *modified scheme* $\widehat{\Pi}$ that replaces some algorithms Π with corresponding ones from the *Anamorphic Triplet* Σ with additional features for protocol participants. The key point is that a powerful observer who learns the private keys used for Π cannot distinguish between Π and $\widehat{\Pi}$. We note here that such an observer is traditionally called a Dictator, but we call him Auditor $\mathcal{A}$ as in our case, it fits better our application scenarios (and also the traditional Adversary notation).

An anamorphic scheme can be then deployed in two modes: (1) the *regular* mode, where the algorithms from the original scheme Π are used; (2) the *anamorphic* mode, where an anamorphic Sender (who is one of the parties of the original protocol) uses the specially crafted algorithms of Σ to transfer an anamorphic message am encrypted using a double key dkey[6] to the anamorphic Receiver so that the Auditor does not recognize that such operation even took place (which is the strongest form of communication confidentiality). More specifically, the Auditor must be unable to distinguish between the anamorphic and the regular operation mode. Consequently, the Auditor cannot distinguish between the modified scheme and the original scheme.

Before we start defining models, let us recall that for anamorphic schemes, there are two parties: the anamorphic Sender who sends am, and the anamorphic Receiver who is the only party that can recover am. In our work, we intend to send anamorphic messages from $\mathcal{D}$, which makes $\mathcal{D}$ the **anamorphic Sender**. We assume the **anamorphic Receiver** to be the Issuer $\mathcal{I}$, since we allow for changing the IssueS algorithm for the issuance process and it fits with all envisioned use-cases. It is possible, though, for an another, external party to be the anamorphic Receiver. For that, aIssueS must be the same as unmodified IssueS and the Receiver must be able to somehow access γ (defined below). Throughout the work, we will be using Device $\mathcal{D}$ interchangeably with anamorphic Sender, and Issuer $\mathcal{I}$ interchangeably with anamorphic Receiver.

We adapt the Anamorphic Signature Scheme model from [29] to a Privacy Pass version in the more capable Public-Key model

[5] Note that this definition accommodates *batch issuance* of tokens.

[6] dkey is assumed to be hidden from the Auditor — due to the properties of anamorphic schemes, the Auditor cannot ask for dkey as the parties can simply claim that they honestly run the protocol Π and dkey is not defined.

(as originally defined for asymmetric encryption schemes in [39]) wherever possible and in the less capable Fully-Asymmetric model, as defined in [14], where necessary. The differences between the two models are described below. We also define two additional security requirements for anamorphic Privacy Pass — robustness and full decryptability.

Fully-Asymmetric Anamorphic Schemes. In this model [14], the double key dkey is a key pair consisting of a public encryption (Sender) key $\mathsf{pk_d}$ and a private decryption (Receiver) key $\mathsf{sk_d}$. The key $\mathsf{pk_d}$ is assumed to be somehow available, so that the Sender can acquire it before running the anamorphic protocol. Later, during an anamorphic protocol execution, $\mathsf{pk_d}$ is used by Sender to encrypt am. dkey being asymmetric, in contrast to the original private key version [38] of anamorphic cryptography, makes the model more applicable to real-life scenarios. In the older symmetric model, the Sender and the Receiver had to somehow establish a shared secret before executing the protocol without arousing the Auditor's suspicion. The asymmetric Sender key $\mathsf{pk_d}$ should be installed inside $\mathcal{D}$ — the advantage over a symmetric dkey is that even if $\mathcal{D}$ is opened for inspection and $\mathsf{pk_d}$ is recovered, it cannot be used to decrypt previous anamorphic messages am.

Since we focus on Privacy Pass, i.e. anonymous token schemes, we use their syntax for anamorphic constructions. It should be noted though that anonymous tokens and blind signature schemes have a similar structure, so the model can be trivially adapted for blind signature schemes as well (the only change is that the message/token is an input to the issuance protocol and not an output).

Definition 2 (Fully-Asymmetric Anamorphic Triplet for Privacy Pass). *Let $\mathcal{I}$ be an anamorphic Receiver and $\mathcal{D}$ be an anamorphic Sender. Then, a Fully-Asymmetric Anamorphic Triplet Σ is a collection of algorithms* $(\mathsf{aGen}, \langle \mathsf{aIssueS}, \mathsf{aIssueU}\rangle, \mathsf{aDec})$*:*

- $(\mathsf{sk}, \mathsf{pk}, \mathsf{sk_d}, \mathsf{pk_d}) \leftarrow\!\$\ \mathsf{aGen}(1^\lambda)$**:** *on input security parameter 1^λ, it outputs a pair of **anamorphic** protocol keys* $(\mathsf{sk}, \mathsf{pk})$ *for $\mathcal{I}$, the **double** secret key $\mathsf{sk_d}$ for $\mathcal{I}$ and the **double** public key $\mathsf{pk_d}$ for $\mathcal{D}$.*
- $\big(\gamma, (\mathbf{t}, \boldsymbol{\tau})\big); \mathcal{T} \leftarrow \langle \mathsf{aIssueS}(\mathsf{sk}), \mathsf{aIssueU}(\mathsf{pk}, \mathsf{am}, \mathsf{pk_d})\rangle$**:** *on input anamorphic signing key sk, anamorphic verification key pk, anamorphic message am from anamorphic message space $\widehat{\mathcal{M}}$ and double public key $\mathsf{pk_d}$, the joint execution returns:*
  - **(for $\mathcal{I}$):** *some receiver protocol output γ;*
  - **(for $\mathcal{D}$):** *an output tuple $(\mathbf{t}, \boldsymbol{\tau})$ containing k-element tuples of tokens and tags $\mathsf{t}_i \in \mathbf{t}, \tau_i \in \boldsymbol{\tau}$.*
- $(\mathsf{am}) \leftarrow \mathsf{aDec}(\mathsf{sk_d}, \gamma, \mathsf{t}_i, \tau_i)$**:** *on input double private key $\mathsf{sk_d}$, $\mathcal{I}$'s protocol output γ and a token/tag pair t_i, τ_i, the algorithm returns an anamorphic message $\mathsf{am} \in \widehat{\mathcal{M}} \cup \{\perp\}$ (*am *can be* $\perp$[7]*).*

*For **correctness**, we require that for any key tuple* $(\mathsf{sk}, \mathsf{pk}, \mathsf{sk_d}, \mathsf{pk_d})$ *generated by* aGen *and any* $\mathsf{am} \in \widehat{\mathcal{M}}$*:*

$$\textbf{if } \big(\gamma, (\mathbf{t}, \boldsymbol{\tau})\big) \leftarrow \langle \mathsf{aIssueS}(\mathsf{sk}), \mathsf{aIssueU}(\mathsf{pk}, \mathsf{t}, \mathsf{am}, \mathsf{pk_d})\rangle$$
$$\textbf{then } \mathsf{am} = \mathsf{aDec}(\mathsf{sk_d}, \gamma, \mathsf{t}_i, \tau_i) \textbf{ for all } i \in [k], \mathsf{t}_i \in \mathbf{t}, \tau_i \in \boldsymbol{\tau}.$$

[7] This is necessary for the *robustness* property defined later.

Note that this definition allows for decryption of am only after the anamorphic Receiver is given t_i, τ_i (e.g. for verification/authorization), due to the fact that t_i, τ_i are required as arguments of aDec. Also note that this definition assumes that all token, tags pairs from the batch decrypt to the same am.

For security, we require that the resulting modified protocol is indistinguishable from the original one (adapted from [9, 29, 39]):

$\mathcal{O}^0_{\mathsf{IndFAA}}(\mathsf{sk}, \mathsf{pk}, \mathsf{pk_d}, \mathsf{am})$

1: $(\gamma, (\mathbf{t}, \boldsymbol{\tau})); \mathcal{T} \leftarrow \langle \mathsf{aIssueS}(\mathsf{sk}), \mathsf{aIssueU}(\mathsf{pk}, \mathsf{am}, \mathsf{pk_d})\rangle$
2: **return** $((\mathbf{t}, \boldsymbol{\tau}), \mathcal{T})$

$\mathcal{O}^1_{\mathsf{IndFAA}}(\mathsf{sk}, \mathsf{pk}, \mathsf{pk_d}, \mathsf{am})$

1: $(\perp, (\mathbf{t}, \boldsymbol{\tau})); \mathcal{T} \leftarrow \langle \mathsf{IssueS}(\mathsf{sk}), \mathsf{IssueU}(\mathsf{pk})\rangle$
2: **return** $((\mathbf{t}, \boldsymbol{\tau}), \mathcal{T})$

Figure 1: Oracles for the Fully-Asymmetric Indistinguishability game.

Definition 3 (Fully-Asymmetric Anamorphic Privacy Pass). *Let Π be a Privacy Pass scheme consisting of* Gen, ⟨IssueS, IssueU⟩, Redeem. *Let Σ be a Fully-Asymmetric Anamorphic Triplet for Π as defined in Definition 2. We then define a following Indistinguishability game for the Auditor $\mathcal{A}$:*

1. *a bit $b \leftarrow\!\$\ \{0, 1\}$ is randomly selected,*
2. *if $b = 0$, anamorphic keys are generated* $(\mathsf{sk}, \mathsf{pk}, \mathsf{sk_d}, \mathsf{pk_d}) \leftarrow\!\$\ \mathsf{aGen}(1^\lambda)$ *and if $b = 1$, regular keys are generated* $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{Gen}(1^\lambda)$ *and* $(\mathsf{pk_d}, \mathsf{sk_d}) := (\perp, \perp)$*,*
3. *The Auditor $\mathcal{A}^{\mathcal{O}^b_{\mathsf{IndFAA}}(\mathsf{sk}, \mathsf{pk}, \mathsf{pk_d}, \cdot)}(\mathsf{sk}, \mathsf{pk})$ is given* $(\mathsf{sk}, \mathsf{pk})$ *and access to the oracle $\mathcal{O}^b_{\mathsf{IndFAA}}(\mathsf{sk}, \mathsf{pk}, \mathsf{pk_d}, \cdot)$ which they can query with anamorphic messages* am *(Figure 1),*
4. *$\mathcal{A}$ outputs bit b',*
5. *$\mathcal{A}$ **wins** if $b' = b$.*

Then, Π is a Fully-Asymmetric Anamorphic Privacy Pass with Triplet Σ if for every PPT Auditor $\mathcal{A}$ the probability of winning in the Fully-Asymmetric Indistinguishability Game is at most $\frac{1}{2} + \varepsilon$, where ε is negligibly small.

Similarly as noted in [29] for private anamorphic signature schemes, there are two consequences of anamorphism: (1) the anamorphic ciphertext of the anamorphic message am is semantically secure and IND-CPA secure with respect to any party that does not have $\mathsf{sk_d}$ (due to indistinguishability property defined above); (2) assuming the original scheme is secure, the modified scheme is one-more-token secure with respect to parties that do not know the signing key sk or $\mathsf{sk_d}$ and unlinkable for any party (besides the Device) that does not know $\mathsf{sk_d}$. This is due to the fact that without the knowledge of $\mathsf{sk_d}$ (and only $\mathcal{I}$ knows $\mathsf{sk_d}$), $\widehat{\Pi}$ is indistinguishable from Π, and therefore all security properties of Π carry over to the modified version (this is in line with observations in [9, 29]).

Public-Key Anamorphic Schemes. Public-Key Anamorphic schemes differ from the Fully-Asymmetric Anamorphic schemes in that no prior communication between the anamorphic Sender and the anamorphic Receiver is required — the Sender does not know the Receiver's public key $\mathsf{pk_d}$ prior to the execution of the protocol and instead $\mathsf{pk_d}$ is somehow derived from the anamorphic protocol execution. This is an advantage from a practical point of view, as, for example, it might be impossible to embed $\mathsf{pk_d}$ in $\mathcal{D}$ during manufacturing.

According to [39], Public-Key Anamorphic Encryption scheme is a (Fully-Asymmetric) Anamorphic Encryption scheme for which aGen returns an empty $\mathsf{pk_d}$. We provide a similar definition for Privacy Pass protocols:

Definition 4 (Public-Key Anamorphic Privacy Pass). *We say that a Fully-Asymmetric Anamorphic Privacy Pass Protocol* Π *with a Fully-Asymmetric Anamorphic Triplet* Σ *is a Public-Key Anamorphic Privacy Pass Protocol and, correspondingly,* Σ *is a Public-Key Anamorphic Triplet if* $\mathsf{aGen}(1^\lambda)$ *returns an empty* $\mathsf{pk_d}$.

It should be noted that it is implicitly assumed that $\mathsf{sk_d}$ is in fact part of a key pair, and its public counterpart $\mathsf{pk_d}$ can be extracted by the Sender from the anamorphic public pk key or elsewhere during protocol execution.

Robustness. The notion of robustness has been introduced in [10]. It captures the property that it should be impossible to recover any anamorphic message am by running the anamorphic decryption algorithm on the output of the unmodified host protocol. In other words, the anamorphic message recovery algorithm should output $\perp$ if the anamorphic Sender has not embedded any anamorphic message. For Privacy Pass, robustness has to be a little stronger. Due to unlinkability, if there were, say, n signing sessions, $\mathcal{I}$ does not know which γ_i to use with a given token, signature pair t^j, τ^j provided by the User for verification (j is unknown to $\mathcal{I}$, so $\mathcal{I}$ does not know which γ_i to use). Moreover, robustness should also ensure that even when $\mathcal{D}$ embeds another anamorphic message in the signing session j, then $(\mathsf{am}) \leftarrow \mathsf{aDec}(\mathsf{sk_d}, \gamma_i, \mathsf{t}^j, \tau^j)$ outputs $\perp$ for $i \neq j$ (even if the anamorphic messages in both sessions are the same). Thus, assuming $\mathcal{D}$ is running in anamorphic mode, for a given token/signature pair t^j, τ^j, the anamorphic Receiver can iterate through γ_i for all $i \in [n]$ to find the only case where the decryption output is not $\perp$. Note that this property on its own **breaks unlinkability** against anamorphic Receiver.

Concluding, *robustness* for an Anamorphic Privacy Pass protocol is defined as follows:

Definition 5 (Robustness). *Let* Π *be a Public-Key or Fully-Asymmetric Anamorphic Privacy Pass protocol such that* $\mathsf{Gen}, \langle \mathsf{IssueS}, \mathsf{IssueU} \rangle, \mathsf{Redeem} \in \Pi$. *Let* $(\mathsf{aGen}, \langle \mathsf{aIssueS}, \mathsf{aIssueU} \rangle, \mathsf{aDec})$ *be a Public-Key or Fully-Asymmetric Anamorphic Triplet. The Robustness Game for a Distinguisher* $\mathcal{D}$ *is executed as follows:*

1. *A bit* $b \leftarrow\!\!\$ \{0,1\}$ *is selected at random,*
2. *if* $b = 0$*, keys are generated with the procedure* aGen*:* $(\mathsf{sk}, \mathsf{pk}, \mathsf{sk_d}, \mathsf{pk_d}$ ***or*** $\perp) \leftarrow\!\!\$ \mathsf{aGen}(1^\lambda)$*;* *if* $b = 1$*, then* $(\mathsf{sk}, \mathsf{pk}, \mathsf{sk_d}, \mathsf{pk_d}) := (\perp, \perp, \perp, \perp)$,
3. *The Distinguisher* $\mathcal{D}^{O^b_{\mathsf{IndRob}}(\mathsf{sk},\mathsf{pk},\mathsf{sk_d},\mathsf{pk_d}\ \mathbf{or}\perp,\cdot,\cdot,\cdot)}(\perp)$ *is given access to* $O^b_{\mathsf{IndRob}}(\mathsf{sk}, \mathsf{pk}, \mathsf{sk_d}, \mathsf{pk_d}\ \mathbf{or}\ \perp, \cdot, \cdot, \cdot)$ *(the robustness oracle, see Figure 2),*
4. $\mathcal{D}$ *outputs bit* b',
5. $\mathcal{D}$ ***wins*** *if* $b' = b$.

Π *with* Σ *is* ***robust*** *if for any PPT Distinguisher* $\mathcal{D}$*, the probability of winning the Robustness Game is at most* $\frac{1}{2} + \varepsilon$*, where* ε *is negligibly small.*

Intuitively, an anamorphic scheme is *robust* if, when in a situation where: (1) $\mathcal{D}$ is using the regular mode of Π and the anamorphic Receiver tries executing aDec on the output of that session; or (2) $\mathcal{D}$ is using either mode but the anamorphic Receiver tries executing aDec with a combination of Issuer signing session output γ_i of one session and token/tag pair t^j, τ^j from any other session ($i \neq j$), the anamorphic Receiver receives $\perp$ from aDec with overwhelming probability.

$O^0_{\mathsf{IndRob}}(\mathsf{sk}, \mathsf{pk}, \mathsf{sk_d}, \mathsf{pk_d}$ **or** $\perp, \mathsf{single}, \mathsf{am}_1, \mathsf{am}_2)$

```
if single = true :
    (γ, (t, τ)) ← ⟨aIssueS(sk), IssueU(pk)⟩
    i ←$ [k], (t, τ) := (t_i ← t, τ_i ← τ)
    γ̂ := γ
else :
    for j = 1, 2 :
        if am_j ≠⊥:
            (γ_j, (t_j, τ_j)) ← ⟨aIssueS(sk), aIssueU(pk, am_j, pk_d)⟩
        else :
            (γ_j, (t_j, τ_j)) ← ⟨aIssueS(sk), IssueU(pk)⟩
    i ←$ [k], (t, τ) := (t_i ← t_1, τ_i ← τ_1)
    γ̂ := γ_2
am ← aDec(sk_d, γ̂, t, τ)
return am
```

$O^1_{\mathsf{IndRob}}(\mathsf{sk}, \mathsf{pk}, \mathsf{sk_d}, \mathsf{pk_d}$ **or** $\perp, \mathsf{single}, \mathsf{am}_1, \mathsf{am}_2)$

```
return ⊥
```

Figure 2: Oracles for the Robustness Game.

Full Decryptability. Full Decryptability is an extension of anamorphic Privacy Pass, not defined before for any other anamorphic scheme in the literature. Intuitively, it aims to help with an *active* Adversary. This Adversary is allowed to arbitrarily modify the messages exchanged between the Device and the Issuer, but with the constraint that the Device receives correct tokens and tags. Full decryptability ensures that if the Device receives proper tokens and tags, then the Issuer can almost always recover the anamorphic message sent by the Device. Effectively, full decryptability means that anamorphic channels cannot be stopped by a watchdog that re-randomizes random values sent between parties.

Definition 6 (Full Decryptability). *Let* Π *be a Public-Key or Fully-Asymmetric Anamorphic Privacy Pass protocol tuple s.t.* $\mathsf{Gen}, \langle \mathsf{IssueS}, \mathsf{IssueU} \rangle, \mathsf{Redeem} \in \Pi$. *Let* $(\mathsf{aGen}, \langle \mathsf{aIssueS}, \mathsf{aIssueU} \rangle, \mathsf{aDec})$ *be a Public-Key or Fully-Asymmetric Anamorphic Triplet. The Full Decryptability Game runs as follows:*

(1) *anamorphic keys are generated* $(\mathsf{sk}, \mathsf{pk}, \mathsf{sk}_\mathsf{d}, \mathsf{pk}_\mathsf{d}$ ***or*** $\perp) \leftarrow\!\$\ \mathsf{aGen}(1^\lambda)$,
(2) *The Adversary selects an anamorphic message* $\mathsf{am} \leftarrow \mathcal{A}(\mathsf{pk})$,
(3) *The Adversary is given the public key* pk *and is ran as proxy:*

$(\gamma, \perp, (\mathbf{t}, \boldsymbol{\tau})) \leftarrow \langle \mathsf{aIssueS}(\mathsf{sk}), \mathcal{A}(\mathsf{pk}), \mathsf{aIssueU}(\mathsf{pk}, \mathsf{m}, \mathsf{am}, \mathsf{pk}_\mathsf{d}$ **or** $\perp)\rangle$,

(4) *The Adversary* $\mathcal{A}$ ***wins*** *if both* aIssueS *and* aIssueU *do not return* $\perp$ *(the protocol concludes correctly for both sides) and:*

$$\mathsf{Redeem}(\mathsf{sk} \textbf{ or } \mathsf{pk}, \mathsf{t}_i, \tau_i) = 1 \textbf{ for all } i \in [k], \mathsf{t}_i \in \mathbf{t}, \tau_i \in \boldsymbol{\tau}$$
$$\textbf{and there exists } j \in [k], \mathsf{t}_j \in \mathbf{t}, \tau_j \in \boldsymbol{\tau} \textbf{ such that}$$
$$\mathsf{am}' \leftarrow \mathsf{aDec}(\mathsf{sk}_\mathsf{d}, \gamma, \mathsf{t}_j, \tau_j) \textbf{ and } \mathsf{am} \neq \mathsf{am}'.$$

Π *with* Σ *is* ***fully-decryptable*** *if for any probabilistic polynomial-time Adversary* $\mathcal{A}$*, the probability of winning the Full Decryptability Game is at most* ε*, where* ε *is negligibly small.*

5 Privacy Pass with Publicly Verifiable Tokens

In this variant of Privacy Pass, the issued tokens can be verified by anyone with the Issuer's public key. Specifically, the standards instantiate this variant using blind RSA signatures [16, 24].

Blind RSA signature scheme [19] is one of the simplest blind signature schemes from the conceptual point of view. Recently, in response to increased interest in blind signature schemes (due to several authorization mechanisms developed in recent years based on blind signatures [7, 25, 33, 42]), they have been standardized in the form of RFC documents [16, 24, 37]. Our analysis is based on this standardized version.

Privacy Pass issuance protocol instantiated with Blind RSA signing procedure is presented on Figure 3. Note that this scheme has only been defined for batch size $k = 1$, i.e., single token issuance. According to RFC 9474 [24], the message to be signed can be optionally padded with a random prefix to increase entropy. This optional part is not present in Privacy Pass, and so we skip it. (Although it can be used as an anamorphic channel as well!)

The verification procedure is presented on Figure 15 (in Section F, due to space constraints).

5.1 Public-Key Anamorphic Model Discussion

In order to facilitate our attacks in the Public-Key Anamorphic model, where $\mathcal{D}$ and $\mathcal{I}$ share no information before the protocol starts, we need to have a way of transmitting pk_d to $\mathcal{D}$ *inside* the protocol. We then borrow a trick that has been proposed for kleptographic attacks on RSA signatures [49, 50]. These attacks utilize the upper half of the RSA modulus to transfer the ephemeral part of the ephemeral-static DH protocol.

We define the aGen function for the modified scheme on Figure 4. Note that n is the bit length of N. On the same Figure 4, we present a slightly simplified version of Gen for generating *probable primes* from FIPS 186-5 [40], as one of the algorithms recommended for key generation in the RFC [24], for comparison. We use an abstract primality constraint check $\mathsf{isPrime}(p, n)$ that returns *True* if number p is an n-bit number and satisfies some primality constraints and *False* otherwise. Additional discussion on prime generation as defined in [40] can be found in Section C.

To recover pk_d, $\mathcal{D}$ runs $\mathsf{pk}_\mathsf{d} \leftarrow \mathsf{Recover}(N)$, which simply takes ℓ highest bits of N and decodes them: $\mathsf{pk}_\mathsf{d} := \mathsf{decode}(N_{\{n-\ell,\ldots,n-1\}})$.

Privacy Pass (Pub. V.) $(\perp, (\mathbf{t}, \boldsymbol{\tau})) \leftarrow \langle \mathsf{IssueS}(\mathsf{sk}), \mathsf{IssueU}(\mathsf{pk})\rangle$

Issuer $\mathcal{I}$		**Device** $\mathcal{D}$	
$(\mathsf{sk} = d)$		$(\mathsf{pk} = (N, e))$; chall	
		$\mathsf{t} \leftarrow\!\$\ \{0,1\}^{256}$	: 1
		$T = \mathsf{t} \Vert \mathsf{chall} \Vert \mathsf{pk}$	: 2
		$h \leftarrow\!\$\ \mathsf{encode}_{\mathsf{PSS}}(T, n)$: $\gcd(h, N) = 1$	: 3
		$r \leftarrow\!\$\ \mathbb{Z}_N : \gcd(r, N) = 1$	: 4
	$\xleftarrow{P}$	$P = h \cdot (r)^e \bmod N$	: 5
6: $s = P^d \bmod N$	$\xrightarrow{s}$		
		$s' = s \cdot r^{-1} \bmod N$	: 7
		if $\mathsf{valid}_{\mathsf{PSS}}(\mathsf{m}, n, (s')^e \bmod N) \neq 1$	: 8
		return $\perp$	: 9
		return $(\boldsymbol{t} := (\mathsf{t}), \boldsymbol{\tau} := (s'))$	: 10

Figure 3: Privacy Pass token signing interaction $\langle \mathsf{IssueS}, \mathsf{IssueU}\rangle$ **for publicly verifiable tokens, based on blind RSA. The argument** chall **(context specific authorization challenge) is a semi-public, auxillary protocol argument.**

5.2 Anamorphism of Privacy Pass with Publicly Verifiable Tokens

In this section we assume that $\mathcal{D}$ already has access to pk_d, either by deriving it from $\mathcal{I}$'s public key generated according to aGen presented in previous section (and then we are working in the Public-Key model), or $\mathcal{D}$ knows it from an out-of-band channel (and then we are working in the Fully-Asymmetric model).

On Figure 5 we present the anamorphic pair of interactive algorithms $\langle \mathsf{aIssueS}, \mathsf{aIssueU}\rangle$ and on Figure 6 we show the corresponding anamorphic decryption function aDec. It should be noted that for the Privacy Pass version based on blind RSA, it might be impossible to define a *fully decryptable* anamorphic instantiation. In fact, we later show that no anamorphic channel can exist in P and s under the presence of a watchdog that randomizes the messages sent between $\mathcal{I}$ and $\mathcal{D}$ (unless a secure channel is assumed).

Let $\mathsf{aGen}_{\mathsf{Id}}(1^\lambda)$ be an *identity* key generation function that outputs $(\mathsf{sk}, \mathsf{pk}, \mathsf{sk}_\mathsf{d} := x, \mathsf{pk}_\mathsf{d} := g^x)$ for $x \leftarrow\!\$\ \mathbb{Z}_q^*$, $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{Gen}(1^\lambda)$.

Theorem 1. *Privacy Pass with Publicly Verifiable Tokens protocol (call it* Π_1*) is a Robust Fully-Asymmetric Anamorphic Privacy Pass protocol with Fully-Asymmetric Anamorphic Triplet* $\Sigma_1 = (\mathsf{aGen}_{\mathsf{Id}}, \langle \mathsf{aIssueS}, \mathsf{aIssueU}\rangle, \mathsf{aDec})$*, assuming that*[8] $n - 4 \cdot \ell > 0$.

Proof for Theorem 1 is presented in Section E.1.

Theorem 2. *Privacy Pass with Publicly Verifiable Tokens protocol* Π_1 *is a Robust Public-Key Anamorphic Privacy Pass protocol*

[8] At least 1 bit of padding is required so that we can always satisfy the first constraint in step 9 on Figure 5. ℓ is the length of the output of the encode function, which is $\sim \lceil \log_2 q \rceil$ (e.g. 256 bits), so for standard RSA modulus lengths, e.g. $n = 2048$, this requirement is easily met.

Privacy Pass (Pub. V.) $\mathsf{Gen}(1^\lambda)$	$\mathsf{aGen}(1^\lambda)$
Issuer $\mathcal{I}$	**Issuer** $\mathcal{I}$
$(1^\lambda); e, n = \lceil \log_2 N \rceil$	$(1^\lambda); e, n = \lceil \log_2 N \rceil$, $\ell = \lvert\mathsf{encode}(\cdot)\rvert$
(probable prime variant)	
$p \leftarrow\$ \mathsf{PRNG}(n/2)$	$\mathsf{sk_d} \leftarrow\$ \mathbb{Z}_q^*$, $\mathsf{pk_d} = g^{\mathsf{sk_d}}$
if $p < \sqrt{2} \cdot (2^{n/2-1})$ **or** **not** $\mathsf{isPrime}(p, n/2)$:	$p \leftarrow\$ \mathsf{PRNG}(n/2)$
goto 1	**if** $p < \sqrt{2} \cdot (2^{n/2-1})$ **or** **not** $\mathsf{isPrime}(p, n/2)$:
$q \leftarrow\$ \mathsf{PRNG}(n/2)$	**goto** 2
if $q < \sqrt{2} \cdot (2^{n/2-1})$ **or** $\lvert p - q\rvert \le 2^{n/2-100}$ **or** **not** $\mathsf{isPrime}(q, n/2)$:	$t = \mathsf{encode}(\mathsf{pk_d})$
goto 3	$u \leftarrow\$ \{0,1\}^{n-\ell}$
$N = p \cdot q$	$\widehat{N} = t \Vert u$
$d = e^{-1} \bmod \varphi(N)$	$q = \widehat{N} // p; r = \widehat{N} \bmod p$
return $(\mathsf{sk} := d, \mathsf{pk} := (N, e))$	**if** $q < \sqrt{2} \cdot (2^{n/2-1})$ **or** $\lvert p - q\rvert \le 2^{n/2-100}$ **or** **not** $\mathsf{isPrime}(q, n/2)$ **or** $(p \cdot q)_{\{n-\ell,\ldots,n-1\}} \neq t$:
	goto 5
	$N = p \cdot q = \widehat{N} - r$
	$d = e^{-1} \bmod \varphi(N)$
	return $(\mathsf{sk} := d, \mathsf{pk} := (N, e), \mathsf{sk_d})$

Figure 4: A modified key generation function $\mathsf{aGen}(1^\lambda)$ for transmitting $\mathsf{pk_d}$ in the Public-Key Anamorphic model (all calculations beside Step 1. of aGen are done over $\mathbb{Z}$ and $a//b$ denotes integer division with no remainder of a by b).

Privacy Pass (Pub. V.) $(\gamma, (\mathbf{t}, \boldsymbol{\tau})) \leftarrow \langle \mathsf{aIssueS}(\mathsf{sk}), \mathsf{aIssueU}(\mathsf{pk}, \mathsf{am}, \mathsf{pk_d}\ \mathbf{or}\ \bot)\rangle$		
Issuer $\mathcal{I}$	**Device** $\mathcal{D}$	
$(\mathsf{sk} = d)$	$(\mathsf{pk} = (N, e), \mathsf{am}, \mathsf{pk_d}); n = \lceil \log_2 N \rceil$, $\ell = \lvert\mathsf{encode}(\cdot)\rvert$	
	if $\mathsf{pk_d} :=\bot: \mathsf{pk_d} \leftarrow \mathsf{Recover}(N)$	: 1
	$y \leftarrow\$ \mathbb{Z}_q^*, Y = g^y; a \leftarrow\$ \mathbb{Z}_q^*, A = g^a$	: 2
	$K := (\mathsf{pk_d})^y$	: 3
	$C := \mathsf{am} \cdot K$	: 4
	$v := \mathsf{encode}(Y) \Vert \mathsf{encode}(C)$	: 5
	$u := \mathsf{encode}((\mathsf{pk_d})^a) \Vert \mathsf{encode}(A)$	: 6
	$pad \leftarrow\$ \{0,1\}^{n-4\cdot\ell}$	: 7
	$r := pad \Vert u \Vert v$	: 8
	if $r \ge N$ **or** $\gcd(r, N) \neq 1$: **goto** 2	: 9
	$\mathsf{t} \leftarrow\$ \{0,1\}^{256}$	: 10
	$T = \mathsf{t} \Vert \mathsf{chall} \Vert \mathsf{pk}$	: 11
	$h \leftarrow\$ \mathsf{encode}_{\mathsf{PSS}}(T, n)$	: 12
	if $\gcd(h, N) \neq 1$: **goto** 12	: 13
$\xleftarrow{P}$	$P := h \cdot (r)^e \bmod N$	: 14
continue the signing protocol from Figure 3 (step 6)…		: 15
return $\gamma := (P, d)$	**return** $(\mathbf{t} := (\mathsf{t}), \boldsymbol{\tau} := (s'))$	: 16

Figure 5: The modified pair of interactive algorithms $\langle \mathsf{aIssueS}, \mathsf{aIssueU}\rangle$ for transmitting an anamorphic message $\mathsf{am} \in \langle g\rangle$ in Publicly Verifiable Privacy Pass. Parts only applicable when working in Public-Key Anamorphic model are highlighted with red.

with Public-Key Anamorphic Triplet $\Sigma_2 = (\mathsf{aGen}, \langle \mathsf{aIssueS}, \mathsf{aIssueU}\rangle, \mathsf{aDec})$*, assuming that* $n - 4 \cdot \ell > 0$.

Proof for Theorem 2 is presented in Section E.2.

5.3 Attacks on Unlinkability in Black-box User Device Model

As we alluded to earlier, a robust anamorphic scheme implies broken unlinkability for the anamorphic Receiver. This comes almost directly from the definition of robustness.

Theorem 3 (Robustness breaks Unlinkability). *Assuming the Black-box User Device model, and any malicious* $\mathcal{D}$ *implementing a Privacy Pass protocol* Π *that is a Robust Fully-Asymmetric or Public-Key Anamorphic Privacy Pass protocol with an Anamorphic Triplet* Σ*, if* $\mathcal{D}$ *is running in anamorphic mode, then the unlinkability of the Privacy Pass protocol* Π *for the User of* $\mathcal{D}$ *is broken from the point of view of the anamorphic Receiver.*

Additionally, the User is not able to distinguish whether $\mathcal{D}$ *is running* Π *with* Σ*, or the regular version of* Π.

Proof of Theorem 3 is presented in Section E.3.

Alternative Linking Mechanism via Single Anamorphic Message. Although we can break unlinkability with robustness, $\mathcal{I}$ and $\mathcal{D}$ might want to run the protocol in the anamorphic mode only once, and then switch to regular mode, instead of relying on linking due to the anamorphic channel. This can be achieved by establishing a shared key.

First, $\mathcal{D}$ uses the anamorphic channel to send an anamorphic message (a shared key) $\mathsf{skey} \leftarrow\$ \langle g\rangle$ to $\mathcal{I}$. Then, after receiving the message/signature pair from that interaction, $\mathcal{I}$ can recover skey. From properties of the anamorphic mode, skey is secret from the Auditor $\mathcal{A}$ and the User. Additionally, $\mathcal{D}$ and $\mathcal{I}$ need to share a (semi-public) counter L that is unique for each interaction. It might contain: (1) the history of previous interactions for issuing blinded signatures for the User; (2) sequential counter of the User's signing interactions; (3) current time; (4) User identity (e.g. email address).

During a signing session, $\mathcal{D}$ calculates r in the following way, instead of choosing it uniformly at random from $\mathbb{Z}_N^*$ ($\mathsf{H}_{\mathbb{Z}_N^*}$ is a hash function modeled as random oracle that maps any bitstring to an element of $\mathbb{Z}_N^*$):

$$r := \mathsf{H}_{\mathbb{Z}_N^*}(\mathsf{skey} \Vert L)$$

Privacy Pass (Pub. V.) $\mathsf{am} \leftarrow \mathsf{aDec}(\mathsf{sk_d}, \gamma, \mathsf{t}, \tau)$

Issuer $\mathcal{I}$

$(\mathsf{sk_d}, \gamma = (P, d), \mathsf{t}, \tau = s'); \ell = |\mathsf{encode}(\cdot)|$

1: $h := (s')^e \bmod N$
2: $R := P \cdot h^{-1} \bmod N$
3: $r := R^d \bmod N$
4: $C := \mathsf{decode}(r_{\{0,\ldots,\ell-1\}})$
5: $Y := \mathsf{decode}(r_{\{\ell,\ldots,2\cdot\ell-1\}})$
6: $A := \mathsf{decode}(r_{\{2\cdot\ell,\ldots,3\cdot\ell-1\}})$
7: $K := Y^{\mathsf{sk_d}}$
8: $\mathsf{am} := C \cdot K^{-1}$
9: $c := A^{\mathsf{sk_d}}$
10: **if** $c = \mathsf{decode}(r_{\{3\cdot\ell,\ldots,4\cdot\ell-1\}})$:
11: **return** am
12: **else** : **return** $\perp$

Figure 6: The modified anamorphic decryption function $\mathsf{am} \leftarrow \mathsf{aDec}(\mathsf{sk_d}, \gamma, \mathsf{t}, \tau)$, for Publicly Verifiable Privacy Pass.

Since $\mathcal{I}$ can compute r' in the same way, when $\mathcal{I}$ receives P from $\mathcal{D}$, they:

(1) calculate $r' := r = \mathsf{H}_{\mathbb{Z}_N^*}(\mathsf{skey} \| L)$,
(2) recover $h := P/r^e \bmod N$,
(3) store h and the $\mathcal{D}$ identifier for future deanonymization.

During verification, the Issuer $\mathcal{I}$ can just check if the signed message m has the same encoding as one of the stored values.

For an Auditor $\mathcal{A}$, the value r is indistinguishable from a uniformly random value as $\mathcal{A}$ does not know skey used as input for the hash function.

5.4 Countermeasures

Transparent Watchdog. Since the presented anamorphic construction is not fully decryptable, it is possible to stop the anamorphic channel with an *active* countermeasure. In order to do that, one can apply a *transparent* Watchdog to randomize r: the Watchdog converts P to $P \cdot (r^*)^e \bmod N$. On the way back, the Watchdog applies the inverse of r^*. The modifications done by the Watchdog and the full scheme are presented on Figure 16 in Section F. Note that this solution cannot function if integrity of exchanged messages is enforced by e.g. a secure channel between $\mathcal{D}$ and $\mathcal{I}$ or by simply having the parties sign their messages.

The resulting signatures remain correct, since $s = (P^*)^d = (P \cdot (r^*)^e)^d = P^d \cdot r^* \bmod N$ and $(s^*)^e = (s \cdot \frac{1}{r^*})^e = (P^d \cdot r^* \cdot \frac{1}{r^*})^e = P \bmod N$.

With these modifications, the Issuer cannot check the format of P as it becomes uniformly distributed and unpredictable. The value r' that can be recalculated by the Issuer during verification changes to $r' = r^* \cdot r \bmod N$, and since r^* is random, then so is r'. Note that *nothing changes* for either $\mathcal{D}$ or $\mathcal{I}$, and they can be oblivious to the existence of the Watchdog.

Unlinkability property for the Publicly Verifiable Privacy Pass protocol with such Watchdog follows from the following property:

Theorem 4. *Consider invocations Π_1, Π_2 of blind RSA signature creation executed by $\mathcal{I}$ with devices $\mathcal{D}_1, \mathcal{D}_2$ (where $\mathcal{D}_1$ and $\mathcal{D}_2$ might be the same). Let the messages seen by $\mathcal{I}$ during execution of Π_1 be P_1, s_1. Let P_2, s_2 be the messages seen by $\mathcal{D}_2$ during execution of Π_2. Then there is a valid execution of the signing procedure with a watchdog, where the messages exchanged are P_1, s_1 from the point of view of $\mathcal{I}$, and P_2, s_2 from the point of view of $\mathcal{D}_2$.*

Proof for Theorem 4 is presented in Section E.4.

Corollary 1. *If $\mathcal{I}$ links a signature (s') and a transcript (P^*, s) with a device $\mathcal{D}$, it must occur independently of (P^*, s). Since r can only be recalculated by $\mathcal{I}$ based on P^*, s', it must mean that the linking must occur independently of r as well.*

6 Privacy Pass with Privately Verifiable Tokens

In this version of the protocol, the Issuer, holding a private key sk and the corresponding public key $\mathsf{pk} = g^{\mathsf{sk}}$, obliviously evaluates for the User a pseudorandom function $F_{\mathsf{sk}}(T) := T^{\mathsf{sk}}$ for the secret input T provided by the User. We show the construction as defined in the RFCs [16, 21] (but generalized to k tokens for consistency), which is slightly different than the original definition in [22]. Token issuance procedure is presented on Figure 7 and the corresponding token redemption procedure is presented on Figure 17 (in Section F, due to space constraints). It is worth noting that the RFC currently only specifies $k = 1$, but there already exists an RFC draft for batch issuance [41].

6.1 Public-Key Anamorphic Model Discussion

For Privacy Pass in the privately verifiable mode, we are in a much tougher situation than for the Blind RSA-based variant if we want to transmit $\mathsf{pk_d}$ in the protocol. First, we cannot use pk to transmit the key, as pk is generated as a result of a one-way function (i.e. g^{sk}), and we cannot embed more than few bits there (using rejection sampling). Second, there is hardly any room in the protocol for subliminal messages, as most values are either deterministic or are a result of a one way function. Of course, we cannot write a proper impossibility result since rejection sampling *does* allow us to transmit a very short $\mathsf{pk_d}$ or a part of a longer one.

We then conclude that the only solutions we are left with are rejection sampling or similar techniques, requiring *many signing sessions* in order to fully transfer a secure $\mathsf{pk_d}$. Since these solutions require additional assumptions, we only briefly describe an exemplary construction in Section D. It should be noted that the presented construction is only a *proof of concept*, showing that it is indeed possible to transmit $\mathsf{pk_d}$ by exploiting the scheme itself. In most cases though, it is more practical to transmit $\mathsf{pk_d}$ using other, more standard means.

6.2 Anamorphism of Privacy Pass with Privately Verifiable Tokens

In this section we again assume that $\mathcal{D}$ has already access to $\mathsf{pk_d}$ (Fully-Asymmetric model).

We use a technique from [10]. Let $\mathbf{T} : \langle g \rangle \to \mathbb{Z}_q$ be a look-up table that is used as a part of the Receiver key $\mathsf{sk_d}$. This look-up

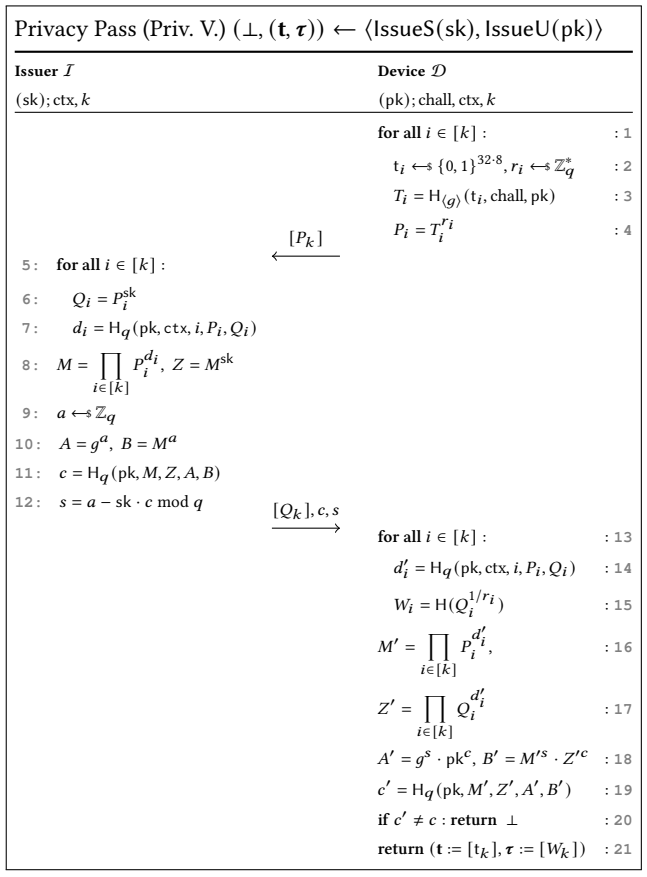

Figure 7: Privacy Pass token signing interaction $\langle \mathsf{IssueS}, \mathsf{IssueU} \rangle$ for privately verifiable tokens. The arguments chall **(context specific authorization challenge provided by the origin) and** ctx **(ciphersuite and mode identifier) are auxillary semi-public arguments used in the protocol.**

table is defined as: $\mathbf{T}[g^{\mathsf{am}}] := \mathsf{am}\ \forall \mathsf{am} \in \widehat{\mathcal{M}}$, i.e. it is a lookup table of discrete logarithms for all possible anamorphic messages in $\widehat{\mathcal{M}} \subset \mathbb{Z}_q$. Of course, this requires $|\widehat{\mathcal{M}}|$ to be sufficiently small to allow all entries of $\mathbf{T}$ to be precomputed in practice (so e.g. $|\widehat{\mathcal{M}}| = 2^{16}$). Note that this precomputation is done only once by $\mathcal{I}$ and not by resource-restrained $\mathcal{D}$.

We assume that the probability $p = |\widehat{\mathcal{M}}|/|\langle g \rangle| = |\widehat{\mathcal{M}}|/q$ that a uniformly sampled $X \leftarrow\$ \langle g \rangle$ satisfies $X = g^m$ for a $m \in \widehat{\mathcal{M}}$ is negligible in λ[9]. The exact composition of $\widehat{\mathcal{M}}$ does not matter for security, so included elements can be chosen arbitrarily.

The key generation function $\mathsf{aGen}(1^\lambda)$ is defined on Figure 8. Anamorphic issuance interactive algorithms $\langle \mathsf{aIssueS}, \mathsf{aIssueU} \rangle$ are presented on Figure 9 while the anamorphic decryption algorithm aDec is presented on Figure 10.

[9]In fact p must be negligible, since if it was not then $\mathbf{T}$ could be used to break DLP, given the assumption that $\mathbf{T}$ can be efficiently computed!

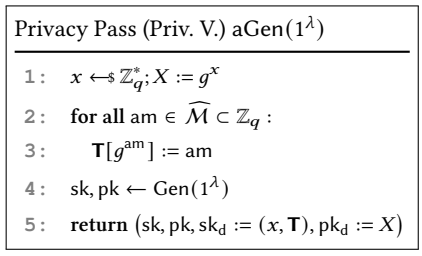

Figure 8: The modified anamorphic key generation function $\mathsf{aGen}(1^\lambda)$ for Privately Verifiable Privacy Pass.

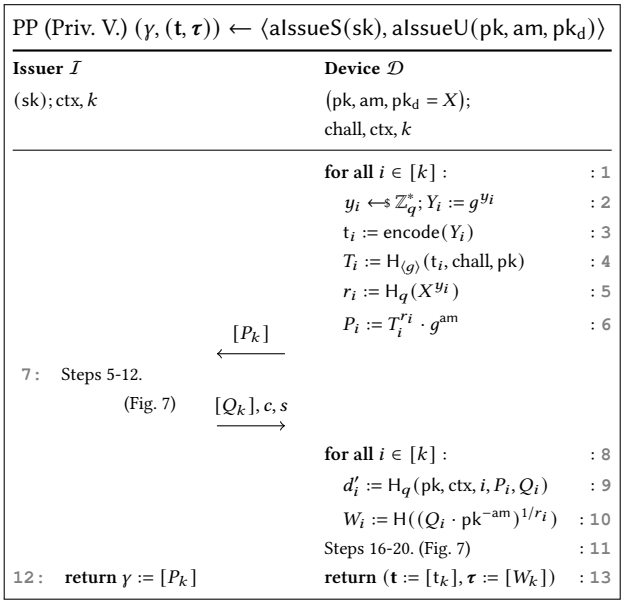

Figure 9: The modified pair of interactive algorithms $\langle \mathsf{aIssueS}, \mathsf{aIssueU} \rangle$ for transmitting an anamorphic message $\mathsf{am} \in \widehat{\mathcal{M}} \subset \mathbb{Z}_q$ in Privately Verifiable Privacy Pass.

Theorem 5. *Privacy Pass with Privately Verifiable Tokens is a Robust, Fully-Decryptable, Fully-Asymmetric Anamorphic Privacy Pass protocol Π_2 with Fully-Asymmetric Anamorphic Triplet $\Sigma_3 = (\mathsf{aGen}, \langle \mathsf{aIssueS}, \mathsf{aIssueU} \rangle, \mathsf{aDec})$ assuming that*[10] *$\ell = |\mathsf{encode}(\cdot)| \approx l_g(\lambda) = 256$.*

Proof of Theorem 5 is presented in Section E.5.

6.3 Attacks on Unlinkability in Black-box User Device Model

For showing that unlinkability is broken, we reuse the results from Theorem 3, since our anamorphic construction for Π_2 is robust and Π_2 is a Privacy Pass protocol.

[10]This assumption is necessary due to the 256-bit bottleneck in t_i. Currently, it is satisfied by ciphersuites [ristretto255, SHA-512] and [P-256, SHA-256] from RFC 9497 [21]. While the constraint could be weakened to $l_g(\lambda) < 256$ and the remaining bits of t_i could be randomly padded, there no ciphersuites for which $l_g(\lambda) < 256$.

Privacy Pass (Priv. V.) $\mathsf{am} \leftarrow \mathsf{aDec}(\mathsf{sk_d}, \gamma, \mathsf{t}_i, \tau_i)$

Issuer $\mathcal{I}$

$(\mathsf{sk_d} = (x, \mathbf{T}), \gamma = [P_k], \mathsf{t}_i, \tau_i)$; chall

1: $x, \mathbf{T} \leftarrow \mathsf{sk_d}$
2: $T := \mathsf{H}_{\langle g \rangle}(\mathsf{t}_i, \mathsf{chall}, \mathsf{pk})$
3: $Y := \mathsf{decode}(\mathsf{t}_i)$
4: $r := \mathsf{H}_q(Y^x)$
5: **for all** $j \in [k]$:
6: $\quad C_j := P_j \cdot T^{-r}$
7: $\quad$ **if** $\mathbf{T}[C_j] \neq \perp$: $\mathsf{am} := \mathbf{T}[C_j]$ **and break**
8: **return** am

Figure 10: The anamorphic decryption function $\mathsf{am} \leftarrow \mathsf{aDec}(\mathsf{sk_d}, \gamma, \mathsf{t}_i, \tau_i)$ **for Privately Verifiable Privacy Pass.**

Alternative Linking Mechanisms. Again, $\mathcal{I}$ and $\mathcal{D}$ might want to run the protocol in the anamorphic mode only a few times, and then switch to regular mode and get immediate deanonimization of all tokens. The attack structure is the same as in Section 5.3. First, $\mathcal{D}$ sends skey of appropriate length over possibly multiple anamorphic messages am_i. Then, let L be defined as in Section 5.3. In subsequent sessions, $\mathcal{D}$ chooses the blinding factors as $r_i \leftarrow \mathsf{H}_q(\mathsf{skey} \| L \| i)$. Then, $\mathcal{I}$ can recalculate r_i and store the unblinded tokens $T_i = P_i^{1/r_i}$.

Additionally, for ciphersuites for which the constraint in Theorem 5 is not satisfied, linking can be done with a simpler attack. We will reuse the assumption that $\mathcal{I}$ and $\mathcal{D}$ share some semi-public counter L_j that uniquely identifies the issuance session j. We also assume that either:

(1) $\mathcal{D}$ and $\mathcal{I}$ are pre-sharing a secret key dkey (like in the first anamorphic encryption definitions from [38]),
(2) or $\mathcal{D}$ has some public key $\mathsf{pk}_{\mathcal{D}}$ and a secret key $\mathsf{sk}_{\mathcal{D}}$ used for identification (e.g. in order to access the signing service provided by $\mathcal{I}$), and then $\mathsf{dkey} := \mathsf{pk}_{\mathcal{D}}^{\mathsf{sk}} = \mathsf{pk}^{\mathsf{sk}_{\mathcal{D}}}$,

Then, $\mathsf{t}_i \leftarrow \mathsf{PRNG}(256, \mathsf{dkey} \| L_j \| i)$, which can be recomputed by $\mathcal{I}$. During verification, $\mathcal{I}$ can find the correct signing session by recalculating t_i for each stored L_j and $i \in [k]$. Since, from assumption, $\mathcal{A}$ can only learn dkey with negligible probability, the attack is undetectable.

6.4 Countermeasures

Transparent Watchdog is Impossible. Observe that for Privacy Pass with Privately Verifiable Tokens it is impossible to define a transparent watchdog since the scheme is *fully-decryptable*.

7 Anonymous Rate-Limited Credentials

Anonymous Rate-Limited Credentials (ARC) [51, 52] is the newest proposal of the Privacy Pass Working Group. It was designed as a replacement to the RlP (*rate-limited Privacy Pass*) draft [26]. ARC provides the additional benefits of multi-use tokens (a single token is unlinkable across multiple presentations) and rate-limiting (a token can be securely used only a fixed number of times). Additionally, unlike RlP, no third party is required. As a drawback, the ARC tokens are only privately verifiable.

ARC is said to be useful in situations where the number of presented zero-knowledge proofs about a token (which is bound to some public information) must not exceed some fixed value. For example, ARC is envisioned for access control of an anonymous user where restrictions on the access rate apply. A simple approach for achieving rate-limiting is to issue tokens with n fixed uses and valid for a fixed time period. If unlinkability is necessary and a simple n-time token does not support unlinkability, one can issue n one-time tokens instead of a single n-time token. However, this approach might be ruled out due to practical issues. Consequently, a more advanced solution, such as ARC, might be necessary.

The ARC scheme of the RFC draft [51] is based on KVAC (*keyed-verification anonymous credentials*) [18, 35]. The ARC scheme, while having slightly different goals due to tokens being multi-use, can be viewed as a (powerful) Privacy Pass scheme as defined in Definition 1 (see discussion below). It also has the same security requirements (unlinkability and one-more-token unforgeability, although the KVAC primitive used in the ARC construction is based on a stronger notion of *extractability*). Formal security of ARC is based on security of KVAC; the draft [51] contains an informal statement of a full-scale security analysis.

Due to space constraints, the original ARC protocol is presented in Section F. The construction requires a second generator in the prime order group $\langle g \rangle$, which we will denote as h. The discrete logarithm $\mathsf{DLog}_g(h)$ is assumed to be unknown to all parties. The key generation function for ARC is presented in Figure 18. The issuance protocol is presented on Figure 19, and the redemption protocol is presented on Figure 20. Note that the token randomization as presented on Figure 19 is normally done for a single token by $\mathcal{D}$ just before presenting that token in a redemption session (the index i is selected at random from a set of indices not used before). However, since the redemption protocol is non-interactive, we use a notational trick and "precompute" all randomized tokens during token issuance. In this way, ARC fits our Privacy Pass syntax perfectly with $k = n$. ARC achieves rate-limiting by making sure that only n nonces/indices are possible and reusing a nonce will result in re-calculating the token already used.

The key component of the ARC scheme are proofs of knowledge created by $\mathcal{D}$ and incorporated into tokens (see Figure 19). These proofs are implemented using Σ-protocols converted to non-interactive proofs via Fiat-Shamir heuristic. More specifically, Schnorr-like proofs are used via a generic Schnorr compiler for linear relations [18, 35, 51]. Since such proofs use quite many random values, there are many opportunities to create anamorphic channels. However, below we show that even without exploiting zero-knowledge proofs, an anamorphic channel can be created without much effort.

7.1 Anamorphism of Anonymous Rate-limited Credentials

In case of ARC, it is very simple to introduce a public-key anamorphic model. Since the ephemeral value u is not revealed to the Auditor, it can be used as the anamorphic Receiver's private key.

In the same spirit, the value r_2 can be used as the sender's ephemeral value, since $M_2 = g^{m_2} \cdot h^{r_2}$ is revealed to the anamorphic

Receiver, and $m_2 = \mathsf{H}_q(\text{chall})$ is semi-public (i.e., at least it is known to the verifier). At this point, both parties have a value that can be used as ephemeral Diffie-Hellman keys, from which the Device can derive a symmetric key. For more details, see Figure 11.

We again use the DLog lookup-table technique from [10] (see Figure 12). Anamorphic encryption is presented in Figure 11, while the anamorphic decryption function is presented in Figure 12. Note that we only modify the part that is run just before token presentation (i.e., the token randomization algorithm).

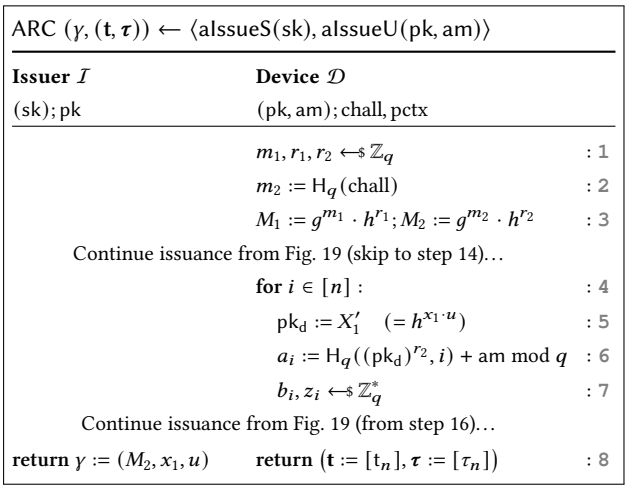

Figure 11: The modified pair of interactive algorithms $\langle \mathsf{aIssueS}, \mathsf{aIssueU} \rangle$ for transmitting an anamorphic message $\mathsf{am} \in \widehat{\mathcal{M}} \subseteq \mathbb{Z}_q$ in ARC.

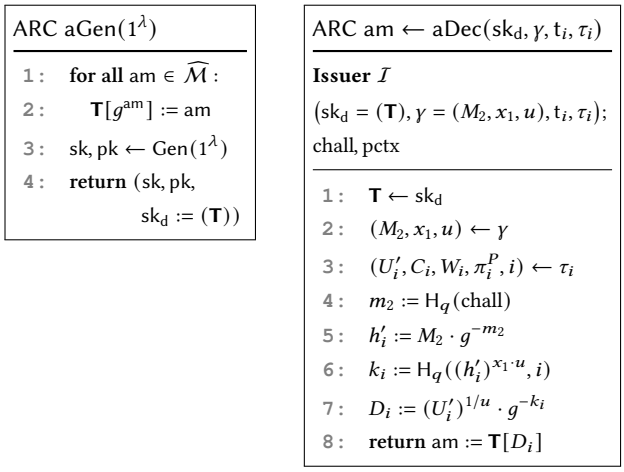

Figure 12: The modified anamorphic key generation function $\mathsf{aGen}(1^\lambda)$ and the anamorphic decryption function $\mathsf{aDec}(\mathsf{sk}_\mathsf{d}, \gamma, \mathsf{t}_i, \tau_i)$ for ARC.

Theorem 6. *ARC is a Robust, Fully-Decryptable, Public-Key Anamorphic Privacy Pass protocol Π_3 with Public-Key Anamorphic Triplet $\Sigma_4 = (\mathsf{aGen}, \langle \mathsf{aIssueS}, \mathsf{aIssueU} \rangle, \mathsf{aDec})$.*

The proof is presented in Section E.6.

7.2 Attacks on Unlinkability in Black-box User Device Model

Again, we reuse the results stated in Theorem 3. Since our construction is robust, and ARC is a Privacy Pass protocol, unlinkability is broken in the Black-box User Device model.

7.3 Countermeasures

Transparent Watchdog is Impossible. Similarly as for Privately Verifiable Privacy Pass, it is impossible to implement a transparent watchdog for Anonymous Rate-limited Credentials, since the anamorphic scheme is *fully-decryptable*.

8 Conclusions

In this paper, we have shown that Privacy Pass, in its three most prominent modes of operation, is indeed anamorphic according to our novel anamorphic model for Privacy Pass-adjacent schemes by constructing practical anamorphic instantiations. As a direct consequence of robust anamorphism, Privacy Pass is susceptible to undetectable attacks on unlinkability in the Black-box User Device model, i.e., when cryptographic operations are executed on behalf of the user by a black-box device.

On the other hand, robust anamorphism can also be used to achieve lawful deanonymization in the Black-box User Device model, given appropriate trust assumptions regarding ownership of the anamorphic decryption key. For example, the key may be shared among a trusted committee that agrees to use it only when properly authorized. Our constructions for anamorphic Publicly and Privately Verifiable Privacy Pass support threshold decryption via, e.g., Lagrange interpolation in the exponent, without recombining sk_d (or $x \in \mathsf{sk}_\mathsf{d}$) in plaintext, due to the ElGamal-like structure of the ciphertexts. Specifically, the committee can compute the shared secret $Y^{\mathsf{sk}_\mathsf{d}}$ in step 7 of aDec from Figure 6, and the shared secret Y^x in step 4 of aDec from Figure 10, directly from their key shares.

Assuming users' devices always operate in anamorphic mode, and given a token/tag pair (t^i, τ^i) corresponding to an offending user, the issuer can provide the committee with the protocol outputs $[\gamma]_n$. This enables the committee to recover the anamorphic message (which itself can contain some metadata regarding the user) $\mathsf{am} \leftarrow \mathsf{aDec}(\mathsf{sk}_\mathsf{d}, \gamma_i, \mathsf{t}^i, \tau^i)$ and thus break unlinkability only for that particular malicious user.

References

[1] California governor signs law requiring online age checks • the register. https://www.theregister.com/2022/09/15/california_aaca_act_signed/. Accessed on 30.08.2025.
[2] Implementing regulation - eu - 2024/2979 - en - eur-lex. https://eur-lex.europa.eu/eli/reg_impl/2024/2979/oj/eng. Accessed on 18.08.2025.
[3] What to know about online age verification laws | ap news. https://apnews.com/article/internet-age-verification-supreme-court-def346d7bf299566a3687d8c4f224fec. Accessed on 30.08.2025.
[4] Why the uk age verification law has led to backlash. https://www.cnbc.com/2025/08/12/why-the-uk-age-verification-law-has-led-to-backlash.html. Accessed on 30.08.2025.
[5] Youtube ai age verification: How it works and what you can do. https://www.msn.com/en-us/news/technology/youtube-ai-age-verification-how-it-works-and-what-you-can-do/ar-AA1LhoJy. Accessed on 30.08.2025.
[6] Michel Abdalla, Chanathip Namprempre, and Gregory Neven. On the (im)possibility of blind message authentication codes. In David Pointcheval,

editor, *Topics in Cryptology – CT-RSA 2006*, pages 262–279, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
[7] Apple. icloud private relay overview. Technical report, December 2021.
[8] Foteini Baldimtsi, Lucjan Hanzlik, Quan Nguyen, and Aayush Yadav. Non-interactive anonymous tokens with private metadata bit. Cryptology ePrint Archive, Paper 2025/430, 2025.
[9] Shalini Banerjee, Tapas Pal, Andy Rupp, and Daniel Slamanig. Simple public key anamorphic encryption and signature using multi-message extensions. Cryptology ePrint Archive, Paper 2025/370, 2025.
[10] Fabio Banfi, Konstantin Gegier, Martin Hirt, Ueli Maurer, and Guilherme Rito. Anamorphic encryption, revisited. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024*, pages 3–32, Cham, 2024. Springer Nature Switzerland.
[11] Carsten Baum, Olivier Blazy, Jan Camenisch, Jaap-Henk Hoepman, Eysa Lee, Anja Lehmann, Anna Lysyanskaya, Renè Mayrhofer, Hart Montgomery, Ngoc Khanh Nguyen, Bart Preneel, abhi shelat, Daniel Slamanig, Stefano Tessaro, Søren Eller Thomsen, and Carmela Troncoso. Cryptographers' feedback on the eu digital identity's arf. https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/discussions/211, 2024. Accessed on 26.08.2025.
[12] Mihir Bellare, Joseph Jaeger, and Daniel Kane. Mass-surveillance without the state: Strongly undetectable algorithm-substitution attacks. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1431–1440, New York, NY, USA, 2015. Association for Computing Machinery.
[13] Daniel J. Bernstein, Mike Hamburg, Anna Krasnova, and Tanja Lange. Elligator: elliptic-curve points indistinguishable from uniform random strings. CCS '13, page 967–980, New York, NY, USA, 2013. Association for Computing Machinery.
[14] Dario Catalano, Emanuele Giunta, and Francesco Migliaro. Anamorphic encryption: New constructions and homomorphic realizations. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024*, pages 33–62, Cham, 2024. Springer Nature Switzerland.
[15] Sofia Celi, Alex Davidson, Steven Valdez, and Christopher A. Wood. Privacy Pass Issuance Protocol. Internet-Draft draft-ietf-privacypass-protocol-16, Internet Engineering Task Force, October 2023. Work in Progress.
[16] Sofia Celi, Alex Davidson, Steven Valdez, and Christopher A. Wood. Privacy Pass Issuance Protocols. RFC 9578, June 2024.
[17] Melissa Chase, F. Betül Durak, and Serge Vaudenay. Anonymous tokens with stronger metadata bit hiding from algebraic macs. In *Advances in Cryptology – CRYPTO 2023: 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20–24, 2023, Proceedings, Part II*, page 418–449, Berlin, Heidelberg, 2023. Springer-Verlag.
[18] Melissa Chase, Sarah Meiklejohn, and Greg Zaverucha. Algebraic macs and keyed-verification anonymous credentials. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, CCS '14, page 1205–1216, New York, NY, USA, 2014. Association for Computing Machinery.
[19] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology*, pages 199–203, Boston, MA, 1983. Springer US.
[20] Henry Corrigan-Gibbs and Dan Boneh. Prio: private, robust, and scalable computation of aggregate statistics. In *Proceedings of the 14th USENIX Conference on Networked Systems Design and Implementation*, NSDI'17, page 259–282, USA, 2017. USENIX Association.
[21] A. Davidson, A. Faz-Hernandez, N. Sullivan, and C. A. Wood. Rfc 9497: Oblivious pseudorandom functions (oprfs) using prime-order groups, 2023.
[22] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, Filippo Valsorda, and Royal Holloway. Privacy pass: Bypassing internet challenges anonymously. *Proceedings on Privacy Enhancing Technologies*, 2018:164–180, 2018.
[23] Alex Davidson, Jana Iyengar, and Christopher A. Wood. The Privacy Pass Architecture. RFC 9576, June 2024.
[24] Frank Denis, Frederic Jacobs, and Christopher A. Wood. RSA Blind Signatures. RFC 9474, October 2023.
[25] Google. How the vpn by google one works | google one. https://one.google.com/about/vpn/howitworks?g1_landing_page=0, 2024. Accessed on 13.03.2024.
[26] Scott Hendrickson, Jana Iyengar, Tommy Pauly, Steven Valdez, and Christopher A. Wood. Rate-Limited Token Issuance Protocol. Internet-Draft draft-ietf-privacypass-rate-limit-tokens-06, Internet Engineering Task Force, April 2024. Work in Progress.
[27] Ari Juels, Michael Luby, and Rafail Ostrovsky. Security of blind digital signatures. In Burton S. Kaliski, editor, *Advances in Cryptology — CRYPTO '97*, pages 150–164, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.
[28] Ben Kreuter, Tancrède Lepoint, Michele Orrù, and Mariana Raykova. Anonymous tokens with private metadata bit. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020*, pages 308–336, Cham, 2020. Springer International Publishing.
[29] Mirosław Kutyłowski, Giuseppe Persiano, Duong Hieu Phan, Moti Yung, and Marcin Zawada. Anamorphic signatures: Secrecy from a dictator who only permits authentication! In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023*, pages 759–790, Cham, 2023. Springer Nature Switzerland.
[30] Mirosław Kutyłowski and Oliwer Sobolewski. Privacy illusion: Subliminal channels in schnorr-like blind-signature schemes. *Applied Sciences*, 15(5), 2025.
[31] Tobias Looker, Vasilis Kalos, Andrew Whitehead, and Mike Lodder. The BBS Signature Scheme. Internet-Draft draft-irtf-cfrg-bbs-signatures-10, Internet Engineering Task Force, January 2026. Work in Progress.
[32] Anna Lysyanskaya. Security analysis of RSA-BSSA. Cryptology ePrint Archive, Paper 2022/895, 2022.
[33] Thibault Meunier, Cefan Daniel Rubin, and Armando Faz-Hernández. Privacy Pass: upgrading to the latest protocol version, 2024.
[34] National Institute of Standards and Technology. Fips 140-3 security requirements for cryptographic modules. 2019.
[35] Michele Orrù. Revisiting keyed-verification anonymous credentials. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, CCS '25, page 1188–1199, New York, NY, USA, 2025. Association for Computing Machinery.
[36] European Parliament. Regulation (eu) no 910/2014 of the european parliament and of the council of 23 july 2014 on electronic identification and trust services for electronic transactions in the internal market and repealing directive 1999/93/ec, July 2014.
[37] Tommy Pauly, Steven Valdez, and Christopher A. Wood. The Privacy Pass HTTP Authentication Scheme. RFC 9577, June 2024.
[38] Giuseppe Persiano, Duong Hieu Phan, and Moti Yung. Anamorphic encryption: Private communication against a dictator. In Orr Dunkelman and Stefan Dziembowski, editors, *Advances in Cryptology - EUROCRYPT 2022, Proceedings, Part II*, volume 13276 of *LNCS*, pages 34–63. Springer, 2022.
[39] Giuseppe Persiano, Duong Hieu Phan, and Moti Yung. Public-key anamorphism in (cca-secure) public-key encryption and beyond. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024*, pages 422–455, Cham, 2024. Springer Nature Switzerland.
[40] Gina M Raimondo and Laurie E Locascio. Fips 186-5 digital signature standard (dss). Technical report, National Institute of Standards and Technology, 2 2023.
[41] Raphael Robert, Christopher A. Wood, and Thibault Meunier. Batched Token Issuance Protocol. Internet-Draft draft-ietf-privacypass-batched-tokens-05, Internet Engineering Task Force, July 2025. Work in Progress.
[42] Paul Schmitt and Barath Raghavan. Pretty good phone privacy. In *30th USENIX Security Symposium (USENIX Security 21)*, pages 1737–1754. USENIX Association, August 2021.
[43] Dominique Schröder and Dominique Unruh. Security of blind signatures revisited. *J. Cryptol.*, 30(2):470–494, April 2017.
[44] Andrew Searles, Renascence Tarafder Prapty, and Gene Tsudik. Dazed & confused: A large-scale real-world user study of recaptchav2. *ArXiv*, abs/2311.10911, 2023.
[45] The European Commission. Regulation (EU) 2024/1183 of the European Parliament and of the Council. https://eur-lex.europa.eu/eli/reg/2024/1183/oj, 2024. Accessed on 11.12.2024.
[46] Luis von Ahn, Benjamin Maurer, Colin McMillen, David Abraham, and Manuel Blum. recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468, 2008.
[47] Adam Young and Moti Yung. Kleptography: Using cryptography against cryptography. In Walter Fumy, editor, *Advances in Cryptology — EUROCRYPT '97*, pages 62–74, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg.
[48] Adam L. Young and Moti Yung. Malicious cryptography - exposing cryptovirology. 2004.
[49] Adam L. Young and Moti Yung. A space efficient backdoor in RSA and its applications. In Bart Preneel and Stafford E. Tavares, editors, *Selected Areas in Cryptography, 12th International Workshop, SAC 2005, Kingston, ON, Canada, August 11-12, 2005, Revised Selected Papers*, volume 3897 of *LNCS*, pages 128–143. Springer, 2005.
[50] Adam L. Young and Moti Yung. A timing-resistant elliptic curve backdoor in RSA. In Dingyi Pei, Moti Yung, Dongdai Lin, and Chuankun Wu, editors, *Information Security and Cryptology, Third SKLOIS Conference, Inscrypt 2007, Xining, China, August 31 - September 5, 2007, Revised Selected Papers*, volume 4990 of *LNCS*, pages 427–441. Springer, 2007.
[51] Cathie Yun, Christopher A. Wood, and Armando Faz-Hernandez. Anonymous Rate-Limited Credentials Cryptography. Internet-Draft draft-ietf-privacypass-arc-crypto-00, Internet Engineering Task Force, February 2026. Work in Progress.
[52] Cathie Yun, Christopher A. Wood, and Armando Faz-Hernandez. Privacy Pass Issuance Protocol for Anonymous Rate-Limited Credentials. Internet-Draft draft-ietf-privacypass-arc-protocol-00, Internet Engineering Task Force, February 2026. Work in Progress.
[53] Cathie Yun, Christopher A. Wood, Mariana Raykova, and Samuel Schlesinger. Anonymous Token with Hidden Metadata Privacy Pass Issuance and Authentication Protocols. Internet-Draft draft-yun-privacypass-athm-00, Internet Engineering Task Force, October 2025. Work in Progress.

A Security Definitions for Blind Signatures and Privacy Pass

In this section we provide formal definitions for blind signature schemes (for completeness) and for the unlinkability property of Privacy Pass (as it is used in proofs).

Blind Signature Schemes. A blind signature scheme is a cryptographic protocol that lets a User $\mathcal{U}$ request a valid signature σ from Issuer[11] $\mathcal{I}$ over a private message t, created with the Issuer's secret key sk in such a way that the Issuer cannot link the signing request with the resulting signature (as long as there is more than one signing request). At the same time, the User cannot create a valid signature without requesting a signing session with the Issuer. In the following definitions we introduce yet another party, the Device $\mathcal{D}$, that runs the User's part of the protocol on their behalf, simulating real-world scenario of a black-box device/application[12].

Definition 7 (Blind Signature Scheme). *A blind signature scheme Π_{BS} is a tuple of algorithms $\Pi_{\mathrm{BS}} = (\mathsf{Gen}, \langle \mathsf{SignS}, \mathsf{SignU}\rangle, \mathsf{Verify})$, where:*

- $\underline{(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{Gen}(1^\lambda)}$**:** *is the key generation procedure that outputs a pair of signing keys* $(\mathsf{sk}, \mathsf{pk})$ *for* $\mathcal{I}$.
- $\underline{(\bot, \sigma); \mathcal{T} \leftarrow \langle \mathsf{SignS}(\mathsf{sk}), \mathsf{SignU}(\mathsf{pk}, \mathsf{t})\rangle}$**:** *is the interactive signing procedure, where $\mathcal{I}$ runs* $\mathsf{SignS}(\mathsf{sk})$*, and $\mathcal{D}$ runs* $\mathsf{SignU}(\mathsf{pk}, \mathsf{t})$ *on behalf of $\mathcal{U}$. The joint execution outputs a signature σ on the side of $\mathcal{D}$ (and $\mathcal{U}$), and nothing on the side of $\mathcal{I}$. A transcript $\mathcal{T}$ of the interaction (consisting of all messages exchanged between $\mathcal{I}$ and $\mathcal{D}$ during an execution) is visible to all parties (including, in particular, $\mathcal{U}$).*
- $\underline{b \leftarrow \mathsf{Verify}(\mathsf{pk}, \mathsf{t}, \sigma)}$**:** *is the verification procedure that takes as input the public key* pk*, a message* t *and a signature σ. It returns $b \in \{0, 1\}$, where* 1 *stands for 'the signature is valid'.*

For ***correctness****, we require that for any keypair* $(\mathsf{sk}, \mathsf{pk})$ *generated by* Gen *and all messages* t *from the message space $\mathcal{M}$:*

$$\textbf{if}\ \ (\bot, \sigma) \leftarrow \langle \mathsf{SignS}(\mathsf{sk}), \mathsf{SignU}(\mathsf{pk}, \mathsf{t})\rangle,\ \ \textbf{then}\ \ \mathsf{Verify}(\mathsf{pk}, \mathsf{t}, \sigma) = 1.$$

We consider the security definition for blindness as defined in [6, 27, 43]. We do not provide a security definition for One-More Unforgeability, as it is not relevant in our work.

Intuitively, blindness means that it should be infeasible for an issuer $\mathcal{I}$ to link a signing session with a (message, signature) pair presented by a User.

Definition 8 (Blindness[13]). *Let Π_{BS} be a blind signature scheme tuple and let Adversary $\mathcal{A}$ be a probabilistic polynomial-time algorithm. $\mathcal{A}$ participates in the following game, where $\mathcal{A}$ takes the role of the Signer:*

1. *a bit $b \leftarrow\!\$\ \{0, 1\}$ is randomly selected,*
2. *$\mathcal{A}$ generates a pair of signing keys* $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathcal{A}_{\mathsf{Gen}}(1^\lambda)$,
3. *$\mathcal{A}$ produces two unique messages $(m_0, m_1) \leftarrow \mathcal{A}$,*
4. *$\mathcal{A}$ runs two parallel signing sessions adaptively and in an arbitrary way (sessions can be started and terminated in any order) with two copies of $\mathcal{D}$, the first session being $(\bot, \sigma_b) \leftarrow \langle \mathcal{A}_{\mathsf{SignS}}(\mathsf{sk}), \mathsf{SignU}(\mathsf{pk}, \mathsf{m}_b)\rangle$ and the second one $(\bot, \sigma_{b-1}) \leftarrow \langle \mathcal{A}_{\mathsf{SignS}}(\mathsf{sk}), \mathsf{SignU}(\mathsf{pk}, \mathsf{m}_{b-1})\rangle$.*
5. *$\mathcal{A}$ is given (σ_0, σ_1) if both sessions terminated correctly (i.e. both $\sigma_0 \neq \bot$ and $\sigma_1 \neq \bot$) and $(\bot, \bot)$ otherwise,*
6. *$\mathcal{A}$ outputs a bit b' and* ***wins*** *if $b' = b$.*

Π_{BS} has the ***blindness*** *property if for any probabilistic polynomial adversary $\mathcal{A}$, the probability of $\mathcal{A}$ winning the game is at most $\frac{1}{2} + \varepsilon$, where ε is negligibly small.*

Unlinkability of Privacy Pass. Here we propose a formalization for the unlinkability property of Privacy Pass.

Definition 9 (Unlinkability). *The Signing and Redemption phases of Π_{PP} are* ***unlinkable****, if each element in the view of the Issuer in the Signing phase is indistinguishable from a uniformly sampled element from its corresponding space (e.g. $\mathbb{G}$ or $\mathbb{Z}_N$) and independent of the view of the Issuer in the Redemption phase.*

Unlinkability can be formalized using the following game. Let $\mathcal{A}$ be a PPT algorithm, who takes the role of the Issuer:

1. *a bit $b \leftarrow\!\$\ \{0, 1\}$ is randomly selected,*
2. *a pair of keys is generated* $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{Gen}(1^\lambda)$,
3. *$\mathcal{A}$ is given* $(\mathsf{sk}, \mathsf{pk})$,
4. *$\mathcal{A}$ runs two parallel Signing sessions adaptively and in an arbitrary way (they can be started and terminated in any order) with two copies of $\mathcal{D}$: $(\bot, (\mathbf{t}^i, \boldsymbol{\tau}^i)) \leftarrow \langle \mathcal{A}_{\mathsf{IssueS}}(\mathsf{sk}), \mathsf{IssueU}(\mathsf{pk})\rangle$ for $i = 0, 1$,*
5. *if both sessions terminated correctly from the point of view of the User (no execution returns $\bot$ to $\mathcal{D}$), $\mathcal{A}_{\mathsf{Redeem}}(\mathsf{sk}$ **or** $\mathsf{pk}, \mathrm{t}_j^b, \tau_j^b)$ is called for $j \in [k]$,*
6. *$\mathcal{A}$ outputs a bit b' and* ***wins*** *if $b' = b$.*

We say that Π_{PP} is ***unlinkable*** *if the probability of $\mathcal{A}$ winning the game is at most $\frac{1}{2} + \varepsilon$, where ε is negligibly small.*

B Instantiations of the Encoding Functions Indistinguishable from Random

Here we present practical implementations of the encode and decode algorithms.

Let us fix $\langle g\rangle \subseteq \mathbb{Z}_p$ for a prime p (e.g. $p = 2 \cdot q + 1$). Then, encode(Y) for a $Y \in \langle g\rangle$ can be implemented using the probabilistic bias removal method (PBRM) [47], where encode and decode are defined on Figure 13. For this construction, $\ell = \lceil \log_2 p\rceil \approx \lceil \log_2 q\rceil = l_g(\lambda)$.

Now, if $\langle g\rangle \subseteq E[\mathbb{F}_p]$, i.e. if we are working on points of elliptic curve defined over a finite field, we suggest using the Elligator 2 mapping [13], for which $\ell = \lfloor \log_2 p\rfloor \approx \lfloor \log_2 q\rfloor \leq l_g(\lambda)$ [13, Theorem 8].

C Discussion on Prime Generation According to FIPS 186-5

As we mentioned in Section 5.1, we need to discuss prime generation as defined in [40].

The FIPS standard provides a number of variants for Gen that produce n-bit numbers in different primality classes, e.g. *probable prime* numbers or *provable prime* numbers. We focus only on the variant that produces *probable prime* numbers since the *provable primes* class as specified in FIPS 186-5 requires that the numbers are

[11] The issuing party is usually named Signer in blind signatures-related literature, but for consistency with Privacy Pass we refer to them as Issuer.

[12] Thus, $\mathcal{D}$ is *in fact* the user party as defined by standard blind signature schemes and $\mathcal{U}$ is more akin to an external observer.

[13] We define the dishonest-key blind variant [6].

encode(x)	decode(x')
1: $b \leftarrow\$ \{0,1\}$	1: **if** $x' \le p$:
2: **if** $b = 1$:	2: **return** x'
3: $x' = x$	3: **else** :
4: **else if** $x \le 2^\ell - p$:	4: **return** $2^\ell - x'$
5: $x' = 2^\ell - x$	
6: **else if** $x > 2^\ell - p$:	
7: **goto** 1	
8: **return** x'	

Figure 13: Probabilistic bias removal method.

always generated from deterministically *seeded* PRNGs (the generation function for the *probable primes* does not mention explicit seeding of PRNG and we assume that it means that the only requirement is sufficient entropy). This is a problem for our construction, since in our modified algorithm, we cannot find the preimage seed of q. This means that if $\mathcal{A}$ requested not only sk, but also the seeds used to generate p, q, they would detect the attack. Fortunately (or unfortunately), as it turns out, FIPS 186-5 does not require storing the seeds even if provable primes are requested: "*Prime number generation seeds shall be kept secret or destroyed when the modulus N is computed*". In the case where provable primes are enforced and $\mathcal{A}$ is always able to request the seeds (for example if implementations without the ability to disclose the seeds cannot be certified), we simply state that $\mathsf{pk_d}$ needs to be communicated to $\mathcal{D}$ in some other way (e.g., embedded in the device during manufacturing), and then our constructions can only work in the Fully-Asymmetric Anamorphic model.

D Informal Construction for Transmitting $\mathsf{pk_d}$ in Issuance Protocol for Privacy Pass in Privately Verifiable Mode

In this section, we briefly describe a construction that could be used to transfer $\mathsf{pk_d} = g^{\mathsf{sk_d}}$ ($\mathsf{sk_d} \in \mathbb{Z}_q^*$) from $\mathcal{I}$ to $\mathcal{D}$ inside the Issuance Protocol for Privacy Pass in Privately Verifiable Mode in a way that is secure against the Auditor $\mathcal{A}$, over n signing sessions.

Our construction makes use of the rejection sampling technique. First notice that the values s, c from Figure 7 are probabilistic — both are dependent on randomly sampled $a \leftarrow\$ \mathbb{Z}_q^*$. We can then use one of them with rejection sampling to transmit b bits of $\mathsf{pk_d}$.

Instead of steps 9-11. of Figure 7, for each session $i \in [n]$, $\mathcal{I}$ encodes a b-bit slice of $\mathsf{pk_d}$:

9. $a \leftarrow\$ \mathbb{Z}_q^*$
10. $A := g^a, B := M^a$
11. $c := \mathsf{H}_q(\mathsf{pk}, M, Z, A, B)$
12. $u_i := \mathsf{encode}(\mathsf{pk_d})_{\{(i-1)\cdot b,\ldots,(i)\cdot b\}}$
13. **if** b bit prefix of $\mathsf{H}(c) \neq u_i$: **goto** 10

Since H, H_q behave like random functions, we should find a proper c in roughly 2^b tries (b has to be appropriately small). $\mathcal{D}$, on receiving c, recovers the b-bit slice of $\mathsf{pk_d}$:

$$u_i := \mathsf{H}(c)_{\{0,\ldots,b-1\}},$$

which is equal to $\mathsf{encode}(\mathsf{pk_d})_{\{(i-1)\cdot b,\ldots,(i)\cdot b-1\}}$.

After $n = \lceil |\mathsf{encode}(\cdot)|/b \rceil$ sessions, $\mathcal{D}$ can reconstruct the entire key:

$$\mathsf{pk_d} := \mathsf{decode}(u_1 \| u_2 \| \ldots \| u_n)$$

For example, if $b = 5$ and $|\mathsf{encode}(\cdot)| = 256$, $\mathsf{pk_d}$ can be transmitted in $n = \lceil 256/5 \rceil = 52$ signing sessions, and each signing session requires $\sim 2^5 = 32$ tries on $\mathcal{I}$'s side. Since encode output distribution is close to uniformly random on ℓ-bit strings and $\mathsf{pk_d}$ is just a random element of $\langle g \rangle$, this construction is indistinguishable from the original protocol except a negligible observer advantage.

Even though the presented construction has severe practical limitations, note that $\mathsf{pk_d}$ needs to be transmitted only once. Rejection sampling might then be a viable strategy of transmitting $\mathsf{pk_d}$, as tokens need to be periodically replenished by $\mathcal{D}$ in the standard use-case.

E Proofs of the Theorems stated in the Work

In this section, we present proofs and proof sketches for the theorems stated throughout the work.

E.1 Proof of Theorem 1

Lemma E.1 (Correctness of anamorphic triplet). *The triplet* $\Sigma_1 = (\mathsf{aGen_{Id}}, \langle \mathsf{aIssueS}, \mathsf{aIssueU} \rangle, \mathsf{aDec})$ *with the identity keygen function and the algorithms defined in Figure 5 and Figure 6 is a correct Fully-Asymmetric Anamorphic Triplet.*

Proof. First, recall that the scheme is defined for $k = 1$ and thus, without loss of generality, $\boldsymbol{\tau} = (\tau)$ and $\mathbf{t} = (\mathrm{t})$. We can then conduct the analysis for a single token/signature pair. Notice that for any key tuple $(\mathsf{sk}, \mathsf{pk}, \mathsf{sk_d} = x, \mathsf{pk_d} = g^x)$ generated by $\mathsf{aGen_{Id}}$ and a message $\mathsf{am} \in \widehat{\mathcal{M}}$, after executing

$$(\gamma, (\mathbf{t}, \boldsymbol{\tau})) \leftarrow \langle \mathsf{aIssueS}(\mathsf{sk}), \mathsf{aIssueU}(\mathsf{pk}, \mathsf{am}, \mathsf{pk_d}) \rangle,$$

we receive $\gamma := (P, d)$, $\boldsymbol{\tau} = (s')$, $\mathbf{t} = (\mathrm{t})$. Then

(1) $P := h \cdot r^e \bmod N$,
(2) $s' := s \cdot r^{-1} = P^d \cdot r^{-1} = h^d \cdot r^{e\cdot d} \cdot r^{-1} = h^d \bmod N$.

So, if we execute the first steps from aDec, i.e. $h := (s')^e \bmod N$ and $R := P/h \bmod N$, we recover the same $r := R^d \bmod N = pad \| u \| v$.

Note that we require that $\gcd(r, N) = 1$ and $r < N$. However, it is easy to fulfil this condition since u, v and pad can be re-randomized after each failed attempt, and we constrain the sizes so that pad is always at least one bit long. aDec can decode the corresponding parts of r and recover the following values:

(1) $\mathsf{decode}(r_{\{0,\ldots,\ell-1\}})$ that equals $\mathsf{decode}(\mathsf{encode}(C)) = C$,
(2) $\mathsf{decode}(r_{\{\ell,\ldots,2\cdot\ell-1\}})$ that equals $\mathsf{decode}(\mathsf{encode}(Y)) = Y$,
(3) $\mathsf{decode}(r_{\{2\cdot\ell,\ldots,3\cdot\ell-1\}})$ that equals $\mathsf{decode}(\mathsf{encode}(A)) = A$,
(4) $\mathsf{decode}(r_{\{3\cdot\ell,\ldots,4\cdot\ell-1\}})$ that equals $\mathsf{decode}(\mathsf{encode}((\mathsf{pk_d})^a)) = (\mathsf{pk_d})^a$.

As $(\mathsf{pk_d})^a = A^{\mathsf{sk_d}}$, the test from line 10 yields the positive result.

In step 7, the ElGamal shared secret is computed $K := Y^{\mathsf{sk_d}} = {\mathsf{pk_d}}^y$ and because $C = \mathsf{am} \cdot K$, aDec outputs the original am. □

Lemma E.2 (Indistinguishability). Π_1 *is a Fully-Asymmetric Anamorphic Privacy Pass protocol with Triplet* Σ_1.

Proof. Consider the Indistinguishability game from Definition 3 with respect to Π_1 with Σ_1.

Through a sequence of games, we show that the anamorphic protocol execution (i.e., when $b = 0$ is selected) is indistinguishable from the regular protocol execution (i.e., when $b = 1$ is selected). Consequently, their outputs and transcripts must also be indistinguishable. In the games below, we only make changes to aIssueU.

G_0: This game is identical to the Indistinguishability game when $b = 0$ is selected and thus first the anamorphic keys are sampled $(\mathsf{sk}, \mathsf{pk}, \mathsf{sk}_d, \mathsf{pk}_d) \leftarrow\$ \mathsf{aGen}(1^\lambda)$. The keys $(\mathsf{sk}, \mathsf{pk})$ are then given to $\mathcal{A}$, and $\mathcal{A}$ can query the oracle for outputs and transcripts of the anamorphic execution

$$(\gamma, (\mathbf{t}, \boldsymbol{\tau})); \mathcal{T} \leftarrow \langle \mathsf{aIssueS}(\mathsf{sk}), \mathsf{aIssueU}(\mathsf{pk}, \mathsf{am}, \mathsf{pk}_d)\rangle$$

for adversarially selected am.

G_1: It is the same as G_0, but instead of setting $u := \mathsf{encode}\big((\mathsf{pk}_d)^a\big)\|\mathsf{encode}(A)$ we sample $B \leftarrow\$ \langle g\rangle$ and set $u := \mathsf{encode}\big(B\big)\|\mathsf{encode}(A)$. If $\mathcal{A}$ can distinguish between G_1 and G_0, then they can break DDH problem with roughly the same advantage, contrary to the assumption about $\langle g\rangle$. Namely, consider a DDH challenge tuple (g^{x_1}, g^{x_2}, g^c) (where $x_1, x_2, c \neq 0$). We can set $\mathsf{pk}_d := g^{x_1}$ and in the oracle queries set $A := (g^{x_2})^{a'}$ for $a' \leftarrow\$ \mathbb{Z}_q^*$ (note that in the reduction $a = a' \cdot x_2$) and $u := \mathsf{encode}\big((g^c)^{a'}\big)\|\mathsf{encode}(A)$. If $c = x_1 \cdot x_2$, then the outcome of the reduction simulates G_0. Indeed, for G_0, $(g^c)^{a'} = (g^{x_1 \cdot x_2})^{a'} = (\mathsf{pk}_d)^{x_2 \cdot a'} = (\mathsf{pk}_d)^a$. On the other hand, it may also correspond to execution of G_1, where B has been chosen as $(g^c)^{a'}$, but this happens with a negligible probability. If $c \neq x_1 \cdot x_2$, then the outcome of the reduction corresponds only to G_1. So, we may conclude that $\mathcal{A}$'s advantage is at most the advantage in breaking DDH plus a negligible value.

G_2: It is the same as G_1, but instead of sampling $B \leftarrow\$ \langle g\rangle$ and setting $u := \mathsf{encode}\big(B\big)\|\ \mathsf{encode}(A)$, we sample $u \leftarrow\$ \{0,1\}^{2\cdot\ell}$ and skip sampling $a \leftarrow\$ \mathbb{Z}_q^*$, $B \leftarrow\$ \langle g\rangle$, and calculating $A := g^a$ altogether. Since a is uniformly sampled, A is indistinguishable from a uniformly sampled group element. B is also randomly sampled, and because of the assumption that encode returns ℓ-bit strings indistinguishable from uniformly random if given uniformly random group elements as input, the encoded group elements $\mathsf{encode}(A)$ and $\mathsf{encode}(B)$ are indistinguishable from uniformly sampled ℓ-bit strings. Hence, the advantage of $\mathcal{A}$ is negligible and bounded by their advantage in distinguishing encoded group elements from random.

G_3: It is the same as G_2, but instead of calculating $K := (\mathsf{pk}_d)^y$ we sample $K \leftarrow\$ \langle g\rangle$. If $\mathcal{A}$ could distinguish between G_2 and G_3 with non-negligible advantage, then $\mathcal{A}$ can be used to break DDH: Consider a DDH challenge tuple (g^{x_1}, g^{x_2}, g^c) (where $x_1, x_2, c \neq 0$). We can set $\mathsf{pk}_d := g^{x_1}$ and in the oracle queries set $Y := (g^{x_2})^{y'}$ for $y' \leftarrow\$ \mathbb{Z}_q^*$ (note that in the reduction $y = y' \cdot x_2$) and $K := (g^c)^{y'}$. If $c = x_1 \cdot x_2$, then the outcome of the reduction simulates G_2. Indeed, for G_2, $(g^c)^{y'} = (g^{x_1 \cdot x_2})^{y'} = (\mathsf{pk}_d)^{x_2 \cdot y'} = (\mathsf{pk}_d)^y$. On the other hand, it may also correspond to execution of G_3, where B has been chosen as $(g^c)^{y'}$, but this happens with a negligible probability. If $c \neq x_1 \cdot x_2$, then the outcome of the reduction corresponds only to G_3. So, we may conclude that $\mathcal{A}$'s advantage is at most the advantage in breaking DDH plus a negligible value.

G_4: It is the same as G_3, but instead of setting $C := \mathsf{am} \cdot K$ we sample $C \leftarrow\$ \langle g\rangle$. Since $K \leftarrow\$ \langle g\rangle$ is randomly sampled, and $\mathsf{am} \in \langle g\rangle$, it follows that $\mathsf{am} \cdot K$ is indistinguishable from a randomly sampled group element. So the games G_3 and G_4 are indistinguishable.

G_5: It is the same as G_4, but instead of sampling $C \leftarrow\$ \langle g\rangle$ and setting $v := \mathsf{encode}(Y)\|\mathsf{encode}(C)$, we sample $v \leftarrow\$ \{0,1\}^{2\cdot\ell}$ and skip sampling $y \leftarrow\$ \mathbb{Z}_q^*$, $K \leftarrow\$ \langle g\rangle$, and calculating $Y := g^y$ altogether. Since C and y (and by extension, Y) are randomly sampled, and we assumed that encode returns ℓ-bit strings indistinguishable from uniformly random if given uniformly random group elements as input, the advantage in distinguishing between G_4 and G_5 is negligible.

G_6: It is the same as G_5, but instead of setting $r := pad\|u\|v$ we sample $r \leftarrow\$ \{0,1\}^n$ and skip sampling $pad \leftarrow\$ \{0,1\}^{n-4\cdot\ell}$, $u \leftarrow\$ \{0,1\}^{2\cdot\ell}$ and $v \leftarrow\$ \{0,1\}^{2\cdot\ell}$. The games G_5 and G_6 are obviously indistinguishable, since the distribution of r is exactly the same from the point of view of $\mathcal{A}$.

G_7: It is the same as G_6, but instead of sampling $r \leftarrow\$ \{0,1\}^n$ and resampling if $r \geq N$, or $\gcd(r, N) \neq 1$, we sample $r \leftarrow\$ \mathbb{Z}_N$ such that $\gcd(r, N) = 1$. Note that n is the bit-length of N. The constraint $\gcd(r, N) = 1$ is the same for both games, so we only need to compare $r \leftarrow\$ \{0,1\}^n$ and $r \leftarrow\$ \mathbb{Z}_N$. Note that in G_6 we reject all samples $r \geq N$, and so, because for all other samples the distribution is uniform, we are really just sampling $r \leftarrow\$ [N]$ which is the same as $r \leftarrow\$ \mathbb{Z}_N$, which means that the two games are actually identical.

G_8: It is the same as G_7, but instead of sampling anamorphic keys, we sample regular keys $(\mathsf{sk}, \mathsf{pk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$. Since $(\mathsf{sk}, \mathsf{pk})$ come from the same distribution and $(\mathsf{sk}_d, \mathsf{pk}_d)$ are not used, G_7 and G_8 are indistinguishable.

The last game, G_8, is identical to the Indistinguishability game when $b = 1$ is selected. As in the sequence of games the advantage of $\mathcal{A}$ is negligible in each step, the advantage in the Indistinguishability game is negligible as well. □

Lemma E.3 (Robustness). *Π_1 is a Robust Fully-Asymmetric Anamorphic Privacy Pass protocol with Triplet Σ_1.*

Proof. Consider the Robustness game specified in Definition 5 with respect to Π_1 with Σ_1.

Through a sequence of games, we will show that the oracle outputting am from a mismatched anamorphic protocol execution (when $b = 0$ is selected) is indistinguishable from the oracle that returns $\perp$ (when $b = 1$ is selected).

G_0: This game is identical to the Robustness game with $b = 0$. Thus, the anamorphic keys are sampled $(\mathsf{sk}, \mathsf{pk}, \mathsf{sk}_d, \mathsf{pk}_d) \leftarrow\$ \mathsf{aGen}(1^\lambda)$, and distinguisher $\mathcal{D}$ can query the robustness oracle for mismatched decryption coming from the execution of the anamorphic protocol for adversarially selected am_1 and am_2.

G_1: It is the same as G_0, but, in aIssueU and IssueU, we replace the mapping $\mathsf{encode}_{\mathsf{PSS}}(-, n)$ (for computing h) by a pseudorandom function that maps into $\{0,1\}^{n-1}$. This is justified by

the construction of $\text{encode}_{\text{PSS}}$, where the output is composed of a hash value and mask generation function. Thus, we can assume that the advantage in distinguishing between $\mathbf{G}_1$ and $\mathbf{G}_0$ is negligibly small.

$\mathbf{G}_2$: It is the same as $\mathbf{G}_1$, but, in aDec, instead of calculating $R := P \cdot h^{-1} \bmod N$ we sample $R \leftarrow\$ \mathbb{Z}_N^*$. Since $P \in \hat{\gamma}$ can either come from IssueU or from aIssueU and $h = (s')^e \bmod N = (\tau)^e \bmod N$ can come from either the same session as P or from a different one (and also either from IssueU or aIssueU), depending on $\mathcal{D}$'s query inputs, we split the cases:

(1) Case: $\mathcal{D}$ *queried on* single = **true**, *so* $P \in \hat{\gamma}$ *comes from* IssueU *and* $h := (\tau)^e \bmod N$ *comes from the same session as* P:
then $P := h \cdot (r)^e$, $R := P \cdot h^{-1} \bmod N = (r)^e \bmod N$. Note that, in IssueU, $r \leftarrow\$ \mathbb{Z}_N^*$ is uniformly random, and then so must be $R = (r)^e \bmod N$.

(2) Case: $\mathcal{D}$ *queried on* single ≠ **true** *and* $\text{am}_2 = \perp$, *so* $P \in \hat{\gamma}$ *comes from* IssueU *and* $h = (\tau)^e \bmod N$ *comes from a session different than* P:
then $P := h' \cdot (r')^e$, and so $R := h' \cdot (r')^e \cdot h^{-1} \bmod N$. Since h and h' have been created by a pseudorandom function for different arguments (with overwhelming probability, since for both sessions $\text{t} \leftarrow\$ \{0,1\}^{256}$), R is indistinguishable from a uniformly chosen random element. Note that it does not matter if h comes from aIssueU or IssueU, since in both h is computed in the same way.

(3) Case: $\mathcal{D}$ *queried on* single ≠ **true** *and* $\text{am}_2 \neq \perp$, *so* $P \in \hat{\gamma}$ *comes from* aIssueU *and* $h = (\tau_1)^e \bmod N$ *comes from a different session than* P:
then the situation is exactly the same as above, since $P := h' \cdot (r')^e$ and h' is computed in exactly the same way as in IssueU. For the same reason as above, R is indistinguishable from a uniformly chosen random element (and it does not matter if h comes from aIssueU or IssueU).

In all cases, R is indistinguishable from a uniformly sampled element from $\mathbb{Z}_N^*$ and thus $\mathbf{G}_2$ and $\mathbf{G}_1$ are indistinguishable except a negligible distinguisher advantage.

$\mathbf{G}_3$: In this game, aDec always returns $\perp$. The games $\mathbf{G}_3$ and $\mathbf{G}_2$ are indistinguishable for the sheer reason that in $\mathbf{G}_2$, the procedure aDec returns the result different from $\perp$ only if $r = R^d \bmod N$ has the proper form. Recall that due to randomness of R, the element r is a random element of $\mathbb{Z}_N^*$ as well. Thus, the test from line 10 in Figure 6 will fail except for an event of a negligible probability. Indeed, it is very unlikely that the element $\text{decode}(r_{\{3\cdot\ell,\ldots,4\cdot\ell-1\}})$ equals $\text{decode}(r_{\{2\cdot\ell,\ldots,3\cdot\ell-1\}})^{\text{sk}_\text{d}}$ for a random r.

The last game, behaves as the game from the Robustness game when $b = 1$ is selected. Thus, we can conclude that the advantage of $\mathcal{D}$ is negligible. □

Theorem 1 follows directly from Theorems E.1 to E.3.

E.2 Proof of Theorem 2

Lemma E.4 (Correctness of anamorphic triplet). *The triplet* $\Sigma_2 = (\text{aGen}, \langle \text{aIssueS}, \text{aIssueU}\rangle, \text{aDec})$ *with algorithms defined in Figure 4, Figure 5 and Figure 6 is a correct Fully-Asymmetric Anamorphic Triplet.*

Proof. The proof is almost the same as for Lemma E.1, with the difference that the keys are now generated by aGen (but still $\text{sk}_\text{d} = x$ and $\text{pk}_\text{d} = g^x$) and in the first step of aIssueU, we recover pk_d with the $\text{Recover}(N)$ algorithm (compare Figure 5, step 1). So it is sufficient to show that $\text{pk}_\text{d} \leftarrow \text{Recover}(N)$ is correct. It is indeed always correct, due to the constraint set in step 9. There should always be a valid solution for q achievable in a reasonable time due to the density of prime numbers and because $\ell \approx \lceil \log_2 q \rceil \ll n$ (e.g., $\ell = 256$ bits while $n = 2048$ bits) and so the fixed part of $\widehat{N}$ is relatively small compared to the random part. Additionally, we set $N = \widehat{N} - r$ and r is at most $n/2$ bits long (since $r < p$), so it will rarely affect the highest ℓ bits of N (there would have to be a chain of $n/2 - \ell + 1$ carry bits). Since q can always be found, the algorithm will not stall except for a negligible probability. □

Lemma E.5 (Indistinguishability). Π_1 *is a Fully-Asymmetric Anamorphic Privacy Pass protocol with Triplet* Σ_2.

Proof. Proof is almost identical to the proof of Lemma E.2. The only difference is in the game $\mathbf{G}_8$:

$\mathbf{G}_8$: It is the same as $\mathbf{G}_7$, but instead of sampling anamorphic keys, we sample regular keys $(\text{sk} = d, \text{pk} = (N, e)) \leftarrow\$ \text{Gen}(1^\lambda)$. Since $\mathcal{A}$ knows the factorization of N, we will show that p, q generated by $\text{aGen}(1^\lambda)$ are indistinguishable from p', q' generated by $\text{Gen}(1^\lambda)$. Indistinguishability of p and p' comes from the construction: they are generated in exactly the same way. The second prime, q, is sampled by taking a bit string $\widehat{N}$ indistinguishable from a random string of length n (because both t and u are indistinguishable from random strings), treating it as a number, then dividing (with integer division) by a random prime p of bit length $n/2$. By [47, Lemma 5.], we know that q calculated this way is random on the set of $n/2$-bit primes, and so is indistinguishable from q'. The only difference is then the additional constraint in step 9, which might change the distribution of q compared to q', but we argue that this change is negligible because as stated in the proof for Theorem E.4, it will very rarely be the case that r affects the upper ℓ-bits of N, which is required for the constraint to take effect. Since every other value is generated as in Gen, we conclude that the two games are indistinguishable.

The last game, $\mathbf{G}_8$, is identical to the Indistinguishability game when $b = 1$ is selected and thus the advantage of $\mathcal{A}$ is bounded by a negligible factor.

□

Lemma E.6 (Robustness). Π_1 *is a Robust Fully-Asymmetric Anamorphic Privacy Pass protocol with Triplet* Σ_2.

Proof. The proof is exactly the same as for Theorem E.3 (with respect to generating a different key tuple lacking pk_d). □

The proof of Theorem 2 comes directly from Theorems E.4 to E.6 and from the fact that aGen returns a void key $\text{pk}_\text{d} = \perp$.

E.3 Proof of Theorem 3

Proof. Consider the Unlinkability game specified as in Definition 9 with respect to a robust, anamorphic instantiation of Π

Issuance session algorithm $\perp \leftarrow \mathcal{A}_{\mathsf{IssueS}}(\mathsf{sk})$

1: Run session 0 $(\gamma_0, \cdot) \leftarrow \langle \mathsf{aIssueS}(\mathsf{sk}), \mathsf{aIssueU}(\cdot)\rangle$
2: Run session 1 $(\gamma_1, \cdot) \leftarrow \langle \mathsf{aIssueS}(\mathsf{sk}), \mathsf{aIssueU}(\cdot)\rangle$
3: Store $\mathsf{state}_0 := (\gamma_0, \gamma_1)$
4: Set $\mathsf{state}_1 := \perp$
5: **return** $\perp$

Redeeming algorithm $\mathcal{A}_{\mathsf{Redeem}}(\mathsf{sk}$ **or** $\mathsf{pk}, \mathsf{t}_j^b, \tau_j^b)$

1: Store $\mathsf{state}_1 \leftarrow \mathsf{state}_1 \cup (\mathsf{t}_j^b, \tau_j^b)$

Finalization algorithm $b \leftarrow \mathcal{A}$

1: Recover $(\gamma_0, \gamma_1) \leftarrow \mathsf{state}_0$
2: Recover $(\mathsf{t}_j^b, \tau_j^b) \leftarrow \mathsf{state}_1, j \leftarrow\!\!\$\ [k]$
3: $\mathsf{am} \leftarrow \mathsf{aDec}(\mathsf{sk}_\mathsf{d}, \gamma_0, \mathsf{t}_j^b, \tau_j^b)$
4: **if** $\mathsf{am} \neq \perp$: **return** $b' := 0$
5: **else** : **return** $b' := 1$

Figure 14: Adversaries' algorithms for breaking unlinkability.

with a fully-asymmetric anamorphic triplet Σ. Since a public key anamorphic triplet is also a fully-asymmetric anamorphic triplet, in this way we also cover the case of public key anamorphic triplets.

First, clearly if Π with Σ is anamorphic, then by Definition 3 we have that it is indistinguishable from a regular execution of Π for the User, who is just an external observer.

Second, we can break unlinkability with the robustness property. Consider the unlinkability game in Definition 9 and an adversary $\mathcal{A}$ who is an anamorphic Receiver (and so they know sk_d). Since $\mathcal{D}$ is running in anamorphic mode (encrypting some anamorphic messages, say $\mathsf{am}_0, \mathsf{am}_1 \in \widehat{\mathcal{M}}$.), in step (4) of the game $\mathcal{A}$ runs two parallel signing sessions for $i = 0, 1$ with $\mathcal{D}$ executing aIssueU:

$$(\perp, (\mathbf{t}^i, \boldsymbol{\tau}^i)) \leftarrow \langle \mathcal{A}_{\mathsf{IssueU}}(\mathsf{sk}), \mathsf{aIssueU}(\mathsf{pk}, \mathsf{am}_i, \mathsf{sk}_\mathsf{d})\rangle.$$

We then consider the following algorithms run by the stateful Adversary, presented in Figure 14, used in the Privacy Pass unlinkability game (Definition 9).

Note that $\mathcal{A}$ wins the unlinkability game with non-negligible probability. Indeed, $\mathcal{A}$ is running $\mathsf{am} \leftarrow \mathsf{aDec}(\mathsf{sk}_\mathsf{d}, \gamma_0, \mathsf{t}_j^b, \tau_j^b)$. By correctness of the anamorphic tuple Σ (Definition 2), if $b = 0$, then $\mathsf{am} = \mathsf{am}_0 \in \widehat{\mathcal{M}}$ and, more importantly, $\mathsf{am} \neq \perp$. In the other case, i.e., when $b = 1$, then γ_0 and t_j^1, τ_j^1 come from different sessions. Note that the Robustness game (Definition 5) requires that for $\mathsf{am} \leftarrow \mathsf{aDec}(\mathsf{sk}_\mathsf{d}, \gamma_0, \mathsf{t}_j^1, \tau_j^1)$ with a uniformly sampled $j \leftarrow\!\!\$\ [k]$, when γ_0 and $(\mathsf{t}_j^1, \tau_j^1)$ come from different sessions and Issuer runs aIssueS, am is indistinguishable from $\perp$. In other words, if $b = 1$, the algorithm of the adversary $\mathcal{A}$ perfectly simulates a Robustness Oracle query with single $\neq$ **true** and $\mathsf{am}_1, \mathsf{am}_2 \neq \perp$, and therefore the resulting am must be indistinguishable with $\perp$ for a robust scheme. So, if $b = 1$, then $\mathsf{am} = \perp$ with overwhelming probability, which concludes the proof. □

E.4 Proof of Theorem 4

Proof. For the proof, we want to find a blinding factor r' that couples both executions. Simply, let $r' = (P_1/P_2)^d \bmod N$. We have to check that $s_2 = s_1 \cdot (r')^{-1}$. Recall that, due to the correctness of RSA blind signatures

$$(s_1)^e = P_1 \bmod N,\ (s_2)^e = P_2 \bmod N\ .$$

Then, we get the following equivalent equalities:

$$\begin{aligned} s_2 \bmod N &\stackrel{?}{=} s_1 \cdot (r')^{-1} \bmod N \\ (s_2)^e \bmod N &\stackrel{?}{=} (s_1)^e \cdot ((r')^{-1})^e \bmod N \\ P_2 \bmod N &\stackrel{?}{=} P_1 \cdot (((P_1/P_2)^d)^{-1})^e \bmod N \\ P_2 \bmod N &\stackrel{?}{=} P_1 \cdot (P_2/P_1) \bmod N \end{aligned}$$

As the last equality is true, so is the first one, and both points of view can originate from the same computation. □

E.5 Proof of Theorem 5

Lemma E.7 (Correctness of anamorphic triplet). *The triplet $\Sigma_3 = (\mathsf{aGen}, \langle \mathsf{aIssueS}, \mathsf{aIssueU}\rangle, \mathsf{aDec})$ with algorithms defined in Figure 8, Figure 9 and Figure 10 is a correct Fully-Asymmetric Anamorphic Triplet.*

Proof. Recall that for $(\mathsf{sk}, \mathsf{pk}, \mathsf{sk}_\mathsf{d} = (x, \mathbf{T}), \mathsf{pk}_\mathsf{d} = g^x)$ generated by aGen and any message $\mathsf{am} \in \widehat{\mathcal{M}}$, if

$$(\gamma, (\mathbf{t}, \boldsymbol{\tau})) \leftarrow \langle \mathsf{aIssueS}(\mathsf{sk}), \mathsf{aIssueU}(\mathsf{pk}, \mathsf{am}, \mathsf{pk}_\mathsf{d})\rangle,$$

then $\gamma := [P_k]$, $\boldsymbol{\tau} := [W_k]$, $\mathbf{t} := [\mathsf{t}_k]$.

Correctness requires that aDec is successful in recovering am for all $j \in [k]$, but let us fix t_j, W_j for a moment. Let us go through the aDec algorithm. First, observe that $T := \mathsf{H}_{\langle g\rangle}(\mathsf{t}_j, \mathsf{chall}, \mathsf{pk}) = T_j$ since both are generated in the same way. Then, note that $\mathsf{t}_j := \mathsf{encode}(Y_j)$ and so $Y := \mathsf{decode}(\mathsf{t}_j) = Y_j$ due to correctness of decode. Then, $r := \mathsf{H}_q(Y^x) = \mathsf{H}_q((g^x)^{y_j}) = r_j$.

Then, we iterate through $i \in [k]$. For $j \neq i$, we have $P_j = T_j^{r_j} \cdot g^{\mathsf{am}}$ and so $C_j = P_j \cdot T^{-r} = P_j \cdot T_i^{-r_i} = T_j^{r_j} \cdot g^{\mathsf{am}} \cdot T_i^{-r_i}$. Since T_j, T_i, r_i, r_j are all independent and indistinguishable from uniformly random strings (all come from hash functions modeled as random oracles with fresh random inputs), we find that C_j is indistinguishable from a uniformly sampled group element. The probability of a uniformly sampled C_j being defined in the lookup table, i.e., $\mathbf{T}[C_j] \neq \perp$, is negligible, since the size of the preimage of $\mathbf{T}$ must be sufficiently small for $\mathbf{T}$ to be efficiently computed in practice. Finally, the probability $p = |\widehat{\mathcal{M}}|/q$ of finding a group element in the preimage of $\mathbf{T}$ by randomly sampling $\langle g\rangle$ is negligible. So, with overwhelming probability, for $j \neq i$, the constraint in step 7 is not satisfied, and the loop iterates further.

For $j = i$, since $P_i = P_j = T_j^{r_j} \cdot g^{\mathsf{am}} = T^r \cdot g^{\mathsf{am}}$, we get that $C_i = P_i \cdot T^{-r} = g^{\mathsf{am}}$. We then return a valid $\mathsf{am} \in \widehat{\mathcal{M}}$ since $\mathbf{T}[g^{\mathsf{am}}] = \mathsf{am}$.

Since correctness holds for an arbitrary fixed index $i \in [k]$, it follows that correctness holds for all $i \in [k]$.

As we see, for all token/tag pairs (t_j, τ_j) for $j \in [k]$, aDec returns a valid am with probability at least $1-\varepsilon$, where ε is negligible. This concludes the proof. □

Lemma E.8 (Indistinguishability). *Π_2 is a Fully-Asymmetric Anamorphic Privacy Pass protocol with Triplet Σ_3.*

Proof. Consider the Indistinguishability game as specified in Definition 3 with respect to Π_2 with Σ_3.

Through a sequence of games, we will show that the anamorphic protocol execution (i.e., when $b=0$ is selected) is indistinguishable from the regular protocol execution (i.e., when $b=1$ is selected). So, it must be that their output and transcripts are also indistinguishable (in the below games we only make changes to aIssueU).

$\mathbf{G}_0$: This game is identical to the Indistinguishability game when $b=0$ is selected and thus anamorphic keys are sampled $(\mathsf{sk}, \mathsf{pk}, \mathsf{sk}_\mathsf{d}, \mathsf{pk}_\mathsf{d}) \leftarrow\!\$\ \mathsf{aGen}(1^\lambda)$, $(\mathsf{sk}, \mathsf{pk})$ are given to $\mathcal{A}$, and $\mathcal{A}$ can query the oracle for outputs and transcripts of the anamorphic execution

$$(\gamma, (\mathbf{t}, \boldsymbol{\tau})); \mathcal{T} \leftarrow \langle \mathsf{aIssueS}(\mathsf{sk}), \mathsf{aIssueU}(\mathsf{pk}, \mathsf{am}, \mathsf{pk}_\mathsf{d}) \rangle$$

for adversarially selected am.

$\mathbf{G}_1$: It is the same as $\mathbf{G}_0$, but instead of setting $r_i := \mathsf{H}_q(X^{y_i})$ we sample $B \leftarrow\!\$\ \langle g \rangle$ and set $r_i := \mathsf{H}_q(B)$. If $\mathcal{A}$ is able to distinguish $\mathbf{G}_1$ and $\mathbf{G}_0$, then they can be used to break DDH problem (recall that we assumed that DDH is hard in $\langle g \rangle$). Consider a DDH challenge tuple (g^{x_1}, g^{x_2}, g^c) (where $x_1, x_2, c \neq 0$). We can set $\mathsf{pk}_\mathsf{d} := g^{x_1} = X$ and in the oracle queries set $Y_i := (g^{x_2})^{y_i'}$ for a randomly sampled $y_i' \leftarrow\!\$\ \mathbb{Z}_q^*$ and set $r_i := \mathsf{H}_q((g^c)^{y_i'})$. Note that in the reduction $y_i = y_i' \cdot x_2$. If $c = x_1 \cdot x_2$, then the outcome of the reduction simulates $\mathbf{G}_0$. Indeed, for $\mathbf{G}_0$, $(g^c)^{y_i'} = (g^{x_1})^{x_2 \cdot y_i'} = (X)^{x_2 \cdot y_i'} = X^{y_i}$. On the other hand, it may also correspond to execution of $\mathbf{G}_1$, where B has been chosen as $(g^c)^{y_i'}$, but this happens with a negligible probability. If $c \neq x_1 \cdot x_2$, then the outcome of the reduction corresponds only to $\mathbf{G}_1$. So, we may conclude that $\mathcal{A}$'s advantage is at most the advantage in breaking DDH plus a negligible value.

$\mathbf{G}_2$: It is the same as $\mathbf{G}_1$, but instead of sampling $y_i \leftarrow\!\$\ \mathbb{Z}_q^*$, calculating $Y_i = g^{y_i}$ and setting $\mathsf{t}_i := \mathsf{encode}(Y_i)$, we sample $\mathsf{t}_i \leftarrow\!\$\ \{0,1\}^{256}$. Note that $\ell = 256$. Since Y_i is uniformly distributed, and we assumed that encode returns ℓ-bit strings indistinguishable from uniformly random strings if given uniformly distributed group elements as input, the advantage of $\mathcal{A}$ can be at most their advantage in distinguishing the output of encode from uniformly random string, which is negligible.

$\mathbf{G}_3$: It is the same as $\mathbf{G}_2$, but instead of setting $r_i := \mathsf{H}_q(B)$, we sample $r_i \leftarrow\!\$\ \mathbb{Z}_q^*$. Recall that we assumed that H_q is a random oracle. Since hash input, B, is uniformly distributed, we conclude that $\mathbf{G}_3$ and $\mathbf{G}_2$ are indistinguishable.

$\mathbf{G}_4$: It is the same as $\mathbf{G}_3$, but instead of setting $P_i := T_i^{r_i} \cdot g^{\mathsf{am}}$ and later $W_i := \mathsf{H}((Q_i \cdot \mathsf{pk}^{-\mathsf{am}})^{1/r_i})$ we set $P_i := T_i^{r_i}$ and later $W_i := \mathsf{H}(Q_i^{1/r_i})$. Note that $W_i = \mathsf{H}(T_i^{\mathsf{sk}})$ in both cases, since $(Q_i \cdot \mathsf{pk}^{-\mathsf{am}})^{1/r_i} = \left((T_i^{r_i} \cdot g^{\mathsf{am}})^{\mathsf{sk}} \cdot g^{-\mathsf{sk}\cdot\mathsf{am}}\right)^{1/r_i} = T_i^{\mathsf{sk}}$. Since $T_i^{r_i}$ is indistinguishable from uniformly random (recall that $r_i \leftarrow\!\$\ \mathbb{Z}_q^*$), and $g^{\mathsf{am}} \in \langle g \rangle$, then $T_i^{r_i}$ and $T_i^{r_i} \cdot g^{\mathsf{am}}$ are actually indistinguishable, and so the two games are indistinguishable.

$\mathbf{G}_5$: It is the same as $\mathbf{G}_4$, but instead of sampling anamorphic keys, we sample regular keys $(\mathsf{sk}, \mathsf{pk}) \leftarrow\!\$\ \mathsf{Gen}(1^\lambda)$. Since $(\mathsf{sk}, \mathsf{pk})$ come from the same distribution and $(\mathsf{sk}_\mathsf{d}, \mathsf{pk}_\mathsf{d})$ are not used anywhere, the two games are indistinguishable.

The last game, $\mathbf{G}_5$, is identical to the Indistinguishability game when $b=1$ is selected and thus the advantage of $\mathcal{A}$ is bounded by a negligible factor.

□

Lemma E.9 (Robustness). *Π_2 is a Robust Fully-Asymmetric Anamorphic Privacy Pass protocol with Triplet Σ_3.*

Proof. Consider the Robustness game specified in Definition 5 with respect to Π_2 with Σ_3.

Through a sequence of games, we will show that the oracle outputting am from a mismatched anamorphic protocol execution (when $b=0$ is selected) is indistinguishable from the oracle that returns $\perp$ (when $b=1$ is selected).

$\mathbf{G}_0$: This game is identical to the Robustness game with $b=0$. Thus, the anamorphic keys are sampled $(\mathsf{sk}, \mathsf{pk}, \mathsf{sk}_\mathsf{d}, \mathsf{pk}_\mathsf{d}) \leftarrow\!\$\ \mathsf{aGen}(1^\lambda)$, and $\mathcal{D}$ can query the robustness oracle for mismatched decryption coming from the execution of the anamorphic protocol for adversarially selected am_1 and am_2.

$\mathbf{G}_1$: It is the same as $\mathbf{G}_0$, but in aDec, instead of setting $C_j := P_j \cdot T^{-r}$ we sample $C_j \leftarrow\!\$\ \langle g \rangle$. Since t_i can either come from IssueU or from aIssueU and $[P_k] = \hat{\gamma}$ can also come from IssueU or from aIssueU and from either the same session as t_i or a different one, depending on $\mathcal{D}$'s query inputs, we split the cases:

(1) Case *$\mathcal{D}$ queried on* $\mathsf{single} = \mathbf{true}$, *so* t_i *comes from* IssueU *and* $\hat{\gamma} = [P_k]$ *comes from the same session as* t_i:
since $\mathsf{t}_i \leftarrow\!\$\ \{0,1\}^{256}$ and $Y := \mathsf{decode}(\mathsf{t}_i)$, it must mean that Y is indistinguishable from uniformly random element due to assumed properties of decode, and so $r := \mathsf{H}_q(Y^{\mathsf{sk}})$ is indistinguishable from uniformly random scalar (the random oracle input is fresh with overwhelming probability). Then $C_j := P_j \cdot T^{-r}$ also must be indistinguishable from a uniformly random group element. So, the advantage of the distinguisher in that case is negligible.

(2) Case: *$\mathcal{D}$ queried on* $\mathsf{single} \neq \mathbf{true}$ *and* $\mathsf{am}_1 \neq \perp$, *so* t_i *comes from* aIssueU *and* $\hat{\gamma} = [P_k]$ *comes from a different session than* t_i *(and* $[P_k]$ *comes either from* IssueU *or* aIssueU*)*:
then, $\mathsf{t}_i = \mathsf{encode}(Y_i)$, and so $Y := \mathsf{decode}(\mathsf{t}_i) = Y_i$. Next, observe that $r := \mathsf{H}_q(Y^{\mathsf{sk}}) = r_i$. Recall that $[P_k]$ and t_i come from different, independent sessions. Since $[P_k]$ can come either from IssueU or aIssueU, we split the sub-cases again:

(a) Sub-case: *$\mathcal{D}$ selected* $\mathsf{am}_2 = \perp$, *so* $[P_k]$ *comes from* IssueU:
then $C_j := P_j \cdot T^{-r} = (T_j')^{r_j'} \cdot T^{-r}$. Note that both $T := \mathsf{H}_q(\mathsf{t}_i, \mathsf{chall}, \mathsf{pk})$ and $T_j' := \mathsf{H}_q(\mathsf{t}_j', \mathsf{chall}, \mathsf{pk})$ are uniformly random group elements, since they both are outputs of the random oracle H_q and their inputs ($\mathsf{t}_i := \mathsf{encode}(Y_i)$ and $\mathsf{t}_j' \leftarrow\!\$\ \{0,1\}^{256}$) are independent and indistinguishable from uniformly sampled, and so have not been used before as input to the random oracle with

overwhelming probability. It must be then that C_j is also indistinguishable from uniformly random, with a negligible distinguisher advantage.

(b) Sub-case: $\mathcal{D}$ *selected* $\mathsf{am}_2 \neq \perp$, *so* $[P_k]$ *comes from* aIssueU: then $C_j := P_j \cdot T^{-r} = (T'_j)^{r'_j} \cdot g^{\mathsf{am}_2} \cdot T^{-r}$. Again, both $T := \mathsf{H}_q(\mathsf{t}_i, \mathsf{chall}, \mathsf{pk})$ and $T'_j := \mathsf{H}_q(\mathsf{t}'_j, \mathsf{chall}, \mathsf{pk})$ are uniformly random group elements, since their inputs ($\mathsf{t}_i := \mathsf{encode}(Y_i)$ and $\mathsf{t}'_j := \mathsf{encode}(Y'_i)$) are indistinguishable from uniformly sampled and have not been used as random oracle query before with overwhelming probability. Exactly as in the case above, C_j is indistinguishable from uniformly random, with a negligible distinguisher advantage.

(3) Case: $\mathcal{D}$ *queried on* $\mathsf{single} \neq \mathbf{true}$ *and* $\mathsf{am}_1 = \perp$, *so* t_i *comes from* IssueU *and* $\hat{\gamma} = [P_k]$ *comes from a different session than* t_i *(and* $[P_k]$ *comes either from* IssueU *or* aIssueU*)*: then, $Y := \mathsf{decode}(\mathsf{t}_i)$ is indistinguishable from a uniformly sampled group element due to assumed properties of decode function (the input $\mathsf{t}_i \leftarrow\!\$\ \{0,1\}^{256}$ is uniformly random). What follows is that $r := \mathsf{H}_q(Y^x)$ must be a uniformly random scalar (since the random oracle H_q has been queried with a fresh input with overwhelming probability) and independent from all other values computed in the protocol. This means that $C_j = P_j \cdot T^{-r}$ must be indistinguishable from a uniformly random group element for any $P_j, T \in \langle g \rangle$. It does not matter whether $[P_k]$ comes from IssueU or aIssueU.

In all cases, C_j is indistinguishable from a uniformly sampled value from $\langle g \rangle$ and thus $\mathbf{G}_1$ and $\mathbf{G}_0$ are indistinguishable with negligible distinguisher's advantage.

$\mathbf{G}_2$: It is the same as $\mathbf{G}_1$, but, in aDec, instead of checking whether $\mathbf{T}[C_j] \neq \perp$ and returning $\mathsf{am} := \mathbf{T}[C_j]$ if the check passes, we always return $\perp$. Since $C_j \leftarrow\!\$\ \langle g \rangle$ is uniformly sampled, the probability that C_j lies in the preimage of $\mathbf{T}$ is equal $|\widehat{\mathcal{M}}|/q$, which is negligible from assumption. The distinguisher's advantage is then also negligible.

$\mathbf{G}_3$: It is the same as $\mathbf{G}_2$, but instead of sampling anamorphic keys, we do not sample any keys. Since keys are not used anywhere, the two games are indistinguishable.

The last game, $\mathbf{G}_3$, is identical to the Robustness game when $b = 1$ is selected and thus the advantage of $\mathcal{D}$ is bounded by a negligible factor.

□

Lemma E.10 (Full Decryptability). *Π_2 is a Fully-Decryptable Fully-Asymmetric Anamorphic Privacy Pass protocol with Triplet Σ_3.*

Proof. We give only a sketch of the proof. The intuition is that the only message sent by $\mathcal{I}$ ($[Q_k]$) is deterministic with respect to the message sent by $\mathcal{D}$ ($[P_k]$) and protected with a non-interactive zero-knowledge proof (of equivalence of logarithms $\pi = (c, s)$). In order to modify any message in the protocol, the adversary would have to forge a non-interactive zero-knowledge proof.

The sketch of the proof in the Random Oracle Model is as follows. Assume that an Adversary $\mathcal{A}$ that wins in the full decryptability game as specified in Definition 6 with regard to Π_2 with Σ_3. We assume that $\mathcal{A}$ modifies at least $[P_k]$ sent by device, as if it is not modified, then the output aDec will be correct and equal to am (it is the only message that transmits anamorphic information).

We will construct a reduction $\mathcal{B}$ that solves the DLP problem by leveraging the extractor for the non-interactive proof of discrete logarithms. Let $(X = g^x)$ be the DLP challenge. Then, we set $\mathsf{pk} := X$ and in the proxy execution of full decryptability game we do the following:

(1) We simulate $\mathcal{D}$ and send $[P_k]$ to $\mathcal{A}$. $\mathcal{A}$ modifies $[P_k]$ to $[P'_k]$ and sends it to the simulated $\mathcal{I}$.
(2) We simulate $\mathcal{I}$ in the Random Oracle Model by sampling $Q_i \leftarrow\!\$\ \langle g \rangle$, calculating d_i, M, Z in the same way as $\mathcal{D}$ in steps 14, 16 and 17 of Figure 7, selecting random $s, c \leftarrow\!\$\ \mathbb{Z}_q^*$, calculating $A := g^s \cdot X^c, B := M^s \cdot Z^c$ and finally setting in the Random Oracle $\mathsf{H}_q(\mathsf{pk}, M, Z, A, B) := c$. Then, on $\mathcal{A}$ input $[P'_k]$, we return $[Q_k], c, s$ to $\mathcal{A}$. Note that, from the point of view of the adversary, random Q_i is indistinguishable from $Q_i = P_i^{\mathsf{sk}}$ from the original protocol, since for them P_i^{sk} is indistinguishable from a uniformly sampled group element (due to DDH), and so it suffices to simulate the proof of equality of discrete logarithms, as is done above.
(3) Note that $\mathcal{A}$ needs to provide the same kind of proof of equivalence of logarithms for $[P_k], [Q'_k]$ for some $[Q'_k]$, because $\mathcal{D}$ expects it and the protocol will fail otherwise. Since $\mathcal{A}$ only knows a valid proof for $[P'_k]$ and $[Q_k]$, they must forge a new one for $[P_k], [Q'_k]$. Since the proof used here is also a proof of knowledge, it must be that $\mathcal{A}$ learns x and indeed $Q'_i = P_i^x$ for each $i \in [k]$. With overwhelming probability $\mathcal{A}$ makes a Random Oracle call for $\mathsf{H}_q(\mathsf{pk}, M', Z', A', B')$, for which we return them $c'_0 \leftarrow\!\$\ \mathbb{Z}_q$. The Adversary returns $[Q'_k], s'_0, c'_0$.
(4) We rewind $\mathcal{A}$ to the point of the Random Oracle call, give them a different value $c'_1 \leftarrow\!\$\ \mathbb{Z}_q$ and receive a different output $[Q'_k], s'_1, c'_1$. So we have $s'_0 = a_0 + x \cdot c'_0, s'_1 = a_1 + x \cdot c'_1$. Since the inputs for both Random Oracle calls are the same (and, in particular $A = g^a = g^{a_0} = g^{a_1}$), we also have $a_0 = a_1$. We then calculate $x = \frac{s'_1 - s'_0}{c'_1 - c'_0}$ and break DLP.

□

Theorem 5 folows directly from Lemmas E.7, E.8, E.9 and E.10.

E.6 Proof of Theorem 6

Lemma E.11 (Correctness of anamorphic triplet). *The triplet $\Sigma_4 = (\mathsf{aGen}, \langle \mathsf{aIssueS}, \mathsf{aIssueU} \rangle, \mathsf{aDec})$ with algorithms defined in Figure 11 and Figure 12 is a correct Fully-Asymmetric Anamorphic Triplet.*

Proof. Reall that for any key tuple $(\mathsf{sk}, \mathsf{pk}, \mathsf{sk_d} = (\mathbf{T}), \perp)$ generated by aGen and any message $\mathsf{am} \in \widehat{\mathcal{M}}$, if we execute

$$(\gamma, (\mathbf{t}, \boldsymbol{\tau})) \leftarrow \langle \mathsf{aIssueS}(\mathsf{sk}), \mathsf{aIssueU}(\mathsf{pk}, \mathsf{am}, \perp) \rangle,$$

then we receive $\gamma := (M_2, x_1, u)$, $\boldsymbol{\tau} := [\tau_k]$, $\mathbf{t} := [\mathsf{t}_k]$.

Correctness requires that aDec is successful for all $i \in [k]$, but let us fix t_i, τ_i for a moment. Let us go through the aDec algorithm. Observe that $m_2 = \mathsf{H}_q(\mathsf{chall})$. Then, $h'_i := M_2 \cdot g^{-m_2}$ and $h - i'$ equals h^{r_2} since $M_2 = g^{m_2} \cdot h^{r_2}$. Next, recall that $\mathsf{pk_d} := X'_1 = h^{x_1 \cdot u}$ and thus $k_i = \mathsf{H}_q((h'_i)^{x_1 \cdot u}, i) = \mathsf{H}_q((h)^{x_1 \cdot u \cdot r_2}, i) = \mathsf{H}_q((\mathsf{pk_d})^{r_2}, i)$. Since $a_i := \mathsf{H}_q((\mathsf{pk_d})^{r_2}, i) + \mathsf{am}$, then it must be that $k_i = a_i - \mathsf{am}$. So,

$D_i := (U_i')^{1/u} \cdot g^{-k_i} = g^{u \cdot a_i \cdot 1/u} \cdot g^{-a_i} \cdot g^{\mathsf{am}} = g^{\mathsf{am}}$, and finally, indeed $\mathbf{T}[D_i] = \mathsf{am}$.

Since correctness holds for an arbitrary fixed index $i \in [k]$, it follows that correctness holds for all $i \in [k]$.

We conclude that Σ_4 is a Fully-Asymmetric Anamorphic Triplet, since for all $i \in [k]$ and all honestly generated token/signature pairs (t_i, τ_i), aDec outputs the correct message am. □

Lemma E.12 (Indistinguishability). *Π_3 is a Fully-Asymmetric Anamorphic Privacy Pass protocol with Triplet Σ_4.*

Proof. (Draft) Consider the Indistinguishability game as specified in Definition 3 with respect to Π_3 with Σ_4.

Through a sequence of games, we will show that the anamorphic protocol execution (i.e., when $b = 0$ is selected) is indistinguishable from the regular protocol execution (i.e., when $b = 1$ is selected) and so it must be that their output and transcripts are also indistinguishable. (Note that in the below games, we only make changes to aIssueU):

- $\mathbf{G}_0$**:** This game is identical to the Indistinguishability game when $b = 0$ is selected and thus anamorphic keys are sampled $(\mathsf{sk}, \mathsf{pk}, \mathsf{sk_d}, \mathsf{pk_d}) \leftarrow\$ \mathsf{aGen}(1^\lambda)$, $(\mathsf{sk}, \mathsf{pk})$ are given to $\mathcal{A}$, and $\mathcal{A}$ can query the oracle for outputs and transcripts of the anamorphic execution

  $$(\gamma, (\mathbf{t}, \boldsymbol{\tau})); \mathcal{T} \leftarrow \langle \mathsf{aIssueS}(\mathsf{sk}), \mathsf{aIssueU}(\mathsf{pk}, \mathsf{am}, \bot) \rangle$$

  for adversarially selected am.
- $\mathbf{G}_1$**:** It is the same as $\mathbf{G}_0$, but instead of calculating a_i according to the formula $a_i := \mathsf{H}_q((\mathsf{pk_d})^{r_2}, i) + \mathsf{am}$ we sample $B \leftarrow\$ \langle g \rangle$, and set $a_i := \mathsf{H}_q(B, i) + \mathsf{am}$. From $\mathcal{A}$'s point of view, the following values related to $\mathsf{pk_d}^{r_2}$ are visible: h^{r_2} (derived easily from M_2), $X_1' = h^{x_1 \cdot u}$, and $X_1 = h^{x_1}$, for r_2, u chosen at random from $\mathbb{Z}_q^*$ and a random secret key x_1. The value $(\mathsf{pk_d})^{r_2} = h^{x_1 \cdot u \cdot r_2}$ is never revealed to $\mathcal{A}$, while $\mathcal{A}$ cannot calculate it due to the CDH assumption. So, from the point of view of $\mathcal{A}$, the random oracle output $\mathsf{H}_q((\mathsf{pk_d})^{r_2}, i)$ is computationally indistinguishable from a fresh random group element under CDH assumption, and so the two games are indistinguishable.
- $\mathbf{G}_2$**:** It is the same as $\mathbf{G}_1$, but instead of setting $a_i := \mathsf{H}_q(B, i) + \mathsf{am} \bmod q$ we sample $a_i \leftarrow\$ \mathbb{Z}_q^*$. Since B is randomly sampled, only used as the input of the random oracle, and independent from $\mathcal{A}$'s view, the output of H_q is uniform except a negligible probability that the random oracle has been queried with the same input before. Adding $\mathsf{am} \in \mathbb{Z}_q$ to a uniformly random scalar yields a value that is still a uniformly random scalar (except a negligible probability that $\mathsf{H}_q(B, i) + \mathsf{am} \bmod q = 0$), and so $\mathbf{G}_2$ and $\mathbf{G}_1$ are indistinguishable.
- $\mathbf{G}_3$**:** It is the same as $\mathbf{G}_2$, but instead of sampling anamorphic keys, we sample regular keys $(\mathsf{sk}, \mathsf{pk}) \leftarrow\$ \mathsf{Gen}(1^\lambda)$. Since $(\mathsf{sk}, \mathsf{pk})$ come from the same distribution and $(\mathsf{sk_d}, \mathsf{pk_d})$ are not used anywhere, the two games are indistinguishable.

The last game, $\mathbf{G}_3$, is identical to the Indistinguishability game when $b = 1$ is selected and thus the advantage of $\mathcal{A}$ is bounded by a negligible factor.

□

Lemma E.13 (Robustness). *Π_3 is a Robust Fully-Asymmetric Anamorphic Privacy Pass protocol with Triplet Σ_4.*

Proof. Consider the Robustness game specified in Definition 5 with respect to Π_3 with Σ_4.

Through a sequence of games, we will show that the oracle outputting am from a mismatched anamorphic protocol execution (when $b = 0$ is selected) is indistinguishable from the oracle that returns $\bot$ (when $b = 1$ is selected).

- $\mathbf{G}_0$**:** This game is identical to the Robustness game with $b = 0$. Thus, the anamorphic keys are sampled $(\mathsf{sk}, \mathsf{pk}, \mathsf{sk_d}, \mathsf{pk_d}) \leftarrow\$ \mathsf{aGen}(1^\lambda)$, and $\mathcal{D}$ can query the robustness oracle for mismatched decryption coming from the execution of the anamorphic protocol for adversarially selected am_1 and am_2.
- $\mathbf{G}_1$**:** It is the same as $\mathbf{G}_0$, but in aDec, instead of setting $D_i := (U_i')^{1/u} \cdot g^{-k_i}$ we sample $D_j \leftarrow\$ \langle g \rangle$. Since t_i, τ_i can either come from IssueU or from aIssueU and $\gamma = (M_2, x_1, u)$ can either come from the same session as t_i, τ_i or from a different one (and M_2 also can come either from a session running with aIssueU or IssueU), depending on $\mathcal{D}$'s query inputs, we split the cases:
  (1) Case: *$\mathcal{D}$ queried on* $\mathsf{single} = \mathbf{true}$*, so* t_i, τ_i *come from* IssueU *and* $\hat{\gamma} = (M_2, x_1, u)$ *comes from the same session as* t_i*:*
  since τ_i comes from IssueU, then $U_i' := U^{a_i} = (g^u)^{a_i}$ for a uniformly sampled $a_i \leftarrow\$ \mathbb{Z}_q^*$, which means that $(U_i')^{1/u} = g^{a_i}$ is indistinguishable from a uniformly random group element. At the same time, $k := \mathsf{H}_q((h_i')^{x_1 \cdot u}, i)$ is independent from a_i. Consequently, $D_i := (U_i')^{1/u} \cdot g^{-k_i} = g^{a_i} \cdot g^{-k_i}$ is indistinguishable from a uniformly random group element.
  (2) Case: *$\mathcal{D}$ queried on* $\mathsf{single} \neq \mathbf{true}$ *and* $\mathsf{am}_1 = \bot$*, so* t_i, τ_i *come from* IssueU *and* $\hat{\gamma} = (M_2, x_1, u)$ *comes from a different session than* t_i, τ_i *(and* M_2 *comes either from* IssueU *or* aIssueU*):*
  then, τ_i again comes from IssueU, and so $U_i' := (U^*)^{a_i^*} = (g^{u^*})^{a_i^*}$ for a uniformly sampled $a_i^* \leftarrow\$ \mathbb{Z}_q^*$. Additionally, in both sessions, $u, u^* \leftarrow\$ \mathbb{Z}_q^*$, which means that $(U_i')^{1/u} = g^{a_i^* \cdot 1/u \cdot u^*}$ is actually indistinguishable from a uniformly random group element. Again, $k := \mathsf{H}_q((h_i')^{x_1 \cdot u}, i)$ is independent from a_i^*, u, u^* and consequently, $D_i := (U_i')^{1/u} \cdot g^{-k_i} = g^{a_i^* \cdot 1/u \cdot u^*} \cdot g^{-k_i}$ is indistinguishable from a uniformly random group element. It does not matter whether M_2 comes from IssueU or aIssueU.
  (3) Case: *$\mathcal{D}$ queried on* $\mathsf{single} \neq \mathbf{true}$ *and* $\mathsf{am}_1 \neq \bot$*, so* t_i, τ_i *come from* aIssueU *and* $\hat{\gamma} = (M_2, x_1, u)$ *comes from a different session than* t_i, τ_i *(and* M_2 *comes either from* IssueU *or* aIssueU*):*
  Recall that $\tau_i := (U_i', C_i, W_i, \pi_i^P, i)$. Then, $U_i' := (U^*)^{a_i^*} = g^{u^* \cdot a_i^*}$ for $a_i^* := \mathsf{H}_q((\mathsf{pk_d}^*)^{r_2^*}, i) + \mathsf{am}_1$. Next, observe that $k_i := \mathsf{H}_q((h_i')^{x_1 \cdot u}, i)$ and let $k_i^* := \mathsf{H}_q((\mathsf{pk_d}^*)^{r_2^*}, i) = a_i^* - \mathsf{am}_1$. Then, $D_i := (U_i')^{1/u} \cdot g^{-k_i} = g^{u^* \cdot a_i^* \cdot 1/u} \cdot g^{-k_i} = (g)^{u^* \cdot 1/u \cdot (k_i^* + \mathsf{am}_1) - k_i}$. Note that for $k_i^* := \mathsf{H}_q(h^{x_1 \cdot u^* \cdot r_2^*}, i)$ and $k_i := \mathsf{H}_q(h^{x_1 \cdot u \cdot r_2}, i)$ the inputs to the random oracle H_q are distinct (except for negligible probability $1/(q-1)$), and hence outputs are independent, since $u, u^*, r_2, r_2^* \leftarrow\$ \mathbb{Z}_q^*$.

Because k_i is also uniformly random (as an output of the random oracle), and independent from the uniformly sampled u, u^* and am_1, then it must mean that D_i is indistinguishable from a uniformly random group element. It does not matter whether M_2 comes from IssueU or aIssueU.

In all cases, D_i is indistinguishable from a uniformly random element of $\langle g \rangle$ and thus $\mathbf{G}_1$ and $\mathbf{G}_0$ are indistinguishable.

$\mathbf{G}_2$: It is the same as $\mathbf{G}_1$, but, in aDec, instead of returning $\mathsf{am} := \mathbf{T}[D_j]$, we always return $\perp$. Since $D_j \leftarrow\!\!\$\ \langle g \rangle$ is uniformly random, the probability that D_j lies in the preimage of $\mathbf{T}$ is equal $|\widehat{\mathcal{M}}|/q$, which is negligible from assumption. The distinguisher's advantage is then negligible and thus $\mathbf{G}_2$ and $\mathbf{G}_1$ are indistinguishable.

The last game, $\mathbf{G}_2$, is identical to the Robustness game when $b = 1$ is selected and thus the advantage of $\mathcal{D}$ is negligible. □

Lemma E.14 (Full Decryptability). *Π_3 is a Fully-Decryptable Fully-Asymmetric Anamorphic Privacy Pass protocol with Triplet Σ_4.*

Sketch. Note that every value in each message of the protocol has its structure proven with a non-interactive zero-knowledge proof. We can then reduce full decryptability to soundness of non-interactive PoK — if it is infeasible to create a convincing proof without the knowledge of corresponding witnesses, then it must be impossible for the adversary $\mathcal{A}$ to forge such a proof, which means that it is impossible for them to modify any message. □

Theorem 6 follows directly from Theorems E.11 to E.14 and from the fact that aGen returns a void key $\mathsf{pk}_\mathsf{d} = \perp$.

F Additional Figures

Privacy Pass (Pub. V.) Redeem($\mathsf{pk}, \mathsf{t}_i, \tau_i$)

Verifier $\mathcal{V}$

$(\mathsf{pk} = (N, e), \mathsf{t}_i, \tau_i)$; chall

1: $T = \mathsf{t}_i \| \text{chall} \| \mathsf{pk}$
2: **if** $\text{valid}_{\mathsf{PSS}}(T, n, (\tau_i)^e \bmod N) = 1$:
3: **return** 1 // "valid"
4: **else** : **return** 0 // "invalid"

Figure 15: Privacy Pass token redemption procedure Redeem **for publicly verifiable tokens.**

Privacy Pass (Pub. V.) + Watchdog $\langle \mathsf{IssueS}(\mathsf{sk}), \mathsf{Watch}(\mathsf{pk}), \mathsf{IssueU}(\mathsf{pk}) \rangle$

Issuer $\mathcal{I}$		**Watchdog** $\mathcal{W}$		**Device** $\mathcal{D}$
$(\mathsf{sk} = d)$		$(\mathsf{pk} = (N, e))$		(pk)
		1: $r^* \leftarrow\$ \mathbb{Z}_N^* : \gcd(r, N) = 1$	$\xleftarrow{P}$	Steps 1-6 of Fig. 3
	$\xleftarrow{P^*}$	2: $P^* := P \cdot (r^*)^e \bmod N$		
Step 7 of Fig. 3	$\xrightarrow{s}$	3: $s^* := s \cdot (r^*)^{-1} \bmod N$	$\xrightarrow{s^*}$	Steps 8-11 of Fig. 3

Figure 16: Token issuance protocol for Publicly Verifiable Privacy Pass with a watchdog $\langle$IssueS, Watch, IssueU$\rangle$**.**

Privacy Pass (Priv. V.) Redeem($\mathsf{sk}, \mathsf{t}_i, \tau_i$)

Verifier $\mathcal{V}$

$(\mathsf{sk}, \mathsf{t}_i, \tau_i)$; pk, chall

$T := \mathsf{H}_{\langle g \rangle}(\mathsf{t}_i, \text{chall}, \mathsf{pk})$	: 1
$W := \mathsf{H}(T^{\mathsf{sk}})$	: 2
if $W \stackrel{?}{=} \tau_i$:	: 3
return 1 //"valid"	: 4
else : **return** 0 //"invalid"	: 5

Figure 17: Privacy Pass token redemption procedure Redeem **for privately verifiable tokens.**

ARC Gen(1^λ)

1: $x_0, x_1, x_2, x_b \leftarrow\$ \mathbb{Z}_q$
2: $X_0 := g^{x_0} \cdot h^{x_b}$
3: $X_1 := h^{x_1}, X_2 = h^{x_2}$
4: $\mathsf{sk} := (x_0, x_1, x_2, x_b)$
5: $\mathsf{pk} := (X_0, X_1, X_2)$
6: **return** $(\mathsf{sk}, \mathsf{pk})$

Figure 18: The key generation function Gen(1^λ) **for ARC.**

$\mathsf{ARC}\ (\bot, (\mathbf{t}, \boldsymbol{\tau})) \leftarrow \langle \mathsf{IssueS}(\mathsf{sk}), \mathsf{IssueU}(\mathsf{pk}) \rangle$

Issuer $\mathcal{I}$		**Device** $\mathcal{D}$	
$(\mathsf{sk} = (x_0, x_1, x_2, x_b)); \mathsf{pk} = (X_0, X_1, X_2)$		$(\mathsf{pk}); \mathsf{chall}, \mathsf{pctx}$	
		$m_1, r_1, r_2 \leftarrow\$ \mathbb{Z}_q$	: 1
		$m_2 := \mathsf{H}_q(\mathsf{chall})$	: 2
		$M_1 := g^{m_1} \cdot h^{r_1}; M_2 := g^{m_2} \cdot h^{r_2}$	: 3
		$\pi_1 \leftarrow \mathsf{PoK.Prove}\left\{(m_1, m_2, r_1, r_2) : \begin{matrix} M_1 = g^{m_1} \cdot h^{r_1} \\ M_2 = g^{m_2} \cdot h^{r_2} \end{matrix}\right\}$	: 4
	$\xleftarrow{M_1, M_2, \pi_1}$		
5: **if** $\mathsf{PoK.Verify}(\pi_1, M_1, M_2) \neq 1$: **return** $\bot$			
6: $u \leftarrow\$ \mathbb{Z}_q^*; U := g^u$			
7: $\widehat{V} := (X_0 \cdot M_1^{x_1} \cdot M_2^{x_2})^u$			
8: $X_0' := h^{x_b \cdot u}; X_1' := X_1^u; X_2' := X_2^u$			
9: $\pi_2 \leftarrow \mathsf{PoK.Prove}\left\{(x_0, x_1, x_2, x_b, u) : \begin{matrix} X_0 = g^{x_0} \cdot h^{x_b} \\ X_1 = h^{x_1} \\ X_2 = h^{x_2} \\ X_0' = h^{x_b \cdot u} \\ X_1' = h^{x_1 \cdot u} \\ X_2' = h^{x_2 \cdot u} \\ U = g^u \\ \widehat{V} = (X_0 \cdot M_1^{x_1} \cdot M_2^{x_2})^u \end{matrix}\right\}$			
10: $R := (X_0', X_1', X_2', U, \widehat{V}, \pi_2)$			
	$\xrightarrow{R, \pi_2}$		
		$(X_0', X_1', X_2', U, \widehat{V}, \pi_2) \leftarrow R$	: 11
		if $\mathsf{PoK.Verify}(\pi_2, X_0, X_1, X_2, X_0', X_1', X_2', U, \widehat{V}) \neq 1$: **return** $\bot$	: 12
		$V := \widehat{V} \cdot \left(X_0' \cdot (X_1')^{r_1} \cdot (X_2')^{r_2}\right)^{-1}$	: 13
		for $i \in [n]$: / Randomized tokens precomputation	: 14
		$a_i, b_i, z_i \leftarrow\$ \mathbb{Z}_q^*$	: 15
		$U_i' := U^{a_i}, V_i' := V^{a_i}$	: 16
		$C_i := V_i' \cdot g^{b_i}$	: 17
		$\mathsf{t}_i := (U')^{m_1} \cdot h^{z_i}$	: 18
		$T_i := \mathsf{H}_{\langle g \rangle}(\mathsf{pctx})$	: 19
		$W_i := T_i^{(m_1+i)^{-1}}; W_i' := (W_i)^{m_1}$	: 20
		$Z_i := X_1^{z_i} \cdot (g^{b_i})^{-1}$	: 21
		$\pi_i^P \leftarrow \mathsf{PoK.Prove}\left\{(m_1, b_i, z_i) : \begin{matrix} \mathsf{t}_i = (U_i')^{m_1} \cdot h^{z_i} \\ Z_i = X_1^{z_i} \cdot (g^{b_i})^{-1} \\ T_i = (W_i)^{m_1+i} \\ W_i' = (W_i)^{m_1} \end{matrix}\right\}$	: 22
		$\tau_i := (U_i', C_i, W_i, \pi_i^P, i)$	: 23
		return $(\mathbf{t} := [\mathsf{t}_n], \boldsymbol{\tau} := [\tau_n])$	: 24

Figure 19: ARC token signing interaction $\langle \mathsf{IssueS}, \mathsf{IssueU} \rangle$. The arguments chall (context specific request token challenge provided by the origin) and pctx (presentation context) are auxiliary semi-public arguments.

ARC Redeem$(\mathsf{sk}, \mathrm{t}_i, \tau_i)$

Verifier $\mathcal{V}$

$\big(\mathsf{sk} = (x_0, x_1, x_2, x_b), \mathrm{t}_i, \tau_i\big)$; chall, pctx

$(U_i', C_i, W_i, \pi_i^P, i) \leftarrow \tau_i$

$m_2 := \mathsf{H}_q(\text{chall})$

$T := \mathsf{H}_{\langle g \rangle}(\text{pctx})$

$W' := T \cdot (W_i)^{-i}$

$Z := (U_i')^{x_0} \cdot (\mathrm{t}_i)^{x_1} \cdot (U_i')^{x_2 \cdot m_2} \cdot C_i^{-1}$

$b \leftarrow \mathsf{PoK.Verify}(\pi_i^P, \mathrm{t}_i, Z, T, W')$

if $b \stackrel{?}{=} 1$ **and** $0 \le i < k$:

  return 1 //"valid"

else : **return** 0 //"invalid"

Figure 20: ARC token redemption procedure Redeem.