No Honor Among Crooks: Non-transferable Anonymous Tokens from Betrayability

David Kretzler
Huawei Technologies
Düsseldorf, Germany
david.kretzler@huawei.com

Yong Li
Huawei Technologies
Düsseldorf, Germany
yong.li1@huawei.com

***Abstract*—In anonymous token protocols, clients obtain access tokens by proving eligibility for the usage of a resource and later get access to the resource by redeeming the token. The server verifying eligibility and providing the resource cannot link the token issuance to its redemption. To prevent ineligible clients from accessing resources, it is crucial to prevent the transfer or sale of tokens. Durak et al. (CCS'24) propose binding tokens to valuable insurance secrets, which must be known to redeem the tokens. The value of the insurance secret deters the vendor from transferring the secret to the token buyer, who cannot redeem the token without the secret. However, the authors do not consider scenarios, where the token vendor assists the buyer during the token redemption. Their construction, therefore, falls short to guarantee non-transferability when facing a vendor-aided token redemption.**

We address this gap by introducing the concept of *anonymous tokens with betrayability*. Our notion ensures that a token buyer, that is able to redeem a bought token, either knows the insurance secret or is able to betray the vendor in a vendor-aided redemption. The betrayal allows the buyer to steal the insurance secret without being detected. This way, we make the support in the token redemption equivalent to a transfer of the insurance secret and, hence, inherit the transfer deterrence of the insurance secret even when considering a vendor-aided token redemption. We formalize our new security notion, present a protocol for anonymous tokens with betrayability, prove its security, and provide an implementation and experimental evaluation.

1. Introduction

In recent years, anonymous tokens attracted a lot of attention, both from academia [1]–[11] and industry [12]–[14]. An anonymous token scheme is a protocol executed between three parties, the client, the issuer and the verifier. A client first requests a token from the issuer via an interactive issuance protocol, in which the issuer verifies the client's eligibility to access a resource and then issues a token to the client. A prominent example for eligibility is humanhood, which can be proven by solving a CAPTCHA. At a later point in time, the client presents the token to a verifier, which verifies authenticity of the token and grants access to a service or resource if the verification is successful. This process is called *token redemption*. Usually, the verifier checks whether the token has already been redeemed to prevent double spending. The basic security properties required by anonymous tokens are accountability and anonymity. The former states that only the issuer can issue valid tokens while the latter requires that the issuer and verifier cannot link the token issuance to its redemption.

Various publications [1]–[11] introduced a variety of protocols for anonymous tokens offering different trade-offs and additional properties. While some constructions [1], [2], [6] require the token issuance key for the verification of tokens and, hence, require the issuer and verifier to be the same entity, others [3]–[5], [7]–[11] enable tokens to be verified publicly and, hence, allow the issuer and verifier to be different entities. Further, there are proposals allowing issuers to include public [4], [10] or private [2], [3], [6], [11] metadata into tokens. Public meta data allows to add a label to tokens such as the revocation information or a usage restriction. Private meta data, on the other hand, allows the issuer to embed an invalidity flag into tokens instead of rejecting token requests. As the meta data is private, the flag can only be checked by the verifier but not by the client. This makes it harder for adversaries to derive information about the eligibility verification process.

One aspect that has received only little attention so far is the prevention of the transfer of tokens. Consider a scenario in which one client, the vendor, receives a token and sells it to another client, the buyer. Such an undesired flow of tokens can grant buyers access to resources for which they are not eligible. Unfortunately, preventing the transfer of tokens while still guaranteeing anonymity is notoriously hard. If the vendor transfers its whole state to the buyer, the only difference between the vendor and the buyer is the identity. A verifier capable of rejecting transferred tokens, hence, needs to know something about the client's identity, which violates anonymity. The only work so far addressing non-transferability of tokens is by Durak et al. [7]. In their work, the authors utilize a technique previously known from [15]–[17] to address the described difficulty. They bind each token to a valuable insurance secret, such as the secret key of a cryptocurrency wallet. The insurance secret needs to be known to redeem the token. However, the client does not present the insurance secret in clear but provides some evidence proving knowledge of the secret. The intuition

behind this approach is that even though a malicious token vendor might be willing to transfer a token it will not transfer its insurance secret due to its value.

One downside of the solution of [7] is that the authors assume clients to redeem tokens on their own, i.e., the vendor and the buyer cannot communicate during the token redemption. Without this assumption, it would be possible for the vendor to assist the buyer in the creation of the evidence, that proves knowledge of the insurance secret, without actually transferring the insurance secret to the buyer. Such an attack undermines the non-transferability properties of their construction. We aim to address this shortcoming. In particular, we allow the token vendor to support the token buyer during the token redemption but enable the token buyer to betray the vendor during the vendor-aided evidence creation. More precisely, we enable the buyer to embed a trapdoor into the evidence creation, which allows it to retrieve the insurance secret from the evidence without being detected. As a vendor cannot know if it will be betrayed by a buyer, the transfer of a token including the aid in the redemption of said token becomes equivalent to transferring the insurance secret itself.

1.1. Contribution

In this work, we propose the notion of *anonymous tokens with betrayability (ATB)*, provide a formal definition, present an instantiation based on BBS signatures [18], prove security of our instantiation, and provide an experimental evaluation. Both, our security notion and our constructions constitute extensions of *non-transferable anonymous tokens (NTAT)* introduced by [7].

Similar to NTAT, anonymous tokens with betrayability bind tokens to valuable insurance secrets and require clients redeeming a token to prove ownership of the associated insurance secret. However, unlike prior work, we explicitly consider the threat of a collusion between the token vendor and the token buyer during redemption. To address this threat, we require betrayability, which allows the token buyer to betray the token vendor by including a trapdoor into the vendor-aided redemption. The trapdoor enables the extraction of the insurance secret without being detected. This way, we make the support in the redemption of a token equivalent to the transfer of the associated insurance secret.

We observe that it is impossible to achieve betrayability in the model proposed by [7] (cf. Appendix A). Therefore, we make a minor sacrifice in terms of unlinkability, i.e., we identify clients during redemption via a pseudonym realized as long-time identification public key. This means that we restrict our scope to applications in which the provided resource or service is bound to an identity, e.g., a user account, but a foreign-issued token is still required to prove claims about the client, e.g., human-hood. However, our scheme still guarantees that the token redemption cannot be linked to its issuance and allows the utilization of multiple pseudonyms for the same or different services. Technically, the introduction of identification keys is necessary as a buyer without a secret, that is capable of betraying the vendor, could be emulated by a malicious verifier to extract insurance secrets from honest clients. By binding the trapdoor to the client's identification secret key, we can prevent the verifier from exploiting the trapdoor. Conceptually, we assess the identification of clients before accessing a resource to fit well into the setting of non-transferable tokens, as any token-authorized access to resources not bound to the client's identity could also be performed by the token vendor on behalf of the buyer, which makes the transfer of the token redundant.

We present an instantiation of our ATB definition and prove its security. Our construction aims to push most of the runtime to the issuance phase, as we envision this phase to be less efficiency critical. Furthermore, we have implemented and evaluated the presented instantiation as well as three alternative constructions offering different trade-offs, i.e., runtime versus communication overhead and issuance versus redemption efficiency. Our evaluation shows that all constructions provide high efficiency in the phase, they are optimized for (1 - 34 % of the overall runtime). Furthermore, the worst-case runtime of a whole protocol execution without network delays in the runtime-optimized versions of our construction is below 200 ms.

1.2. Technical Overview

In the following, we provide a technical overview over our construction.

BBS signatures. Analogously to [7], [19], tokens in our work constitute BBS signatures [18], [20], [21]. Let $\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$ be a tuple of groups of prime order p with generators $G_0, H_0, H_1, \ldots, H_n \in \mathbb{G}_1$ and $G_2 \in \mathbb{G}_2$ and map $\mathsf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear pairing. A BBS signature on a message array $\{x_i\}_{i\in[0,L]}$ under a secret signing key $z \in \mathbb{Z}_p$ is a tuple (σ, e) with $\sigma = \frac{1}{z+e}(G_0 + \sum_{i\in[0,L]} x_i H_i)$ and $e \xleftarrow{\$} \mathbb{Z}_p$. The signature is verified under public verification key $Z = zG_2$ by checking whether $\mathsf{e}(\sigma, Z + eG_2) = \mathsf{e}(G_0 + \sum_{i\in[0,L]} x_i H_i, G_2)$.

Insurance secret and blind issuance. The insurance secret is a secret key $\mathsf{sk}^{\mathsf{In}}$ associated with a public insurance key $\mathsf{pk}^{\mathsf{In}} = \mathsf{sk}^{\mathsf{In}} G_1$. In order to bind $\mathsf{sk}^{\mathsf{In}}$ to a token, it is embedded into the token's signature together with a mask $r \xleftarrow{\$} \mathbb{Z}_p$, i.e., $\sigma = \frac{1}{z+e}(G_0 + rH_0 + \mathsf{sk}^{\mathsf{In}} H_1)$. We call σ the token and (r, e) the witness denoted as ω.

To prevent an issuer from learning the token, which would harm the unlinkability guarantees of the construction, issuers are required to sign tokens blindly. Clients send a signing request $T = \delta(G_0 + rH_0 + \mathsf{sk}^{\mathsf{In}} H_1)$ for $\delta \xleftarrow{\$} \mathbb{Z}_p$ and issuers provide a blind signature $S = \frac{1}{z+e} T$ which is unblinded to token $\sigma = \frac{1}{\delta} S$. To prevent maliciously crafted requests, we require that clients append zero-knowledge proofs of correct request creation to their message.

Naive token redemption. The naive way to realize the token redemption is to send the token and the public identification key $\mathsf{pk}^{\mathsf{Id}}$ together with a zero-knowledge proof of knowledge of the witnesses $(e, r, \mathsf{sk}^{\mathsf{In}}, \mathsf{sk}^{\mathsf{Id}})$ to the verifier.

While this naive approach is sufficient for schemes not considering non-transferability, it does not ensure that the client submitting the proof, the buyer, indeed knows the witnesses. The proof can also be generated by another client, the vendor, and be transferred to the buyer together with the token. Even if the zero-knowledge proof is interactive and needs to be generated based on a challenge issued by the verifier, as done by [7], it is possible for the buyer to still redeem the token without knowledge of the witnesses by forwarding the challenge to the vendor and sending the vendor's response to the verifier.

Token redemption with betrayability. We mitigate the problems of the naive approach by including a trapdoor into the token redemption that can be used by a token buyer to betray a vendor and steal the vendor's insurance key. In particular, we design a three-message redemption protocol as follows. The redemption is initiated by the client by sending the token σ and its identity public key $\mathsf{pk}^{\mathsf{Id}}$. The verifier responds with an encryption key K. The client encrypts the secret insurance key $\mathsf{sk}^{\mathsf{In}}$ under K and sends the ciphertexts together with a zero-knowledge proof ensuring the verifier that the client knows the witnesses $(e, r, \mathsf{sk}^{\mathsf{In}}, \mathsf{sk}^{\mathsf{Id}})$ and that the ciphertexts have been created correctly. If a buyer redeeming a token wants to steal the associated insurance key from a token vendor assisting in the proof creation, the buyer sends an encryption key K', for which it knows the secret decryption key k', to the vendor. If the vendor creates a valid proof, the client can use the decryption key k' to extract the insurance secret $\mathsf{sk}^{\mathsf{In}}$.

To ensure that the verifier cannot retrieve the insurance secret $\mathsf{sk}^{\mathsf{In}}$ while still guaranteeing that a buyer can do so, we require that the verifier proves that K is useless, i.e., that it does not know the corresponding decryption key, in a way that can only be verified by the client redeeming a token. Technically, we require the verifier to prove that either K is useless or it knows the secret identification key $\mathsf{sk}^{\mathsf{Id}}$ ($\mathsf{pk}^{\mathsf{Id}} = \mathsf{sk}^{\mathsf{Id}} G_1$). As the verifier does not know $\mathsf{sk}^{\mathsf{Id}}$, it can only send a useless key K to the client. However, the buyer knowing its identification key $\mathsf{sk}^{\mathsf{Id}}$, can replace K with K' and forge a valid proof. Although the vendor could prevent the betrayal by specifying the identity key for which the token should be redeemed, this would prevent the token buyer from redeeming the token for one of its own identities and, hence, render the bought token useless.

Zero-knowledge friendly trapdoor. We utilize the exponential ElGamal cryptosystem [22] for the encryption of the insurance secret as it harmonizes well with the Schnorr-style [23] zero-knowledge proofs we utilize in our construction. The client computes a ciphertext of a message x under public key K as $(A = r'G_1, B = r'K + xG_1)$. Further, we realize the proof of uselessness of K by sampling K as valid ElGamal key but with respect to generator G_0, i.e., $K = kG_0$, and proving knowledge of $k = \mathsf{dlog}_{G_0}(K)$. As ciphertexts are created with respect to G_1, the adversary would have to know $\mathsf{dlog}_{G_1}(G_0)$ to compute the decryption key associated with K.

Unfortunately, exponential ElGamal does not allow for direct decryption of high entropy secrets. The buyer can compute xG_1 but would need to be able to compute $\mathsf{dlog}_{G_1}(xG_1)$ to retrieve x. If x has low entropy it can be retrieved with brute-forcing techniques. This is not possible for $\mathsf{sk}^{\mathsf{In}}$. To address this problem, we split the bit representation of $\mathsf{sk}^{\mathsf{In}}$ into chunks and encrypt the chunks individually, i.e., we split $\mathsf{sk}^{\mathsf{In}}$ into $x_1, \ldots, x_L$ such that $x_i \in [0, 2^\ell - 1]$ and $\mathsf{sk}^{\mathsf{In}} = \sum_{i \in [L]} 2^{(i-1)\ell} x_i$ and pick ℓ such that $\mathsf{dlog}_{G_1}(x_i G_i)$ can be computed efficiently. To ensure that ciphertexts can be decrypted, the client does not only prove correct creation of the ciphertexts but also that the chunks are in the correct range and add up to the insurance key $\mathsf{sk}^{\mathsf{In}}$. In our evaluation, we consider instantiations of the range proof based on Bulletproof++ [24] and Groth16 general-purpose SNARKs [25].

Pushing the bottleneck to the issuance phase. We assess the issuance phase to be less efficiency critical than the redemption phase. This is due to the fact that the issuance phase usually includes the verification of the client's eligibility to receive a token such as the solving of a CAPTCHA. Therefore, we aim to push the major protocol overhead of our construction, the creation of the range proofs, to the issuance phase. We do so by including the insurance key's chunk decomposition instead of the key itself into the token. BBS signatures support the efficient creation of signatures on message arrays without an increase in the signature size and, hence, the token size. When sending a token request, the client proves in zero-knowledge that the request contains the chunk decomposition of the insurance secret corresponding to its public insurance key and that all chunks are in the correct range. During redemption, the client only needs to prove that the provided ciphertexts match the secret key chunks embedded into its token.

1.3. Related Work

The concept of anonymous tokens is closely related to the long-studied concept of anonymous credentials. In the following, we provide an overview of the literature on anonymous credentials and anonymous tokens.

Anonymous credentials. Anonymous credentials have first been introduced by Chaum [26]–[28] in the form of blind signatures and subsequently refined by a long line of work including [17], [19], [29]–[39]. An anonymous credential can be used to prove statements about the credential holder based on attributes encoded into the credential. In contrast to anonymous tokens, anonymous credentials are designed to be used multiple times. The credential ensures that a utilization of the credential does not leak any information about the holder or the credential's attributes beyond the proven statement. Furthermore, it is guaranteed that a utilization of a credential cannot be linked to its issuance or any other utilization of the same credential. Anonymous credentials are usually realized as blindly issued signatures on lists of attributes. A popular signature scheme for building anonymous credentials is the BBS(+) signature scheme studied in [18], [20], [21], [40]–[42]. The features making BBS(+) signatures appealing for anonymous credentials are the capability to sign arrays of messages without increasing

the signature size, the support of blind signature issuance with minimal overhead, and the existence of efficient zero-knowledge proofs for the disclosure of individual attributes.

Non-transferability in the context of anonymous credentials has been studied by [17], [30], [38], [39]. In [17], the authors introduced the concept of insurance secrets previously known from [15], [16] to the domain of anonymous credentials. Subsequently, [30] proposed an alternative notion of non-transferability, in which a client can only share one of its credentials by sharing all of its credentials. The credentials the client is not willing to share act as an insurance disincentivizing the transfer of the credentials it is willing to share. UProve [39], is an industrial instantiation of anonymous credentials that adapts the insurance secret approach of [17] to achieve non-transferability. While UProve is highly efficient, it only supports one-show credentials (similar to tokens) and lacks a formal security proof. An alternative to insurance secrets is to store the credential within a secure hardware device. In [38], the authors proposed an anonymous credential scheme tailored to resource constrained secure hardware devices. In their protocol, the secure hardware device generates the proofs to use the credential, but relies on the unprotected hardware to improve efficiency. None of the works on non-transferable credentials considers the risk of a collusion between the credential vendor and buyer during the credential's utilization. While we assess that a vendor-aided utilization of credentials can be prevented with the use of secure hardware, we are interested in a solution that can be implemented in software.

Anonymous tokens. Anonymous tokens have been studied by [1]–[11]. One approach adopted by [1], [2], [4], [9] is to construct anonymous tokens based on Oblivious Pseudo Random Functions (OPRFs) [43]. While [1] formally introduced the concept of anonymous tokens, [2] extended their work with private metadata bits and [4] added public metadata and public verifiability. Subsequently, [9] proposed an OPRF-based token scheme providing public verifiability without requiring bilinear pairings. An alternative approach to OPRFs proposed by [3], [8], [10], [11] is to instantiate publicly verifiable anonymous tokens based on blind signatures [26], [27]. In particular, [3] utilizes blind Schnorr signatures [23], [44] and supports private metadata bits, [8] presents constructions based on blind BLS multi-signatures [45]–[47] and Snowblind [48] supporting a decentralized issuance of tokens, [10] proposes a construction supporting public metadata that is based on RSA signatures [49], and [11] utilizes so-called non-interactive blind signatures [50] to enable the issuance of tokens independently of a specific client request.

Another line of work [6], [7] instantiates anonymous tokens without public verifiability based on algebraic message authentication codes (MACs). Technically, the MACs are realized as blindly issued BBS signatures in a group without a pairing and, hence, without public verifiability of signatures. In [6], the authors identified weaknesses in the secrecy of the metadata bit in [2], [4] and proposed a MAC-based construction with a private metadata bit providing stronger secrecy guarantees. Subsequently, Durak et al. [7] introduced the notion of non-transferability to the domain of anonymous tokens by adapting the insurance secret technique from [17], [39] to an algebraic MAC-based anonymous token construction derived from [19].

As [7] is the work most closely related to ours, we will discuss it in more detail. The authors extend prior definitions of anonymous tokens with a non-transferability property that requires clients to embed an insurance secret into tokens during issuance, which must also be known during redemption. To formalize this property, they require the existence of an extractor such that, for every client capable of successfully redeeming a token, the extractor–given black-box access to the client–can recover the corresponding insurance secret. The authors instantiate non-transferable anonymous tokens by realizing tokens as blindly issued BBS-based MACs on the insurance secret and including an interactive zero-knowledge proof of knowledge of the redeemed token's secret into the redemption protocol. The interactivity ensures freshness of the proof, making it impossible for a vendor to precompute a proof and transfer it together with the token. While the primary construction of [7] requires the issuer and the verifier to be the same party, the authors propose an extension with public verifiability that replaces the BBS-based MACs with BBS signatures. Furthermore, they show how their scheme can be extended to support public metadata.

Unfortunately, Durak et al. [7] do not consider a vendor-aided token redemption. If a vendor assists a buyer during the redemption of a transferred token, it can help generate the proof without revealing the insurance secret to the buyer, despite the proof being interactive. When considering such an attack, the extractor in the security definition effectively extracts the secret from the vendor rather than the buyer. Our work builds on [7] but addresses this limitation. Conceptually, we replace the non-transferability property with a betrayability property, which explicitly models collusion between a token vendor and a token buyer during redemption and requires the buyer to be able to steal the insurance secret from the vendor. Technically, we adopt the approach of the public verifiable protocol variant of [7], realizing tokens as blindly issued BBS signatures. However, we extend the token structure and embed a trapdoor into the redemption protocol that can be exploited only in a vendor-aided token redemption and enables the buyer to steal the vendor's secret. As in [7], our scheme can be extended to support public metadata by encoding additional data into tokens, as explained in Appendix E.

2. Preliminaries

We denote the security parameter by $\lambda \in \mathbb{N}$, the set $\{1, ..., n\}$ as $[n]$, and the set $\{a, a+1, \ldots, b-1, b\}$ as $[a, b]$. All considered sets are ordered multisets. We denote a negligible function by negl and computational indistinguishability of two distributions by $\approx_c$.

We use bilinear pairings as defined by [51]. A bilinear pairing group is described by three cyclic groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ of prime order p, generators $G_1 \in \mathbb{G}_1, G_2 \in$

$\mathbb{G}_2$ and a mapping $\mathsf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The map e is called a bilinear map iff it can be computed efficiently, $\mathsf{e}(aU, bV) = (ab)\mathsf{e}(U, V)$ for all $(U, V, a, b) \in \mathbb{G}_1 \times \mathbb{G}_2 \times \mathbb{Z}_p \times \mathbb{Z}_p$, and $\mathsf{e}(G_1, G_2) \neq 1$ for all generators G_1 and G_2. To generate a bilinear pairing group, we make use of a group generation algorithm GGen that on input 1^λ outputs a group description $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e})$, where p is a λ-bit prime and the groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ are of order p. We denote by $\mathbb{G}^* = \mathbb{G} \setminus \{1_\mathbb{G}\}$ the set of all generators of $\mathbb{G}$ for $\mathbb{G} \in \{\mathbb{G}_1, \mathbb{G}_2\}$. In this work, we consider Type-2 or Type-3 pairings, i.e., pairings, in which no efficient homomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ is known. For every bilinear pairing group $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}) \leftarrow \mathsf{GGen}(1^\lambda)$ we assume the discrete logarithm to be hard in $\mathbb{G}_1, \mathbb{G}_2$, and $\mathbb{G}_T$ and the external Diffie-Hellman (XDH) assumption to hold in $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e})$. The latter requires the decisional Diffie-Hellman (DDH) assumption to hold in $\mathbb{G}_1$. We denote an algorithm that attempts to compute the discrete logarithm of element $H \in \mathbb{G}_1$ to the basis G with a brute-forcing strategy over exponents $x \in [0, 2^i]$ by $\mathsf{dlog}_G(h)[0, 2^i]$.[1] We abbreviate probabilistic polynomial time algorithms by PPT and denote the execution of a PPT algorithm on input b by $a \leftarrow \mathcal{A}(b)$. To fix the random tape r of algorithm $\mathcal{A}$, we write $a \leftarrow \mathcal{A}(b; r)$. Finally, we treat $\frac{1}{0}$ as 0.

2.1. Zero-Knowledge Proofs of Knowledge

Zero-knowledge proofs of knowledge allow a prover to convince a verifier that it knows a witness w for a statement s of a NP-language $\mathcal{L}_\mathcal{R}$ with associated relation $\mathcal{R}$ such that $(s, w) \in \mathcal{R}$. We use the Camenisch-Stadler notation [54] $\mathsf{PK}_\mathcal{R} = [w : \mathcal{R}(s, w) = 1]$ to denote the relation of a proof system proving knowledge of a witness w such that $\mathcal{R}(s, w) = 1$. All components of the condition $\mathcal{R}(s, w) = 1$ that are not explicitly declared as witnesses are assumed to be part of the statement. We define non-interactive zero-knowledge proofs in the random oracle model as follows.

Definition 1. *Let $\mathcal{R}$ be a NP-relation, $\mathcal{L}_\mathcal{R} = \{s : (s, w) \in \mathcal{R}\}$ be the associated NP-language and $\mathcal{H}$ be a global random oracle. A non-interactive zero-knowledge proof of knowledge for relation $\mathcal{R}$ consists of the following two polynomial time algorithms* $(\mathsf{Prove}, \mathsf{Verify})$ *with access to a random oracle $\mathcal{H}$.*

- $\mathsf{Prove}(s, w) \to \pi$*: a probabilistic algorithm that takes as input a statement s and a witness w and outputs a proof π.*
- $\mathsf{Verify}(s, \pi) \to 0/1$*: a deterministic algorithm that takes as input a statement s and a proof π and outputs either 1 accepting the proof or 0 rejecting it.*

The algorithms satisfy the following properties:

- *Completeness: For all $(s, w) \in \mathcal{R}$, it holds that*

$$\Pr\left[\mathsf{Verify}(s, \pi) = 1 : \ \pi \leftarrow \mathsf{Prove}(s, w)\ \right] = 1.$$

1. There are algorithms such as baby-step giant-step [52] or Pollard's Kangaroo [53] that offer better efficiency than the naive approach of testing all exponents in the range individually.

Experiment: $\mathsf{Exp}^{\mathsf{P\text{-}BBS\text{-}SUF}}_{\mathcal{A},L}(\lambda)$

$\mathsf{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}) \leftarrow \mathsf{GGen}(1^\lambda),\ G_0, G_1 \xleftarrow{\$} \mathbb{G}_1^*$
$\{H_i\}_{i\in[L]} \xleftarrow{\$} \mathbb{G}_1^L,\ G_2 \leftarrow \mathbb{G}_2^*,\ z \xleftarrow{\$} \mathbb{Z}_p,\ Z \leftarrow zG_2,\ \mathcal{S} \leftarrow \emptyset$
$(\{m_i^*\}_{i\in[L]}, A^*, e^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Sign}}}(\mathsf{pp}, G_0, G_1, \{H_i\}_{i\in[L]}, G_2, Z)$
if $\mathsf{e}(A^*, eG_2 + Z) \neq \mathsf{e}(G_0 + \sum_{i\in[L]} m_i^* H_i, G_2)$ **then return** 0
if $(\{m_i^*\}_{i\in[L]}, A^*, e^*) \in \mathcal{S}$ **then return** 0 **else return** 1

$\mathcal{O}^{\mathsf{Sign}}(\{m_i\}_{i\in[L]})$:

$e \xleftarrow{\$} \mathbb{Z}_p,\ A \leftarrow \frac{1}{z+e}(G_0 + \sum_{i\in[L]} m_i H_i)$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{(\{m_i\}_{i\in[L]}, A, e)\},$ **return** (A, e)

Figure 1: Security game for strong unforgeability of BBS signatures.

- *Knowledge soundness: For every $\lambda \in \mathbb{N}$, every auxiliary input $\mathsf{aux} \in \{0,1\}^*$ of size polynomial in λ, every PPT adversary $\mathcal{A}$, and every random tape r, there exists a PPT extractor $\mathcal{E}$ with oracle access to $\mathcal{A}$ and control over random oracle $\mathcal{H}$, as well as a negligible function negl, such that*

$$\Pr\left[\begin{array}{l|c|l} \mathsf{Verify}(s, \pi) = 1 & : & (s, \pi) \leftarrow \mathcal{A}(\mathsf{aux}; r) \\ \wedge (s, w) \notin \mathcal{R} & & w \leftarrow \mathcal{E}^{\mathcal{H}, \mathcal{A}}(\mathsf{aux}, r) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

  Since $\mathcal{A}$ is polynomial-time and its input size is polynomial in the security parameter λ, the definition implicitly requires the size of statement s and proof π to be polynomial in λ.
- *Zero-knowledge: There exists a PPT simulator $\mathcal{S}$ controlling random oracle $\mathcal{H}$ such that for every $\lambda \in \mathbb{N}$ and for every $(s, w) \in \mathcal{R}$ of size polynomial in λ, it holds that*

$$\{\mathsf{Prove}(s, w)\} \approx_c \{\mathcal{S}^\mathcal{H}(1^\lambda, s)\}.$$

Remarks. Formally, a proof requires perfect soundness while an argument tolerates a negligible soundness error. However, in this work, we will utilize the term proof for both arguments and proofs. As all the proofs of knowledge, we utilize in our constructions, rely on the random oracle model, we choose to incorporate the random oracle into the formalization in order to ease the definition. One of the proof systems of our construction incorporates a range proof, which we will realize with the Bulletproof++ [24] proof system. This proof system requires a dedicated setup. We formalize zero-knowledge proofs of knowledge with setup in Appendix C.

2.2. BBS unforgeability

We base our construction on BBS signatures, as discussed in the technical overview (Section 1.2). However, in order to efficiently instantiate our zero-knowledge proofs, we make non-black box use of BBS signatures. Therefore, we do not need a formal definition of the abstract primitive instantiated by BBS signatures. Instead, we state Theorem 1.

Theorem 1. *Let* GGen *be a as above. For every* $\lambda \in \mathbb{N}$, $L = L(\lambda)(L \in \mathbb{N} : L > 1)$ *and* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* negl *such that*

$$\Pr[\mathsf{Exp}^{\mathsf{P\text{-}BBS\text{-}SUF}}_{\mathcal{A},L}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{P\text{-}BBS\text{-}SUF}}_{\mathcal{A},L}$ *is as defined in Figure 1.*

Theorem 1 is straightforwardly implied by Theorem 1 in [18].

3. Anonymous Tokens with Betrayability

Analogously to [7], *anonymous tokens with betrayability (ATB)* deter clients from transferring tokens by binding each token to a valuable insurance secret that needs to be known during both token issuance and token redemption. Even though a token vendor might be willing to sell a token, it would not sell the corresponding secret due to its value, which renders transferred tokens useless. While [7] assumed that the token buyer and the vendor cannot communicate during the token redemption, we get rid of this assumption. Instead, we guarantee that a client redeeming a token either knows the corresponding insurance secret or is able to include a trapdoor into a vendor-aided token redemption that allows it to steal the vendor's insurance secret without being detected.

Formally, we define ATB as below. In the following we denote both the token issuer and token verifier as server even though they can be instantiated by different parties. As our definition extends the one of [7], we take parts of their definition (almost) verbatim. We discuss the differences in the notions in Appendix B.

Definition 2. *An* anonymous token with betrayability *scheme is defined as a tuple of the following* PPT *algorithms and protocols:*

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*: a randomized algorithm that takes as input the security parameter* λ *and returns public parameters* pp.
- $\mathsf{S.KeyGen}(\mathsf{pp}) \to (\mathsf{sk}_s, \mathsf{pk}_s)$*: a randomized algorithm that takes as input the public parameters and generates a key pair for the server.*
- $\mathsf{C.KeyGen}(\mathsf{pp}) \to (\mathsf{sk}^{\mathsf{In}}_c, \mathsf{pk}^{\mathsf{In}}_c, \mathsf{sk}^{\mathsf{Id}}_c, \mathsf{pk}^{\mathsf{Id}}_c)$*: a randomized algorithm that takes as input the public parameters and generates two key pairs for the client, one for insurance and one for identification.*
- $\mathsf{C.Query}(\mathsf{pp}, \mathsf{sk}^{\mathsf{In}}_c, \mathsf{pk}^{\mathsf{In}}_c, \mathsf{pk}_s) \to (\mathsf{query}, \mathsf{st})$*: a randomized algorithm constituting the first step of an interactive issuance protocol between client and server. The client uses the public parameters, its insurance keys and the server's public key to generate a token request and saves its state* st *for the token finalization.*
- $\mathsf{S.Issue}(\mathsf{pp}, \mathsf{sk}_s, \mathsf{pk}^{\mathsf{In}}_c, \mathsf{query}) \to \mathsf{resp}/\perp$*: a randomized algorithm constituting the second step of the issuance protocol. The server takes the public parameters, its key pair, the client's public insurance key and the client query as input and produces a response* resp*. If the query is not valid, the algorithm returns* $\perp$*.*
- $\mathsf{C.Final}(\mathsf{pp}, \mathsf{st}, \mathsf{resp}) \to (\sigma, \omega)$*: a deterministic algorithm constituting the third and final step of the issuance protocol. The client takes the public parameters, the previous state and the server's response as input and generates a token* σ *as well as a witness* ω*.*
- $\langle \mathsf{C.Red}(\mathsf{pp}, \sigma, \omega, \mathsf{sk}^{\mathsf{Id}}_c, \mathsf{pk}^{\mathsf{Id}}_c, \mathsf{pk}_s), \mathsf{S.Red}(\mathsf{pp}, \mathsf{pk}_s)\rangle \to 1/0$*: An interactive protocol executed between the client and the server, in which the client, identified by its identification public key* $\mathsf{pk}^{\mathsf{Id}}_c$*, shows a token* σ *to the server and proves that it knows a corresponding witness* ω*. We assume that the protocol is initiated by the client by sending* $(\sigma, \mathsf{pk}^{\mathsf{Id}}_c)$ *to the server.*[2]
  Formally, the computations of each party are modeled via a next message randomized PPT *algorithm* S.NextMsg *resp.* C.NextMsg *that takes an instance specific state of the party and an incoming message as input and produces a new state, an outgoing message, and a flag. If not required, we neglect the flag. The initial state is the input to the protocol. If the flag indicates a protocol termination, i.e., flag = 1, the protocol ends. The output* 0 *or* 1 *is the new state and the final message (if any) is the new message. To simplify the security games, we assume that the intermediate states cannot be* 0 *or* 1 *and that the server sends its output as final message.*
- $\mathsf{KeyV}(\mathsf{pp}, \mathsf{sk}, \mathsf{pk}) \to 1/0$*: a deterministic algorithm that takes as input the public parameters and a key pair and outputs* 1 *iff the key pair is valid.*

An ATB scheme needs to satisfy correctness *(Definition 3),* accountability *(Definition 4),* unlinkability *(Definition 5) and* betrayability *(Definition 6).*

We define the correctness property as follows:

Definition 3 (Correctness)**.** *We call an ATB scheme* correct*, if for all* $\lambda \in \mathbb{N}$, $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{sk}_s, \mathsf{pk}_s) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$, $(\mathsf{sk}^{\mathsf{Id}}_c, \mathsf{pk}^{\mathsf{Id}}_c, \mathsf{sk}^{\mathsf{In}}_c, \mathsf{pk}^{\mathsf{In}}_c) \leftarrow \mathsf{C.KeyGen}(\mathsf{pp})$ *it holds that if* $(\sigma, \omega) \leftarrow \mathsf{C.Final}(\mathsf{st}, \mathsf{resp})$ *for* $(\mathsf{query}, \mathsf{st}) \leftarrow \mathsf{C.Query}(\mathsf{pp}, \mathsf{sk}^{\mathsf{In}}_c, \mathsf{pk}^{\mathsf{In}}_c \mathsf{pk}_s)$ *and* $\mathsf{resp} \leftarrow \mathsf{S.Issue}(\mathsf{pp}, \mathsf{sk}_s, \mathsf{pk}^{\mathsf{In}}_c, \mathsf{query})$ *then*

$$\mathsf{KeyV}(\mathsf{pp}, \mathsf{sk}^{\mathsf{In}}_c, \mathsf{pk}^{\mathsf{In}}_c) = \mathsf{KeyV}(\mathsf{pp}, \mathsf{sk}^{\mathsf{Id}}_c, \mathsf{pk}^{\mathsf{Id}}_c) = 1$$
$$\langle \mathsf{C.Red}(\mathsf{pp}, \sigma, \omega, \mathsf{sk}^{\mathsf{Id}}, \mathsf{pk}^{\mathsf{Id}}, \mathsf{pk}_s), \mathsf{S.Red}(\mathsf{pp}, \mathsf{pk}_s)\rangle = 1.$$

3.1. Accountability

Accountability requires three properties, *one-more unforgeability (OMUF)*, *request unforgeability (RUF)*, and *identity unforgeability (IUF)*. One-more unforgeability states that it is impossible for an adversary to redeem q tokens without making q successful token queries. Request unforgeability states that an adversary cannot request a token under the insurance key of an honest client. Identity unforgeability requires it to be impossible for an adversary to redeem a token under the identity of an honest client. Note that these

2. The protocol does not take any additional states as input or alter the parties' long-time states. If such effects need to be captured, e.g., to prevent tokens from being used multiple times, they need to be captured by the programs surrounding the anonymous token scheme.

properties imply that secret keys of honest parties cannot be leaked. This is due to the fact that any adversary capable of stealing a secret key from an honest party can either forge tokens or request or redeem a token in the name of another honest party. Formally, we define accountability as follows.

Definition 4. *An ATB scheme is* accountable *if it satisfies* one-more unforgeability, token consistency, request unforgeability, *and* identity unforgeability, *i.e., if for every* PPT *adversary* $\mathcal{A}$ *there exists a negligible function* negl *such that* $\mathsf{Pr}[\mathsf{Exp}^{*}_{\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$ *for* $\mathsf{Exp}^{*} \in \{\mathsf{Exp}^{\mathsf{OMUF}}, \mathsf{Exp}^{\mathsf{RUF}}, \mathsf{Exp}^{\mathsf{IUF}}\}$ *where* $\mathsf{Exp}^{\mathsf{OMUF}}, \mathsf{Exp}^{\mathsf{RUF}}$ *and* $\mathsf{Exp}^{\mathsf{IUF}}$ *are as defined in Figure 2.*

3.2. Unlinkability

Intuitively, *unlinkability (UL)* requires that a malicious server is not able to link the issuance of a token to its redemption, even if it controls both the verifier and the issuer. Formally, we define a security game in which the adversary has full control over both the issuer and the verifier. The adversary can initiate issuance sessions, each of which returns a unique identifier, and can later initiate redemption sessions for tokens using the identifiers of the corresponding issuance sessions. We require that the adversary cannot determine whether two tokens—selected based on their identifiers—have been swapped.

Definition 5. *An ATB scheme is* unlinkable *if for every* PPT *adversary* $\mathcal{A}$*, there exists a negligible function* negl *such that* $|\mathsf{Pr}[\mathsf{Exp}^{\mathsf{UL}}_{\mathcal{A}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$ *where* $\mathsf{Exp}^{\mathsf{UL}}_{\mathcal{A}}$ *is defined as in Figure 3.*

3.3. Betrayability

Intuitively, *betrayability (BT)* requires that a client redeeming a bought token, potentially with the help of the token vendor, can *betray* the vendor and steal the insurance secret associated with the token.

Formally, we define a security game with two adversaries, the token vendor and the token buyer. The vendor first specifies a particular token it aims to sell, (σ^*), and the token buyer an identification key pair it aims to redeem the token with, $(\mathsf{sk}^{\mathsf{Id}}_*, \mathsf{pk}^{\mathsf{Id}}_*)$. Afterwards, the adversaries engage in an interactive protocol. Whenever, the adversaries are executed, they also have access to a token request and redemption oracle representing concurrent interaction of the parties with the server. The security game has two modes. In the colluding mode $(b = 0)$, the buyer does not betray the vendor and the adversaries win if the buyer successfully redeems σ^* with identification key $\mathsf{pk}^{\mathsf{Id}}_*$. In the betraying mode $(b = 1)$, we replace the token buyer with an extractor that attempts to steal the insurance secret embedded into σ^* from the vendor. Since the game cannot directly verify whether a given insurance secret belongs to σ^*, we keep track of all the insurance public keys used by the vendor prior to specifying σ^* and consider the extractor to be successful if it extracts an insurance secret corresponding to

Experiment: $\mathsf{Exp}^{\mathsf{OMUF}}_{\mathcal{A}}(\lambda)$

$q_{\mathsf{Rq}}, q_{\mathsf{Rd}} \leftarrow 0,\ \Gamma \leftarrow \emptyset,\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ (\mathsf{sk}_s, \mathsf{pk}_s) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
$\mathcal{A}^{\mathcal{O}^{\mathsf{Rq}}, \mathcal{O}^{\mathsf{Rd}}}(\mathsf{pp}, \mathsf{pk}_s)$, **if** $q_{\mathsf{Rq}} < q_{\mathsf{Rd}}$ **return** 1 **else return** 0

$\underline{\mathcal{O}^{\mathsf{Rq}}(\mathsf{pk}, \mathsf{query}) :}$
$q_{\mathsf{Rq}} \leftarrow q_{\mathsf{Rq}} + 1$, **return** $\mathsf{S.Issue}(\mathsf{pp}, \mathsf{sk}_s, \mathsf{pk}, \mathsf{query})$

$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathsf{sid}, \mathsf{m}) :}$
if $\mathsf{st}_{\mathsf{sid}} \in \{0, 1\}$ **then return** $\perp$
if $\mathsf{st}_{\mathsf{sid}} = \mathbf{null}$ **then** $\mathsf{st}_{\mathsf{sid}} \leftarrow (\mathsf{pp}, \mathsf{pk}_s),\ (\sigma_{\mathsf{sid}}, \cdot) \leftarrow \mathsf{m}$
$(\mathsf{st}, \mathsf{m}', \mathsf{flag}) \leftarrow \mathsf{S.NextMsg}(\mathsf{st}_{\mathsf{sid}}, \mathsf{m})$
if $\mathsf{flag} = \mathsf{st} = 1 \wedge \sigma_{\mathsf{sid}} \notin \Gamma$ **then** $q_{\mathsf{Rd}} \leftarrow q_{\mathsf{Rd}} + 1, \Gamma \leftarrow \Gamma \cup \{\sigma_{\mathsf{sid}}\}$
$\mathsf{st}_{\mathsf{sid}} \leftarrow \mathsf{st}$, **return** m'

Experiment: $\mathsf{Exp}^{\mathsf{RUF}}_{\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{sk}^{\mathsf{In}}_c, \mathsf{pk}^{\mathsf{In}}_c, \mathsf{sk}^{\mathsf{Id}}_c, \mathsf{pk}^{\mathsf{Id}}_c) \leftarrow \mathsf{C.KeyGen}(\mathsf{pp})$
$(\mathsf{query}, \mathsf{sk}_s) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Qu}}, \mathcal{O}^{\mathsf{Fin}}, \mathcal{O}^{\mathsf{Rd}}}(\mathsf{pp}, \mathsf{pk}^{\mathsf{In}}_c, \mathsf{pk}^{\mathsf{Id}}_c)$
$\mathsf{resp} = \mathsf{S.Issue}(\mathsf{pp}, \mathsf{sk}_s, \mathsf{pk}^{\mathsf{In}}_c, \mathsf{query})$
if $\exists i$ s.t. $\mathsf{query} = \mathsf{query}_i \vee \mathsf{resp} = \perp$ **then return** 0
return 1

$\underline{\mathcal{O}^{\mathsf{Qu}}(i, \mathsf{pk}_s):}$
if $\mathsf{query}_i \neq \mathbf{null}$ **then abort**
$(\mathsf{query}_i, \mathsf{st}^Q_i) \leftarrow \mathsf{C.Query}(\mathsf{pp}, \mathsf{sk}^{\mathsf{In}}_c, \mathsf{pk}^{\mathsf{In}}_c, \mathsf{pk}_s),\ \mathsf{pk}_i \leftarrow \mathsf{pk}_s$
return query_i

$\underline{\mathcal{O}^{\mathsf{Fin}}(i, \mathsf{resp}):}$
if $\mathsf{query}_i = \mathbf{null}$ or $\sigma_i \neq \mathbf{null}$ **then abort**
$(\sigma_i, \omega_i) \leftarrow \mathsf{C.Final}(\mathsf{pp}, \mathsf{st}^Q_i, \mathsf{resp}),$ **return** σ_i

$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathsf{sid}, \mathsf{m}) :}$
if $\mathsf{st}^R_{\mathsf{sid}} = \mathbf{null}$ **then**
  $i \leftarrow m,$ **if** $\sigma_i \in \{\mathbf{null}, \perp\}$ **then return** $\perp$
  $\mathsf{st}^R_{\mathsf{sid}} \leftarrow (\mathsf{pp}, \sigma_i, \omega_i, \mathsf{sk}^{\mathsf{Id}}_c, \mathsf{pk}^{\mathsf{Id}}_c, \mathsf{pk}_i),$ **return** $(\sigma_i, \mathsf{pk}^{\mathsf{Id}}_c)$
else $(\mathsf{st}^R_{\mathsf{sid}}, \mathsf{m}') \leftarrow \mathsf{C.NextMsg}(\mathsf{st}^R_{\mathsf{sid}}, \mathsf{m}),$ **return** m'

Experiment: $\mathsf{Exp}^{\mathsf{IUF}}_{\mathcal{A}}(\lambda)$

$\mathsf{res} \leftarrow 0,\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\cdot, \cdot, \mathsf{sk}^{\mathsf{Id}}_c, \mathsf{pk}^{\mathsf{Id}}_c) \leftarrow \mathsf{C.KeyGen}(\mathsf{pp})$
$(\mathsf{pk}_s, \mathsf{st}_{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Qu}}, \mathcal{O}^{\mathsf{Fin}}, \mathcal{O}^{\mathsf{Rd}}}(1, \mathsf{pp}, \mathsf{pk}^{\mathsf{Id}}_c),\ \mathcal{A}^{\mathcal{O}^*}(2, \mathsf{st}_{\mathcal{A}})$
return res

$\underline{\mathcal{O}^{\mathsf{Qu}}(i, \mathsf{pk}^{\mathsf{In}}_c, \mathsf{sk}^{\mathsf{In}}_c, \mathsf{pk}):}$
if $\mathsf{query}_i \neq \mathbf{null} \vee \mathsf{KeyV}(\mathsf{pp}, \mathsf{sk}^{\mathsf{In}}_c, \mathsf{pk}^{\mathsf{In}}_c) = 0$ **then return** $\perp$
$(\mathsf{query}_i, \mathsf{st}^Q_i) \leftarrow \mathsf{C.Query}(\mathsf{pp}, \mathsf{sk}^{\mathsf{In}}_c, \mathsf{pk}^{\mathsf{In}}_c, \mathsf{pk}),\ \mathsf{pk}_i \leftarrow \mathsf{pk}$
return query_i

$\underline{\mathcal{O}^{\mathsf{Fin}}(i, \mathsf{resp}):}$
if $\mathsf{query}_i = \mathbf{null}$ or $\sigma_i \neq \mathbf{null}$ **then abort**
$(\sigma_i, \omega_i) \leftarrow \mathsf{C.Final}(\mathsf{pp}, \mathsf{st}^Q_i, \mathsf{resp}),$ **return** σ_i

$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathsf{sid}, \mathsf{m}) :}$
if $\mathsf{st}^R_{\mathsf{sid}} = \mathbf{null}$ **then**
  $i \leftarrow \mathsf{m},$ **if** $\sigma_i \in \{\mathbf{null}, \perp\}$ **then return** $\perp$
  $\mathsf{st}^R_{\mathsf{sid}} \leftarrow (\mathsf{pp}, \sigma_i, \omega_i, \mathsf{sk}^{\mathsf{Id}}_c, \mathsf{pk}^{\mathsf{Id}}_c, \mathsf{pk}_i),$ **return** $(\sigma_i, \mathsf{pk}^{\mathsf{Id}}_c)$
else $(\mathsf{st}^R_{\mathsf{sid}}, \mathsf{m}') \leftarrow \mathsf{C.NextMsg}(\mathsf{st}^R_{\mathsf{sid}}, \mathsf{m}),$ **return** m'

$\underline{\mathcal{O}^*(\mathsf{m}) :}$
if $\mathsf{st} \in \{0, 1\}$ **then return** $\perp$
if $\mathsf{st} = \mathbf{null}$ **then** $\mathsf{st} \leftarrow (\mathsf{pp}, \mathsf{pk}_s)$, replace pk^* in m by $\mathsf{pk}^{\mathsf{Id}}_c$
$(\mathsf{st}, \mathsf{m}', \mathsf{flag}) \leftarrow \mathsf{S.NextMsg}(\mathsf{st}, \mathsf{m})$
if $\mathsf{flag} = \mathsf{st} = 1$ **then** $\mathsf{res} \leftarrow 1,$ **return** $1,$ **else return** m'

Figure 2: One-more unforgeability, request unforgeability and identity unforgeability game.

any of the vendor's insurance public keys. For betrayability, we require that the extractor's success probability in the betraying mode is at least as high as the adversaries' winning probability in the colluding mode.

Definition 6. *An ATB scheme is* betrayable *if for every pair of* PPT *adversaries $\mathcal{A}_B$ and $\mathcal{A}_V$, there exists an extractor $\mathcal{E}$ with black-box access to $\mathcal{A}_B$, integer $n = n(\lambda)(n \in \mathbb{N})$ and negligible function* negl *such that*

$$\mathsf{Pr}[\mathsf{Exp}^{\mathsf{BT}}_{\mathcal{A}_B,\mathcal{A}_V,\mathcal{E},n,1}(\lambda) = 1] \geq \mathsf{Pr}[\mathsf{Exp}^{\mathsf{BT}}_{\mathcal{A}_B,\mathcal{A}_V,\mathcal{E},n,0}(\lambda) = 1] - \mathsf{negl}(\lambda)$$

where $\mathsf{Exp}^{\mathsf{BT}}_{\mathcal{A}_B,\mathcal{A}_V,\mathcal{E},n,b}$ is defined as in Figure 3.

4. Our ATB scheme

This section presents our construction for anonymous tokens with betrayability. We refer the reader to the technical overview (cf. Section 1.2) for a high-level description of our construction and focus on the technical presentation in the following.

4.1. Zero-Knowledge Proofs

We make use of three different zero-knowledge proof systems. We introduce the proof systems in the following and present instantiations in Appendix D. We slightly abuse the notation and omit statement components that are part of the protocol's public parameters pp when invoking algorithms $\mathsf{PK}_*.\mathsf{Prove}$ and $\mathsf{PK}_*.\mathsf{Verify}$.

All of the proof systems but the range proofs used within the request proof are instantiated with generalized Schnorr proofs [23]. The range proofs are instantiated with the Bulletproof++ proof system [24]. In the evaluation, we also consider instantiations of the range proofs based on Groth16 generalized purpose SNARKs [25].

Request proof. When sending a request, the client has to prove that the request is well-formed, i.e., that the request's blinded commitment T contains the chunk decomposition $\{x_i\}_{i\in[L]}$ of the secret insurance key $\mathsf{sk}^{\mathsf{In}}$ and that all chunks are in the expected range. Formally, we define the request proof relation as

$$\mathsf{PK}_{\mathsf{Rq}} = \mathsf{PK}\left[\begin{array}{l}(\{x_i\}_{i\in[L]}, r, \delta) : \\ \mathsf{pk}^{\mathsf{In}}_c = (\sum_{i\in[L]} 2^{(i-1)\ell} x_i)G_1 \\ T = \delta(G_0 + rH_0 + \sum_{i\in[L]} x_i H_i) \\ x_i \in [0, 2^\ell - 1] \text{ for } i \in [L]\end{array}\right].$$

As we will instantiate the range proofs with the Bulletproof++ system, we require an explicit setup, which fixes the statement components $(p, \mathbb{G}_1, G_0, H_0, L, \ell)$. The setup returns proof parameters rpp that are input to both $\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Prove}$ and $\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Verify}$. We assume $(\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Setup}, \mathsf{PK}_{\mathsf{Rq}}.\mathsf{Prove}, \mathsf{PK}_{\mathsf{Rq}}.\mathsf{Verify})$ to be a zero-knowledge proof of knowledge with setup for relation $\mathsf{PK}_{\mathsf{Rq}}$.

Challenge proof. During redemption, the server sends an encryption key $K = kG_0$ to the client and proves that

Experiment: $\mathsf{Exp}^{\mathsf{UL}}_{\mathcal{A}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ j' \leftarrow 0,\ b \xleftarrow{\$} \{0,1\},\ \Gamma \leftarrow \emptyset$
$(\mathsf{pk}_s, \mathsf{st}^{\mathcal{A}}_0) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KG}}}(0, \mathsf{pp})$
$(\alpha, \beta, \mathsf{st}^{\mathcal{A}}_1) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KG}},\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Fin}},\mathcal{O}^{\mathsf{Rd}}}(1, \mathsf{st}^{\mathcal{A}}_0)$
if $\exists k \in \{\alpha, \beta\}$ s.t. $\sigma_k \in \{\mathbf{null}, \perp\} \vee k \in \Gamma$ **then abort**
if $b = 1$ **then** swap values of $(\sigma_\alpha, \omega_\alpha)$ and $(\sigma_\beta, \omega_\beta)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KG}},\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Fin}},\mathcal{O}^{\mathsf{Rd}}}(2, \mathsf{st}^{\mathcal{A}}_1)$ **if** $b = b'$ **return** 1 **else** 0

$\underline{\mathcal{O}^{\mathsf{KG}}():}$
$j' \leftarrow j' + 1,\ (\mathsf{sk}^{\mathsf{Id}}_{j'}, \mathsf{pk}^{\mathsf{Id}}_{j'}, \mathsf{sk}^{\mathsf{In}}_{j'}, \mathsf{pk}^{\mathsf{In}}_{j'}) \leftarrow \mathsf{C.KeyGen}(\mathsf{pp})$
return $(\mathsf{pk}^{\mathsf{Id}}_{j'}, \mathsf{sk}^{\mathsf{In}}_{j'}, \mathsf{pk}^{\mathsf{In}}_{j'})$

$\underline{\mathcal{O}^{\mathsf{Rq}}(i, j):}$
if $\mathsf{query}_i \neq \mathbf{null}$ **or** $\mathsf{sk}^{\mathsf{In}}_j = \mathbf{null}$ **then abort**
$(\mathsf{query}_i, \mathsf{st}^Q_i) \leftarrow \mathsf{C.Query}(\mathsf{pp}, \mathsf{sk}^{\mathsf{In}}_j, \mathsf{pk}^{\mathsf{In}}_j, \mathsf{pk}_s),\ \mathbf{return}\ \mathsf{query}_i$

$\underline{\mathcal{O}^{\mathsf{Fin}}(i, \mathsf{resp}):}$
if $\mathsf{query}_i = \mathbf{null} \vee \sigma_i \neq \mathbf{null}$ **then return** $\perp$
$(\sigma_i, \omega_i) \leftarrow \mathsf{C.Final}(\mathsf{pp}, \mathsf{st}^Q_i, \mathsf{resp}),\ \mathbf{if}\ \sigma_i = \perp\ \mathbf{return}\ 0\ \mathbf{else}\ 1$

$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathsf{sid}, \mathsf{m}):}$
if $\mathsf{st}^R_{\mathsf{sid}} = \mathbf{null}$ **then**
  $(i, j) \leftarrow \mathsf{m},\ \mathbf{if}\ \sigma_i \in \{\mathbf{null}, \perp\} \vee \mathsf{pk}^{\mathsf{Id}}_j = \mathbf{null}$ **then abort**
  $\Gamma \leftarrow \Gamma \cup \{i\}$
  $\mathsf{st}^R_{\mathsf{sid}} \leftarrow (\mathsf{pp}, \sigma_i, \omega_i, \mathsf{sk}^{\mathsf{Id}}_j, \mathsf{pk}^{\mathsf{Id}}_j, \mathsf{pk}_s),\ \mathbf{return}\ (\sigma_j, \mathsf{pk}^{\mathsf{Id}}_j)$
else $(\mathsf{st}^R_{\mathsf{sid}}, \mathsf{m}') \leftarrow \mathsf{C.NextMsg}(\mathsf{st}^R_{\mathsf{sid}}, \mathsf{m}),\ \mathbf{return}\ \mathsf{m}'$

Experiment: $\mathsf{Exp}^{\mathsf{BT}}_{\mathcal{A}_B,\mathcal{A}_V,\mathcal{E},n,b}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{sk}_s, \mathsf{pk}_s) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
$(\mathcal{K}, \mathcal{T}, \mathsf{m}^V_0) \leftarrow \emptyset^3, \mathsf{st}_1 = \{\mathsf{pp}, n, \mathsf{pk}_s\}$
if $b = 0$ **then** $\tilde{\mathcal{A}}_B \leftarrow \mathcal{A}_B$ **else** $\tilde{\mathcal{A}}_B \leftarrow \mathcal{E}^{\mathcal{A}_B}$
$(\sigma^*, \mathsf{st}^B_{(2,0)}) \leftarrow \mathcal{A}_V^{\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Rd}}(V,\cdot,\cdot)}(1, \mathsf{st}_1),\ \mathcal{K}^* \leftarrow \mathcal{K}$
$(\mathsf{pk}^{\mathsf{Id}}_*, \mathsf{sk}^{\mathsf{Id}}_*, \mathsf{st}^V_{(2,0)}) \leftarrow \tilde{\mathcal{A}}_B^{\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Rd}}(B,\cdot,\cdot)}(1, \sigma^*, \mathsf{st}_1)$
if $\mathsf{KeyV}(\mathsf{pp}, \mathsf{sk}^{\mathsf{Id}}_*, \mathsf{pk}^{\mathsf{Id}}_*) = 0$ **then return** 0
for $i = 1$ **until** $i = n$ **do**
  $(\mathsf{m}^B_i, \mathsf{st}^B_{(2,i)}) \leftarrow \tilde{\mathcal{A}}_B^{\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Rd}}(B,\cdot,\cdot)}(2, i, \mathsf{m}^V_{i-1}, \mathsf{st}^B_{(2,i-1)})$
  $(\mathsf{m}^V_i, \mathsf{st}^V_{(2,i)}) \leftarrow \mathcal{A}_V^{\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Rd}}(V,\cdot,\cdot)}(2, i, \mathsf{m}^B_i, \mathsf{st}^V_{(2,i-1)})$
if $b = 0 \wedge \mathsf{success} = \mathbf{true}$ **then return** 1
if $b = 1$
  $\mathsf{sk}^* \leftarrow \tilde{\mathcal{A}}_B(2, \mathsf{st}^B_{(3,n)})$
  if $\exists \mathsf{pk} \in \mathcal{K}^* : \mathsf{KeyV}(\mathsf{pp}, \mathsf{sk}^*, \mathsf{pk}) = 1$ **return** 1
return 0

$\underline{\mathcal{O}^{\mathsf{Rq}}(\mathsf{pk}, \mathsf{query}):}$
$\mathcal{K} \leftarrow \mathcal{K} \cup \{\mathsf{pk}\}$
return $\mathsf{S.Issue}(\mathsf{pp}, \mathsf{sk}_s, \mathsf{pk}, \mathsf{query})$

$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathcal{P}, \mathsf{sid}, \mathsf{m}):}$
if $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}} = \mathbf{null}$ **then**
  $(\sigma, \mathsf{pk}) \leftarrow \mathsf{m}, \mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}} \leftarrow (\mathsf{pp}, \mathsf{sk}_s)$
  if $\mathcal{P} = B \wedge (\sigma, \mathsf{pk}) = (\sigma^*, \mathsf{pk}^{\mathsf{Id}}_*)$ **then** $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathsf{sid}\}$
$(\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}}, \mathsf{m}', \mathsf{flag}) \leftarrow \mathsf{S.NextMsg}(\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}}, \mathsf{m})$
if $\mathcal{P} = B \wedge \mathsf{flag} = \mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}} = 1 \wedge \mathsf{sid} \in \mathcal{T}$ **then** $\mathsf{success} \leftarrow \mathbf{true}$
return m'

Figure 3: Unlinkability and betrayability game.

it either knows $k = \mathsf{dlog}_{G_0}(K)$ or $\mathsf{sk}^{\mathsf{Id}} = \mathsf{dlog}_{G_1}(\mathsf{pk}^{\mathsf{Id}})$. The first condition proves that encryption key K is useless as encryptions are created with respect to G_1. The latter condition allows the holder of $\mathsf{sk}^{\mathsf{Id}}$ to replace the useless key K with a useful K^* and forge a valid proof. We define the proof relation

$$\mathsf{PK}_{\mathsf{Ch}} = \mathsf{PK}[(d) : K = dG_0 \vee \mathsf{pk}_c^{\mathsf{Id}} = dG_1],$$

where d represents either the decryption key k or the client's secret identification key $\mathsf{sk}^{\mathsf{Id}}$. We assume $(\mathsf{PK}_{\mathsf{Ch}}.\mathsf{Prove}, \mathsf{PK}_{\mathsf{Ch}}.\mathsf{Verify})$ to be a zero-knowledge proof of knowledge for relation $\mathsf{PK}_{\mathsf{Ch}}$.

Redemption proof. Via the redemption proof, the client proves that it knows the witness ω for the presented token σ, the secret key $y = \mathsf{sk}^{\mathsf{Id}}$ corresponding to the presented public identification key $Y = \mathsf{pk}^{\mathsf{Id}}$ and that the created ElGamal ciphertexts $\{(A_i, B_i)\}_{i\in[L]}$ encrypt the chunks $\{x_i\}_{i\in[L]}$ that have been committed to within σ. As the client does not know the secret BBS signing key z, we cannot directly proof knowledge of the witnesses $(z, r, e, \{x_i\}_{i\in[L]})$ such that $\sigma = \frac{1}{z+e}(G_0 + rH_0 + \sum_{i\in[L]} x_i H_i)$. Instead, we require the client to send $\sigma' = G_0 + rH_0 - e\sigma + \sum_{i\in[L]} x_i H_i$ and proof knowledge of the witnesses $(r, e, \{x_i\}_{i\in[L]})$ required to compute σ'. Note that if σ' is computed as supposed (which is proven by the zero-knowledge proof) and σ is valid it needs to hold that $\sigma' = z\sigma$ which can be verified by the server by checking whether $\mathsf{e}(\sigma, Z) = \mathsf{e}(\sigma', G_2)$. Therefore, it suffices to use the following proof system:

$$\mathsf{PK}_{\mathsf{Rd}} = \mathsf{PK}\left[\begin{array}{l}(y, r, e, \{r_i, x_i\}_{i\in[L]}) : \\ Y = yG_1 \\ \sigma' = G_0 + rH_0 - e\sigma + \sum_{i\in[L]} x_i H_i \\ A_i = r_i G_1, B_i = r_i K + x_i G_1 \text{ for } i \in [L]\end{array}\right].$$

We assume $(\mathsf{PK}_{\mathsf{Rd}}.\mathsf{Prove}, \mathsf{PK}_{\mathsf{Rd}}.\mathsf{Verify})$ to be a zero-knowledge proof of knowledge for relation $\mathsf{PK}_{\mathsf{Rd}}$.

4.2. The construction

Our construction for anonymous tokens with betrayability is specified in Figure 4. In order to betray a token vendor during the token redemption, the buyer replaces the received encryption key K with $K^* = k^* G_1$ for a randomly sampled but known k^*. As the replacement of K invalidates the zero-knowledge proof π_{Ch}, the buyer also replaces π_{Ch} by $\pi^*_{\mathsf{Ch}} = \mathsf{PK}_{\mathsf{Ch}}.\mathsf{Prove}((K^*, \mathsf{pk}_c^{\mathsf{Id}}), \mathsf{sk}_c^{\mathsf{Id}})$. Even though the buyer does not know $\mathsf{dlog}_{G_0}(K^*)$ it does know $\mathsf{dlog}_{G_1}(\mathsf{pk}_c^{\mathsf{Id}})$ and, therefore, can create a valid zero-knowledge proof indistinguishable from the one created by the server. If the vendor (or some jointly executed protocol) returns the ciphertexts required for the token redemption, the buyer verifies the zero-knowledge proof and if successful uses k^* to decrypt the ciphertexts.

To facilitate the recovery of the insurance secret while employing exponential ElGamal for encryption, we split the insurance secret $\mathsf{sk}^{\mathsf{In}}$ into chunks $x_1, \ldots x_L \in [0, 2^\ell - 1]$, which are encrypted and decrypted individually. The betraying buyer recovers the secret chunks x_i from the decryptions

Construction 1: BBS-based ATB scheme

The scheme is parameterized with a complexity bound $\mathcal{C}$ describing a trade-off between protocol efficiency and runtime of betraying buyers. It is necessary that computations of complexity $\mathcal{C}$ are feasible for reasonably equipped clients.

Setup(1^λ):

1) Compute $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T) \leftarrow \mathsf{GGen}(1^\lambda)$, sample $G_0, G_1 \xleftarrow{\$} \mathbb{G}_1^*$, $\mathbb{G}_2 \xleftarrow{\$} \mathbb{G}_2^*$ and $H_0, H_1, \ldots, H_L \xleftarrow{\$} \mathbb{G}_1$, select ℓ such that $\mathsf{dlog}_{G_1}(G)[0, 2^\ell - 1]$ can be computed with complexity of at most $\mathcal{C}$ for any $G \in \mathbb{G}_1$, and set $L \leftarrow \lceil \frac{|p|}{\ell} \rceil$.
2) Compute $\mathsf{rpp} = \mathsf{PK}_{\mathsf{Rq}}.\mathsf{Setup}(p, \mathbb{G}_1, G_0, H_0, L, \ell)$.
3) Return $\mathsf{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_0, G_1, G_2, \{H_i\}_{i\in[0,L]}, L, \ell, \mathsf{rpp})$.

C.KeyGen(pp):

4) Sample $x, y \xleftarrow{\$} \mathbb{Z}_p$, compute $X = xG_1, Y = yG_1$, and return $(\mathsf{sk}_c^{\mathsf{In}}, \mathsf{pk}_c^{\mathsf{In}}, \mathsf{sk}_c^{\mathsf{Id}}, \mathsf{pk}_c^{\mathsf{Id}}) = (x, X, y, Y)$.

S.KeyGen(pp):

5) Sample $z \xleftarrow{\$} \mathbb{Z}_p$, compute $Z = zG_2$, and return $(\mathsf{sk}_s, \mathsf{pk}_s) = (z, Z)$.

C.Query(pp, $\mathsf{sk}_c^{\mathsf{In}}$, $\mathsf{pk}_c^{\mathsf{In}}$, pk_s):

6) Split the bit representation of $\mathsf{sk}_c^{\mathsf{In}}$ into chunks $x_1, \ldots, x_L \in [0, 2^\ell - 1]$ such that $\mathsf{sk}_c^{\mathsf{In}} = \sum_{i\in[L]} 2^{(i-1)\ell} x_i$ holds without modulo reduction.
7) Sample $r, \delta \xleftarrow{\$} \mathbb{Z}_p$
8) Compute $C = G_0 + rH_0 + \sum_{i\in[L]} x_i H_i$ and $T = \delta C$
9) Compute $\pi_{\mathsf{Rq}} = \mathsf{PK}_{\mathsf{Rq}}.\mathsf{Prove}(\mathsf{rpp}, (\mathsf{pk}_c^{\mathsf{In}}, T), (\{x_i\}_{i\in[L]}, r, \delta))$.
10) Return $(\mathsf{query}, \mathsf{st}) = ((T, \pi_{\mathsf{Rq}}), (\mathsf{pk}_s, r, \delta, \{x_i\}_{i\in[L]}, T))$.

S.Issue(pp, sk_s, $\mathsf{pk}_c^{\mathsf{In}}$, query):

11) If $\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Verify}(\mathsf{rpp}, (\mathsf{pk}_c^{\mathsf{In}}, T), \pi_{\mathsf{Rq}}) = 0$ return $\perp$.
12) Sample $e \xleftarrow{\$} \mathbb{Z}_p$ and compute $S = \frac{1}{\mathsf{sk}_s + e} T$.
13) Return $\mathsf{resp} = (e, S)$.

C.Final(pp, st, resp):

14) If $\mathsf{e}(S, \mathsf{pk}_s + eG_2) \neq \mathsf{e}(T, G_2)$ return $\perp$.
15) Otherwise, return $(\sigma, \omega) = (\frac{1}{\delta} S, (r, e, \{x_i\}_{i\in[L]}))$.

⟨C.Red(pp, σ, ω, $\mathsf{sk}_c^{\mathsf{Id}}$, $\mathsf{pk}_c^{\mathsf{Id}}$, pk_s), S.Red(pp, pk_s)⟩:

16) The protocol is initiated by C sending $(\mathsf{pk}_c^{\mathsf{Id}}, \sigma)$.
17) S samples $k \xleftarrow{\$} \mathbb{Z}_p$, computes $K = kG_0$ and $\pi_{\mathsf{Ch}} = \mathsf{PK}_{\mathsf{Ch}}.\mathsf{Prove}((K, \mathsf{pk}_c^{\mathsf{Id}}), k)$, and sends (K, π_{Ch}) to C.
18) If $\mathsf{PK}_{\mathsf{Ch}}.\mathsf{Verify}((K, \mathsf{pk}_c^{\mathsf{Id}}), \pi_{\mathsf{Ch}}) = 0$ or $K = 0$ then C outputs 0.
19) C samples $r_1, \ldots, r_L \xleftarrow{\$} \mathbb{Z}_p$ and computes
    - $A_i = r_i G_1$ and $B_i = r_i K + x_i G_1$ for $i \in [L]$,
    - $\sigma' = G_0 + rH_0 - e\sigma + \sum_{i\in[L]} x_i H_i$ and
    - $\pi_{\mathsf{Rd}} = \mathsf{PK}_{\mathsf{Rd}}.\mathsf{Prove}((\mathsf{pk}_c^{\mathsf{Id}}, K, \sigma, \sigma', \{A_i, B_i\}_{i\in[L]}), (\mathsf{sk}_c^{\mathsf{Id}}, r, e, \{r_i, x_i\}_{i\in[L]}))$.
20) C sends $(\pi_{\mathsf{Rd}}, \sigma', \{A_i, B_i\}_{i\in[L]})$ to S and outputs 1.
21) S outputs and sends 1 if $\mathsf{e}(\sigma, Z) = \mathsf{e}(\sigma', G_2)$ and $\mathsf{PK}_{\mathsf{Rd}}.\mathsf{Verify}((\mathsf{pk}_c^{\mathsf{Id}}, K, \sigma, \sigma', \{A_i, B_i\}_{i\in[L]}), \pi_{\mathsf{Rd}}) = 1$ and 0 otherwise.

KeyV(pp, sk, pk):

22) If $\mathsf{pk} = \mathsf{sk} G_1$ return 1. Otherwise, return 0.

Figure 4: Our ATB scheme.

$D_i = x_i G_1$ by brute-forcing the discrete logarithm of D_i over the interval $[0, 2^\ell - 1]$, i.e., $x_i = \mathsf{dlog}_{G_1}(D_i)[0, 2^\ell - 1]$. Although there exists algorithms accelerating the computation of the discrete logarithm [52], [53], the complexity of that computation still grows exponentially in ℓ. This results in an inherent trade-off between the protocol's efficiency and the betraying buyer's computational complexity. Larger values of ℓ improve protocol efficiency, since they reduce L and thereby lower the complexity of the protocol. However, increasing ℓ also raises the cost of brute-forcing $x_i = \mathsf{dlog}_{G_1}(D_i)[0, 2^\ell - 1]$. We configure this trade-off by parametrizing the protocol with a complexity bound $\mathcal{C}$ and choosing ℓ in the setup such that $\mathsf{dlog}_{G_1}(G)[0, 2^\ell - 1]$ can be computed with complexity below $\mathcal{C}$. Naturally, we require that computations of complexity $\mathcal{C}$ are feasible for reasonably equipped clients. As the complexity bound $\mathcal{C}$ is a fixed protocol parameter, it follows that $L = \lceil \frac{|p|}{\ell} \rceil$ scales with the security parameter λ (which is equal to the bit-size of p). The overall complexity of our protocol, therefore, scales quadratically in the security parameter.

We note that the betrayal is not completely for free, as the buyer will not necessarily be able to redeem the bought token if it decides to betray the vendor. The created redemption proof is with respect to a replaced encryption key K^* and, therefore, will not be accepted by the server. Furthermore, the buyer needs the complete token witness $(r, e, \{x_i\}_{i\in[L]})$ to redeem the token itself while it steals only $\{x_i\}_{i\in[L]}$. Nevertheless, we remark that our constructions can straightforwardly be extended with the same techniques used for the extraction of $\{x_i\}_{i\in[L]}$ to allow the extraction of r and e.

5. Security

In this section, we show security of the ATB construction presented in Section 4.

Theorem 2. *The construction described in Construction 1 (Figure 4) is a secure ATB scheme according to Definition 2.*

We prove the theorem by proving the necessary security properties individually.

5.1. Correctness

Theorem 3. *The construction described in Construction 1 satisfies the* correctness *property defined in Definition 3.*

From the definition of algorithms Setup, $\mathsf{S.KeyGen}$, $\mathsf{S.Issue}$, $\mathsf{C.KeyGen}$, $\mathsf{C.Final}$, and $\mathsf{C.Query}$ and the completeness of the request zero-knowledge proof system it follows that

$$\mathsf{pk}_c^{\mathsf{In}} = \mathsf{sk}_c^{\mathsf{In}} G_1,\ \mathsf{pk}_c^{\mathsf{Id}} = \mathsf{sk}_c^{\mathsf{Id}} G_1,\ \mathsf{pk}_s = \mathsf{sk}_s G_2$$
$$\mathsf{sk}_c^{\mathsf{In}} = \sum_{i\in[L]} 2^{(i-1)\ell} x_i,\ x_i \in [0, 2^\ell - 1] \text{ for } i \in [L]$$
$$\sigma = \frac{1}{\mathsf{sk}_s + e}(G_0 + rH_0 + \sum_{i\in[L]} x_i H_i)$$

for each $(\sigma, \omega) = (\sigma, (r, e, \{x_i\}_{i\in[L]}))$ produced by $\mathsf{C.Final}$. We can conclude that $\mathsf{KeyV}(\mathsf{pp}, \mathsf{sk}_c^{\mathsf{In}}, \mathsf{pk}_c^{\mathsf{In}}) = \mathsf{KeyV}(\mathsf{pp}, \mathsf{sk}_c^{\mathsf{Id}}, \mathsf{pk}_c^{\mathsf{Id}}) = 1$. As σ is as defined as above and $\sigma' = G_0 + rH_0 - e\sigma + \sum_{i\in[L]} x_i H_i$ it needs to hold that $\sigma' = \mathsf{sk}_s \sigma$ and, hence, $\mathsf{e}(\sigma, \mathsf{pk}_s) = \mathsf{e}(\sigma', G_2)$. Finally, as all proof components of $\mathsf{PK}_{\mathsf{Rd}}$ are computed as defined by the proof relation, it follows from the proof system's completeness property that $\mathsf{PK}_{\mathsf{Rd}}.\mathsf{Verify}((\mathsf{pk}_c^{\mathsf{Id}}, K, \sigma, \sigma', \{A_i, B_i\}_{i\in[L]}), \pi_{\mathsf{Rd}}) = 1$. Therefore, we can conclude that $\langle \mathsf{C.Red}(\mathsf{pp}, \sigma, \omega, \mathsf{sk}^{\mathsf{Id}}, \mathsf{pk}^{\mathsf{Id}}, \mathsf{pk}_s), \mathsf{S.Red}(\mathsf{pp}, \mathsf{pk}_s)\rangle = 1$.

5.2. Accountability

Theorem 4. *The scheme described in Construction 1 satisfies the accountability property defined in Definition 4.*

To show accountability, we prove that for any PPT adversary $\mathcal{A}$, the probability of winning the one-more unforgeability game, the request unforgeability game, and the identity unforgeability game is at most negligible in the security parameter. We provide proof sketches in the following and defer the full proof to Appendix F.

One-more unforgeability. The goal of our proof is to simplify the security game such that the resulting game can only be won by an adversary providing a BBS forgery. In two game hops, we adapt the request oracle to employ the request proof's extractor to extract the witnesses $(\{x_i\}_{i\in[L]}, r, \delta)$ from the blinded request (T, π_{Rq}) and adapt the redemption oracle to employ the redemption proof's extractor to extract the redemption proof's witnesses $(y, r, e, \{r_i, x_i\}_{i\in[L]})$ from the final redemption message $(\pi_{\mathsf{Rd}}, \sigma', \{A_i, B_i\}_{i\in[L]})$. Computational indistinguishability between the adapted and the original game follows from the knowledge soundness property of the request proof system and the redemption proof system.

In the resulting game, we can show that the adversary's success probability is negligible by reducing to the BBS unforgeability game (cf. Figure 1). First note that we can use the BBS signing oracle to answer token requests in the reduction without knowing the secret issuance key sk_s. As the experiment extracts the witnesses $(\{x_i\}_{i\in[L]}, r, \delta)$ within the request oracle, it can request signatures (σ, e) on $(\{x_i\}_{i\in[L]} \cup \{r\})$ from the BBS signing oracle and create the token responses as $(S = \delta\sigma, e)$. Next note that the adversary has to successfully redeem at least $q_{\mathsf{Rq}} + 1$ distinct tokens while only querying q_{Rq} tokens. Therefore, it needs to hold that at least one of the redeemed tokens has not been issued by the request oracle and, hence, constitutes a valid BBS forgery (together with the extracted witnesses). As no PPT adversary can win the BBS unforgeability game with probability higher than negligible (cf. Theorem 1) and the difference in the success probability in the original and the simplified game is negligible, we conclude that every PPT adversary wins the original one-more unforgeability game with at most negligible probability.

Request unforgeability. The goal of the request unforgeability proof is to gradually simplify the security game until

we can show that the adversary's success probability is at most negligible by reducing to the discrete logarithm assumption.

In the first four game hops, we adapt the redemption oracle to (i) use the redemption proof's simulator instead of the proof system's witnesses to create redemption proofs, (ii) employ the challenge proof's extractor to extract k such that $K = kG_0 \vee \mathsf{pk}_j^{\mathsf{Id}} = kG_1$, (iii) randomly sample the ciphertexts if $K = kG_0$, and (iv) ignore the case that $\mathsf{pk}_j^{\mathsf{Id}} = kG_1$. All changes introduce only a negligible difference in the adversary's success probability; (i) due to the zero-knowledge property of the redemption proof, (ii) due to the knowledge soundness property of the challenge proof, (iii) due to DDH assumption in $\mathbb{G}_1$ (which also implies semantic security of the ElGamal ciphertexts), and (iv) due to the discrete logarithm assumption in $\mathbb{G}_1$ making it impossible for an adversary to find $\mathsf{dlog}_{G_1}(\mathsf{pk}_j^{\mathsf{Id}})$.

Next, we adapt the request oracle to create request proofs with the proof system's simulator instead of the witnesses. Computational indistinguishability follows from the zero-knowledge property of the request proof system. Subsequently, we randomly sample $C_i \xleftarrow{\$} \mathbb{G}_1$ instead of computing $C_i \leftarrow G_0 + r_i H_0 + \sum_{j\in[L]} x_j H_j$. This change does not affect the adversary's view and, hence, the output distribution of the experiment as r_i acts as a one-time pad ensuring a uniform distribution of C_i. Finally, we adapt the experiment's main thread to employ the request proof's extractor to extract $(\{x_j\}_{j\in[L]}, \cdot, \cdot)$ such that $\mathsf{pk}_c^{\mathsf{In}} = (\sum_{j\in[L]} 2^{(j-1)\ell} x_j)G_1$. Computational indistinguishability follows from the knowledge soundness property of the request proof system.

In the final game, we can show that any PPT adversary has at most a negligible success probability via a reduction to the discrete logarithm assumption. As the adversary has to provide $\{x_j\}_{j\in[L]}$ such that $\mathsf{pk}_c^{\mathsf{In}} = (\sum_{j\in[L]} x_j 2^{(j-1)\ell})G_1$ without getting any extra information about $\mathsf{pk}_c^{\mathsf{In}}$ beyond $\mathsf{pk}_c^{\mathsf{In}}$ itself (all oracles are simulated with randomly sampled dummy-values), a winning adversary can be used to compute $\mathsf{dlog}_{G_1}(\mathsf{pk}_c^{\mathsf{In}})$. By assumption it holds that no PPT adversary can compute the discrete logarithm in $\mathbb{G}_1$ with probability higher than negligible, which allows us to conclude that every PPT adversary wins the simplified request unforgeability game with at most negligible probability. As there is at most a negligible gap in the success probability between the final and the original game, we conclude that our construction satisfies request unforgeability.

Identity unforgeability. Analogously to our request unforgeability proof, the identity unforgeability proof gradually simplifies the identity unforgeability security game to enable a reduction to the discrete logarithm assumption.

First, we adopt the request unforgeability proof's first simplification of the redemption oracle and use the redemption proof's simulator instead of the proof system's witnesses to create redemption proofs within $\mathcal{O}^{\mathsf{Rd}}$. Next, we use the redemption proof's extractor to extract the witnesses of the redemption proof submitted to $\mathcal{O}^*$. Computational indistinguishability follows from the knowledge soundness property of the redemption proof system.

In the resulting game we can show that any PPT adversary has at most a negligible success probability by reducing to the discrete logarithm assumption. As the adversary has to provide a valid zero-knowledge proof proving knowledge of sk^* such that $\mathsf{pk}_c^{\mathsf{Id}} = \mathsf{sk}^* G_1$ without getting any information about $\mathsf{pk}_c^{\mathsf{Id}}$ beyond $\mathsf{pk}_c^{\mathsf{Id}}$ itself a winning adversary can be used to compute $\mathsf{dlog}_{G_1}(\mathsf{pk}_c^{\mathsf{Id}})$ – the discrete logarithm of $\mathsf{pk}_c^{\mathsf{Id}}$ needs to be contained in the witnesses extracted from the redemption proof. As there is at most a negligible gap in the success probability between the simplified and the original identity unforgeability game and the adversary wins in the simplified game with at most negligible probability due to the discrete logarithm assumption, we conclude that our construction satisfies identity unforgeability.

5.3. Unlinkability

Theorem 5. *The scheme described in Construction 1 satisfies the* unlinkability *property defined in Definition 5.*

We provide a proof sketch in the following and defer the full proof to Appendix G. The goal of our unlinkability proof is to gradually simplify the unlinkability security game until the game no longer uses the challenge bit b. If the game does not use bit b, the adversary cannot do better than guessing.

Analogously to the request unforgeability proof we simplify the oracles, i.e., we employ the redemption proof's simulator to create redemption proofs, we use the challenge proof's extractor to extract k such that $K = kG_0$ or $\mathsf{pk}_j^{\mathsf{Id}} = kG_1$, we create ciphertexts of randomly sampled dummy insurance key chunks if $K = kG_0$, we ignore the case that $\mathsf{pk}_j^{\mathsf{Id}} = kG_1$, we employ the request proof's simulator to create request proofs, and finally sample all commitments randomly, i.e., $C_i \xleftarrow{\$} \mathbb{G}_1$.

In the resulting experiment, we can discard the storage of token witnesses from the experiment as they are no longer used after being stored. Subsequently, we resample the selected tokens σ_α and σ_β randomly from $\mathbb{G}_1$ after potentially swapping them. This change does not affect the view of the adversary and, hence, the experiment's output distribution because in both, the new and the previous experiment, all tokens are uniform random elements in $\mathbb{G}_1$ and the associated T_i values are distributed independently from the tokens due to the random masks δ_i. Finally, we get rid of the swap of the tokens, which does not influence the experiment anymore because the selected tokens are overwritten after potentially being swapped. This change yields a final experiment, in which the challenge bit b, that needs to be guessed, is not used and, hence, is guessed with probability of exactly $\frac{1}{2}$. As all game hops introduce at most a negligible difference in the success probability, we conclude that $|\mathsf{Pr}[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{UL}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda)$ for every PPT adversary $\mathcal{A}$.

5.4. Betrayability

We state the following theorem:

Theorem 6. *The scheme described in Construction 1 satisfies the* betrayability *property defined in Definition 6.*

To show betrayability, we define the following extractor $\mathcal{E}$ with access to the token buying adversary $\mathcal{A}_B$ and oracles $\mathcal{O}^{\mathsf{Rq}}$ and $\mathcal{O}^{\mathsf{Rd}}$. We denote the extractor as if it would have an internal storage even though it is required to store its internal state within the state returned by its execution. Formally, all stored variables are appended to the returned state and removed before being passed to the internally executed adversary. We provide a proof sketch in the following and defer the complete proof to Appendix H.

1) Define $\mathcal{O}^{\mathsf{Rq}}_{\mathsf{ext}}$ to answer queries $(\mathsf{pk}, \mathsf{query})$ by returning the output of $\mathcal{O}^{\mathsf{Rq}}(\mathsf{pk}, \mathsf{query})$.
2) Define $\mathcal{O}^{\mathsf{Rd}}_{\mathsf{ext}}$ to answer queries $(B, \mathsf{sid}, \mathsf{m})$ as follows:
   a) If $\mathsf{st}^{(\mathcal{S},B)}_{\mathsf{sid}}$ is not defined, and $\mathsf{m} = (\sigma^*, \mathsf{pk}^{\mathsf{Id}}_*)$, then store $k_{\mathsf{sid}} \xleftarrow{\$} \mathbb{Z}_p$, compute $K_{\mathsf{sid}} = k_{\mathsf{sid}} G_1$ and $\pi_{\mathsf{Ch}} = \mathsf{PK}_{\mathsf{Ch}}.\mathsf{Prove}((K_{\mathsf{sid}}, \mathsf{pk}^{\mathsf{Id}}_*), \mathsf{sk}^{\mathsf{Id}}_*)$, add sid to $\mathcal{T}^*$, store $\mathsf{st}^{(\mathcal{S},B)} = (k_{\mathsf{sid}}, K_{\mathsf{sid}}, \sigma^*, \mathsf{pk}^{\mathsf{Id}}_*)$ and respond with $(K_{\mathsf{sid}}, \pi_{\mathsf{Ch}})$.
   b) If $\mathsf{st}^{(\mathcal{S},B)}_{\mathsf{sid}} \in \{0,1\}$ then return $\emptyset$.
   c) If $\mathsf{sid} \in \mathcal{T}^*$, then parse $(\pi_{\mathsf{Rd}}, \sigma', \{A_i, B_i\}_{i\in[L]}) = \mathsf{m}$.
      If $\mathsf{e}(\sigma^*, \mathsf{pk}_s) \neq \mathsf{e}(\sigma', G_2)$ or $\mathsf{PK}_{\mathsf{Rd}}.\mathsf{Verify}((\mathsf{pk}^{\mathsf{Id}}_*, K_{\mathsf{sid}}, \sigma^*, \sigma', \{A_i, B_i\}_{i\in[L]}), \pi_{\mathsf{Rd}}) = 0$ then set $\mathsf{st}^{(\mathcal{S},B)}_{\mathsf{sid}} = 0$ and return 0.
      If x^* is undefined then compute $D_i = B_i - kA_i$, $x_i = \mathsf{dlog}_{G_1}(D_i)[0; 2^\ell - 1]$. If $\nexists i \in [L]$ such that $x_i = \perp$ then define $x^* = \sum_{i\in[L]} 2^{(i-1)\ell} x_i$.
      Set $\mathsf{st}^{(\mathcal{S},B)}_{\mathsf{sid}} = 1$ and return 1.
   d) Otherwise, return $\mathcal{O}^{\mathsf{Rd}}(B, \mathsf{sid}, \mathsf{m})$.
3) On input $(1, \sigma^*, \{\mathsf{pp}, n, \mathsf{pk}_s\})$, execute $(\mathsf{pk}^{\mathsf{Id}}_*, \mathsf{sk}^{\mathsf{Id}}_*, \mathsf{st}^B_{(2,0)}) \leftarrow \mathcal{A}_B^{\mathcal{O}^{\mathsf{Rq}}_{\mathsf{ext}}, \mathcal{O}^{\mathsf{Rd}}_{\mathsf{ext}}}(1, \{\mathsf{pp}, n, \mathsf{pk}_s\})$, store $\mathsf{pp}, \mathsf{pk}_s, \mathsf{pk}^{\mathsf{Id}}_*, \mathsf{sk}^{\mathsf{Id}}_*, \sigma^*, \mathcal{T}^* \leftarrow \emptyset$ and return $(\mathsf{pk}^{\mathsf{Id}}_*, \mathsf{sk}^{\mathsf{Id}}_*, \mathsf{st}^B_{(2,0)})$.
4) On input $(2, i, \mathsf{m}, \mathsf{st})$, execute $(\mathsf{m}', \mathsf{st}') \leftarrow \mathcal{A}_B^{\mathcal{O}^{\mathsf{Rq}}_{\mathsf{ext}}, \mathcal{O}^{\mathsf{Rd}}_{\mathsf{ext}}(B,\cdot,\cdot)}(2, i, \mathsf{m}, \mathsf{st})$ and return $(\mathsf{m}', \mathsf{st}')$.
5) On input $(3, \mathsf{st}^B_{(3,n)})$ return x^* if defied or $\perp$ otherwise.

The aim of the betrayability proof is to gradually simplify the betrayability security game until the game's only dependence on b is the win condition and the win condition for $b = 1$ is contained in the win condition for $b = 0$.

First, we inline the actions of the extractor $\mathcal{E}$ as defined above into the security game. Next, we make the whole redemption oracle independent from coin b. To do so, we use the challenge proof's simulator instead of the proof system's witnesses to create challenge proofs and adapt the actions inherent to the $(b = 1)$-execution to the $(b = 0)$-execution and vice versa. In particular, we also include a trapdoor into the challenge $(K \leftarrow kG_1)$ and attempt the decryption of the received ciphertexts to x^* if $(b = 0) \wedge \mathcal{P} = B \wedge (\sigma, \mathsf{pk}) = (\sigma^*, \mathsf{pk}^{\mathsf{Id}}_*)$. In the other direction, we define the $\mathsf{success}$ variable if $(b = 1)$ analogously to how it is defined in the $(b = 0)$-case. The simulation of the challenge proof introduces only a negligible difference to the output distribution due to the zero-knowledge property of the challenge proof system. The other changes do not influence the output distribution, as K is still distributed uniformly random and the subsequent steps do not influence the experiment ($\mathsf{success}$ is not used in the $(b = 1)$-case and x^* is not used in the $(b = 0)$-case).

In the next step, we use the request proof's extractor to extract the witnesses of the request proof and abort if the extraction fails. This step introduces only a negligible difference in the output distribution due to the knowledge soundness property of the request proof system. With the extracted witnesses, we compute the resulting token σ and store the token together with the witnesses in a set Γ as long as σ^* (or $\mathcal{K}^*$) has not been defined, yet.

Subsequently, we use the redemption proof's extractor to extract the witnesses of the redemption proof and abort if (i) the extraction fails or (ii) σ^* has not been issued by the request oracle (which can be checked via Γ). This step introduces only a negligible difference in the output distribution due to (i) the knowledge soundness property of the redemption proof system and (ii) the unforgeability property of the BBS signatures. To see the latter, note that the successful extraction of the redemption proof provides valid witnesses $\omega' = (\cdot, r', e', \{\cdot, x'_i\}_{i\in[L]})$ such that (σ^*, ω') constitutes a valid BBS signature. If σ^* has not been issued by the oracle, the tuple (σ^*, ω') needs to be a BBS forgery.

Next, we check if the token-witness pairs $(\sigma^*, \omega^* = (e^*, r^*, \{x^*_i\}_{i\in[L]}))$ stored in Γ match the witnesses ω' extracted from the redemption proof after the extraction of ω'. If this is not the case, we abort the experiment. Here, we show that the change introduces only a negligible difference in the output distribution via a reduction to the discrete logarithm assumption. In particular, we show that an adversary that is capable of computing two distinct but valid witnesses (ω^*, ω') for the same token σ^* can be used to compute the discrete logarithm in $\mathbb{G}_1$.

In a next step, we compute x^* based on ω^* instead of the ciphertexts included in the redemption message. This change does not influence the output distribution. To see this, note that $\omega' = \omega^*$ (when considering only the relevant components of ω') and $x'_i = x^*_i \in [0, 2^\ell - 1]$ for $i \in [L]$. It follows that the computation of $\mathsf{dlog}_{G_1}(D_i)[0; 2^\ell - 1]$ succeeds with probability 1. Consequently, it needs to hold that both ways of computing x^*, based on the ciphertexts or based on ω^*, yield the same x^*.

Finally, we modify the win condition of the $(b = 0)$-case to include the win condition of the $(b = 1)$-case, i.e., $x^*G_1 \in \mathcal{K}^*$. This change does not influence the output distribution. Note that whenever $\mathsf{success}$ is set to $\mathbf{true}$ (the only win condition in the previous experiment), the experiment also defines x^* (based on ω^*). Further, observe that for every token-witness tuple in Γ, it needs to hold that a public key $\mathsf{pk} = xG_1$ is added to $\mathcal{K}^*$. Therefore, it holds that if $\mathsf{success} = \mathbf{true}$ then $x^*G_1 \in \mathcal{K}^*$, which shows that the additional win condition does not influence the output distribution.

As all game hops introduce at most a negligible difference in the success probability and the success probability of the $(b = 1)$-case in the final experiment is clearly higher or equal to the one of the $(b = 0)$-case (the win condition of the $(b = 1)$-case is included into the one of the $(b = 0)$-case), we can conclude that $\mathsf{Pr}[\mathsf{Exp}^{\mathsf{BT}}_{\mathcal{A}_B, \mathcal{A}_V, \mathcal{E}, n, 1}(\lambda) = 1] \geq \mathsf{Pr}[\mathsf{Exp}^{\mathsf{BT}}_{\mathcal{A}_B, \mathcal{A}_V, \mathcal{E}, n, 0}(\lambda)] = 1 - \mathsf{negl}(\lambda)$.

6. Evaluation

We implemented several versions of our construction, executed benchmarks and determined the communication complexity. For comparison, we have also implemented and benchmarked the publicly verifiable non-transferable anonymous token (PV-NTAT) construction of [7].

The versions of our construction differentiate in the instantiation of the range proofs and the phase, in which the range proofs are executed. We either execute the range proofs in the issuance phase as part of the client's request proof or in the redemption phase as part of the client's redemption proof. In the latter case, we also postpone the chunk decomposition of the insurance secret to the redemption phase. For the instantiation of the range proofs, we consider the Bulletproofs++ proof system [24] and Groth16 general-purpose zk-SNARKs [25]. Bulletproofs++ have the advantage that they plug-well into the generalized Schnorr proofs, we use for the remaining relations of the respective proof system, and support the efficient aggregation of proofs, which is desirable as we require one range proof for every insurance key chunk. Zk-SNARKS, on the other hand, have a constant proof size and, hence, are highly efficient in terms of communication. However, the proof creation and verification is much less efficient than the creation of Bulletproofs++ proofs. The Groth16 SNARK proofs fit well into our setting as there is one dedicated verifier for the range proofs, the server. Therefore, we do not need to deal with the toxic waste produced during the generation of the SNARK's public parameters. In the SNARK-based variants, we realize the whole proof, not only the part concerning the range relation, with one SNARK.

We call the variants Req-BP++, Red-BP++, Req-SNARK, and Red-SNARK, based on the phase that includes the range proofs and the instantiation of the range proofs. The Req-BP++ variant is the one presented in Construction 1 in Section 4.

6.1. Benchmarks

Experimental setup. Our implementation and benchmarks are written in Rust (compiled with rustc 1.86.0-nightly (124cc9219)). We execute all our benchmarks on a Laptop with an Intel(R) Core(TM) i7-1360P @ 2.20 GHz processor and 16GB of RAM. We repeat all benchmarks 100 times and report the average result.

In our benchmarks, we measure the execution time of contract and client in both the request phase and the redemption phase. We bench the parties individually and add up non-continuously executed parts if necessary. The client's execution during the request phase includes the computation of C.Query and C.Final and the server's execution the computation of S.Issue. During the redemption phase, the client executes Step 18 - Step 20 while the server executes Step 17 and Step 21 of Construction 1. Our benchmarks do not consider the overhead of message transmission. Network affects can be incorporated based on the respective network conditions and the reported communication complexity.

All the variants are based on the *BLS12-377* curve. The SNARKs are computed over the *BW6-761* curve, as this curve's scalar field is equal to the base field of the *BLS12-377* curve. We utilize Arkworks' crates for curve operations and SNARK computation and the *bulletproofs-plus-plus* crate for the Bulletproof++ range proofs.

We repeat all benchmarks for $L \in \{4, 8, 16, 32\}$. By focusing on powers of 2 when considering 256-bit Scalar elements, we can ensure that the secret insurance key is split into equally sized chunks, which is convenient for implementation. We do not consider $L = 2$ as this would require a buyer betraying a token vendor to brute force 128-bit scalars, which is not feasible. Further, we note that an increase of the security parameter (which is equivalent to an increase of group order p) requires to also increase L in order to ensure that the chunks can still be brute-forced in reasonable time. The complexity of our protocol, therefore, scales quadratically with the security parameter.

Experimental Results. We report the results of the benchmarks of all our constructions in Figure 5, split in the runtime of the client and the server in the request phase and the redemption phase. In addition, we directly compare all variants in Figure 6.

The individual results show that the major overhead is due to the creation of the range proofs. In the Req-BP++ and Req-SNARK variant, the range proof is created by the client in the request phase. In the Red-BP++ and Red-SNARK construction, the range proof is created by the client in the redemption phase. In all variants, the phase including the creation of the range proof dominates the overall execution time of the variant. Further, the comparison highlights that the runtime of the variants based on Bulletproofs++ is much lower, which is expected to compensate the additional communication complexity in most network settings. When comparing the SNARK-based variants it is clearly evident that the Req-SNARK variant is much more efficient than the Red-SNARK.

Our implementation of the PV-NTAT construction of [7] yields 3.9 ms and 1 ms for the client in the request and redemption phase and 1.1 ms and 2.7 ms for the server in the request and redemption phase. This totals to a total execution time below 9 ms. Even though our extension comes with significant overhead, we still assess the variants based on Bulletproof++ with a total execution time below 50 ms for $L = 4$ to be feasible for real-world deployments.

6.2. Communication Complexity

Let L be the number of chunks, $|G|$ be the size of (encoded) group elements of $\mathbb{G}_1$, $|p|$ the size of field elements, $\mathsf{RP}[|p|, L]$ the size of a Bulletproof++ range proof for L elements in range $[0, 2^{\frac{|p|}{L}} - 1]$, and $|\mathsf{SNARK}|$ the size of a Groth16 SNARK. Based on these parameters, we report the communication complexity of the variants of our construction and compare them to the public verifiable protocol version of [7] in Table 1.

In the Bulletproof++ system instantiated over the BLS12-377 curve, four range proofs for interval $[0, 2^{64} - 1]$

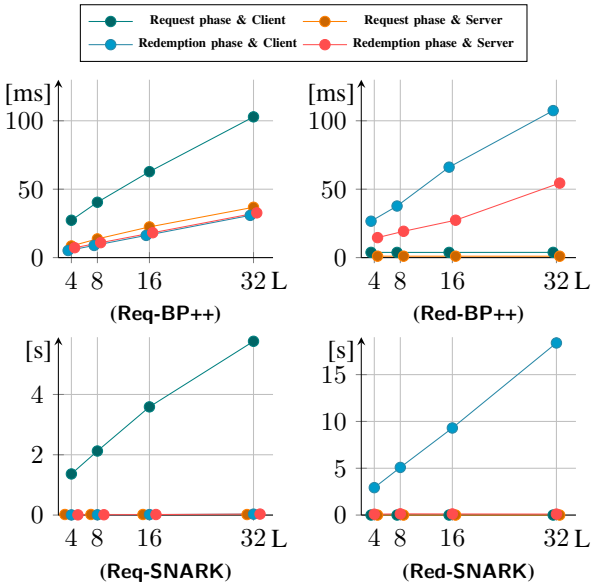

Figure 5: Execution cost of client and server in the request and redemption phase of our constructions dependent on the number of chunks L. To differentiate overlapping data points, we slightly shift them in x-direction, even though they still belong to $L = 4, 8, 16, 32$.

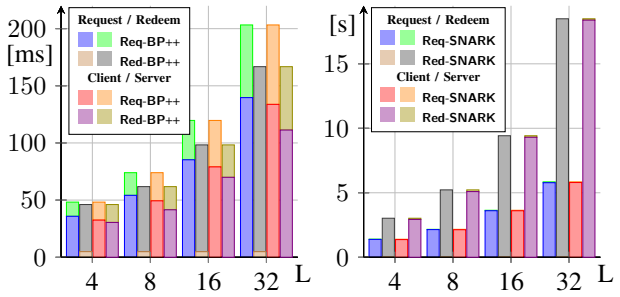

Figure 6: Comparison of the total runtime of all variants of our construction. The stacked bars differentiate between the request and redemption phase (the two left bars) and the client and server overhead (the two right bars).

require $\mathsf{RP}[|256|, 4] = 736$ bytes $(12|G| + 5|p|)$. One Groth16 proof instantiated over the BW6-761 curve has $|\mathsf{SNARK}| = 288$ bytes (two elements of the curve pairing's first group and one of its second group).

The overhead of our constructions when compared to [7] is mainly due to the chunk decomposition of the secret insurance key that allows decryption of exponential ElGamal ciphertexts of the insurance secret chunks, which in turn, enables us to perform efficient zero-knowledge proofs over ElGamal ciphertexts.

TABLE 1: Communication complexity of our Bulletproof++-based and SNARK-based instantiation compared to prior work on non-transferable tokens.

	Issuance		Redemption	
	Client	Server	Client	Server
PV-NTAT [7][3]	$2\|G\| + 4\|p\|$	$\|G\| + \|p\|$	$2\|G\| + 5\|p\|$	$\|p\|$
Req-BP++	$(2+L)\|G\|$ $+(3+2L)\|p\|$ $+\mathsf{RP}[p, L]$	$\|G\| + \|p\|$	$(3+2L)\|G\|$ $+(4+2L)\|p\|$	$3\|G\| + \|p\|$
Red-BP++	$2\|G\| + 4\|p\|$	$\|G\| + \|p\|$	$(3+2L)\|G\|$ $+(4+2L)\|p\|$ $+\mathsf{RP}[p, L]$	$3\|G\| + \|p\|$
Req-SNARK	$2\|G\|$ $+\|\mathsf{SNARK}\|$	$\|G\| + \|p\|$	$(3+2L)\|G\|$ $+(4+2L)\|p\|$	$3\|G\| + \|p\|$
Red-SNARK	$2\|G\| + 4\|p\|$	$\|G\| + \|p\|$	$(3+2L)\|G\|$ $+\|\mathsf{SNARK}\|$	$3\|G\| + \|p\|$

References

[1] A. Davidson, I. Goldberg, N. Sullivan, G. Tankersley, and F. Valsorda, "Privacy pass: Bypassing internet challenges anonymously," *Proc. Priv. Enhancing Technol.*, vol. 2018, no. 3, pp. 164–180, 2018. [Online]. Available: https://doi.org/10.1515/popets-2018-0026

[2] B. Kreuter, T. Lepoint, M. Orrù, and M. Raykova, "Anonymous tokens with private metadata bit," in *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, ser. Lecture Notes in Computer Science, D. Micciancio and T. Ristenpart, Eds., vol. 12170. Springer, 2020, pp. 308–336. [Online]. Available: https://doi.org/10.1007/978-3-030-56784-2_11

[3] F. Benhamouda, T. Lepoint, M. Orrù, and M. Raykova, "Publicly verifiable anonymous tokens with private metadata bit," *IACR Cryptol. ePrint Arch.*, p. 4, 2022. [Online]. Available: https://eprint.iacr.org/2022/004

[4] T. Silde and M. Strand, "Anonymous tokens with public metadata and applications to private contact tracing," in *Financial Cryptography and Data Security - 26th International Conference, FC 2022, Grenada, May 2-6, 2022, Revised Selected Papers*, ser. Lecture Notes in Computer Science, I. Eyal and J. A. Garay, Eds., vol. 13411. Springer, 2022, pp. 179–199. [Online]. Available: https://doi.org/10.1007/978-3-031-18283-9_9

[5] F. Benhamouda, M. Raykova, and K. Seth, "Anonymous counting tokens," in *Advances in Cryptology - ASIACRYPT 2023 - 29th International Conference on the Theory and Application of Cryptology and Information Security, Guangzhou, China, December 4-8, 2023, Proceedings, Part II*, ser. Lecture Notes in Computer Science, J. Guo and R. Steinfeld, Eds., vol. 14439. Springer, 2023, pp. 245–278. [Online]. Available: https://doi.org/10.1007/978-981-99-8724-5_8

[6] M. Chase, F. B. Durak, and S. 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*, ser. Lecture Notes in Computer Science, H. Handschuh and A. Lysyanskaya, Eds., vol. 14082. Springer, 2023, pp. 418–449. [Online]. Available: https://doi.org/10.1007/978-3-031-38545-2_14

3. We add the transmission of the insurance public key during the request phase and the transmission of the token during the redemption phase. Further, we assume the hash-based commitment utilized in [7] to receive a field element in $\mathbb{Z}_p$ as random nonce and return a field element in $\mathbb{Z}_p$ as commitment.

[7] F. B. Durak, L. Marco, A. Talayhan, and S. Vaudenay, "Non-transferable anonymous tokens by secret binding," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, B. Luo, X. Liao, J. Xu, E. Kirda, and D. Lie, Eds. ACM, 2024, pp. 2460–2474. [Online]. Available: https://doi.org/10.1145/3658644.3670338

[8] I. Karantaidou, O. Renawi, F. Baldimtsi, N. Kamarinakis, J. Katz, and J. Loss, "Blind multisignatures for anonymous tokens with decentralized issuance," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS 2024, Salt Lake City, UT, USA, October 14-18, 2024*, B. Luo, X. Liao, J. Xu, E. Kirda, and D. Lie, Eds. ACM, 2024, pp. 1508–1522. [Online]. Available: https://doi.org/10.1145/3658644.3690364

[9] M. Orrù, S. Tessaro, G. Zaverucha, and C. Zhu, "Oblivious issuance of proofs," in *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part IX*, ser. Lecture Notes in Computer Science, L. Reyzin and D. Stebila, Eds., vol. 14928. Springer, 2024, pp. 254–287. [Online]. Available: https://doi.org/10.1007/978-3-031-68400-5_8

[10] G. Amjad, K. Yeo, and M. Yung, "RSA blind signatures with public metadata," *Proc. Priv. Enhancing Technol.*, vol. 2025, no. 1, pp. 37–57, 2025. [Online]. Available: https://doi.org/10.56553/popets-2025-0004

[11] F. Baldimtsi, L. Hanzlik, Q. Nguyen, and A. Yadav, "Non-interactive anonymous tokens with private metadata bit," *IACR Cryptol. ePrint Arch.*, p. 430, 2025. [Online]. Available: https://eprint.iacr.org/2025/430

[12] Cloudflare. (2025) Privacy pass. [Online]. Available: https://developers.cloudflare.com/waf/tools/privacy-pass/

[13] Google. (2024) Private state tokens. [Online]. Available: https://developers.google.com/privacy-sandbox/protections/private-state-tokens

[14] IETF. (2025) Privacy pass (privacypass). [Online]. Available: https://datatracker.ietf.org/wg/privacypass/about/

[15] C. Dwork, J. B. Lotspiech, and M. Naor, "Digital signets: Self-enforcing protection of digital information (preliminary version)," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, G. L. Miller, Ed. ACM, 1996, pp. 489–498. [Online]. Available: https://doi.org/10.1145/237814.237997

[16] O. Goldreich, B. Pfitzmann, and R. L. Rivest, "Self-delegation with controlled propagation - or - what if you lose your laptop," in *Advances in Cryptology - CRYPTO '98, 18th Annual International Cryptology Conference, Santa Barbara, California, USA, August 23-27, 1998, Proceedings*, ser. Lecture Notes in Computer Science, H. Krawczyk, Ed., vol. 1462. Springer, 1998, pp. 153–168. [Online]. Available: https://doi.org/10.1007/BFb0055726

[17] A. Lysyanskaya, R. L. Rivest, A. Sahai, and S. Wolf, "Pseudonym systems," in *Selected Areas in Cryptography, 6th Annual International Workshop, SAC'99, Kingston, Ontario, Canada, August 9-10, 1999, Proceedings*, ser. Lecture Notes in Computer Science, H. M. Heys and C. M. Adams, Eds., vol. 1758. Springer, 1999, pp. 184–199. [Online]. Available: https://doi.org/10.1007/3-540-46513-8_14

[18] S. Tessaro and C. Zhu, "Revisiting BBS signatures," in *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, ser. Lecture Notes in Computer Science, C. Hazay and M. Stam, Eds., vol. 14008. Springer, 2023, pp. 691–721. [Online]. Available: https://doi.org/10.1007/978-3-031-30589-4_24

[19] A. Barki, S. Brunet, N. Desmoulins, and J. Traoré, "Improved algebraic macs and practical keyed-verification anonymous credentials," in *Selected Areas in Cryptography - SAC 2016 - 23rd International Conference, St. John's, NL, Canada, August 10-12, 2016, Revised Selected Papers*, ser. Lecture Notes in Computer Science, R. Avanzi and H. M. Heys, Eds., vol. 10532. Springer, 2016, pp. 360–380. [Online]. Available: https://doi.org/10.1007/978-3-319-69453-5_20

[20] D. Boneh, X. Boyen, and H. Shacham, "Short group signatures," in *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, ser. Lecture Notes in Computer Science, M. K. Franklin, Ed., vol. 3152. Springer, 2004, pp. 41–55. [Online]. Available: https://doi.org/10.1007/978-3-540-28628-8_3

[21] M. H. Au, W. Susilo, and Y. Mu, "Constant-size dynamic k-taa," in *Security and Cryptography for Networks, 5th International Conference, SCN 2006, Maiori, Italy, September 6-8, 2006, Proceedings*, ser. Lecture Notes in Computer Science, R. D. Prisco and M. Yung, Eds., vol. 4116. Springer, 2006, pp. 111–125. [Online]. Available: https://doi.org/10.1007/11832072_8

[22] T. E. Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Advances in Cryptology, Proceedings of CRYPTO '84, Santa Barbara, California, USA, August 19-22, 1984, Proceedings*, ser. Lecture Notes in Computer Science, G. R. Blakley and D. Chaum, Eds., vol. 196. Springer, 1984, pp. 10–18. [Online]. Available: https://doi.org/10.1007/3-540-39568-7_2

[23] C. Schnorr, "Efficient identification and signatures for smart cards," in *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, ser. Lecture Notes in Computer Science, G. Brassard, Ed., vol. 435. Springer, 1989, pp. 239–252. [Online]. Available: https://doi.org/10.1007/0-387-34805-0_22

[24] L. Eagen, S. Kanjalkar, T. Ruffing, and J. Nick, "Bulletproofs++: Next generation confidential transactions via reciprocal set membership arguments," in *Advances in Cryptology - EUROCRYPT 2024 - 43rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zurich, Switzerland, May 26-30, 2024, Proceedings, Part V*, ser. Lecture Notes in Computer Science, M. Joye and G. Leander, Eds., vol. 14655. Springer, 2024, pp. 249–279. [Online]. Available: https://doi.org/10.1007/978-3-031-58740-5_9

[25] J. Groth, "On the size of pairing-based non-interactive arguments," in *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, ser. Lecture Notes in Computer Science, M. Fischlin and J. Coron, Eds., vol. 9666. Springer, 2016, pp. 305–326. [Online]. Available: https://doi.org/10.1007/978-3-662-49896-5_11

[26] D. Chaum, "Blind signatures for untraceable payments," in *Advances in Cryptology: Proceedings of CRYPTO '82, Santa Barbara, California, USA, August 23-25, 1982*, D. Chaum, R. L. Rivest, and A. T. Sherman, Eds. Plenum Press, New York, 1982, pp. 199–203. [Online]. Available: https://doi.org/10.1007/978-1-4757-0602-4_18

[27] ——, "Blind signature system," in *Advances in Cryptology, Proceedings of CRYPTO '83, Santa Barbara, California, USA, August 21-24, 1983*, D. Chaum, Ed. Plenum Press, New York, 1983, p. 153.

[28] ——, "Security without identification: Transaction systems to make big brother obsolete," *Commun. ACM*, vol. 28, no. 10, pp. 1030–1044, 1985. [Online]. Available: https://doi.org/10.1145/4372.4373

[29] L. Chen, "Access with pseudonyms," in *Cryptography: Policy and Algorithms, International Conference, Brisbane, Queensland, Australia, July 3-5, 1995, Proceedings*, ser. Lecture Notes in Computer Science, E. Dawson and J. D. Golic, Eds., vol. 1029. Springer, 1995, pp. 232–243. [Online]. Available: https://doi.org/10.1007/BFb0032362

[30] J. Camenisch and A. Lysyanskaya, "An efficient system for non-transferable anonymous credentials with optional anonymity revocation," in *Advances in Cryptology - EUROCRYPT 2001, International Conference on the Theory and Application of Cryptographic Techniques, Innsbruck, Austria, May 6-10, 2001, Proceeding*, ser. Lecture Notes in Computer Science, B. Pfitzmann, Ed., vol. 2045. Springer, 2001, pp. 93–118. [Online]. Available: https://doi.org/10.1007/3-540-44987-6_7

[31] ——, "Signature schemes and anonymous credentials from bilinear maps," in *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, ser. Lecture Notes in Computer Science, M. K. Franklin, Ed., vol. 3152. Springer, 2004, pp. 56–72. [Online]. Available: https://doi.org/10.1007/978-3-540-28628-8_4

[32] J. Camenisch, "Anonymous credentials: Opportunities and challenges," in *Security and Privacy in Dynamic Environments, Proceedings of the IFIP TC-11 21st International Information Security Conference (SEC 2006), 22-24 May 2006, Karlstad, Sweden*, ser. IFIP, S. Fischer-Hübner, K. Rannenberg, L. Yngström, and S. Lindskog, Eds., vol. 201. Springer, 2006, p. 460. [Online]. Available: https://doi.org/10.1007/0-387-33406-8_42

[33] J. Camenisch, M. Dubovitskaya, K. Haralambiev, and M. Kohlweiss, "Composable and modular anonymous credentials: Definitions and practical constructions," in *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, ser. Lecture Notes in Computer Science, T. Iwata and J. H. Cheon, Eds., vol. 9453. Springer, 2015, pp. 262–288. [Online]. Available: https://doi.org/10.1007/978-3-662-48800-3_11

[34] J. Camenisch, S. Krenn, A. Lehmann, G. L. Mikkelsen, G. Neven, and M. Ø. Pedersen, "Formal treatment of privacy-enhancing credential systems," in *Selected Areas in Cryptography - SAC 2015 - 22nd International Conference, Sackville, NB, Canada, August 12-14, 2015, Revised Selected Papers*, ser. Lecture Notes in Computer Science, O. Dunkelman and L. Keliher, Eds., vol. 9566. Springer, 2015, pp. 3–24. [Online]. Available: https://doi.org/10.1007/978-3-319-31301-6_1

[35] J. Blömer, J. Bobolz, D. Diemert, and F. Eidens, "Updatable anonymous credentials and applications to incentive systems," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, L. Cavallaro, J. Kinder, X. Wang, and J. Katz, Eds. ACM, 2019, pp. 1671–1685. [Online]. Available: https://doi.org/10.1145/3319535.3354223

[36] A. Sonnino, M. Al-Bassam, S. Bano, S. Meiklejohn, and G. Danezis, "Coconut: Threshold issuance selective disclosure credentials with applications to distributed ledgers," in *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019. [Online]. Available: https://www.ndss-symposium.org/ndss-paper/coconut-threshold-issuance-selective-disclosure-credentials-with-applications-to-distributed-ledgers/

[37] Z. Yu, M. H. Au, and R. Yang, "Accountable anonymous credentials," in *Advances in Cyber Security: Principles, Techniques, and Applications*, K. Li, X. Chen, and W. Susilo, Eds. Springer, 2019, pp. 49–68. [Online]. Available: https://doi.org/10.1007/978-981-13-1483-4_3

[38] L. Hanzlik and D. Slamanig, "With a little help from my friends: Constructing practical anonymous credentials," in *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, Y. Kim, J. Kim, G. Vigna, and E. Shi, Eds. ACM, 2021, pp. 2004–2023. [Online]. Available: https://doi.org/10.1145/3460120.3484582

[39] C. Paquin and G. Zaverucha. (2023) U-prove cryptographic specification v1.1 (revision 5). [Online]. Available: https://github.com/microsoft/uprove-node-reference/blob/main/doc/U-Prove%20Cryptographic%20Specification%20V1.1%20Revision%205.pdf

[40] J. Camenisch, M. Drijvers, and A. Lehmann, "Anonymous attestation using the strong diffie hellman assumption revisited," in *Trust and Trustworthy Computing - 9th International Conference, TRUST 2016, Vienna, Austria, August 29-30, 2016, Proceedings*, ser. Lecture Notes in Computer Science, M. Franz and P. Papadimitratos, Eds., vol. 9824. Springer, 2016, pp. 1–20. [Online]. Available: https://doi.org/10.1007/978-3-319-45572-3_1

[41] J. Doerner, Y. Kondi, E. Lee, A. Shelat, and L. Tyner, "Threshold BBS+ signatures for distributed anonymous credential issuance," in *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, May 21-25, 2023*. IEEE, 2023, pp. 773–789. [Online]. Available: https://doi.org/10.1109/SP46215.2023.10179470

[42] S. Faust, C. Hazay, D. Kretzler, L. Rometsch, and B. Schlosser, "Non-interactive threshold BBS+ from pseudorandom correlations," *IACR Cryptol. ePrint Arch.*, p. 1076, 2023. [Online]. Available: https://eprint.iacr.org/2023/1076

[43] S. Jarecki, A. Kiayias, and H. Krawczyk, "Round-optimal password-protected secret sharing and T-PAKE in the password-only model," in *Advances in Cryptology - ASIACRYPT 2014 - 20th International Conference on the Theory and Application of Cryptology and Information Security, Kaoshiung, Taiwan, R.O.C., December 7-11, 2014, Proceedings, Part II*, ser. Lecture Notes in Computer Science, P. Sarkar and T. Iwata, Eds., vol. 8874. Springer, 2014, pp. 233–253. [Online]. Available: https://doi.org/10.1007/978-3-662-45608-8_13

[44] G. Fuchsbauer, A. Plouviez, and Y. Seurin, "Blind schnorr signatures and signed elgamal encryption in the algebraic group model," in *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part II*, ser. Lecture Notes in Computer Science, A. Canteaut and Y. Ishai, Eds., vol. 12106. Springer, 2020, pp. 63–95. [Online]. Available: https://doi.org/10.1007/978-3-030-45724-2_3

[45] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Advances in Cryptology - ASIACRYPT 2001, 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, December 9-13, 2001, Proceedings*, ser. Lecture Notes in Computer Science, C. Boyd, Ed., vol. 2248. Springer, 2001, pp. 514–532. [Online]. Available: https://doi.org/10.1007/3-540-45682-1_30

[46] A. Boldyreva, "Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme," in *Public Key Cryptography - PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography, Miami, FL, USA, January 6-8, 2003, Proceedings*, ser. Lecture Notes in Computer Science, Y. Desmedt, Ed., vol. 2567. Springer, 2003, pp. 31–46. [Online]. Available: https://doi.org/10.1007/3-540-36288-6_3

[47] D. Boneh, M. Drijvers, and G. Neven, "Compact multi-signatures for smaller blockchains," in *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, ser. Lecture Notes in Computer Science, T. Peyrin and S. D. Galbraith, Eds., vol. 11273. Springer, 2018, pp. 435–464. [Online]. Available: https://doi.org/10.1007/978-3-030-03329-3_15

[48] E. C. Crites, C. Komlo, M. Maller, S. Tessaro, and C. Zhu, "Snowblind: A threshold blind signature in pairing-free groups," in *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part I*, ser. Lecture Notes in Computer Science, H. Handschuh and A. Lysyanskaya, Eds., vol. 14081. Springer, 2023, pp. 710–742. [Online]. Available: https://doi.org/10.1007/978-3-031-38557-5_23

[49] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978. [Online]. Available: https://doi.org/10.1145/359340.359342

[50] L. Hanzlik, "Non-interactive blind signatures for random messages," in *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, ser. Lecture Notes in Computer Science, C. Hazay and M. Stam, Eds., vol. 14008. Springer, 2023, pp. 722–752. [Online]. Available: https://doi.org/10.1007/978-3-031-30589-4_25

[51] D. Boneh and M. K. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*, ser. Lecture Notes in Computer Science, J. Kilian, Ed., vol. 2139. Springer, 2001, pp. 213–229. [Online]. Available: https://doi.org/10.1007/3-540-44647-8_13

[52] D. Shanks, "Class number, a theory of factorization, and genera," in *Proceedings of Symposia in Pure Mathematics*, vol. 20. American Mathematical Society, 1971, pp. 415–440.

[53] J. M. Pollard, "Monte carlo methods for index computation (mod p)," *Mathematics of computation*, vol. 32, no. 143, pp. 918–924, 1978.

[54] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups (extended abstract)," in *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, ser. Lecture Notes in Computer Science, B. S. K. Jr., Ed., vol. 1294. Springer, 1997, pp. 410–424. [Online]. Available: https://doi.org/10.1007/BFb0052252

[55] Full version. [Online]. Available: https://anonymous.4open.science/r/AnonymousTokensWithBetrayability

Appendix A. Necessity of Identification Keys

In this section, we show that our betrayability notion cannot be achieved in the model of [7], in which the redemption of a token requires nothing but the token and its witness including the corresponding insurance secret.

First, consider a scenario Exp_1, in which a token vendor V with a valid token supports an honest buyer B (that has no private state and does not try to betray V) in the redemption of a token at a server S that constitutes both issuer and verifier. Client V only participates in the process if B does nothing but forwarding messages between V and S and responds to the messages received by B exactly as it would respond to messages by S in an honest token redemption. If the vendor detects that a message received from B does not originate from S, it aborts the redemption.

We observe that in Exp_1, the redemption of the token is always successful. To see this, consider a second scenario Exp_2, in which a client C^* internally executes both V and B and engages in a redemption with server S. Client C^* sends the messages produced by B to S and the messages received by S to B. It is obvious that the redemption in Exp_1 is successful if it is in Exp_2. We simplify Exp_2 to Exp_2', in which C^* directly forwards messages between V and S without making use of B. Clearly, this does not affect the outcome of the scenario, as the messages produced by C^* are exactly the same in Exp_2 and Exp_2'. As Exp_2' constitutes an honestly executed token redemption, the correctness of the anonymous token scheme guarantees that the redemption in Exp_2' is successful. Therefore, we can conclude that the redemption in Exp_1 is successful as well.

Next, we observe that the betrayability property requires that a buyer B' is able to betray any vendor V' in any joint redemption protocol without being detected and that the betrayal is successful with probability equal or higher than the success probability of the joint redemption protocol (minus negligible). Therefore, it is guaranteed that in an experiment Exp_1', in which B is replaced by a betraying buyer B', B' obtains the insurance secret with non-negligible probability without being detected.

Finally, we consider a scenario Exp_3, in which an honest client C redeems a token from a malicious server S'. The malicious server internally executes an honest server S and a betraying buyer B'. It sends the messages produced by B' to C, inputs the message received from C to B' and forwards messages between B' and S. Note that client C is not able to differentiate if it talks to B' or S as otherwise V would be able to do so in experiment Exp_1'. As B' extracts the client's secret with non-negligible probability in Exp_1', it holds that S' can do so in scenario Exp_3. However, if S' is able to extract an honest client's insurance secret, S' can be used to attack the request unforgeability property of the token scheme by first extracting the insurance secret with S' and then forging requests with the extracted secret. Similarly, S' can be used to break the unlinkability property.

Consequently, we can conclude that in the model of [7], betrayability and request unforgeability as well as betrayability and unlinkability contradict each other. In our model, we circumvent this impossibility by providing an identification key to the buyer that needs to be known for the token redemption and the insertion of the trapdoor into the redemption. This makes it impossible for a malicious server to emulate a betraying buyer.

Appendix B. Comparison of our security notion to [7]

In our definition, we inherit the one-more unforgeability, request unforgeability and unlinkability property from [7] and adapt them to our definition. Furthermore, we replace the non-transferability and uniqueness property of [7] by our betrayability property.

A first difference to [7] is that we introduce identification keys to identify clients during redemption and enable token buyers to incorporate a trapdoor into a vendor-aided token redemption. We include the identification keys in the redemption protocols of the inherited security games. Furthermore, we need to ensure that a client does not redeem a token under the identity of another client. Therefore, we introduce the identity unforgeability property.

Secondly, we observed a weakness in the one-more unforgeability notion of [7]. In particular, the adversary is required to provide token-witnesses pairs that can be redeemed in an honest execution of the redemption protocol in order to win the experiment. However, we assess it to be important that the tokens cannot even be redeemed in a malicious execution of the redemption protocol. Therefore, we adapt the definition accordingly. In our definition, the adversary has access to a query and a redemption oracle and wins if it successfully redeems more distinct tokens than it queried. We note that this definition is much closer to the original one-more unforgeability definition of anonymous tokens introduced by [1].

Third, we explicitly consider a collusion between a token vendor and a token buyer during token redemption. Instead

of requiring that the client redeeming a token knows the corresponding insurance secret as done in [7], we require that the buyer either knows the secret or is able to extract it from the vendor, if the vendor-aided redemption would be successful. Therefore, we have to extend the security game to incorporate two different types of adversaries and allow one of them to betray the other. As our notion of betrayability implies token uniqueness, we do no longer need this property.

Finally, we allow the adversary in the unlinkability game to select the public and private key of the server.

Appendix C. Zero-Knowledge Proofs with Setup

For the range proofs utilized in our construction, we require a zero-knowledge proof with a dedicated setup creating public parameters that are necessary for the proof creation and verification. The setup fixes some parts of the proof statements, e.g, the underlying algebraic group. We formalize zero-knowledge proofs with setup in the random oracle model as follows.

Definition 7. *Let $\mathcal{R}$ be a NP-relation, $\mathcal{L}_{\mathcal{R}} = \{(s_0, s_1) : ((s_0, s_1), w) \in \mathcal{R}\}$ be the associated NP-language, and $\mathcal{H}$ be a global random oracle. A non-interactive zero-knowledge proof of knowledge with setup for relation $\mathcal{R}$ consists of the following three polynomial time algorithms* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *with access to a random oracle $\mathcal{H}$.*

- $\mathsf{Setup}(s_0) \to \mathsf{pp}$*: a probabilistic algorithm that takes as input a statement constant s_0 and outputs the public proof parameters* rpp.
- $\mathsf{Prove}(\mathsf{rpp}, s_1, w) \to \pi$*: a probabilistic algorithm that takes as input the public proof parameters* rpp*, a statement component s_1, and a witness w and outputs a proof π.*
- $\mathsf{Verify}(\mathsf{rpp}, s_1, \pi) \to 0/1$*: a deterministic algorithm that takes as input the public proof parameters, a statement component s_1, and a proof π and outputs either 1 accepting the proof or 0 rejecting it.*

The algorithms satisfy the following properties:

- *Completeness: For all $((s_0, s_1), w) \in \mathcal{R}$ it holds that*

$$\Pr\left[\mathsf{Verify}(\mathsf{rpp}, s_1, \pi) = 1 : \begin{array}{l}\mathsf{rpp} \leftarrow \mathsf{Setup}(s_0),\\ \pi \leftarrow \mathsf{Prove}(\mathsf{rpp}, s_1, w)\end{array}\right] = 1.$$

- *Knowledge soundness: For every $\lambda \in \mathbb{N}$, every auxiliary input $\mathsf{aux} \in \{0,1\}^*$ of size polynomial in λ, every statement constant $s_0 \in \{0,1\}^*$ of size polynomial in λ, every* PPT *adversary $\mathcal{A}$, and every random tape r, there exists a* PPT *extractor $\mathcal{E}$ with oracle access to $\mathcal{A}$ and control over random oracle $\mathcal{H}$, as well as a negligible function* negl*, such that*

$$\Pr\left[\begin{array}{l}\mathsf{Verify}((s_0, s_1), \pi) = 1\\ \wedge((s_0, s_1), w) \notin \mathcal{R}\end{array} : \begin{array}{l}\mathsf{rpp} \leftarrow \mathsf{Setup}(s_0)\\ (s_1, \pi) \leftarrow \mathcal{A}(\mathsf{rpp}, \mathsf{aux}; r)\\ w \leftarrow \mathcal{E}^{\mathcal{H}, \mathcal{A}}(\mathsf{rpp}, \mathsf{aux}, r)\end{array}\right] \leq \mathsf{negl}(\lambda).$$

Construction 2: Partial request proof

$\mathsf{PK}^1_{\mathsf{Rq}}.\mathsf{Prove}((\mathsf{pp}, \boldsymbol{X}, \boldsymbol{T}, \{\boldsymbol{D_i}\}_{\boldsymbol{i}\in[\boldsymbol{L}]}), (\boldsymbol{r}, \boldsymbol{\delta}, \{\boldsymbol{r_i}, \boldsymbol{x_i}\}_{\boldsymbol{i}\in[\boldsymbol{L}]}))$:

1) Sample $\alpha_\delta, \alpha_1, \ldots, \alpha_L, \beta_0, \beta_1, \ldots, \beta_L \xleftarrow{\$} \mathbb{Z}_p$.
2) Compute $U_T = \alpha_\delta T + \beta_0 H_0 + \sum_{i\in[L]} \alpha_i H_i$ and $U_X = \sum_{i\in[L]} (\alpha_i \cdot 2^{(i-1)\ell}) G_1$
3) For $i \in [L]$ compute $U_i = \alpha_i G_0 + \beta_i H_0$.
4) Compute $c = \mathcal{H}(\mathsf{pp}, X, T, \{D_i\}_{i\in[L]}, U_T, U_X, \{U_i\}_{i\in[L]})$
5) Compute $\gamma_\delta = \alpha_\delta + c \cdot \frac{1}{\delta}$ and $\gamma_0^R = \alpha_0 - c \cdot r$.
6) For $i \in [L]$ compute $\gamma_i^X = \alpha_i - c \cdot x_i$ and $\gamma_i^R = \beta_i - c \cdot r_i$.
7) Return $\pi = (c, \gamma_\delta, \gamma_0^R, \{(\gamma_i^X, \gamma_i^R)\}_{i\in[L]})$.

$\mathsf{PK}^1_{\mathsf{Rq}}.\mathsf{Verify}((\mathsf{pp}, \boldsymbol{X}, \boldsymbol{T}, \{\boldsymbol{D_i}\}_{\boldsymbol{i}\in[\boldsymbol{L}]}), \boldsymbol{\pi})$:

8) Compute $W_T = -cG_0 + \gamma_\delta T + \gamma_0^R H_0 + \sum_{i\in[L]} \gamma_i^X H_i$.
9) Compute $W_X = cX + \sum_{i\in[L]} (\gamma_i^X \cdot 2^{(i-1)\ell}) \cdot G_1$
10) For $i \in [L]$ compute $W_i = cD_i + \gamma_i^X G_0 + \gamma_i^R H_0$.
11) Compute $c' = \mathcal{H}(\mathsf{pp}, X, T, \{D_i\}_{i\in[L]}, W_T, W_X, \{W_i\}_{i\in[L]})$.
12) Output 1 if $c = c$ and 0 otherwise.

Construction 3: Request proof Wrapper

$\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Setup}(\boldsymbol{p}, \boldsymbol{\mathbb{G}_1}, \boldsymbol{G_0}, \boldsymbol{H_0}, \boldsymbol{L}, \boldsymbol{\ell})$

1) Compute $\mathsf{rpp}' \leftarrow \mathsf{PK}^2_{\mathsf{Rq}}.\mathsf{Setup}(p, \mathbb{G}_1, G_0, H_0, L, \ell)$.
2) Output $\mathsf{rpp} \leftarrow (p, \mathbb{G}_1, G_0, H_0, L, \ell, \mathsf{rpp}')$.

$\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Prove}(\mathsf{rpp}, (\mathsf{pp}, \mathsf{pk}^{\mathsf{In}}_c, \boldsymbol{T}), (\{\boldsymbol{x_i}\}_{\boldsymbol{i}\in[\boldsymbol{L}]}, \boldsymbol{r}, \boldsymbol{\delta}))$:

3) For $i \in [L]$ sample $r_i \xleftarrow{\$} \mathbb{Z}_p$ and compute $D_i = x_i G_0 + r_i H_0$.
4) Compute $\pi_1 = \mathsf{PK}^1_{\mathsf{Rq}}.\mathsf{Prove}((\mathsf{pp}, \mathsf{pk}^{\mathsf{In}}_c, T, \{D_i\}_{i\in[L]}), (r, \delta, \{r_i, x_i\}_{i\in[L]}))$.
5) Compute $\pi_2 = \mathsf{PK}^2_{\mathsf{Rq}}.\mathsf{Prove}(\mathsf{rpp}, \{D_i\}_{i\in[L]}, \{x_i, r_i\}_{i\in[L]})$
6) Output $\pi = (\pi_1, \pi_2, \{D_i\}_{i\in[L]})$.

$\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Verify}(\mathsf{rpp}, (\mathsf{pp}, \mathsf{pk}^{\mathsf{In}}_c, \boldsymbol{T}), \boldsymbol{\pi})$:

7) Output 1 if $\mathsf{PK}^1_{\mathsf{Rq}}.\mathsf{Verify}((\mathsf{pp}, \mathsf{pk}^{\mathsf{In}}_c, T, \{D_i\}_{i\in[L]}), \pi_2) = 1$ and $\mathsf{PK}^2_{\mathsf{Rq}}.\mathsf{Verify}(\mathsf{rpp}, \{D_i\}, \pi_2) = 1$. Otherwise, output 0.

Figure 7: Instantiations of the request proof systems.

- *Zero-knowledge: There exists a PPT simulator $\mathcal{S}$ controlling oracle $\mathcal{H}$ such that for every group $\lambda \in \mathbb{N}$ and for every $((s_0, s_1), w) \in \mathcal{R}$ of size polynomial in λ, it holds that*

$$\{(\mathsf{rpp}, \pi) : \mathsf{rpp} \leftarrow \mathsf{Setup}(s_0), \pi \leftarrow \mathsf{Prove}(\mathsf{rpp}, s_1, w)\} \approx_c \left\{\mathcal{S}^{\mathcal{H}}(\mathsf{rpp}, (s_0, s_1))\right\}.$$

Appendix D. Instantiations of our Zero-Knowledge Proof Systems

This section presents instantiations for the proof systems introduced in Section 4.1. We slightly abuse the notation and assume the protocol's public parameters $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_0, G_1, G_2, \{H_1\}_{i\in[0,L]}, L, \ell, \mathsf{rpp})$ to be part of every statement even though the statement only contains the relevant components of the public parameters

Within the request proof system, we make use of two different proof types, a zero-knowledge proof of knowledge to prove knowledge of $(\{x_i\}_{i\in[L]}, r, \delta)$ as well as a range proof to prove that the chunks are in the correct range ($x_i \in [0, 2^\ell - 1]$ for $i \in [L]$). Therefore, we split the request

Construction 4: Challenge Proof

$\mathsf{PK}_{\mathsf{Ch}}.\mathsf{Prove}((\mathsf{pp}, K, \mathsf{pk}_c^{\mathsf{Id}}), d)$:
1) Set $D_0 = K$ and $D_1 = \mathsf{pk}_c^{\mathsf{Id}}$.
2) Set $b = 0$ if $D_0 = dG_0$ and $b = 1$ otherwise. Set $\bar{b} = 1 - b$.
3) Sample $\alpha_b, \gamma_{\bar{b}}, c_{\bar{b}} \xleftarrow{\$} \mathbb{Z}_p$ and compute $U_b = \alpha_b G_b$ and $U_{\bar{b}} = \gamma_{\bar{b}} G_{\bar{b}} - c_{\bar{b}} D_{\bar{b}}$.
4) Compute $c = \mathcal{H}(\mathsf{pp}, D_0, D_1, U_0, U_1)$ and $c_b = c - c_{\bar{b}}$.
5) Compute $\gamma_b = \alpha_b + d \cdot c_b$ and output $\pi = (U_0, U_1, \gamma_0, \gamma_1, c_0)$.

$\mathsf{PK}_{\mathsf{Ch}}.\mathsf{Verify}((\mathsf{pp}, K, \mathsf{pk}_c^{\mathsf{Id}}), \pi)$:
6) Compute $c' = \mathcal{H}(\mathsf{pp}, K, \mathsf{pk}_c^{\mathsf{Id}}, U_0, U_1)$ and $c'_1 = c' - c_0$.
7) Output 1 if $\gamma_0 G_0 = U_0 + c_0 K$ and $\gamma_1 G_1 = U_1 + c'_1 \mathsf{pk}_c^{\mathsf{Id}}$ and 0 otherwise.

Construction 5: Redemption Proof

$\mathsf{PK}_{\mathsf{Rd}}.\mathsf{Prove}((\mathsf{pp}, Y, K, \sigma, \sigma', \{A_i, B_i\}_{i\in[L]}), (y, r, e, \{r_i, x_i\}_{i\in[L]}))$:
1) Sample $\alpha_y, \alpha_r, \alpha_e, \alpha_1, \ldots, \alpha_L, \beta_1, \ldots, \beta_L \xleftarrow{\$} \mathbb{Z}_p$.
2) Compute $U_Y = \alpha_y G_1$ and $U_T = \alpha_r H_0 + \sum_{i\in[L]} \alpha_i H_i - \alpha_e \sigma$
3) For $i \in [L]$ compute $U_i^A = \beta_i G_1$ and $U_i^B = \beta_i K + \alpha_i G_1$.
4) Compute $c = \mathcal{H}(\mathsf{pp}, Y, \sigma, \sigma', K, \{A_i, B_i\}_{i\in[L]}, U_Y, U_T, \{U_i^A, U_i^B\}_{i\in[L]})$.
5) Compute $\gamma_y = \alpha_y - c \cdot y$, $\gamma_0^R = \alpha_r - c \cdot r$ and $\gamma_e = \alpha_e - c \cdot e$.
6) For $i \in [L]$ compute $\gamma_i^X = \alpha_i - c \cdot x_i$ and $\gamma_i^R = \beta_i - c \cdot r_i$.
7) Output $\pi = (c, \gamma_y, \gamma_0^R, \gamma_e, \{(\gamma_i^X, \gamma_i^R)\}_{i\in[L]})$.

$\mathsf{PK}_{\mathsf{Rd}}.\mathsf{Verify}((\mathsf{pp}, Y, K, \sigma, \sigma', \{A_i, B_i\}_{i\in[L]}), \pi)$:
8) Compute $W_Y = cY + \gamma_y G_1$ and $W_T = c(z\sigma - G_0) + \gamma_0^R H_0 - \gamma_e \sigma + \sum_{i\in[L]} \gamma_i^X H_i$.
9) For $i \in [L]$ compute $W_i^A = cA_i + \gamma_i^R G_1$ and $W_i^B = cB_i + \gamma_i^R K + \gamma_i^X G_1$.
10) Compute $c' = \mathcal{H}(\mathsf{pp}, Y, \sigma, \sigma', K, \{A_i, B_i\}_{i\in[L]}, W_Y, W_T, \{W_i^A, W_i^B\}_{i\in[L]})$.
11) Output 1 if $c = c'$ and 0 otherwise.

Figure 8: Instantiations of the challenge and redemption proof systems.

proof into two separate components with the following proof relations:

$$\mathsf{PK}^1_{\mathsf{Rq}} = \mathsf{PK}\left[\begin{array}{l}(r, \delta, \{x_i, r_i\}_{i\in[L]}) : \\ X = (\sum_{i\in[L]} 2^{(i-1)\ell} x_i) G_1 \\ T = \delta(G_0 + rH_0 + \sum_{i\in[L]} x_i H_i) \\ D_i = x_i G_0 + r_i H_0 \text{ for } i \in [L]\end{array}\right]$$

$$\mathsf{PK}^2_{\mathsf{Rq}} = \mathsf{PK}\left[\begin{array}{l}(\{x_i, r_i\}_{i\in[L]}) : \\ D_i = x_i G_0 + r_i H_0 \text{ for } i \in [L] \\ x_i \in [0, 2^\ell - 1] \text{ for } i \in [L]\end{array}\right].$$

As we instantiate the range proof with the Bulletproof++ proof system [24], we have to incorporate Pedersen commitments to the chunks into both relations. In order to instantiate a zero-knowledge proof for the poof relation $\mathsf{PK}_{\mathsf{Rq}}$, we define a wrapper proof system ($\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Setup}$, $\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Prove}$, $\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Verify}$) that creates the required commitments to the chunks and creates proofs for the two proof relations $\mathsf{PK}^1_{\mathsf{Rq}}$ and $\mathsf{PK}^2_{\mathsf{Rq}}$ individually. We assume ($\mathsf{PK}^2_{\mathsf{Rq}}.\mathsf{Setup}$, $\mathsf{PK}^2_{\mathsf{Rq}}.\mathsf{Prove}$, $\mathsf{PK}^2_{\mathsf{Rq}}.\mathsf{Verify}$) to be a secure zero-knowledge proof of knowledge with setup for relation $\mathsf{PK}^2_{\mathsf{Rq}}$, e.g., instantiated with the Bulletproof++ proof system, and instantiate the proof systems for proof relation $\mathsf{PK}^1_{\mathsf{Rq}}$ and the wrapper as in Figure 8. The challenge proof system and the redemption proof system are instantiated as in Figure 8.

Appendix E. Incorporating Metadata

Given that our scheme builds on BBS signatures, it can easily be extended to support public metadata and client-side private metadata.

Public metadata can be added by requiring both client and server to replace G_0 by $G'_0 = G_0 + \mathcal{H}_{\mathsf{pm}}(\mathsf{md}) \cdot H_{\mathsf{pm}}$ for additional public parameter $H_{\mathsf{pm}} \xleftarrow{\$} \mathbb{G}_1$, collision resistant hash function $\mathcal{H}_{\mathsf{pm}} : \{0,1\}^* \to \mathbb{Z}_p$, and metadata md per session. We note that public metadata limits the anonymity set of a redeemed token to all other tokens issued for the same metadata. Therefore, it is recommended to pick coarse grained metadata that is shared with sufficiently many other clients in order to sustain the anonymity guarantees of the scheme.

Client-side private metadata can be added by replacing the commitment $C_i = G_0 + \mathcal{H}_{\mathsf{pm}}(\mathsf{md}) \cdot H_{\mathsf{pm}} + rH_0 + \sum_{i\in L} x_i H_i$ with $C'_i = C_i + \sum_{j\in[N]} s_j H_j^{\mathsf{sm}}$ for N metadata elements s_j encoded as field elements and additional public parameters $H_1^{\mathsf{sm}}, \ldots, H_N^{\mathsf{sm}} \xleftarrow{\$} \mathbb{G}_1$. Obviously, we have to extend the request and redemption proof such that the client additionally proves knowledge of the secret metadata. This can be done with the same technique used to prove knowledge of the insurance key chunks. If necessary, the scheme can be extended to require clients to prove statements about the private metadata with zero-knowledge proof techniques already known from BBS(+) signatures [18], [20], [21].

Appendix F. Proof of Theorem 4

This section provides a detailed proof of Theorem 4. Let $\mathcal{A}$ be a generic PPT adversary. We prove one-more unforgeability

$$\mathsf{Pr}[\mathsf{Exp}^{\mathsf{OMUF}}_{\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

request unforgeability

$$\mathsf{Pr}[\mathsf{Exp}^{\mathsf{RUF}}_{\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

and identity unforgeability

$$\mathsf{Pr}[\mathsf{Exp}^{\mathsf{IUF}}_{\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

separately. By combining the three, we can conclude that Theorem 4 holds.

F.1. One-more Unforgeability

We prove one-more unforgeability via a sequence of games. Let $\mathsf{Exp}^{\mathsf{OMUF}}_0 = \mathsf{Exp}^{\mathsf{OMUF}}_{\mathcal{A}}$ be the initial security game. The subsequent games are defined as follows.

Inline construction. In the first game, we inline our concrete construction into the security game (where relevant). The game is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{OMUF}}_1(\lambda)$

$q_{\mathsf{Rq}}, q_{\mathsf{Rd}} = 0,\ \Gamma \leftarrow \emptyset,\ \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ (\mathsf{sk}_s, \mathsf{pk}_s) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$
$\mathcal{A}^{\mathcal{O}^{\mathsf{Rq}}, \mathcal{O}^{\mathsf{Rd}}}(\mathsf{pp}, \mathsf{pk}_s)$, **if** $q_{\mathsf{Rq}} < q_{\mathsf{Rd}}$ **return** 1 **else return** 0

$\underline{\mathcal{O}^{\mathsf{Rq}}(\mathsf{pk}, \mathsf{query}) :}$
$q_{\mathsf{Rq}} \leftarrow q_{\mathsf{Rq}} + 1,\ (T, \pi_{\mathsf{Rq}}) \leftarrow \mathsf{query}$
if $\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Verify}(\mathsf{rpp}, (\mathsf{pk}^{\mathsf{In}}, T), \pi_{\mathsf{Rq}}) = 0$ **then return** $\emptyset$.
$e \xleftarrow{\$} \mathbb{Z}_p, S \leftarrow \frac{1}{\mathsf{sk}_s + e} T$
return (e, S)

$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathsf{sid}, \mathsf{m}) :}$
if $\mathsf{st}_{\mathsf{sid}} \in \{0, 1\}$ **then return** $\perp$
if $\mathsf{st}_{\mathsf{sid}} = \mathbf{null}$ **then**
  $(\sigma_{\mathsf{sid}}, \mathsf{pk}^{\mathsf{Id}}_{\mathsf{sid}}) \leftarrow \mathsf{m}, k_{\mathsf{sid}} \xleftarrow{\$} \mathbb{Z}_p, K_{\mathsf{sid}} \leftarrow k_{\mathsf{sid}} G_0$
  $\pi_{\mathsf{sid}} \leftarrow \mathsf{PK}_{\mathsf{Ch}}.\mathsf{Prove}((K_{\mathsf{sid}}, \mathsf{pk}^{\mathsf{Id}}_{\mathsf{sid}}), k_{\mathsf{sid}})$
  $\mathsf{st}_{\mathsf{sid}} \leftarrow (\sigma_{\mathsf{sid}}, \mathsf{pk}^{\mathsf{Id}}_{\mathsf{sid}}, K_{\mathsf{sid}})$
  return $(K_{\mathsf{sid}}, \pi_{\mathsf{sid}})$
else
  $(\pi_{\mathsf{Rd}}, \sigma', \{A_i, B_i\}_{i \in [L]}) \leftarrow \mathsf{m}$
  if $\mathsf{e}(\sigma, \mathsf{pk}_s) \neq \mathsf{e}(\sigma', G_2)$ **then** $\mathsf{st}_{\mathsf{sid}} \leftarrow 0,$ **return** 0
  if $\mathsf{PK}_{\mathsf{Rd}}.\mathsf{Verify}((\mathsf{pk}_{\mathsf{sid}}, K_{\mathsf{sid}}, \sigma_{\mathsf{sid}}, \sigma', \{A_i, B_i\}_{i \in [L]}), \pi_{\mathsf{Rd}}) = 0$ **then** $\mathsf{st}_{\mathsf{sid}} \leftarrow 0,$ **return** 0
  if $\sigma_{\mathsf{sid}} \notin \Gamma$ **then** $q_{\mathsf{Rd}} \leftarrow q_{\mathsf{Rd}} + 1, \Gamma \leftarrow \Gamma \cup \{\sigma_{\mathsf{sid}}\}$
  $\mathsf{st}_{\mathsf{sid}} \leftarrow 1,$ **return** 1

This step does not change the output behavior. It follows that

$$\mathsf{Pr}[\mathsf{Exp}^{\mathsf{OMUF}}_1(\lambda) = 1] = \mathsf{Pr}[\mathsf{Exp}^{\mathsf{OMUF}}_0(\lambda) = 1]$$

Extract request proofs. In the next experiment, we adapt the request oracle. After successfully verifying the request proof, we employ the extractor $\mathcal{E}_{\mathsf{Rq}}$ of the request proof system ($\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Setup}$, $\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Prove}$, $\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Verify}$) to extract the witness of the request proof. The extractor has access to the adversary $\mathcal{A}$, receives the transcript, trans, of the experiment execution up to the extraction including the adversary's random tape, and has the power to program the random oracle. It extracts the witnesses $(\{x_i\}_{i \in [L]}, r, \delta)$ that are all either $\perp$ or satisfy the proof system relation $\mathsf{PK}_{\mathsf{Rq}}$. If the witnesses do not satisfy the proof relation, we abort the experiment. The resulting experiment is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{OMUF}}_2(\lambda)$

$\ldots$ $\underline{\mathcal{O}^{\mathsf{Rq}}(\mathsf{pk}^{\mathsf{In}}, \mathsf{query}) :}$
$q_{\mathsf{Rq}} \leftarrow q_{\mathsf{Rq}} + 1,\ (T, \pi_{\mathsf{Rq}}) \leftarrow \mathsf{query}$
if $\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Verify}(\mathsf{rpp}, (\mathsf{pk}^{\mathsf{In}}, T), \pi_{\mathsf{Rq}}) = 0$ **then return** $\emptyset$.
$(\{x_i\}_{i \in [L]}, r, \delta) \leftarrow \mathcal{E}^{\mathcal{A}(\cdot)}_{\mathsf{Rq}}(\mathsf{trans}),$ **if** $r = \perp$ **then return** $\perp$
$e \xleftarrow{\$} \mathbb{Z}_p, S \leftarrow \frac{1}{\mathsf{sk}_s + e} T$
return (e, S)
$\ldots$

It is obvious that $\mathsf{Exp}^{\mathsf{OMUF}}_2$ and $\mathsf{Exp}^{\mathsf{OMUF}}_1$ only differ if the extractor is not successful, which happens with negligible probability due to the knowledge soundness property of the request proof system. This allows us to use the difference lemma to conclude that

$$|\Pr[\mathsf{Exp}^{\mathsf{OMUF}}_2(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{OMUF}}_1(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

Extract redemption proofs. Next, we adapt the redemption oracle to extract the witness of the redemption proof π_{Rd} after successfully verifying π_{Rd}. To do so, we use $\mathcal{E}_{\mathsf{Rd}}$ of the redemption proof system, that succeeds expect with negligible probability. As before, the extractor has access to the adversary $\mathcal{A}$, receives the transcript, trans, of the experiment execution up to the extraction including the adversary's random tape, and has the power to program the random oracle. It extracts the witnesses $(\mathsf{sk}^{\mathsf{Id}}, r, e, \{r_i, x_i\}_{i \in [L]})$ that are either all $\perp$ or satisfy the redemption proof relation $\mathsf{PK}_{\mathsf{Rd}}$. The resulting experiment is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{OMUF}}_3(\lambda)$

$\ldots$ $\underline{\mathcal{O}^{\mathsf{Rd}}(\mathsf{sid}, \mathsf{m}) :}$
$\ldots$
else
  $(\pi_{\mathsf{Rd}}, \sigma', \{A_i, B_i\}_{i \in [L]}) \leftarrow \mathsf{m}$
  if $\mathsf{e}(\sigma, \mathsf{pk}_s) \neq \mathsf{e}(\sigma', G_2)$ **then** $\mathsf{st}_{\mathsf{sid}} \leftarrow 0,$ **return** 0
  if $\mathsf{PK}_{\mathsf{Rd}}.\mathsf{Verify}((\mathsf{pk}_{\mathsf{sid}}, K_{\mathsf{sid}}, \sigma_{\mathsf{sid}}, \sigma', \{A_i, B_i\}_{i \in [L]}), \pi_{\mathsf{Rd}}) = 0$ **then** $\mathsf{st}_{\mathsf{sid}} \leftarrow 0,$ **return** 0
  $(\mathsf{sk}^{\mathsf{Id}}, r, e, \{r_i, x_i\}_{i \in [L]}) \leftarrow \mathcal{E}^{\mathcal{A}(\cdot)}_{\mathsf{Rd}}(\mathsf{trans})$
  if $e = \perp$ **then** $\mathsf{st}_{\mathsf{sid}} \leftarrow 0,$ **return** 0
  if $\sigma_{\mathsf{sid}} \notin \Gamma$ **then** $q_{\mathsf{Rd}} \leftarrow q_{\mathsf{Rd}} + 1, \Gamma \leftarrow \Gamma \cup \{\sigma_{\mathsf{sid}}\}$
  $\mathsf{st}_{\mathsf{sid}} \leftarrow 1,$ **return** 1

It is obvious that $\mathsf{Exp}^{\mathsf{OMUF}}_3$ and $\mathsf{Exp}^{\mathsf{OMUF}}_2$ only differ if the extractor is not successful, which happens with negligible probability due to the knowledge soundness property of the redemption proof system. This allows us to use the difference lemma to conclude that

$$|\Pr[\mathsf{Exp}^{\mathsf{OMUF}}_3(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{OMUF}}_2(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

Simplification. In a bridging step, we adapt the issuance oracle to compute the signatures based on the extracted

witnesses of the request proof. As both ways of computing the (blinded) token result in exactly the same token, this change does does not influence the output distribution. The resulting experiment is defined as follows:

Experiment: $\mathsf{Exp}_4^{\mathsf{OMUF}}(\lambda)$

$\dots$ $\underline{\mathcal{O}^{\mathsf{Rq}}(\mathsf{pk}^{\mathsf{In}}, \mathsf{query}):}$
3: $q_{\mathsf{Rq}} \leftarrow q_{\mathsf{Rq}} + 1,\ (T, \pi_{\mathsf{Rq}}) \leftarrow \mathsf{query}$
4: **if** $\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Verify}(\mathsf{rpp}, (\mathsf{pk}^{\mathsf{In}}, T), \pi_{\mathsf{Rq}}) = 0$ **then return** $\emptyset$.
5: $(\{x_i\}_{i\in[L]}, r, \delta) \leftarrow \mathcal{E}_{\mathsf{Rq}}^{\mathcal{A}(\cdot)}(\mathsf{trans}),$ **if** $r = \perp$ **then return** $\perp$
6: $e \xleftarrow{\$} \mathbb{Z}_p, \sigma \leftarrow \frac{1}{\mathsf{sk}_s + e}(G_0 + rH_0 + \sum_{j\in[L]} x_j H_j), S \leftarrow \delta \cdot \sigma$
7: **return** (e, S)
$\dots$

We conclude that

$$\Pr[\mathsf{Exp}_4^{\mathsf{OMUF}}(\lambda) = 1] = \Pr[\mathsf{Exp}_3^{\mathsf{OMUF}}(\lambda) = 1].$$

Reduction to BBS signatures. Note that in $\mathsf{Exp}_4^{\mathsf{OMUF}}$, the adversary can only receive q_{Rq} signatures but has to provide $q+1$ signatures with distinct first signature components σ in order to win the experiment. Therefore, it follows that at least one of the submitted tokens is a fresh BBS signature. This allows us to prove that $\Pr[\mathsf{Exp}_4^{\mathsf{OMUF}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$ via a reduction to Theorem 1. Formally, we pick $L'(\lambda)$ such that, for $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}) \leftarrow \mathsf{GGen}(1^\lambda)$, $G_1 \xleftarrow{\$} \mathbb{G}_1^*$, and $\ell = \lceil \frac{|p|}{L'(\lambda)} \rceil$, the computation of $\mathsf{dlog}_{G_1}(G)[0, 2^\ell - 1]$ has complexity below $\mathcal{C}$ for all $G \in \mathbb{G}_1$. Then, we define an adversary $\mathcal{B}$ on security game $\mathsf{Exp}_{\mathcal{B},L'+1}^{\mathsf{P\text{-}BBS\text{-}SUF}}(\lambda)$ that internally executes $\mathcal{A}$ on security game $\mathsf{Exp}_4^{\mathsf{OMUF}}(\lambda)$ and wins with the same probability as $\mathcal{A}$. We define $\mathcal{B}$ as follows:

- Receive $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_0, G_1, \{H_i\}_{i\in[L'+1]}, G_2, Z)$ from the $\mathsf{Exp}_{\mathcal{B},L'+1}^{\mathsf{P\text{-}BBS\text{-}SUF}}$ security game.
- Pick ℓ such that $\mathsf{dlog}_{G_1}(G)[0, 2^{\ell-1}]$ can be computed with complexity of at most $\mathcal{C}$ for any $G \in \mathbb{G}_1$ and set $L \leftarrow \lceil \frac{|p|}{\ell} \rceil$. Based on the definition of L' it holds that $L \leq L'$.
- Define $H_0 \leftarrow H_{L+1}$ and compute $\mathsf{rpp} = \mathsf{RP}.\mathsf{Setup}(p, \mathbb{G}_1, G_0, H_0, L, \ell)$.
- Define $\mathsf{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_0, G_1, H_0, H_1, \dots, H_L, L, \ell, \mathsf{rpp})$.
- Initialize $\Gamma, \Gamma_{\mathsf{Rq}}, \Gamma_{\mathsf{Rd}} \leftarrow \emptyset$.
- Execute $\mathcal{A}^{\mathcal{O}^{\mathsf{Rq}}, \mathcal{O}^{\mathsf{Rd}}}(\mathsf{pp}, Z)$ and answer oracle queries as follows:
  - To answer $\mathcal{O}^{\mathsf{Rq}}$, execute the steps of the oracle as defined in $\mathsf{Exp}_4^{\mathsf{OMUF}}$ but instead of sampling e and calculating σ (which is not possible, as sk_s is not known), query $\mathcal{O}^{\mathsf{Sign}}(\{x_i\}_{i\in[L]} \cup \{r\} \cup \{0\}_{i\in L'\setminus L})$ of the $\mathsf{Exp}_{\mathcal{B},L'+1}^{\mathsf{P\text{-}BBS\text{-}SUF}}$ game to receive (e, σ) and store $(\{x_i\}_{i\in[L]} \cup \{r\}, \sigma, e)$ in Γ_{Rq}.
  - $\mathcal{O}^{\mathsf{Rd}}$ can be executed as defined in $\mathsf{Exp}_4^{\mathsf{OMUF}}$ since it does not need knowledge of the secret key. However, when adding σ_{sid} to Γ, $\mathcal{B}$ also adds $(\{x_i\}_{i\in[L]} \cup \{r\} \cup \{0\}_{i\in L'\setminus L}, \sigma_{\mathsf{sid}}, e)$ to Γ_{Rd}.
- If $q_{\mathsf{Rq}} \geq q_{\mathsf{Rd}}$ return $\perp$.
- Otherwise, define $(\{x_i^*\}_{i\in[L]} \cup \{r^*\}, \sigma^*, e^*)$ to be the first element of $\Gamma_{\mathsf{Rd}} \setminus \Gamma_{\mathsf{Rq}}$ and return $(\{x_i^*\}_{i\in[L]} \cup \{r^*\}, \sigma^*, e^*)$.

First note, that the view of $\mathcal{A}$ in the experiment simulated by $\mathcal{B}$ is identical to the view of $\mathcal{A}$ in experiment $\mathsf{Exp}_4^{\mathsf{OMUF}}$. The public parameters are generated identically in both experiments and oracle queries are answered identically as well. To analyze the success probability of $\mathcal{B}$, we first note that all elements in Γ_{Rd} are valid BBS message-signature tuples under public key $Z = zG_2$ (where z is unknown). Note that for each tuple $(\{x_i\}_{i\in[L]} \cup \{r\}, \sigma, e)$ (and associated σ') that is added to Γ_{Rd} it holds that $\mathsf{e}(\sigma, Z) = \mathsf{e}(\sigma', G_2)$ (explicitly checked) and $\sigma' = G_0 + rH_0 - e\sigma + \sum_{i\in[L]} x_i H_i$ (due to the success of the knowledge extractor). The former implies that $\sigma' = z\sigma$ which allows us to conclude that $\sigma = \frac{1}{z+e}(G_0 + rH_0 + \sum_{i\in[L]} x_i H_i)$ and, hence, that $(\{x_i\}_{i\in[L]} \cup \{r\}, \sigma, e)$ is a valid BBS message-signature tuple if added to Γ_{Rd}.

Next, we note that any element in $\Gamma_{\mathsf{Rd}} \setminus \Gamma_{\mathsf{Rq}}$ (if existent) is a fresh BBS signature. This follows straightforwardly from the fact that Γ_{Rq} contains all issued BBS signatures.

Finally, we not that $\Gamma_{\mathsf{Rd}} \setminus \Gamma_{\mathsf{Rq}}$ contains at least one element if $\mathcal{A}$ would win $\mathsf{Exp}_4^{\mathsf{OMUF}}$. In order for $\mathcal{A}$ to win, it needs to hold that $q_{\mathsf{Rq}} \geq q_{\mathsf{Rd}}$. Furthermore, the size of Γ_{Rq} is at most q_{Rq} (if the zero-knowledge proof is invalid, q_{Rq} is incremented but Γ_{Rq} remains untouched) and the size of Γ_{Rd} is equal to q_{Rd}. Therefore, it needs to hold that Γ_{Rd} is larger than Γ_{Rq} and, hence, that $\Gamma_{\mathsf{Rd}} \setminus \Gamma_{\mathsf{Rq}}$ contains at least one element, if $\mathcal{A}$ would win $\mathsf{Exp}_4^{\mathsf{OMUF}}$.

As we have shown that Γ_{Rd} contains only valid BBS signatures, $\Gamma_{\mathsf{Rd}} \setminus \Gamma_{\mathsf{Rq}}$ contains only fresh BBS signatures (if any), and that $\Gamma_{\mathsf{Rd}} \setminus \Gamma_{\mathsf{Rq}}$ contains at least one element if $\mathcal{A}$ would win in $\mathsf{Exp}_4^{\mathsf{OMUF}}$, we conclude that $\mathcal{B}$ outputs a valid forgery if $\mathcal{A}$ would win in $\mathsf{Exp}_4^{\mathsf{OMUF}}$ and, hence that

$$\mathsf{Pr}[\mathsf{Exp}_{\mathcal{B},L'+1}^{\mathsf{P\text{-}BBS\text{-}SUF}}(\lambda) = 1] = \mathsf{Pr}[\mathsf{Exp}_4^{\mathsf{OMUF}}(\lambda) = 1].$$

Putting all steps together. Above, we have shown that

$$\begin{aligned}
\Pr[\mathsf{Exp}_1^{\mathsf{OMUF}}(\lambda) = 1] &= \Pr[\mathsf{Exp}_0^{\mathsf{OMUF}}(\lambda) = 1] \\
|\Pr[\mathsf{Exp}_2^{\mathsf{OMUF}}(\lambda) = 1] - \Pr[\mathsf{Exp}_1^{\mathsf{OMUF}}(\lambda) = 1]| &\leq \mathsf{negl}(\lambda). \\
|\Pr[\mathsf{Exp}_3^{\mathsf{OMUF}}(\lambda) = 1] - \Pr[\mathsf{Exp}_2^{\mathsf{OMUF}}(\lambda) = 1]| &\leq \mathsf{negl}(\lambda). \\
\Pr[\mathsf{Exp}_4^{\mathsf{OMUF}}(\lambda) = 1] &= \Pr[\mathsf{Exp}_3^{\mathsf{OMUF}}(\lambda) = 1] \\
\mathsf{Pr}[\mathsf{Exp}_{\mathcal{B},L'+1}^{\mathsf{P\text{-}BBS\text{-}SUF}}(\lambda) = 1] &= \mathsf{Pr}[\mathsf{Exp}_4^{\mathsf{OMUF}}(\lambda) = 1]
\end{aligned}$$

Further Theorem 1 states that $\mathsf{Pr}[\mathsf{Exp}_{\mathcal{B},L'+1}^{\mathsf{P\text{-}BBS\text{-}SUF}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$. Therefore, we conclude that

$$\mathsf{Pr}[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{OMUF}}(\lambda) = 1] \leq \mathsf{negl}(\lambda).$$

F.2. Request Unforgeability

We prove request unforgeability via a sequence of games. Let $\mathsf{Exp}_0^{\mathsf{RUF}} = \mathsf{Exp}_{\mathcal{A}}^{\mathsf{RUF}}$ be the initial security game. The subsequent games are defined as follows.

Inline construction. In the first game, we inline our concrete construction into the security game's main thread. The game is defined as follows:

Experiment: $\mathsf{Exp}_1^{\mathsf{RUF}}(\lambda)$

1: $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{sk}_c^{\mathsf{In}}, \mathsf{pk}_c^{\mathsf{In}}, \mathsf{sk}_c^{\mathsf{Id}}, \mathsf{pk}_c^{\mathsf{Id}}) \leftarrow \mathsf{C.KeyGen}(\mathsf{pp})$
2: $((T^*, \pi^*), \mathsf{sk}_s) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Qu}}, \mathcal{O}^{\mathsf{Fin}}, \mathcal{O}^{\mathsf{Rd}}}(\mathsf{pp}, \mathsf{pk}_c^{\mathsf{In}}, \mathsf{pk}_c^{\mathsf{Id}})$
3: **if** $\exists i$ s.t. $(T^*, \pi^*) = (T_i, \pi_i)$ **then return** 0
4: **if** $\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Verify}(\mathsf{rpp}, (\mathsf{pk}_c^{\mathsf{In}}, T^*), \pi^*) = 0$ **then return** 0
5: **return** 1

$\underline{\mathcal{O}^{\mathsf{Qu}}(i, \mathsf{pk}):}$

if $T_i \neq \textbf{null}$ **then return** $\perp$
Select $x_1, \ldots, x_L \in [0, 2^\ell - 1]$ such that $\mathsf{sk}_c^{\mathsf{In}} = \sum_{j\in[L]} 2^{(j-1)\ell} x_j$ holds without modulo reduction (This representation is unique and independent of i)
$r_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p,\ C_i = G_0 + r_i H_0 + \sum_{j\in[L]} x_j H_j,\ \mathsf{pk}_i \leftarrow \mathsf{pk}$
$T_i \leftarrow \delta_i C_i, \pi_i \leftarrow \mathsf{PK}_{\mathsf{Rq}}.\mathsf{Prove}(\mathsf{rpp}, (\mathsf{pk}_c^{\mathsf{In}}, T_i), (\{x_j\}_{j\in[L]}, r_i, \delta_i))$
return (T_i, π_i)

$\underline{\mathcal{O}^{\mathsf{Fin}}(i, \mathsf{resp} = (e, S)):}$

if $T_i = \textbf{null} \vee \sigma_i \neq \textbf{null}$ **then return** $\perp$
if $\mathsf{e}(S, \mathsf{pk}_i + eG_2) \neq \mathsf{e}(T_i, G_2)$ **then** $\sigma_i \leftarrow \perp$, **return** σ_i
else $\sigma_i \leftarrow \frac{1}{\delta_i} S,\ e_i \leftarrow e,$ **return** σ_i

$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathsf{sid}, \mathsf{m}):}$

if $\mathsf{st}_{\mathsf{sid}} = \textbf{null}$ **then**
  $i \leftarrow \mathsf{m},$ **if** $\sigma_i \in \{\textbf{null}, \perp\}$ **then return** $\perp$
  $\mathsf{st}_{\mathsf{sid}} \leftarrow i,$ **return** $(\mathsf{pk}_c^{\mathsf{Id}}, \sigma_i)$
elseif $\mathsf{st}_{\mathsf{sid}} \neq \perp$ **then**
  $i \leftarrow \mathsf{st}_{\mathsf{sid}}, (K, \pi_{\mathsf{Ch}}) \leftarrow \mathsf{m}, \mathsf{st}_{\mathsf{sid}} \leftarrow \perp$
  if $\mathsf{PK}_{\mathsf{Ch}}.\mathsf{Verify}((K, \mathsf{pk}_c^{\mathsf{Id}}), \pi_{\mathsf{Ch}}) = 0 \vee K = 0$ **return** $\perp$
  $\sigma' \leftarrow G_0 + r_i H_0 - e_i \sigma_i + \sum_{j\in[L]} x_j H_j$
  $\forall j \in [L] : r_j \xleftarrow{\$} \mathbb{Z}_p,\ A_j \leftarrow r_j G_1,\ B_j \leftarrow r_j K + x_j G_1$
  $\pi_{\mathsf{Rd}} \leftarrow \mathsf{PK}_{\mathsf{Rd}}.\mathsf{Prove}((\mathsf{pk}_c^{\mathsf{Id}}, K, \sigma_i, \sigma', \{r_j, x_j\}_{j\in[L]}), (\mathsf{sk}_c^{\mathsf{Id}}, r_i, e_i, \{r_j, x_j\}_{j\in[L]}))$
  return $(\pi_{\mathsf{Rd}}, \sigma', \{A_j, B_j\}_{j\in[L]})$
else return $\emptyset$

This step does not change the output distribution. It follows that

$$\mathsf{Pr}[\mathsf{Exp}_1^{\mathsf{RUF}}(\lambda) = 1] = \mathsf{Pr}[\mathsf{Exp}_0^{\mathsf{RUF}}(\lambda) = 1].$$

Simulate redemption proofs. First, we simplify the redemption oracle by using the redemption proof's simulator $\mathcal{S}_{\mathsf{Rd}}$ instead of the proof system's witnesses to create redemption proofs. The simulator receives the proof's statement $(\mathsf{pp}, \mathsf{pk}_c^{\mathsf{Id}}, K, \sigma_i, \{A_j, B_j\}_{j\in[L]})$ and has the capability of programming the random oracle. It creates a valid zero knowledge proof π_{Rd}, except with negligible probability. The resulting experiment is defined as follows:

Experiment: $\mathsf{Exp}_2^{\mathsf{RUF}}(\lambda)$

$\ldots$
$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathsf{sid}, m):}$
14: $\ldots$
17: **elseif** $\mathsf{st}_{\mathsf{sid}} \neq \perp$ **then**
18: $\quad i \leftarrow \mathsf{st}_{\mathsf{sid}}, (K, \pi_{\mathsf{Ch}}) \leftarrow \mathsf{m}, \mathsf{st}_{\mathsf{sid}} \leftarrow \perp$
19: $\quad$ **if** $\mathsf{PK}_{\mathsf{Ch}}.\mathsf{Verify}((K, \mathsf{pk}_c^{\mathsf{Id}}), \pi_{\mathsf{Ch}}) = 0 \vee K = 0$ **return** $\perp$
20: $\quad \sigma' \leftarrow G_0 + r_i H_0 - e_i \sigma_i + \sum_{j\in[L]} x_j H_j$
21: $\quad \forall j \in [L] : r_j \xleftarrow{\$} \mathbb{Z}_p,\ A_j \leftarrow r_j G_1,\ B_j \leftarrow r_j K + x_j G_1$
22: $\quad \pi_{\mathsf{Rd}} \leftarrow \mathcal{S}_{\mathsf{Rd}}(\mathsf{pp}, \mathsf{pk}_c^{\mathsf{Id}}, K, \sigma_i, \sigma', \{A_j, B_j\}_{j\in[L]})$
23: $\quad$ **return** $(\pi_{\mathsf{Rd}}, \sigma', \{A_j, B_j\}_{j\in[L]})$
24: **else return** $\emptyset$

Note that the redemption proofs in $\mathsf{Exp}_1^{\mathsf{RUF}}$ are always valid due to the completeness of the proof system and the fact that the proof relation always holds (signatures have been verified before defining $\sigma \neq \perp$, identity keys are generated honestly, and ciphertexts are created as expected). It follows that $\mathsf{Exp}_2^{\mathsf{RUF}}$ and $\mathsf{Exp}_1^{\mathsf{RUF}}$ only differ if the simulator is not successful, which happens with at most negligible probability due to the zero-knowledge property of the redemption proof system. We use the difference lemma to conclude that

$$|\mathsf{Pr}[\mathsf{Exp}_2^{\mathsf{RUF}}(\lambda) = 1] - \mathsf{Pr}[\mathsf{Exp}_1^{\mathsf{RUF}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

Extract challenge proof. In the next game, we further adapt the redemption oracle to extract the witness of the challenge proof. After successfully verifying the challenge proof π_{Ch}, we employ extractor $\mathcal{E}_{\mathsf{Ch}}$ of the challenge proof system to extract the witness of the challenge proof. The extractor has access to the adversary $\mathcal{A}$, receives the transcript trans of the experiment execution up to the extraction including the adversary's random tape, and has the power to program the random oracle. It extracts the witness k that is either $\perp$ or satisfies the proof system relation $\mathsf{PK}_{\mathsf{Ch}}$. If the witness does not satisfy the proof relation, we return an empty message. The resulting experiment is defined as follows:

Experiment: $\mathsf{Exp}_3^{\mathsf{RUF}}(\lambda)$

$\ldots$
$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathsf{sid}, m), i):}$
14: $\ldots$
17: **elseif** $\mathsf{st}_{\mathsf{sid}} \neq \perp$ **then**
18: $\quad i \leftarrow \mathsf{st}_{\mathsf{sid}}, (K, \pi_{\mathsf{Ch}}) \leftarrow \mathsf{m}, \mathsf{st}_{\mathsf{sid}} \leftarrow \perp$
19: $\quad$ **if** $\mathsf{PK}_{\mathsf{Ch}}.\mathsf{Verify}((K, \mathsf{pk}_c^{\mathsf{Id}}), \pi_{\mathsf{Ch}}) = 0 \vee K = 0$ **return** $\perp$
20: $\quad \sigma' \leftarrow G_0 + r_i H_0 - e_i \sigma_i + \sum_{j\in[L]} x_j H_j$
21: $\quad k \leftarrow \mathcal{E}_{\mathsf{Ch}}^{\mathcal{A}(\cdot)}(\mathsf{trans}),$ **if** $k = \perp$ **then return** $\perp$
22: $\quad \forall j \in [L] : r_j \xleftarrow{\$} \mathbb{Z}_p,\ A_j \leftarrow r_j G_1,\ B_j \leftarrow r_j K + x_j G_1$
23: $\quad \pi_{\mathsf{Rd}} \leftarrow \mathcal{S}_{\mathsf{Rd}}(\mathsf{pp}, \mathsf{pk}_c^{\mathsf{Id}}, K, \sigma_i, \sigma', \{A_j, B_j\}_{j\in[L]})$
24: $\quad$ **return** $(\pi_{\mathsf{Rd}}, \sigma', \{A_j, B_j\}_{j\in[L]})$
25: **else return** $\emptyset$

It is obvious that $\mathsf{Exp}_3^{\mathsf{RUF}}$ and $\mathsf{Exp}_2^{\mathsf{RUF}}$ only differ if the extractor is not successful, which happens with negligible probability due to the knowledge soundness property of the challenge proof system. This allows us to use the difference lemma to conclude that

$$|\mathsf{Pr}[\mathsf{Exp}_3^{\mathsf{RUF}}(\lambda) = 1] - \mathsf{Pr}[\mathsf{Exp}_2^{\mathsf{RUF}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

Dummy ciphertexts if the decryption key is useless. Next, we further adapt the redemption oracle to randomly sample the second ciphertext component if the ElGamal key K is useless, i.e., if $K = kG_0$ (remember that ciphertexts are computed with respect to G_1). The resulting experiment is defined as follows:

Experiment: $\mathsf{Exp}_4^{\mathsf{RUF}}(\lambda)$

$\ldots$
$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathsf{sid}, m), i):}$
14: $\ldots$
17: **elseif** $\mathsf{st}_{\mathsf{sid}} \neq \perp$ **then**
18: $\quad i \leftarrow \mathsf{st}_{\mathsf{sid}}, (K, \pi_{\mathsf{Ch}}) \leftarrow \mathsf{m}, \mathsf{st}_{\mathsf{sid}} \leftarrow \perp$
19: $\quad$ **if** $\mathsf{PK}_{\mathsf{Ch}}.\mathsf{Verify}((K, \mathsf{pk}_c^{\mathsf{Id}}), \pi_{\mathsf{Ch}}) = 0 \vee K = 0$ **return** $\perp$
20: $\quad k \leftarrow \mathcal{E}_{\mathsf{Ch}}^{\mathcal{A}(\cdot)}(\mathsf{trans}),$ **if** $k = \perp$ **then return** $\perp$
21: $\quad \sigma' \leftarrow G_0 + r_i H_0 - e_i \sigma_i + \sum_{j\in[L]} x_j H_j$
22: $\quad$ **if** $K = kG_0$ **then**
23: $\quad\quad \forall j \in [L] : r_j, \bar{r}_j \xleftarrow{\$} \mathbb{Z}_p,\ A_j \leftarrow r_j G_1,\ B_j \leftarrow \bar{r}_j G_1$
24: $\quad$ **else then** (requires that $\mathsf{pk}_c^{\mathsf{Id}} = kG_1$)
25: $\quad\quad \forall j \in [L] : r_j \xleftarrow{\$} \mathbb{Z}_p,\ A_j \leftarrow r_j G_1,\ B_j \leftarrow r_j K + x_j G_1$

26: $\pi_{\mathsf{Rd}} \leftarrow \mathcal{S}_{\mathsf{Rd}}(\mathsf{pp}, \mathsf{pk}_c^{\mathsf{Id}}, K, \sigma_i, \sigma', \{A_j, B_j\}_{j\in[L]})$
27: **return** $(\pi_{\mathsf{Rd}}, \sigma', \{A_j, B_j\}_{j\in[L]})$
28: **else return** $\emptyset$

Indistinguishability between $\mathsf{Exp}_4^{\mathsf{RUF}}$ from $\mathsf{Exp}_3^{\mathsf{RUF}}$ follows intuitively from the semantic security of the ciphertexts and formally from the Decisional Diffie–Hellman assumption in $\mathbb{G}_1$.[4] To see this, note that (G_1, A_*, G_0, B_*) can be constructed from a DDH tuple $(G_1, A = aG_1, B = bG_1, C = cG_1)$ with c being either random or equal to (ab). To do so, we define $G_1 = G_1, A_* = A, G_0 = B, B_* = kC + x_*G_1$. If $c = ab$, it holds that $B_* = kaG_0 + \bar{x}_*G_1$, which makes the experiment derived from the DDH tuple equivalent to $\mathsf{Exp}_3^{\mathsf{RUF}}$. If c is random, it holds that B_* is distributed uniformly random, which makes the derived experiment equivalent to $\mathsf{Exp}_4^{\mathsf{RUF}}$. As we can only replace one ciphertext after another, we utilize a sequence of $n \cdot L$ hybrids, assuming that the adversary can query the redemption oracle at most n times. As the DDH assumption guarantees that the difference in the success probability between any two adjacent intermediate hybrids is at most negligible and there are only a polynomial number of hybrids between $\mathsf{Exp}_3^{\mathsf{RUF}}$ and $\mathsf{Exp}_4^{\mathsf{RUF}}$, we can conclude that the output distribution in $\mathsf{Exp}_4^{\mathsf{RUF}}$ and $\mathsf{Exp}_3^{\mathsf{RUF}}$ is computationally indistinguishable, i.e.,

$$|\mathsf{Pr}[\mathsf{Exp}_4^{\mathsf{RUF}}(\lambda) = 1] - \mathsf{Pr}[\mathsf{Exp}_3^{\mathsf{RUF}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

Assume encryption keys to be useless. In the next experiment, we get rid of the differentiation between $K = kG_0$ and $\mathsf{pk}_c^{\mathsf{Id}} = kG_1$ in the redemption oracle. We assume that $K = kG_0$ if the extractor for k is successful. The experiment is defined as follows:

Experiment: $\mathsf{Exp}_5^{\mathsf{RUF}}(\lambda)$
...
$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathsf{sid}, m), i):}$
14: ...
68: **elseif** $\mathsf{st}_{\mathsf{sid}} \neq \bot$ **then**
69: $(i, \mathsf{sk}_c^{\mathsf{Id}}, \mathsf{pk}_c^{\mathsf{Id}}) \leftarrow \mathsf{st}_{\mathsf{sid}}, (K, \pi_{\mathsf{Ch}}) \leftarrow \mathsf{m}, \mathsf{st}_{\mathsf{sid}} \leftarrow \bot$
70: **if** $\mathsf{PK}_{\mathsf{Ch}}.\mathsf{Verify}((K, \mathsf{pk}_c^{\mathsf{Id}}), \pi_{\mathsf{Ch}}) = 0 \vee K = 0$ **return** $\bot$
71: $k \leftarrow \mathcal{E}_{\mathsf{Ch}}^{\mathcal{A}(\cdot)}(\mathsf{trans})$, **if** $k = \bot$ **then return** $\bot$
72: $\sigma' \leftarrow G_0 + r_iH_0 - e_i\sigma_i + \sum_{j\in[L]} x_jH_j$
73: $\forall j \in [L] : r_j, \bar{r}_j \xleftarrow{\$} \mathbb{Z}_p,\ A_j \leftarrow r_jG_1,\ B_j \leftarrow \bar{r}_jG_1$
74: $\pi_{\mathsf{Rd}}^{\mathsf{sid}} \leftarrow \mathcal{S}_{\mathsf{Rd}}(\mathsf{pp}, \mathsf{pk}_c^{\mathsf{Id}}, K, \sigma_i, \sigma', \{A_j, B_j\}_{j\in[L]})$
75: **return** $(\pi_{\mathsf{Rd}}, \sigma', \{A_j, B_j\}_{j\in[L]})$
76: **else return** $\emptyset$

It is obvious that $\mathsf{Exp}_5^{\mathsf{RUF}}$ and $\mathsf{Exp}_4^{\mathsf{RUF}}$ only differ if $K \neq kG_0$ resp. if $\mathsf{pk}_c^{\mathsf{Id}} = kG_1$. We call this event F. We can show that event F cannot happen expect with negligible probability via a reduction to the discrete logarithm problem. Note that, in case of event F, $k = \mathsf{sk}_c^{\mathsf{Id}}$ is the discrete logarithm of $\mathsf{pk}_c^{\mathsf{Id}}$ to the basis of G_1 and that $\mathsf{sk}_c^{\mathsf{Id}}$ is generated by the experiment but not used at all (redemption zero-knowledge proofs are simulated). The only information about $\mathsf{sk}^{\mathsf{Id}}$ available to the adversary is $\mathsf{pk}_c^{\mathsf{Id}}$. This allows us to use an adversary from which we can extract k such that $\mathsf{pk}_c^{\mathsf{Id}} = kG_1$ to solve the discrete logarithm problem in $\mathbb{G}_1$. As we assume the discrete logarithm assumption to be hard in $\mathbb{G}_1$, we can apply the difference lemma to conclude that

$$|\mathsf{Pr}[\mathsf{Exp}_5^{\mathsf{RUF}}(\lambda) = 1] - \mathsf{Pr}[\mathsf{Exp}_4^{\mathsf{RUF}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

Simulate requests proofs. Next, we tackle the request oracle and create the request proofs returned by the oracle with simulator $\mathcal{S}_{\mathsf{Rq}}$ instead of the proof system's witnesses. The simulator receives the proof's statement $(\mathsf{pp}, \mathsf{pk}_c^{\mathsf{In}}, T)$ and has the capability of programming the random oracle. It creates a valid zero knowledge proof π_{Rq}, except with negligible probability. The resulting experiment is defined as follows:

Experiment: $\mathsf{Exp}_6^{\mathsf{RUF}}(\lambda)$
...
$\underline{\mathcal{O}^{\mathsf{Qu}}(i, \mathsf{pk}):}$
6: **if** $T_i \neq$ **null then return** $\bot$
7: Select $x_1, \ldots, x_L \in [0, 2^\ell - 1]$ such that $\mathsf{sk}_c^{\mathsf{In}} = \sum_{j\in[L]} x_j \cdot 2^{(j-1)\ell}$ holds without modulo reduction (This representation is unique and independent of i)
8: $r_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p,\ C_i = G_0 + r_iH_0 + \sum_{j\in[L]} x_jH_j,\ \mathsf{pk}_i \leftarrow \mathsf{pk}$
9: $T_i \leftarrow \delta_iC_i,\ \pi_i \leftarrow \mathcal{S}_{\mathsf{Rq}}(\mathsf{pp}, \mathsf{pk}_c^{\mathsf{In}}, T_i)$
10: **return** (T_i, π_i)
...

In $\mathsf{Exp}_5^{\mathsf{RUF}}$, the requests proofs are always valid due to the completeness property of the proof system. Therefore, it follows that the experiments $\mathsf{Exp}_6^{\mathsf{RUF}}$ and $\mathsf{Exp}_5^{\mathsf{RUF}}$ only differ if the simulator is not successful, which happens with at most negligible probability due to the zero-knowledge property of the request proof system. We use the difference lemma to conclude that

$$|\mathsf{Pr}[\mathsf{Exp}_6^{\mathsf{RUF}}(\lambda) = 1] - \mathsf{Pr}[\mathsf{Exp}_5^{\mathsf{RUF}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

Simplification. In a bridging step, we simplify the request oracle by sampling $C_i \xleftarrow{\$} \mathbb{G}_1$ instead of computing $C_i \leftarrow G_0 + r_iH_0 + \sum_{j\in[L]} x_jH_\varphi$. As the uniformly randomly sampled mask r_i acts as a one-time pad and is not used further, C_i is distributed uniformly random in both experiments. The game is defined as follows:

Experiment: $\mathsf{Exp}_7^{\mathsf{RUF}}(\lambda)$
...
$\underline{\mathcal{O}^{\mathsf{Qu}}(i, \mathsf{pk}_s):}$
6: **if** $T_i \neq$ **null then return** $\bot$
7: $r_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p,\ C_i \xleftarrow{\$} \mathbb{G}_1,\ \mathsf{pk}_i \leftarrow \mathsf{pk},\ T_i \leftarrow \delta_iC_i,\ \pi_i \leftarrow \mathcal{S}_{\mathsf{Rq}}(\mathsf{pp}, \mathsf{pk}_c^{\mathsf{In}}, T_i)$
8: **return** (T_i, π_i)
...

As the simplification does not affect the output distribution, we conclude that

$$\mathsf{Pr}[\mathsf{Exp}_7^{\mathsf{RUF}}(\lambda) = 1] = \mathsf{Pr}[\mathsf{Exp}_6^{\mathsf{RUF}}(\lambda) = 1].$$

4. A direct reduction to the ElGamal cryptosystem is cumbersome as the second ciphertext component B needs to be multiplied by k to transform the ciphertext component $B = r\mathsf{pk} + M$ received from the security game into the component $B' = rkG_0 + M'$ ($\mathsf{pk} = G_0$) compatible with our security game. This obviously tampers with the message M.

Extract request proof. We adapt the main thread to extract the witness of the request proof π^* after successfully verifying the proof. To do so, we employ extractor $\mathcal{E}_{\mathsf{Rq}}$ of the request proof system. The extractor has access to the adversary $\mathcal{A}$, receives the transcript trans of the experiment execution up to the extraction including the adversary's random tape, and has the power to program the random oracle. It extracts the witnesses $(\{x_j\}_{j\in[L]}, r, \delta)$ that are either all $\perp$ or satisfy the proof system's relation $\mathsf{PK}_{\mathsf{Rq}}$. The resulting game is defined as follows:

Experiment: $\mathsf{Exp}_8^{\mathsf{RUF}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\mathsf{sk}_c^{\mathsf{In}}, \mathsf{pk}_c^{\mathsf{In}}, \mathsf{sk}_c^{\mathsf{Id}}, \mathsf{pk}_c^{\mathsf{Id}}) \leftarrow \mathsf{C.KeyGen}(\mathsf{pp})$
$((T^*, \pi^*), \mathsf{sk}_s) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Qu}}, \mathcal{O}^{\mathsf{Fin}}, \mathcal{O}^{\mathsf{Rd}}}(\mathsf{pp}, \mathsf{pk}_c^{\mathsf{In}}, \mathsf{pk}_c^{\mathsf{Id}})$
if $\exists i$ s.t. $(T^*, \pi^*) = (T_i, \pi_i)$ **then return** 0
if $\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Verify}(\mathsf{rpp}, (\mathsf{pk}_c^{\mathsf{In}}, T^*), \pi^*) = 0$ **then return** 0
$(\{x_j\}_{j\in[L]}, r, \delta) \leftarrow \mathcal{E}_{\mathsf{Rq}}^{\mathcal{A}(\cdot)}(\mathsf{trans})$, **if** $r = \perp$ **then return** 0
return 1
...

It is obvious that $\mathsf{Exp}_8^{\mathsf{RUF}}$ and $\mathsf{Exp}_7^{\mathsf{RUF}}$ only differ if the extractor is not successful. Note that, we ensure that the request proof has not been issued by the request oracle. Therefore, it needs to originate from the adversary such that the proofs issued by the query oracle cannot tamper with the extraction. As the extractor is successful expect with negligible probability, due to the knowledge soundness property, we can apply the difference lemma to conclude that

$$|\mathsf{Pr}[\mathsf{Exp}_8^{\mathsf{RUF}}(\lambda) = 1] - \mathsf{Pr}[\mathsf{Exp}_7^{\mathsf{RUF}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

Reduction to the discrete logarithm assumption. If the adversary $\mathcal{A}$ wins $\mathsf{Exp}_8^{\mathsf{RUF}}$, then the game extracted $\{x_j\}_{j\in[L]}$ from $\mathcal{A}$ such that $\mathsf{pk}_c^{\mathsf{In}} = (\sum_{j\in[L]} 2^{(j-1)\cdot\ell} x_j) G_1$. Therefore, we can use $\mathcal{A}$ to compute the discrete logarithm of $\mathsf{pk}_c^{\mathsf{In}}$ to the basis of G_1 which contradicts the discrete logarithm assumption in $\mathbb{G}_1$. Formally, we define an adversary $\mathcal{B}$, that receives a discrete logarithm challenge $(G, H) \in \mathbb{G}_1^2$, internally executes $\mathcal{A}$, and returns $x = \mathsf{dlog}_G(H)$ with probability $\mathsf{Pr}[\mathsf{Exp}_8^{\mathsf{RUF}}(\lambda) = 1]$. $\mathcal{B}$ is defined as follows:

- Execute $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and send $(\mathbb{G}_1, p)$ to the discrete logarithm challenger to receive (G, H).
- Overwrite $G_1 \leftarrow G$ and sample $\mathsf{pk}^{\mathsf{Id}} \xleftarrow{\$} \mathbb{G}_1$.
- Execute $((T^*, \pi^*), \cdot) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Qu}}, \mathcal{O}^{\mathsf{Fin}}, \mathcal{O}^{\mathsf{Rd}}}(\mathsf{pp}, H, \mathsf{pk}^{\mathsf{Id}})$.
  - Answer oracle queries as defined by experiment $\mathsf{Exp}_8^{\mathsf{RUF}}$.
- If $\exists i$ s.t. $(T^*, \pi^*) = (T_i, \pi_i)$ or $\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Verify}(\mathsf{rpp}, (\mathsf{pk}_c^{\mathsf{In}}, T^*), \pi^*) = 0$ then return $\perp$.
- Otherwise, employ the request proof extractor to compute $(\{x_j\}_{j\in[L]}, r, \delta) \leftarrow \mathcal{E}_{\mathsf{Rq}}^{\mathcal{A}(\cdot)}(\mathsf{trans})$.
- If $r = \perp$ then return $\perp$.
- Otherwise, return $x = \sum_{j\in[L]} 2^{(j-1)\cdot\ell} x_j$.

First note that the view of $\mathcal{A}$ in the experiment simulated by $\mathcal{B}$ is identical to the view of $\mathcal{A}$ in experiment $\mathsf{Exp}_8^{\mathsf{RUF}}$. The public parameters, the insurance public key, and the oracle responses are distributed identically in both experiments. By definition of the extractor $\mathcal{E}_{\mathsf{Rq}}$ it needs to hold that either $r = \perp$ (extraction fails) or $\mathsf{pk}_c^{\mathsf{In}} = (\sum_{j\in[L]} 2^{(j-1)\cdot\ell} x_j) G_1$. Further, if $\mathcal{A}$ wins $\mathsf{Exp}_8^{\mathsf{RUF}}$ it needs to hold that $r \neq \perp$ and, therefore, that $x = \sum_{j\in[L]} 2^{(j-1)\cdot\ell} x_j$ is the discrete logarithm of $\mathsf{pk}_c^{\mathsf{In}}$. It follows that $\mathcal{B}$ wins the discrete logarithm experiment if $\mathcal{A}$ would win $\mathsf{Exp}_8^{\mathsf{RUF}}$. As we assume that no PPT adversary wins the discrete logarithm experiment in $\mathbb{G}_1$ expect with negligible probability, it follows that

$$\mathsf{Pr}[\mathsf{Exp}_8^{\mathsf{RUF}}(\lambda) = 1] \leq \mathsf{negl}(\lambda).$$

Putting all steps together. As we use a polynomial number of game hops (in the number of oracle queries and the chunk size L), the difference in the success probability of the adversary in any two adjacent experiments is at most negligible, and the adversary can win in the final game $\mathsf{Exp}_{10}^{\mathsf{RUF}}$ with at most negligible probability, we can conclude that

$$\mathsf{Pr}[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{RUF}}(\lambda) = 1] \leq \mathsf{negl}(\lambda).$$

F.3. Identity Unforgeability

We prove identity unforgeability via a sequence of games. Let $\mathsf{Exp}_0^{\mathsf{IUF}} = \mathsf{Exp}_{\mathcal{A}}^{\mathsf{IUF}}$ be the initial security game. The subsequent games are defined as follows.

Inline construction. In the first game, we inline our concrete construction into the security game. The game is defined as follows:

Experiment: $\mathsf{Exp}_1^{\mathsf{IUF}}(\lambda)$

$\mathsf{res} \leftarrow 0, \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda), (\cdot, \cdot, \mathsf{sk}_c^{\mathsf{Id}}, \mathsf{pk}_c^{\mathsf{Id}}) \leftarrow \mathsf{C.KeyGen}(\mathsf{pp})$
$(\mathsf{pk}_s, \mathsf{st}^{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Qu}}, \mathcal{O}^{\mathsf{Fin}}, \mathcal{O}^{\mathsf{Rd}}}(1, \mathsf{pp}, \mathsf{pk}_c^{\mathsf{Id}}),\ \mathcal{A}^{\mathcal{O}^*}(2, \mathsf{st}^{\mathcal{A}})$
return res

$\underline{\mathcal{O}^{\mathsf{Qu}}(i, \mathsf{pk}_c^{\mathsf{In}}, \mathsf{sk}_c^{\mathsf{In}}, \mathsf{pk}):}$
if $T_i \neq \mathbf{null} \vee \mathsf{KeyV}(\mathsf{pp}, \mathsf{sk}_c^{\mathsf{In}}, \mathsf{pk}_c^{\mathsf{In}}) = 0$ **then return** $\perp$
Select $x_1^i, \ldots, x_L^i \in [0, 2^\ell - 1]$ such that $\mathsf{sk}_c^{\mathsf{In}} = \sum_{j\in[L]} x_j \cdot 2^{(j-1)\ell}$ holds without modulo reduction (This representation is unique and independent of i).
$r_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p,\ C_i = G_0 + r_i H_0 + \sum_{j\in[L]} x_j^i H_j,\ \mathsf{pk}_i \leftarrow \mathsf{pk}$
$T_i \leftarrow \delta_i C_i, \pi_i \leftarrow \mathsf{PK}_{\mathsf{Rq}}.\mathsf{Prove}(\mathsf{rpp}, (\mathsf{pk}_c^{\mathsf{In}}, T_i), (\{x_j^i\}_{j\in[L]}, r_i, \delta_i))$
return (T_i, π_i)

$\underline{\mathcal{O}^{\mathsf{Fin}}(i, \mathsf{resp} = (e, S)):}$
if $T_i = \mathbf{null} \vee \sigma_i \neq \mathbf{null}$ **then return** $\perp$
if $\mathsf{e}(S, \mathsf{pk}_i + eG_2) \neq \mathsf{e}(T_i, G_2)$ **then** $\sigma_i \leftarrow \perp$, **return** σ_i
else $\sigma_i \leftarrow \frac{1}{\delta_i} S,\ e_i \leftarrow e,$ **return** σ_i

$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathsf{sid}, \mathsf{m}):}$
if $\mathsf{st}_{\mathsf{sid}} = \mathbf{null}$ **then**
  $i \leftarrow \mathsf{m}$, **if** $\sigma_i \in \{\mathbf{null}, \perp\}$ **then return** $\perp$
  $\mathsf{st}_{\mathsf{sid}} \leftarrow i,$ **return** $(\mathsf{pk}_c^{\mathsf{Id}}, \sigma_i)$
elseif $\mathsf{st}_{\mathsf{sid}} \neq \perp$ **then**
  $i \leftarrow \mathsf{st}_{\mathsf{sid}}, (K, \pi_{\mathsf{Ch}}) \leftarrow \mathsf{m}, \mathsf{st}_{\mathsf{sid}} \leftarrow \perp$
  if $\mathsf{PK}_{\mathsf{Ch}}.\mathsf{Verify}((K, \mathsf{pk}_c^{\mathsf{Id}}), \pi_{\mathsf{Ch}}) = 0 \vee K = 0$ **return** $\perp$
  $\sigma' \leftarrow G_0 + r_i H_0 - e_i \sigma + \sum_{j\in[L]} x_j^i H_j$
  $\forall j \in [L] : r_j \xleftarrow{\$} \mathbb{Z}_p,\ A_j \leftarrow r_j G_1,\ B_j \leftarrow r_j K + x_j^i G_1$
  $\pi_{\mathsf{Rd}} \leftarrow \mathsf{PK}_{\mathsf{Rd}}.\mathsf{Prove}((\mathsf{pk}_c^{\mathsf{Id}}, K, \sigma_i, \sigma', \{A_j, B_j\}_{j\in[L]}), (r_i, e_i, \mathsf{sk}_c^{\mathsf{Id}}, \{r_j, x_j^i\}_{j\in[L]}))$
  return $(\pi_{\mathsf{Rd}}, \sigma', \{A_j, B_j\}_{j\in[L]})$
else return $\emptyset$

$\underline{\mathcal{O}^*(\mathsf{m}):}$

23: **if** $\mathsf{st} \in \{0,1\}$ **then return** $\bot$
24: **if** $\mathsf{st} = \textbf{null}$ **then**
25: $(\sigma^*, \cdot) \leftarrow m, k^* \xleftarrow{\$} \mathbb{Z}_p, K^* \leftarrow k^* G_0, \mathsf{st} \leftarrow (\sigma^*, \mathsf{pk}_s, K^*)$
26: $\pi_{\mathsf{Ch}} \leftarrow \mathsf{PK}_{\mathsf{Ch}}.\mathsf{Prove}((K^*, \mathsf{pk}_c^{\mathsf{Id}}), k^*)$
27: **return** (K^*, π_{Ch})
28: **else**
29: $(\pi^*, \sigma'_*, \{A_i^*, B_i^*\}_{i\in[L]}) \leftarrow \mathsf{m}$
30: **if** $\mathsf{e}(\sigma^*, \mathsf{pk}_s) \neq \mathsf{e}(\sigma'_*, G_2)$ **then** $\mathsf{st}_{\mathsf{sid}} \leftarrow 0$, **return** 0
31: **if** $\mathsf{PK}_{\mathsf{Rd}}.\mathsf{Verify}((\mathsf{pk}_c^{\mathsf{Id}}, K^*, \sigma^*, \sigma'_*, \{A_i^*, B_i^*\}_{i\in[L]}), \pi^*) = 0$ **then** $\mathsf{st} \leftarrow 0$, **return** 0
32: $(\mathsf{res}, \mathsf{st}) \leftarrow (1,1)$, **return** 1

This step does not affect the output distribution. It follows that

$$\mathsf{Pr}[\mathsf{Exp}_1^{\mathsf{IUF}}(\lambda) = 1] = \mathsf{Pr}[\mathsf{Exp}_0^{\mathsf{IUF}}(\lambda) = 1].$$

Simulate redemption proofs. Analogously to $\mathsf{Exp}_2^{\mathsf{RUF}}$, we use the redemption proof's simulator to create redemption proofs π_{Rd} within the redemption oracle. The resulting experiment is defined as follows:

Experiment: $\mathsf{Exp}_2^{\mathsf{IUF}}(\lambda)$

...
$\mathcal{O}^{\mathsf{Rd}}(\mathsf{sid}, \mathsf{m})$:
12: ...
15: **elseif** $\mathsf{st}_{\mathsf{sid}} \neq \bot$ **then**
16: $i \leftarrow \mathsf{st}_{\mathsf{sid}}, (K, \pi_{\mathsf{Ch}}) \leftarrow \mathsf{m}, \mathsf{st}_{\mathsf{sid}} \leftarrow \bot$
17: **if** $\mathsf{PK}_{\mathsf{Ch}}.\mathsf{Verify}((K, \mathsf{pk}_c^{\mathsf{Id}}), \pi_{\mathsf{Ch}}) = 0$ **then return** $\bot$
18: $\sigma' \leftarrow G_0 + r_i H_0 - e_i \sigma + \sum_{j\in[L]} x_j^i H_j$
19: $\forall j \in [L] : r_j \xleftarrow{\$} \mathbb{Z}_p,\ A_j \leftarrow r_j G_1,\ B_j \leftarrow r_j K + x_j^i G_1$
20: $\pi_{\mathsf{Rd}} \leftarrow \mathcal{S}_{\mathsf{Rd}}(\mathsf{pp}, \mathsf{pk}_c^{\mathsf{Id}}, K, \sigma_i, \sigma', \{A_j^{\mathsf{sid}}, B_j^{\mathsf{sid}}\}_{j\in[L]})$
21: **return** $(\pi_{\mathsf{Rd}}, \sigma', \{A_j, B_j\}_{j\in[L]})$
22: **else return** $\emptyset$
...

Analogously to $\mathsf{Exp}_2^{\mathsf{RUF}}$ it can be shown that

$$|\mathsf{Pr}[\mathsf{Exp}_2^{\mathsf{IUF}}(\lambda) = 1] - \mathsf{Pr}[\mathsf{Exp}_1^{\mathsf{IUF}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$

based on the zero-knowledge property of the redemption proof system.

Extraction of the redemption proof. We adapt $\mathcal{O}^*$ to extract the witnesses of the redemption proof analogously to $\mathsf{Exp}_3^{\mathsf{OMUF}}$. The resulting experiment is defined as follows:

Experiment: $\mathsf{Exp}_3^{\mathsf{IUF}}(\lambda)$

...
$\mathcal{O}^*(\mathsf{m})$:
23: ...
28: **else**
29: $(\pi^*, \sigma'_*, \{A_i^*, B_i^*\}_{i\in[L]}) \leftarrow \mathsf{m}$
30: **if** $\mathsf{e}(\sigma^*, \mathsf{pk}_s) \neq \mathsf{e}(\sigma'_*, G_2)$ **then** $\mathsf{st}_{\mathsf{sid}} \leftarrow 0$, **return** 0
31: **if** $\mathsf{PK}_{\mathsf{Rd}}.\mathsf{Verify}((\mathsf{pk}_c^{\mathsf{Id}}, K^*, \sigma^*, \sigma'_*, \{A_i^*, B_i^*\}_{i\in[L]}), \pi^*) = 0$ **then** $\mathsf{st} \leftarrow 0$, **return** 0
32: $(\mathsf{sk}^*, r^*, e^*, \{r_i^*, x_i^*\}_{i\in[L]}) \leftarrow \mathcal{E}_{\mathsf{Rd}}^{\mathcal{A}(\cdot)}(\mathsf{trans})$
33: **if** $e^* = \bot$ **then** $\mathsf{st}_{\mathsf{sid}} \leftarrow 0$, **return** 0
34: $(\mathsf{res}, \mathsf{st}) \leftarrow (1,1)$, **return** 1

It is obvious that $\mathsf{Exp}_3^{\mathsf{IUF}}$ and $\mathsf{Exp}_2^{\mathsf{IUF}}$ only differ if the extractor is not successful. As the challenge K^* ensures that the zero-knowledge proof is fresh and cannot be one that has been send from the oracle to the adversary, we conclude from the knowledge soundness property of the redemption proof system that the extractor fails only with negligible probability. Therefore, we can apply the difference lemma to conclude that

$$|\mathsf{Pr}[\mathsf{Exp}_3^{\mathsf{IUF}}(\lambda) = 1] - \mathsf{Pr}[\mathsf{Exp}_2^{\mathsf{IUF}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

Reduction to the discrete logarithm assumption. In order to win the experiment, the adversary $\mathcal{A}$ has to provide a zero-knowledge proof of knowledge of secret key $\mathsf{sk}^* = \mathsf{dlog}_{G_1}(\mathsf{pk}_c^{\mathsf{Id}})$ to the oracle $\mathcal{O}^*$. However, it does not receive any information about sk^* beyond $\mathsf{pk}_c^{\mathsf{Id}}$. This is due to the fact that the redemption proofs in oracle $\mathcal{O}^{\mathsf{Rd}}$ are simulated. Hence, we can use $\mathcal{A}$ to compute the discrete logarithm of $\mathsf{pk}_c^{\mathsf{Id}}$ to the basis of G_1 which contradicts the discrete logarithm assumption in $\mathbb{G}_1$. Formally, we define an adversary $\mathcal{B}$, that receives a discrete logarithm challenge $(G, H) \in \mathbb{G}_1^2$, internally executes $\mathcal{A}$, and returns $x = \mathsf{dlog}_G(H)$ with probability $\mathsf{Pr}[\mathsf{Exp}_3^{\mathsf{IUF}}(\lambda) = 1]$. $\mathcal{B}$ is defined as follows:

- Execute $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and send $(\mathbb{G}_1, p)$ to the discrete logarithm challenger to receive (G, H).
- Overwrite $G_1 \leftarrow G$.
- Execute $(\mathsf{pk}_s, \mathsf{st}^{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{Qu}}, \mathcal{O}^{\mathsf{Fin}}, \mathcal{O}^{\mathsf{Rd}}}(1, \mathsf{pp}, H)$
  - Answer oracle queries as defined by experiment $\mathsf{Exp}_3^{\mathsf{IUF}}$.
- Execute $\mathcal{A}^{\mathcal{O}^*}(2, \mathsf{st}^{\mathcal{A}})$.
  - Answer oracle queries as defined by experiment $\mathsf{Exp}_3^{\mathsf{IUF}}$.
- If sk^* is not defined or equals $\bot$ then return $\bot$.
- Otherwise return sk^*.

First note that the view of $\mathcal{A}$ in the experiment simulated by $\mathcal{B}$ is identical to the view of $\mathcal{A}$ in experiment $\mathsf{Exp}_3^{\mathsf{IUF}}$. The public parameters, the identification public key, and the oracle responses are distributed identically in both experiments. Further, note that $\mathcal{A}$ can only win in $\mathsf{Exp}_3^{\mathsf{IUF}}$ if extractor $\mathcal{E}_{\mathsf{Rd}}$ successfully extracts a witness $(\mathsf{sk}^*, \dots)$ from the redemption proof such that $H = \mathsf{sk}^* G_1$. We conclude that $\mathcal{B}$ wins the discrete logarithm experiment exactly if $\mathcal{A}$ would win $\mathsf{Exp}_3^{\mathsf{IUF}}$. As no PPT adversary can win the discrete logarithm experiment expect with negligible probability by assumption, it follows that

$$\mathsf{Pr}[\mathsf{Exp}_3^{\mathsf{IUF}}(\lambda) = 1] \leq \mathsf{negl}(\lambda).$$

Putting all steps together. As we use a polynomial number of game hops (in the number of oracle queries), the difference in the success probability of the adversary in any two adjacent experiments is at most negligible, and the adversary can win in the final game $\mathsf{Exp}_3^{\mathsf{IUF}}$ with at most negligible probability, we can conclude that

$$\mathsf{Pr}[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{IUF}}(\lambda) = 1] \leq \mathsf{negl}(\lambda).$$

Appendix G. Proof of Theorem 5

This section provides a detailed proof of Theorem 5. To show unlinkability, we have to show that for every PPT adversaries $\mathcal{A}$ it holds that

$$|\mathsf{Pr}[\mathsf{Exp}_{\mathcal{A}}^{\mathsf{UL}}(\lambda) = 1] - \frac{1}{2}| \leq \mathsf{negl}(\lambda).$$

We do so via a sequence of games. Let $\mathcal{A}$ be an arbitrary PPT adversaries and $\mathsf{Exp}_0^{\mathsf{UL}} = \mathsf{Exp}_{\mathcal{A}}^{\mathsf{UL}}$ be the initial security game. The subsequent games are defined as follows.

Inline construction. In the first game, we inline our construction into the security game. The game is defined as follows:

Experiment: $\mathsf{Exp}_1^{\mathsf{UL}}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ j' \leftarrow 0,\ b \xleftarrow{\$} \{0,1\},\ \Gamma \leftarrow \emptyset$
$(\mathsf{pk}_s, \mathsf{st}_0^{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KG}}}(0, \mathsf{pp})$
$(\alpha, \beta, \mathsf{st}_1^{\mathcal{A}}) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KG}}, \mathcal{O}^{\mathsf{Rq}}, \mathcal{O}^{\mathsf{Fin}}, \mathcal{O}^{\mathsf{Rd}}}(1, \mathsf{st}_0^{\mathcal{A}})$
if $\exists k \in \{\alpha, \beta\}$ s.t. $\sigma_k \in \{\mathbf{null}, \bot\} \vee k \in \Gamma$ **then abort**
if $b = 1$ **then** swap values of $(\sigma_\alpha, \omega_\alpha)$ and $(\sigma_\beta, \omega_\beta)$
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KG}}, \mathcal{O}^{\mathsf{Rq}}, \mathcal{O}^{\mathsf{Fin}}, \mathcal{O}^{\mathsf{Rd}}}(2, \mathsf{st}_1^{\mathcal{A}})$
if $b = b'$ **then return** 1 **else return** 0

$\underline{\mathcal{O}^{\mathsf{KG}}():}$

$j' \leftarrow j' + 1,\ (\mathsf{sk}_{j'}^{\mathsf{Id}}, \mathsf{pk}_{j'}^{\mathsf{Id}}, \mathsf{sk}_{j'}^{\mathsf{In}}, \mathsf{pk}_{j'}^{\mathsf{In}}) \leftarrow \mathsf{C.KeyGen}(\mathsf{pp})$
return $(\mathsf{pk}_{j'}^{\mathsf{Id}}, \mathsf{sk}_{j'}^{\mathsf{In}}, \mathsf{pk}_{j'}^{\mathsf{In}})$

$\underline{\mathcal{O}^{\mathsf{Rq}}(i, j):}$

if $T_i \neq \mathbf{null} \vee j > j'$ **then abort**
Split bit representation of $\mathsf{sk}_j^{\mathsf{In}}$ into $x_1^i, \ldots, x_L^i \in [0, 2^\ell - 1]$ such that $\mathsf{sk}_j^{\mathsf{In}} = \sum_{\varphi \in [L]} 2^{(\varphi-1)\ell} x_\varphi$ without modulo reduction.
$r_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p,\ C_i = G_0 + r_i H_0 + \sum_{\varphi \in [L]} x_\varphi H_\varphi,\ T_i \leftarrow \delta_i C_i$
$\pi_i \leftarrow \mathsf{PK}_{\mathsf{Rq}}.\mathsf{Prove}(\mathsf{rpp}, (\mathsf{pk}_j^{\mathsf{In}}, T_i), (\{x_\varphi\}_{\varphi \in [L]}, r_i, \delta_i))$
return (T_i, π_i)

$\underline{\mathcal{O}^{\mathsf{Fin}}(i, \mathsf{resp} = (e, S)):}$

if $T_i = \mathbf{null} \vee \sigma_i \neq \mathbf{null}$ **then return** $\bot$
if $\mathsf{e}(S, \mathsf{pk}_s + eG_2) \neq \mathsf{e}(T_i, G_2)$ **then** $\sigma_i \leftarrow \bot$, **return** σ_i
else $\sigma_i \leftarrow \frac{1}{\delta_i} S,\ e_i \leftarrow e,$ **return** 1

$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathsf{sid}, \mathsf{m}):}$

if $\mathsf{st}_{\mathsf{sid}} = \mathbf{null}$ **then**
  $(i, j) \leftarrow \mathsf{m}$, **if** $\sigma_i \in \{\mathbf{null}, \bot\} \vee \mathsf{pk}_j^{\mathsf{Id}} = \mathbf{null}$ **then return** $\bot$
  $\Gamma \leftarrow \Gamma \cup \{i\}, \mathsf{st}_{\mathsf{sid}} \leftarrow (i, j),$ **return** $(\mathsf{pk}_j^{\mathsf{Id}}, \sigma_i)$
elseif $\mathsf{st}_{\mathsf{sid}} \neq \bot$ **then**
  $(i, j) \leftarrow \mathsf{st}_{\mathsf{sid}}, (K, \pi_{\mathsf{Ch}}) \leftarrow \mathsf{m}, \mathsf{st}_{\mathsf{sid}} \leftarrow \bot$
  if $\mathsf{PK}_{\mathsf{Ch}}.\mathsf{Verify}((K, \mathsf{pk}_j^{\mathsf{Id}}), \pi_{\mathsf{Ch}}) = 0 \vee K = 0$ **return** $\bot$
  $\sigma' \leftarrow G_0 + r_i H_0 - e_i \sigma + \sum_{\phi \in [L]} x_\phi^i H_i$
  $\forall \varphi \in [L] : r_\varphi \xleftarrow{\$} \mathbb{Z}_p,\ A_\varphi \leftarrow r_\varphi G_1,\ B_\varphi \leftarrow r_\varphi K + x_\varphi^i G_1$
  $\pi_{\mathsf{Rd}} \leftarrow \mathsf{PK}_{\mathsf{Rd}}.\mathsf{Prove}((\mathsf{pk}_j^{\mathsf{Id}}, K, \sigma_i, \sigma', \{A_\varphi, B_\varphi\}_{\varphi \in [L]}), (\mathsf{sk}_j^{\mathsf{Id}}, r_i, e_i, \{r_\varphi, x_\varphi^i\}_{\varphi \in [L]}))$
  return $(\pi_{\mathsf{Rd}}, \sigma', \{A_\varphi, B_\varphi\}_{\varphi \in [L]})$
else return $\emptyset$

This step does not change the output distribution. It follows that

$$\mathsf{Pr}[\mathsf{Exp}_1^{\mathsf{UL}}(\lambda) = 1] = \mathsf{Pr}[\mathsf{Exp}_0^{\mathsf{UL}}(\lambda) = 1].$$

Simplify the redemption oracle. Analogously to $\mathsf{Exp}_2^{\mathsf{RUF}}$, $\mathsf{Exp}_3^{\mathsf{RUF}}$, $\mathsf{Exp}_4^{\mathsf{RUF}}$ and $\mathsf{Exp}_5^{\mathsf{RUF}}$, we define games $\mathsf{Exp}_2^{\mathsf{UL}}$, $\mathsf{Exp}_3^{\mathsf{UL}}$, $\mathsf{Exp}_4^{\mathsf{UL}}$, and $\mathsf{Exp}_5^{\mathsf{UL}}$ that simplify the redemption oracle as follows:

- $\mathsf{Exp}_2^{\mathsf{UL}}$ creates the redemption proofs using the proof system's simulator $\mathcal{S}_{\mathsf{Rd}}$ instead of creating them honestly.
- $\mathsf{Exp}_3^{\mathsf{UL}}$ employs an extractor for the challenge proof to extract k such that $K = kG_1$ or $\mathsf{pk}_j^{\mathsf{Id}} = kG_1$ after π_{Ch} has been verified successfully.
- $\mathsf{Exp}_4^{\mathsf{UL}}$ randomly samples the second ciphertext component if $K = kG_1$. If $\mathsf{pk}_j^{\mathsf{Id}} = kG_1$, the ciphertexts are still created correctly.
- $\mathsf{Exp}_5^{\mathsf{UL}}$ removes the $\mathsf{pk}_j^{\mathsf{Id}} = kG_1$ case, i.e., it always creates ciphertexts on randomly sampled dummy values.

$\mathsf{Exp}_5^{\mathsf{UL}}$ is defined as follows:

Experiment: $\mathsf{Exp}_5^{\mathsf{UL}}(\lambda)$

$\ldots$
$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathsf{sid}, \mathsf{m}):}$
$\ldots$
elseif $\mathsf{st}_{\mathsf{sid}} \neq \bot$ **then**
  $(i, j) \leftarrow \mathsf{st}_{\mathsf{sid}}, (K, \pi_{\mathsf{Ch}}) \leftarrow \mathsf{m}, \mathsf{st}_{\mathsf{sid}} \leftarrow \bot$
  if $\mathsf{PK}_{\mathsf{Ch}}.\mathsf{Verify}((K, \mathsf{pk}_j^{\mathsf{Id}}), \pi_{\mathsf{Ch}}) = 0 \vee K = 0$ **return** $\bot$
  $k \leftarrow \mathcal{E}_{\mathsf{Ch}}^{\mathcal{A}(\cdot)}$, **if** $k = \bot$ **then return** $\bot$.
  $\forall \varphi \in [L] : r_\varphi, \bar{r}_\phi \xleftarrow{\$} \mathbb{Z}_p,\ A_\varphi \leftarrow r_\varphi G_1,\ B_\varphi \leftarrow \bar{r}_\phi G_1$
  $\pi_{\mathsf{Rd}}^{\mathsf{sid}} \leftarrow \mathcal{S}_{\mathsf{Rd}}(\mathsf{pp}, \mathsf{pk}_j^{\mathsf{Id}}, K, \sigma_i, \{A_\varphi^{\mathsf{sid}}, B_\varphi^{\mathsf{sid}}\}_{\varphi \in [L]})$
  return $(\pi_{\mathsf{Rd}}, \{A_\varphi, B_\varphi\}_{\varphi \in [L]})$
else return $\emptyset$

Analogously to $\mathsf{Exp}_2^{\mathsf{RUF}}$, $\mathsf{Exp}_3^{\mathsf{RUF}}$, $\mathsf{Exp}_4^{\mathsf{RUF}}$, and $\mathsf{Exp}_5^{\mathsf{RUF}}$, it can be shown that

$$|\mathsf{Pr}[\mathsf{Exp}_2^{\mathsf{UL}}(\lambda) = 1] - \mathsf{Pr}[\mathsf{Exp}_1^{\mathsf{UL}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$
$$|\mathsf{Pr}[\mathsf{Exp}_3^{\mathsf{UL}}(\lambda) = 1] - \mathsf{Pr}[\mathsf{Exp}_2^{\mathsf{UL}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$
$$|\mathsf{Pr}[\mathsf{Exp}_4^{\mathsf{UL}}(\lambda) = 1] - \mathsf{Pr}[\mathsf{Exp}_2^{\mathsf{UL}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$
$$|\mathsf{Pr}[\mathsf{Exp}_5^{\mathsf{UL}}(\lambda) = 1] - \mathsf{Pr}[\mathsf{Exp}_2^{\mathsf{UL}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

Simplify the request oracle. Analogously to $\mathsf{Exp}_6^{\mathsf{RUF}}$, we define game $\mathsf{Exp}_6^{\mathsf{UL}}$ that simplifies the request oracle by using the proof system's simulator $\mathcal{S}_{\mathsf{Rq}}$ to create request proofs instead of creating them honestly. $\mathsf{Exp}_6^{\mathsf{UL}}$ is defined as follows:

Experiment: $\mathsf{Exp}_6^{\mathsf{UL}}(\lambda)$

$\ldots$
$\underline{\mathcal{O}^{\mathsf{Qu}}(i, j):}$
if $T_i \neq \mathbf{null} \vee j > j'$ **then abort**
Split bit representation of $\mathsf{sk}_j^{\mathsf{In}}$ into $x_1^i, \ldots, x_L^i \in [0, 2^\ell - 1]$ such that $\mathsf{sk}_j^{\mathsf{In}} = \sum_{\varphi \in [L]} 2^{(\varphi-1)\ell} x_\varphi$ without modulo reduction.
$r_i, \delta_i \xleftarrow{\$} \mathbb{Z}_p,\ C_i = G_0 + r_i H_0 + \sum_{\varphi \in [L]} x_\varphi^i H_\varphi,\ \mathsf{pk}_i \leftarrow \mathsf{pk}$
$T_i \leftarrow \delta_i C_i,\ \pi_i \leftarrow \mathcal{S}_{\mathsf{Rq}}(\mathsf{pp}, \mathsf{pk}_j^{\mathsf{In}}, T_i)$
return (T_i, π_i)
$\ldots$

Analogously to $\mathsf{Exp}_6^{\mathsf{RUF}}$, it can be shown that

$$|\mathsf{Pr}[\mathsf{Exp}_6^{\mathsf{UL}}(\lambda) = 1] - \mathsf{Pr}[\mathsf{Exp}_5^{\mathsf{UL}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$

based on the zero-knowledge property of the request proof system.

Simplification. In a bridging step, we simplify the experiment without changing the output distribution.

- We compute $C_i \xleftarrow{\$} \mathbb{G}_1$ instead of $C_i \leftarrow G_0 + r_i H_0 + \sum_{\varphi \in [L]} x_\varphi^i H_\varphi$. As the uniformly random sampled mask r_i acts as a one-time pad and is not used further, C_i is distributed uniformly random in both

experiments. This means that the change in the computation of C_i does not alter the output distribution.

- We remove the assignment of the token witness, i.e., the storage of variables $r_i, e_i, \{x^i_\varphi\}_{\varphi\in[L]}$ from the experiments. As these variables are used at most once (r_i and x^i_φ are not used at all), they can be discarded after being used without affecting the output distribution.

$\mathsf{Exp}^{\mathsf{UL}}_7$ is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{UL}}_7(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ j' \leftarrow 0,\ b \xleftarrow{\$} \{0,1\},\ \Gamma \leftarrow \emptyset$
$(\mathsf{pk}_s, \mathsf{st}^{\mathcal{A}}_0) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KG}}}(0, \mathsf{pp})$
$(\alpha, \beta, \mathsf{st}^{\mathcal{A}}_1) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KG}},\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Fin}},\mathcal{O}^{\mathsf{Rd}}}(1, \mathsf{st}^{\mathcal{A}}_0)$
if $\exists k \in \{\alpha, \beta\}$ s.t. $\sigma_k \in \{\mathbf{null}, \perp\} \vee k \in \Gamma$ **then abort**
if $b = 1$ **then** swap values of (σ_α) and (σ_β)
$b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KG}},\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Fin}},\mathcal{O}^{\mathsf{Rd}}}(2, \mathsf{st}^{\mathcal{A}}_1)$
if $b = b'$ **then return** 1 **else return** 0

$\underline{\mathcal{O}^{\mathsf{KG}}():}$

$j' \leftarrow j' + 1,\ (\mathsf{sk}^{\mathsf{Id}}_{j'}, \mathsf{pk}^{\mathsf{Id}}_{j'}, \mathsf{sk}^{\mathsf{In}}_{j'}, \mathsf{pk}^{\mathsf{In}}_{j'}) \leftarrow \mathsf{C.KeyGen}(\mathsf{pp})$
return $(\mathsf{pk}^{\mathsf{Id}}_{j'}, \mathsf{sk}^{\mathsf{In}}_{j'}, \mathsf{pk}^{\mathsf{In}}_{j'})$

$\underline{\mathcal{O}^{\mathsf{Rq}}(i, j):}$

if $T_i \neq \mathbf{null} \vee j > j'$ **then abort**
$\delta_i \xleftarrow{\$} \mathbb{Z}_p, C_i \xleftarrow{\$} \mathbb{G}_1, T_i \leftarrow \delta_i C_i,$
$\pi_i \leftarrow \mathcal{S}_{\mathsf{Rq}}(\mathsf{pp}, \mathsf{pk}^{\mathsf{In}}_c, T_i), \mathbf{return}\ (T_i, \pi_i)$

$\underline{\mathcal{O}^{\mathsf{Fin}}(i, (e, S)):}$

if $T_i = \mathbf{null} \vee \sigma_i \neq \mathbf{null}$ **then return** $\perp$
if $\mathsf{e}(S, \mathsf{pk}_s + eG_2) \neq \mathsf{e}(T_i, G_2)$ **then** $\sigma_i \leftarrow \perp, \mathbf{return}\ \sigma_i$
else $\sigma_i \leftarrow \frac{1}{\delta_i} S,\ e_i \leftarrow e,$ **return** 1

$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathsf{sid}, \mathsf{m}):}$

if $\mathsf{st}_{\mathsf{sid}} = \mathbf{null}$ **then**
  $(i, j) \leftarrow \mathsf{m}$, **if** $\sigma_i \in \{\mathbf{null}, \perp\} \vee \mathsf{pk}^{\mathsf{Id}}_j = \mathbf{null}$ **then return** $\perp$
  $\Gamma \leftarrow \Gamma \cup \{i\}, \mathsf{st}_{\mathsf{sid}} \leftarrow (i, j),$ **return** $(\mathsf{pk}^{\mathsf{Id}}_j, \sigma_i)$
elseif $\mathsf{st}_{\mathsf{sid}} \neq \perp$ **then**
  $(i, j) \leftarrow \mathsf{st}_{\mathsf{sid}}, (K, \pi_{\mathsf{Ch}}) \leftarrow \mathsf{m}, \mathsf{st}_{\mathsf{sid}} \leftarrow \perp$
  if $\mathsf{PK}_{\mathsf{Ch}}.\mathsf{Verify}((K, \mathsf{pk}^{\mathsf{Id}}_j), \pi_{\mathsf{Ch}}) = 0$ **then return** $\perp$
  $k \leftarrow \mathcal{E}^{\mathcal{A}(\cdot)}_{\mathsf{Ch}}$, **if** $k = \perp$ **then return** $\perp$.
  $\forall \varphi \in [L] : r_\varphi, \bar{x}_\phi \xleftarrow{\$} \mathbb{Z}_p,\ A_\varphi \leftarrow r_\varphi G_1,\ B_\varphi \leftarrow r_\varphi K + \bar{x}_\phi G_1$
  $\pi^{\mathsf{sid}}_{\mathsf{Rd}} \leftarrow \mathcal{S}_{\mathsf{Rd}}(\mathsf{pp}, \mathsf{pk}^{\mathsf{Id}}_j, K, \sigma_i, \{A_\varphi, B_\varphi\}_{\varphi\in[L]})$
  return $(\pi_{\mathsf{Rd}}, \{A_\varphi, B_\varphi\}_{\varphi\in[L]})$
else return $\emptyset$

As the modifications do not affect the output distribution, we conclude that

$$\mathsf{Pr}[\mathsf{Exp}^{\mathsf{UL}}_7(\lambda) = 1] = \mathsf{Pr}[\mathsf{Exp}^{\mathsf{UL}}_6(\lambda) = 1].$$

Randomize selected tokens. In the next experiment, we overwrite the selected tokens σ_α and σ_β with uniformly random sampled elements in $\mathbb{G}_1$, i.e., $\sigma_\alpha, \sigma_\beta \xleftarrow{\$} \mathbb{G}_1$. The game is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{UL}}_8(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ j' \leftarrow 0,\ b \xleftarrow{\$} \{0,1\},\ \Gamma \leftarrow \emptyset$
$(\mathsf{pk}_s, \mathsf{st}^{\mathcal{A}}_0) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KG}}}(0, \mathsf{pp})$
$(\alpha, \beta, \mathsf{st}^{\mathcal{A}}_1) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KG}},\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Fin}},\mathcal{O}^{\mathsf{Rd}}}(1, \mathsf{st}^{\mathcal{A}}_0)$
if $\exists k \in \{\alpha, \beta\}$ s.t. $\sigma_k \in \{\mathbf{null}, \perp\} \vee k \in \Gamma$ **then abort**
if $b = 1$ **then** swap values of (σ_α) and (σ_β)
$\sigma_\alpha, \sigma_\beta \xleftarrow{\$} \mathbb{G}_1,\ b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KG}},\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Fin}},\mathcal{O}^{\mathsf{Rd}}}(2, \mathsf{st}^{\mathcal{A}}_1)$
if $b = b'$ **then return** 1 **else return** 0
...

First, note that the request proofs (π_{Rq}) and redemption proofs (π_{Rd}) are simulated without using the witnesses and that the ciphertexts $\{A_\phi, B_\phi\}_{\phi\in[L]}$ are sampled uniformly random. These messages provide no information to the adversary. What remains are the blinded token requests T_i and tokens σ_i. In $\mathsf{Exp}^{\mathsf{UL}}_7$, it holds that $T_i = \delta_i C_i$ and $\sigma_i = \frac{1}{\mathsf{sk}_s + e_i} C_i$ for randomly sampled $e_i, \delta_i \xleftarrow{\$} \mathbb{Z}$ (both e_i and δ_i are freshly sampled per i) and $C_i \xleftarrow{\$} \mathbb{G}_1$. As C_i is sampled uniformly random, it holds that σ_i is distributed uniformly random. Further, δ_i acts as a random one-time pad such that T_i is distributed uniformly random and independent of σ_i. Obviously, σ_i is the same every time it is queried (T_i can only be queried once). In $\mathsf{Exp}^{\mathsf{UL}}_8$, the only difference is that σ_i is sampled uniformly random from $\mathbb{G}_1$ once α, β are defined, i.e., if $i \in \alpha, \beta$ and $\{\alpha, \beta\} \not\subseteq \Gamma$. The latter implies that σ_α and σ_β have not been observed before they are re-sampled, which allows us to conclude that σ_α and σ_β are the same every time, they are queried. As in both experiments all observed T_i are independent uniform random elements in $\mathbb{G}_1$ that can only be queried once and σ_i are independent uniform random elements in $\mathbb{G}_1$, that are the same every time they are queried, we can conclude that the view of the adversary is identical in both experiments and, hence, that

$$\mathsf{Pr}[\mathsf{Exp}^{\mathsf{UL}}_8(\lambda) = 1] = \mathsf{Pr}[\mathsf{Exp}^{\mathsf{UL}}_7(\lambda) = 1].$$

Do not swap the selected tokens. In the final experiment, we get rid of the swap and, therefore, of any utilization of the bit b that is to be guessed. The experiment is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{UL}}_9(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ j' \leftarrow 0,\ b \xleftarrow{\$} \{0,1\},\ \Gamma \leftarrow \emptyset$
$(\mathsf{pk}_s, \mathsf{st}^{\mathcal{A}}_0) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KG}}}(0, \mathsf{pp})$
$(\alpha, \beta, \mathsf{st}^{\mathcal{A}}_1) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KG}},\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Fin}},\mathcal{O}^{\mathsf{Rd}}}(1, \mathsf{st}^{\mathcal{A}}_0)$
if $\exists k \in \{\alpha, \beta\}$ s.t. $\sigma_k \in \{\mathbf{null}, \perp\} \vee k \in \Gamma$ **then abort**
$\sigma_\alpha, \sigma_\beta \xleftarrow{\$} \mathbb{G}_1,\ b' \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{KG}},\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Fin}},\mathcal{O}^{\mathsf{Rd}}}(2, \mathsf{st}^{\mathcal{A}}_1)$
if $b = b'$ **then return** 1 **else return** 0
...

As σ_α and σ_β are overwritten after being swapped, the swap (or its removal) does not have any influence of the experiment's output distribution. We can conclude that:

$$\mathsf{Pr}[\mathsf{Exp}^{\mathsf{UL}}_9(\lambda) = 1] = \mathsf{Pr}[\mathsf{Exp}^{\mathsf{UL}}_8(\lambda) = 1].$$

Further, in $\mathsf{Exp}^{\mathsf{UL}}_9$, the randomly sampled bit b is never used. Therefore, we can conclude that

$$\mathsf{Pr}[\mathsf{Exp}^{\mathsf{UL}}_9(\lambda) = 1] = \frac{1}{2}.$$

Putting all steps together. As we have a polynomial number of game hops (in the number of oracle queries and the chunk size L), the difference in the success probability

of the adversary in any two adjacent experiments is at most negligible, and the adversary wins in the final game with probability $\frac{1}{2}$, we conclude that

$$|\mathsf{Pr}[\mathsf{Exp}^{\mathsf{UL}}_{\mathcal{A}}(\lambda)=1]-\frac{1}{2}|\leq \mathsf{negl}(\lambda).$$

Appendix H. Proof of Theorem 6

This section provides a detailed proof of Theorem 6. To show betrayability, we have to show that for every pair of PPT adversaries $(\mathcal{A}_B,\mathcal{A}_V)$, there exists an extractor $\mathcal{E}$ and $n=n(\lambda)(n\in\mathbb{N})$ such that

$$\mathsf{Pr}[\mathsf{Exp}^{\mathsf{BT}}_{\mathcal{A}_B,\mathcal{A}_V,\mathcal{E},n,1}(\lambda)=1]\geq \mathsf{Pr}[\mathsf{Exp}^{\mathsf{BT}}_{\mathcal{A}_B,\mathcal{A}_V,\mathcal{E},n,0}(\lambda)=1]-\mathsf{negl}(\lambda).$$

We show betrayability via a sequence of games. Let $\mathcal{A}_B$ and $\mathcal{A}_V$ be arbitrary PPT adversaries, $\mathcal{E}$ be the extractor defined in Section 5.4, and $\mathsf{Exp}^{\mathsf{BT}}_{0,b}=\mathsf{Exp}^{\mathsf{BT}}_{\mathcal{A}_B,\mathcal{A}_V,\mathcal{E},n,b}$ be the initial security game. The subsequent games are defined as follows.

Inline construction and extractor. In game $\mathsf{Exp}^{\mathsf{BT}}_1$, we inline the computation of our concrete instantiation as well as the actions of the extractor into the security game. To make the presentation more compact, we denote the interactive protocol between $\mathcal{A}_B$ and $\mathcal{A}_V$, in which the experiment passes the messages between the parties as

$$\langle \mathcal{A}_B^{\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Rd}}(B,\cdot,\cdot)}(2,\cdot,\cdot,\cdot),\mathcal{A}_V^{\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Rd}}(V,\cdot,\cdot)}(2,\cdot,\cdot,\cdot)\rangle.$$

The game is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{BT}}_{1,b}(\lambda)$

$\mathsf{pp}\leftarrow\mathsf{Setup}(1^\lambda),\ (\mathsf{sk}_s,\mathsf{pk}_s)\leftarrow\mathsf{S.KeyGen}(\mathsf{pp}),\ b\xleftarrow{\$}\{0,1\}$
$(\mathcal{K},\mathcal{T},\mathcal{T}^*,\mathsf{m}_0^V)\leftarrow\emptyset^4,\ \mathsf{st}_1\leftarrow\{\mathsf{pp},n,\mathsf{pk}_s\}$
$(\sigma^*,\mathsf{st}^B_{(2,0)})\leftarrow\tilde{\mathcal{A}}_V^{\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Rd}}(V,\cdot,\cdot)}(1,\mathsf{st}_1),\ \mathcal{K}^*\leftarrow\mathcal{K}$
$(\mathsf{pk}^{\mathsf{Id}}_*,\mathsf{sk}^{\mathsf{Id}}_*,\mathsf{st}^V_{(2,0)})\leftarrow\mathcal{A}_B^{\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Rd}}(B,\cdot,\cdot)}(1,\sigma^*,\mathsf{st}_1)$
if $\mathsf{KeyV}(\mathsf{pp},\mathsf{sk}^{\mathsf{Id}}_*,\mathsf{pk}^{\mathsf{Id}}_*)=0$ **then return** 0
$\langle\mathcal{A}_B^{\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Rd}}(B,\cdot,\cdot)}(2,\cdot,\cdot,\cdot),\mathcal{A}_V^{\mathcal{O}^{\mathsf{Rq}},\mathcal{O}^{\mathsf{Rd}}(V,\cdot,\cdot)}(2,\cdot,\cdot,\cdot)\rangle$
if $b=0\wedge\mathsf{success}=\mathbf{true}$ **then return** 1
if $b=1\wedge(x^*G_1)\in\mathcal{K}^*$ **then return** 1
else return 0

$\underline{\mathcal{O}^{\mathsf{Rq}}(\mathsf{pk},\mathsf{query}=(T,\pi_{\mathsf{Rq}})):}$
$\mathcal{K}\leftarrow\mathcal{K}\cup\{\mathsf{pk}\}$
if $\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Verify}(\mathsf{rpp},(\mathsf{pk},T),\pi_{\mathsf{Rq}})=0$ **then return** $\emptyset$
$e\xleftarrow{\$}\mathbb{Z}_p,S\leftarrow\frac{1}{\mathsf{sk}_s+e}T$
return (e,S)

$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathcal{P},\mathsf{sid},\mathsf{m}):}$
if $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}}\in\{0,1\}$ **then return** $\emptyset$
elseif $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}}=\mathbf{null}$ **then**
  $(\sigma,\mathsf{pk})\leftarrow\mathsf{m},\ k\xleftarrow{\$}\mathbb{Z}_p$
  if $b=1\wedge\mathcal{P}=B\wedge(\sigma,\mathsf{pk})=(\sigma^*,\mathsf{pk}^{\mathsf{Id}}_*)$ **then**
    $K\leftarrow kG_1,\ \pi\leftarrow\mathsf{PK}_{\mathsf{Ch}}.\mathsf{Prove}((K,\mathsf{pk}^{\mathsf{Id}}_*),\mathsf{sk}^{\mathsf{Id}}_*)$
    $\mathcal{T}^*\leftarrow\mathcal{T}^*\cup\{\mathsf{sid}\}$
  else
    $K\leftarrow kG_0,\ \pi=\mathsf{PK}_{\mathsf{Ch}}.\mathsf{Prove}((K,\mathsf{pk}),k)$
    if $\mathcal{P}=B\wedge(\sigma,\mathsf{pk})=(\sigma^*,\mathsf{pk}^{\mathsf{Id}}_*)$ **then** $\mathcal{T}\leftarrow\mathcal{T}\cup\{\mathsf{sid}\}$
  $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}}\leftarrow(k,K,\sigma,\mathsf{pk}),$ **return** (K,π)
else
  $(k,K,\sigma,\mathsf{pk})\leftarrow\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}},\ (\pi,\sigma',\{A_i,B_i\}_{i\in[L]})\leftarrow\mathsf{m}$
  if $\mathsf{PK}_{\mathsf{Rd}}.\mathsf{Verify}((\mathsf{pk},\ K,\ \sigma,\sigma',\ \{A_i,B_i\}_{i\in[L]}),\ \pi_{\mathsf{Rd}})\ =\ 0\ \vee\ \mathsf{e}(\sigma,\mathsf{pk}_s)\neq\mathsf{e}(\sigma',G_2)$ **then** $\mathsf{st}^{(\mathcal{S},\mathcal{P})}_{\mathsf{sid}}\leftarrow0,$ **return** 0
  if $b=1\wedge\mathcal{P}=B\wedge\mathsf{sid}\in\mathcal{T}^*\wedge x^*=\mathbf{null}$ **then**
    $\forall i\in[L]:D_i\leftarrow B_i-kA_i,\ x_i\leftarrow\mathsf{dlog}_{G_1}(D_i)[0,2^\ell-1]$
    if $\nexists i\in[L]:x_i=\perp$ **then** $x^*\leftarrow\sum_{i\in[L]}2^{(i-1)\cdot\ell}x_i$
  if $b=0\wedge\mathcal{P}=B\wedge\mathsf{sid}\in\mathcal{T}$ **then** $\mathsf{success}\leftarrow\mathbf{true}$
  $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}}\leftarrow1,$ **return** 1

This step does not affect the output distributions. It follows that

$$\Pr[\mathsf{Exp}^{\mathsf{BT}}_{1,b}(\lambda)=1]=\Pr[\mathsf{Exp}^{\mathsf{BT}}_{0,b}(\lambda)=1]\text{ for }b\in\{0,1\}.$$

Simulate challenge proofs. First, we create the challenge proofs inside the redemption oracle with the challenge proof's simulator $\mathcal{S}_{\mathsf{Ch}}$ instead of the proof system's witnesses. The simulator receives the proof's statement $(\mathsf{pp},K,\mathsf{pk}^{\mathsf{Id}}_*)$ and has the capability of programming the random oracle. It creates a valid zero knowledge proof π_{Ch}, except with negligible probability. The resulting experiment is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{BT}}_{2,b}(\lambda)$

...
$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathcal{P},\mathsf{sid},\mathsf{m}):}$
if $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}}\in\{0,1\}$ **then return** $\emptyset$
elseif $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}}=\mathbf{null}$ **then**
  $(\sigma,\mathsf{pk})\leftarrow\mathsf{m},\ k\xleftarrow{\$}\mathbb{Z}_p$
  if $b=1\wedge\mathcal{P}=B\wedge(\sigma,\mathsf{pk})=(\sigma^*,\mathsf{pk}^{\mathsf{Id}}_*)$ **then**
    $K\leftarrow kG_1,\ \pi\leftarrow\mathcal{S}_{\mathsf{Ch}}(\mathsf{pp},K,\mathsf{pk}^{\mathsf{Id}}_*)$
    $\mathcal{T}^*\leftarrow\mathcal{T}^*\cup\{\mathsf{sid}\}$
  else
    $K\leftarrow kG_0,\ \pi=\mathcal{S}_{\mathsf{Ch}}(\mathsf{pp},K,\mathsf{pk}^{\mathsf{Id}}_*)$
    if $\mathcal{P}=B\wedge(\sigma,\mathsf{pk})=(\sigma^*,\mathsf{pk}^{\mathsf{Id}}_*)$ **then** $\mathcal{T}\leftarrow\mathcal{T}\cup\{\mathsf{sid}\}$
  $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}}\leftarrow(k,K,\sigma,\mathsf{pk}),$ **return** (K,π)
else ...

As the challenge proofs created in the redemption oracle of $\mathsf{Exp}^{\mathsf{BT}}_{1,b}$ are always valid, it follows that $\mathsf{Exp}^{\mathsf{BT}}_{2,b}$ and $\mathsf{Exp}^{\mathsf{BT}}_{1,b}$ only differ if the simulator is not successful, which happens with negligible probability due to the zero-knowledge property of the challenge proof system. We use the difference lemma to conclude that

$$|\mathsf{Pr}[\mathsf{Exp}^{\mathsf{BT}}_{2,b}(\lambda)=1]-\mathsf{Pr}[\mathsf{Exp}^{\mathsf{BT}}_{1,b}(\lambda)=1]|\leq\mathsf{negl}(\lambda)\ (\text{for }b\in\{0,1\}).$$

Coin independent oracles. Next, we include a bridging step to make the whole redemption oracle independent of the coin b. The following changes are only of conceptual nature and have no influence on the output distribution.

- We always include the trapdoor into the challenge message, i.e., always define $K=kG_1$. In both experiments, this and the previous, K is an independent random group element and the challenge proof is a simulated valid proof. Therefore, it follows that the adversary's view and, hence, the output distribution is not affected by this change.

- We merge $\mathcal{T}$ and $\mathcal{T}^*$ and utilize the same set $\mathcal{T}$ regardless of b. If $b = 1$, $\mathcal{T}$ is never populated. The same holds for $\mathcal{T}^*$ if $b = 0$. Therefore, it does not affect the experiment to use one set for $\mathcal{T}$ and $\mathcal{T}^*$.
- We try to compute x^* independent of b, i.e., also if $(b = 0 \wedge \mathcal{P} = B \wedge \mathsf{sid} \in \mathcal{T})$. This does not change the output distribution as x^* is not further used if $b = 0$.
- We set $\mathsf{success} \leftarrow \mathbf{true}$ if $\mathcal{P} = B \wedge \mathsf{sid} \in \mathcal{T} \wedge x^* = \mathbf{null}$ but before the (potential) decryption of the ciphertexts $\{A_i, B_i\}$. As success is not further used if $b = 1$, defining success for $b = 1$ has no influence on the output distribution. Requiring $x^* = \mathbf{null}$ for setting $\mathsf{success} \leftarrow \mathbf{true}$ does not influence the output distribution as, in the new experiment, x^* can only be defined to be not $\mathbf{null}$ if success has been set to true before.
- We merge redundant program flows instructions.

The resulting experiment is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{BT}}_{3,b}(\lambda)$

...

$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathcal{P}, \mathsf{sid}, \mathsf{m}) :}$

14: **if** $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}} \in \{0,1\}$ **then return** $\emptyset$
15: **elseif** $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}} = \mathbf{null}$ **then**
16: $(\sigma, \mathsf{pk}) \leftarrow \mathsf{m},\ k \xleftarrow{\$} \mathbb{Z}_p,\ K \leftarrow kG - 1,\ \pi \leftarrow \mathcal{S}_{\mathsf{Ch}}(\mathsf{pp}, K, \mathsf{pk}^{\mathsf{Id}}_*)$
17: **if** $\mathcal{P} = B \wedge (\sigma, \mathsf{pk}) = (\sigma^*, \mathsf{pk}^{\mathsf{Id}}_*)$ **then** $\mathcal{T} \leftarrow \mathcal{T} \cup \{\mathsf{sid}\}$
18: $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}} \leftarrow (k, K, \sigma, \mathsf{pk}),\ \mathbf{return}\ (K, \pi)$
19: **else**
20: $(k, K, \sigma, \mathsf{pk}) \leftarrow \mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}},\ (\pi, \sigma', \{A_i, B_i\}_{i\in[L]}) \leftarrow \mathsf{m}$
21: **if** $\mathsf{PK}_{\mathsf{Rd}}.\mathsf{Verify}((\mathsf{pk},\ K,\ \sigma, \sigma',\ \{A_i, B_i\}_{i\in[L]}),\ \pi_{\mathsf{Rd}}) = 0 \vee \mathsf{e}(\sigma, \mathsf{pk}_s) \neq \mathsf{e}(\sigma', G_2)$ **then** $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}} \leftarrow 0$, **return** 0
22: **if** $\mathcal{P} = B \wedge \mathsf{sid} \in \mathcal{T} \wedge x^* = \mathbf{null}$ **then**
23: $\mathsf{success} \leftarrow \mathbf{true}$
24: $\forall i \in [L] : D_i \leftarrow B_i - kA_i,\ x_i \leftarrow \mathsf{dlog}_{G_1}(D_i)[0, 2^\ell - 1]$
25: **if** $\nexists i \in [L] : x_i = \perp$ **then** $x^* \leftarrow \sum_{i\in[L]} 2^{(i-1)\cdot\ell} x_i$
26: $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}} \leftarrow 1,\ \mathbf{return}\ 1$

These changes do not affect the output distributions. It follows that

$$\Pr[\mathsf{Exp}^{\mathsf{BT}}_{3,b}(\lambda) = 1] = \Pr[\mathsf{Exp}^{\mathsf{BT}}_{2,b}(\lambda) = 1] \text{ for } b \in \{0,1\}.$$

Extract redemption proofs. In the next game, we employ the extractor of the redemption proof system $\mathcal{E}_{\mathsf{Rd}}$ to extract the redemption proof's witnesses if the proof has been validated correctly and $(\mathcal{P} = B \wedge \mathsf{sid} \in \mathcal{T} \wedge x^* = \mathbf{null})$. The extractor has access to the adversaries $(\mathcal{A}_V, \mathcal{A}_B)$, receives the transcript trans of the experiment execution up to the extraction including the adversaries' random tapes, and has the power to program the random oracle. It extracts the witnesses $(y', r', e', \{x'_i, r'_i\}_{i\in[L]})$ that are either all $\perp$ or satisfy the proof system's relation $\mathsf{PK}_{\mathsf{Rd}}$. If the extraction is not successful, we abort the experiment with output 0.

Experiment: $\mathsf{Exp}^{\mathsf{BT}}_{4,b}(\lambda)$

...

$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathcal{P}, \mathsf{sid}, \mathsf{m}) :}$

15: ...
20: **else**
21: $(k, K, \sigma, \mathsf{pk}) \leftarrow \mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}},\ (\pi, \sigma', \{A_i, B_i\}_{i\in[L]}) \leftarrow \mathsf{m}$
22: **if** $\mathsf{PK}_{\mathsf{Rd}}.\mathsf{Verify}((\mathsf{pk},\ K,\ \sigma, \sigma',\ \{A_i, B_i\}_{i\in[L]}),\ \pi_{\mathsf{Rd}}) = 0 \vee \mathsf{e}(\sigma, \mathsf{pk}_s) \neq \mathsf{e}(\sigma', G_2)$ **then** $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}} \leftarrow 0$, **return** 0
23: **if** $\mathcal{P} = B \wedge \mathsf{sid} \in \mathcal{T} \wedge x^* = \mathbf{null}$ **then**
24: $(y', r', e', \{x'_i, r'_i\}_{i\in[L]}) \leftarrow \mathcal{E}^{\mathcal{A}_V(\cdot), \mathcal{A}_B(\cdot)}_{\mathsf{Rd}}(\mathsf{trans})$
25: **if** $e' = \perp$ **then abort** experiment with output 0
26: $\mathsf{success} \leftarrow \mathbf{true}$
27: $\forall i \in [L] : D_i \leftarrow B_i - kA_i,\ x_i \leftarrow \mathsf{dlog}_{G_1}(D_i)[0, 2^\ell - 1]$
28: **if** $\nexists i \in [L] : x_i = \perp$ **then** $x^* \leftarrow \sum_{i\in[L]} 2^{(i-1)\cdot\ell} x_i$
29: $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}} \leftarrow 1,\ \mathbf{return}\ 1$

It is obvious that $\mathsf{Exp}^{\mathsf{BT}}_{4,b}$ and $\mathsf{Exp}^{\mathsf{BT}}_{3,b}$ only differ if the extractor is not successful, which happens with at most negligible probability due to the knowledge soundness property of the redemption proof system. This allows us to use the difference lemma to conclude that

$$|\Pr[\mathsf{Exp}^{\mathsf{BT}}_{4,b}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{BT}}_{3,b}(\lambda) = 1]| \leq \mathsf{negl}(\lambda) \quad (\text{for } b \in \{0,1\}).$$

Extract request proofs and store issued tokens. Next, we adapt the request oracle to employ the request proof system's extractor $\mathcal{E}_{\mathsf{Rq}}$ to extract the witnesses of the request proofs $(\{x_i\}_{i\in[L]}, r, \delta)$. The extractor has access to the adversaries $(\mathcal{A}_V, \mathcal{A}_B)$, receives the transcript trans of the experiment execution up to the extraction including the adversaries' random tapes, and has the power to program the random oracle. The extracted witnesses are either all $\perp$ or satisfy the proof system's relation. If the extraction is not successful, we abort the experiment with output 0.

Additionally, we compute the resulting token σ and if $\mathcal{K}^*$ is not defined, yet, i.e., if σ^* has not been submitted, store the token with the witnesses in an extra set Γ. The resulting experiment is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{BT}}_{5,b}(\lambda)$

1: $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda),\ (\mathsf{sk}_s, \mathsf{pk}_s) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp}),\ b \xleftarrow{\$} \{0,1\}$
2: $(\mathcal{K}, \mathcal{T}, \mathcal{T}^*, \Gamma, \mathsf{m}^V_0) \leftarrow \emptyset^5,\ \mathsf{st}_1 \leftarrow \{\mathsf{pp}, n, \mathsf{pk}_s\}$
3: ...

$\underline{\mathcal{O}^{\mathsf{Rq}}(\mathsf{pk}, \mathsf{query} = (T, \pi_{\mathsf{Rq}})) :}$

10: $\mathcal{K} \leftarrow \mathcal{K} \cup \{\mathsf{pk}\}$
11: **if** $\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Verify}(\mathsf{rpp}, (\mathsf{pk}, T), \pi_{\mathsf{Rq}}) = 0$ **then return** $\emptyset$
12: $(\{x_i\}_{i\in[L]}, r, \delta) \leftarrow \mathcal{E}^{\mathcal{A}_V(\cdot), \mathcal{A}_B(\cdot)}_{\mathsf{Rq}}(\mathsf{trans})$
13: **if** $r = \perp$ **then abort** experiment with output 0
14: $e \xleftarrow{\$} \mathbb{Z}_p, S \leftarrow \frac{1}{\mathsf{sk}_s + e} T$
15: $\sigma \leftarrow \frac{1}{\mathsf{sk}_s + e}(G_0 + rH_0 + \sum_{i\in[L]} x_i H_i)$
16: **if** $\mathcal{K}^* = \mathbf{null}$ **then** $\Gamma \leftarrow \Gamma \cup \{(\sigma, e, r, \{x_i\}_{i\in L})\}$
17: **return** (e, S)

...

As Γ is only populated but not used, it is clearly evident that $\mathsf{Exp}^{\mathsf{BT}}_{5,b}$ and $\mathsf{Exp}^{\mathsf{BT}}_{4,b}$ only differ if the extractor is not successful, which happens with at most negligible probability due to the knowledge soundness property of the request proof system. This allows us to use the difference lemma to conclude that

$$|\Pr[\mathsf{Exp}^{\mathsf{BT}}_{5,b}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{BT}}_{4,b}(\lambda) = 1]| \leq \mathsf{negl}(\lambda) \quad (\text{for } b \in \{0,1\}).$$

Abort if σ^* is re-issued. Next, we check for every (blindly) issued token if it is equal to σ^* and abort the

experiment with output 0 if this is the case. The resulting experiment is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{BT}}_{6,b}(\lambda)$

...

$\underline{\mathcal{O}^{\mathsf{Rq}}(\mathsf{pk}, \mathsf{query} = (T, \pi_{\mathsf{Rq}})) :}$

10: $\mathcal{K} \leftarrow \mathcal{K} \cup \{\mathsf{pk}\}$
11: **if** $\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Verify}(\mathsf{rpp}, (\mathsf{pk}, T), \pi_{\mathsf{Rq}}) = 0$ **then return** $\emptyset$
12: $(\{x_i\}_{i\in[L]}, r, \delta) \leftarrow \mathcal{E}^{\mathcal{A}_V(\cdot),\mathcal{A}_B(\cdot)}_{\mathsf{Rq}}(\mathsf{trans})$
13: **if** $r = \perp$ **then abort** experiment with output 0
14: $e \xleftarrow{\$} \mathbb{Z}_p, S \leftarrow \frac{1}{\mathsf{sk}_s+e}T$
15: $\sigma \leftarrow \frac{1}{\mathsf{sk}_s+e}(G_0 + rH_0 + \sum_{i\in[L]} x_i H_i)$
16: **if** $\mathcal{K}^* = \mathbf{null}$ **then** $\Gamma \leftarrow \Gamma \cup \{(\sigma, e, r, \{x_i\}_{i\in L})\}$
17: **if** $\sigma^* \neq \mathbf{null} \wedge \sigma = \sigma^*$ **then abort** experiment with output 0
18: **return** (e, S)

...

It is evident that $\mathsf{Exp}^{\mathsf{BT}}_{6,b}$ and $\mathsf{Exp}^{\mathsf{BT}}_{5,b}$ only differ if any token computed within the request oracle is equal to σ^* after σ^* has already been defined. As e is sampled uniformly random, it holds that $(\mathsf{sk}_s + e)$, $\frac{1}{\mathsf{sk}_s+e}$ and, hence, σ is distributed uniformly random. Therefore, it can only happen with negligible probability that the request oracle (blindly) issues a token equal to σ^* (after σ^* has already been defined). We apply the difference lemma to conclude that

$$|\Pr[\mathsf{Exp}^{\mathsf{BT}}_{6,b}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{BT}}_{5,b}(\lambda) = 1]| \leq \mathsf{negl}(\lambda) \quad (\text{for } b \in \{0,1\}).$$

Abort if σ^* is a forgery. Before setting success to **true**, we check if there is an element $(\sigma^*, \cdot, \cdot, \cdot)$ in Γ and abort the experiment with output 0, otherwise. The resulting game is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{BT}}_{7,b}(\lambda)$

...

$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathcal{P}, \mathsf{sid}, \mathsf{m}) :}$

19: ...
24: **else**
25: $(k, K, \sigma, \mathsf{pk}) \leftarrow \mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}}$, $(\pi, \sigma', \{A_i, B_i\}_{i\in[L]}) \leftarrow \mathsf{m}$
26: **if** $\mathsf{PK}_{\mathsf{Rd}}.\mathsf{Verify}((\mathsf{pk}, K, \sigma, \sigma', \{A_i, B_i\}_{i\in[L]}), \pi_{\mathsf{Rd}}) = 0 \vee \mathsf{e}(\sigma, \mathsf{pk}_s) \neq \mathsf{e}(\sigma', G_2)$ **then** $\mathsf{st}^{(\mathcal{S},\mathcal{P})}_{\mathsf{sid}} \leftarrow 0$, **return** 0
27: **if** $\mathcal{P} = B \wedge \mathsf{sid} \in \mathcal{T} \wedge x^* = \mathbf{null}$ **then**
28: $(y', r', e', \{x'_i, r'_i\}_{i\in[L]}) \leftarrow \mathcal{E}^{\mathcal{A}_V(\cdot),\mathcal{A}_B(\cdot)}_{\mathsf{Rd}}(\mathsf{trans})$
29: **if** $e' = \perp$ **then abort** experiment with output 0
30: **if** $\nexists(\sigma^*, \cdot, \cdot, \cdot) \in \Gamma$ **then abort** experiment with output 0
31: $\mathsf{success} \leftarrow \mathbf{true}$
32: $\forall i \in [L] : D_i \leftarrow B_i - kA_i,\ x_i \leftarrow \mathsf{dlog}_{G_1}(D_i)[0, 2^\ell - 1]$
33: **if** $\nexists i \in [L] : x_i = \perp$ **then** $x^* \leftarrow \sum_{i\in[L]} 2^{(i-1)\cdot\ell} x_i$
34: $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}} \leftarrow 1$, **return** 1

It is obvious that $\mathsf{Exp}^{\mathsf{BT}}_{7,b}$ and $\mathsf{Exp}^{\mathsf{BT}}_{6,b}$ only differ if the newly inserted abort is executed. Let's call this event F. For F to happen, the adversary $\mathcal{A}_V$ has to submit a token at the beginning of the experiment, that has not been issued by the request oracle, for which the other adversary $\mathcal{A}_B$ learns the corresponding witnesses (the extraction of the redemption proof needs to be successful for the newly inserted abort to be executed). This means that the token-witness pair constitutes a forgery if F happens. Formally, we can show that F cannot happen expect with negligible probability via a reduction to the BBS unforgeability game. We pick $L'(\lambda)$ such that, for $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}) \leftarrow \mathsf{GGen}(1^\lambda)$, $G_1 \xleftarrow{\$} \mathbb{G}_1^*$, and $\ell = \lceil \frac{|p|}{L'(\lambda)} \rceil$, the computation of $\mathsf{dlog}_{G_1}(G)[0, 2^\ell - 1]$ has complexity below $\mathcal{C}$ for all $G \in \mathbb{G}_1$. Then, we define an adversary $\mathcal{B}$ on security game $\mathsf{Exp}^{\mathsf{P\text{-}BBS\text{-}SUF}}_{\mathcal{B},L'+1}$ that internally executes $\mathsf{Exp}^{\mathsf{BT}}_{7,b}$ with $(\mathcal{A}_V, \mathcal{A}_B)$ and wins with the probability of F occurring in $\mathsf{Exp}^{\mathsf{BT}}_{7,b}$. We define $\mathcal{B}$ as follows:

- Receive $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_0, G_1, \{H_i\}_{i\in[L'+1]}, G_2, Z)$ from the $\mathsf{Exp}^{\mathsf{P\text{-}BBS\text{-}SUF}}_{\mathcal{B},L'+1}$ security game.
- Pick ℓ such that $\mathsf{dlog}_{G_1}(G)[0, 2^{\ell-1}]$ can be computed with complexity of at most $\mathcal{C}$ for any $G \in \mathbb{G}_1$ and set $L \leftarrow \lceil \frac{|p|}{\ell} \rceil$. Based on the definition of L' it holds that $L \leq L'$.
- Define $H_0 \leftarrow H_{L+1}$ and compute $\mathsf{rpp} = \mathsf{RP.Setup}(p, \mathbb{G}_1, G_0, H_0, L, \ell)$.
- Define $\mathsf{pp} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, G_0, G_1, H_0, H_1, \ldots, H_L, L, \ell, \mathsf{rpp})$.
- Execute $\mathsf{Exp}^{\mathsf{BT}}_{6,b}$ with $(\mathcal{A}_V, \mathcal{A}_B)$ based on public parameters pp, server public key $\mathsf{pk}_s = Z$ and with unknown server secret key sk_s and answer oracle queries as follows:
  - To answer $\mathcal{O}^{\mathsf{Rq}}$, execute the steps of the oracle as defined in $\mathsf{Exp}^{\mathsf{BT}}_{6,b}$ but instead of sampling e and calculating σ and S (which is not possible, as sk_s is not known), query $\mathcal{O}^{\mathsf{Sign}}(\{x_i\}_{i\in[L]} \cup \{r\} \cup \{0\}_{i\in L'\setminus L})$ of the $\mathsf{Exp}^{\mathsf{P\text{-}BBS\text{-}SUF}}_{\mathcal{B},L'+1}$ game to receive (e, σ) and compute $S = \delta\sigma$.
  - $\mathcal{O}^{\mathsf{Rd}}$ can be executed as defined in $\mathsf{Exp}^{\mathsf{BT}}_{6,b}$ since it does not require knowledge of the server secret key. However, if the abort in line 30 is executed, i.e., if F occurs, return the forgery $(\{x'_i\}_{i\in[L]} \cup \{r'\} \cup \{0\}_{i\in L'\setminus L}, \sigma^*, e')$.
- Return $\perp$.

First note, that the view of the adversaries in the experiment simulated by $\mathcal{B}$ is identical to the view of the adversaries in $\mathsf{Exp}^{\mathsf{BT}}_{7,b}$. The public parameters are generated identically and oracle queries are answered identically in both experiments. Therefore, it also needs to hold that F occurs in the experiment simulated by $\mathcal{B}$ with the same probability as in $\mathsf{Exp}^{\mathsf{BT}}_{7,b}$. Further, we observe that (a) $\mathcal{B}$ outputs a tuple $\neq \perp$ if and only if F occurs, (b) every tuple $\neq \perp$ returned by $\mathcal{B}$ is a valid BBS signature, and (c) every output $\neq \perp$ has not been issued by the signing oracle. Observation (a) follows straight-forwardly from the definition of $\mathcal{B}$. Observation (b) follows from the fact that the non-$\perp$ output of $\mathcal{B}$ is only produced if F occurs and F can only occur if $\mathsf{e}(\sigma^*, Z) = \mathsf{e}(\sigma', G_2)$ and $\sigma' = G_0 + r'H_0 - e'\sigma^* + \sum_{i\in[L]} x'_i H_i$ (the extraction of the redemption proof needs to be successful), which implies that $\sigma^* = \frac{1}{x+e'}(G_0 + r'H_0 + \sum_{i\in[L]} x'_i H_i)$. Finally, observation (c) follows from the fact that $\mathsf{Exp}^{\mathsf{BT}}_{7,b}$ aborts if a signature with $\sigma = \sigma^*$ is issued after σ^* is defined, every token issued before σ^* is defined is added to Γ and F cannot occur if σ^* has been added to Γ. From observations (a), (b), and (c), we can conclude that $\mathcal{B}$ wins with the same probability as F occurs in $\mathsf{Exp}^{\mathsf{BT}}_{7,b}$. As the success probability of $\mathcal{B}$ is bound to be negligible by assumption (cf. Theorem 1), we can use the difference lemma to conclude that

$$|\Pr[\mathsf{Exp}^{\mathsf{BT}}_{7,b}(\lambda) = 1] - \Pr[\mathsf{Exp}^{\mathsf{BT}}_{6,b}(\lambda) = 1]| \leq \mathsf{negl}(\lambda) \quad (\text{for } b \in \{0,1\}).$$

Abort if σ^* is a collision. In the next game, we check if the witnesses extracted in the redemption proof for token

σ^*, i.e., $(e', r', \{x'_i\}_{i\in[L]})$ match the witnesses stored in Γ for token σ^*. If this is the case, we abort the experiment. The resulting experiment is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{BT}}_{8,b}(\lambda)$

...

$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathcal{P}, \mathsf{sid}, \mathsf{m}):}$

19: ...
24: **else**
25: $(k, K, \sigma, \mathsf{pk}) \leftarrow \mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}}$, $(\pi, \sigma', \{A_i, B_i\}_{i\in[L]}) \leftarrow \mathsf{m}$
26: **if** $\mathsf{PK}_{\mathsf{Rd}}.\mathsf{Verify}((\mathsf{pk}, K, \sigma, \sigma', \{A_i, B_i\}_{i\in[L]}), \pi_{\mathsf{Rd}}) = 0 \vee \mathsf{e}(\sigma, \mathsf{pk}_s) \neq \mathsf{e}(\sigma', G_2)$ **then** $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}} \leftarrow 0$, **return** 0
27: **if** $\mathcal{P} = B \wedge \mathsf{sid} \in \mathcal{T} \wedge x^* =$ **null then**
28: $(y', r', e', \{x'_i, r'_i\}_{i\in[L]}) \leftarrow \mathcal{E}^{\mathcal{A}_V(\cdot),\mathcal{A}_B(\cdot)}_{\mathsf{Rd}}(\mathsf{trans})$
29: **if** $e' = \perp$ **then abort** experiment with output 0
30: **if** $\nexists(\sigma^*, \cdot, \cdot, \cdot) \in \Gamma$ **then abort** experiment with output 0
31: Let $(e^*, r^*, \{x^*_i\}_{i\in[L]})$ be such that $(\sigma^*, e^*, r^*, \{x^*_i\}_{i\in[L]}) \in \Gamma$ (first occurrence, if multiple)
32: **if** $(e', r', \{x'_i\}_{i\in[L]}) \neq (e^*, r^*, \{x^*_i\}_{i\in[L]})$ **then abort** experiment with output 0
33: $\mathsf{success} \leftarrow$ **true**
34: $\forall i \in [L] : D_i \leftarrow B_i - kA_i$, $x_i \leftarrow \mathsf{dlog}_{G_1}(D_i)[0, 2^\ell - 1]$
35: **if** $\nexists i \in [L] : x_i = \perp$ **then** $x^* \leftarrow \sum_{i\in[L]} 2^{(i-1)\cdot\ell} x_i$
36: $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}} \leftarrow 1$, **return** 1

Again, it is obvious that $\mathsf{Exp}^{\mathsf{BT}}_{8,b}$ and $\mathsf{Exp}^{\mathsf{BT}}_{7,b}$ only differ if the newly inserted abort is executed. Let's call this event F. We show that the probability of F is negligible by building a PPT discrete logarithm solver in $\mathbb{G}_1$, $\mathcal{B}$, that internally simulates $\mathsf{Exp}^{\mathsf{BT}}_{8,b}$ and solves discrete logarithm problems with probability

$$\mathsf{Pr}[\mathsf{Exp}^{\mathsf{dlog}}_{\mathcal{B}}(1^\lambda) = 1] = \mathsf{Pr}[F \text{ in } \mathsf{Exp}^{\mathsf{BT}}_{8,b}(1^\lambda)](\frac{p-1}{p})^{L+3}.$$

We define the solver $\mathcal{B}$ as follows:

- Execute $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$ and send $(\mathbb{G}_1, p)$ to the discrete logarithm challenger to receive (G, H).
- Sample $\alpha_g, \alpha_0, \alpha_1, \ldots \alpha_L \xleftarrow{\$} \mathbb{Z}_p$ and $\beta_g, \beta_0, \beta_1, \ldots, \beta_L \xleftarrow{\$} \mathbb{Z}_p$.
- Compute $G_0 = \alpha_g G + \beta_g H$ and sample $G_1 \xleftarrow{\$} \mathbb{G}^*_1$.
- For $i \in [0, L]$ compute $H_i = \alpha_i G + \beta_i H$.
- If $G_0 = 0$ or $\exists i \in [0, L] : H_i = 0$ then output $\perp$.
- Sample $\mathsf{rpp} \leftarrow \mathsf{RP}.\mathsf{Setup}(p, \mathbb{G}_1, G_0, H_0, L, \ell)$.
- Define $\mathsf{pp}' \leftarrow (p, \mathbb{G}_1, G_0, G_1, H_0, H_1, \ldots, H_L, L, \ell, \mathsf{rpp})$.
- Execute experiment $\mathsf{Exp}^{\mathsf{BT}}_{8,b}$ with the following modifications:
  1) Instead of computing $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, set $\mathsf{pp} \leftarrow \mathsf{pp}'$.
  2) If the newly inserted abort condition is executed, i.e., if Line 32 is executed with $(e^*, r^*, \{x^*_i\}_{i\in[L]}) \neq (e', r', \{x^*_i\}_{i\in[L]})$, then store $z = \mathsf{sk}_s, (e^*, r^*, \{x^*_i\}_{i\in[L]})$ and $(e', r', \{x'_i\}_{i\in[L]})$ and abort the experiment execution.
  3) If the experiments ends without being aborted by the solver then output $\perp$.
- Compute:
  $u \leftarrow (z+e')(\alpha_g + \alpha_0 r^* + \sum_{i\in[L]} \alpha_i x^*_i) - (z+e^*)(\alpha_g + \alpha_0 r' + \sum_{i\in[L]} \alpha_i x'_i)$
  $v \leftarrow (z+e')(\beta_g + \beta_0 r^* + \sum_{i\in[L]} \beta_i x^*_i) - (z+e^*)(\beta_g + \beta_0 r' + \sum_{i\in[L]} \beta_i x'_i)$.
- If $v = 0$ then output $\perp$. Otherwise, output $h = -\frac{u}{v}$.

First, note that $\mathcal{B}$ perfectly simulates the experiment to the adversaries (until the abort). This is due to the fact that the solver's only change to the experiment is the way the generators $G_0, H_0, H_1, \ldots, H_L$ are sampled. However, if $\mathcal{B}$ simulates the experiment, i.e., if none of the generators is equal to 0, the generators $G_0, H_0, H_1, \ldots, H_L$ are uniformly random generators of $\mathbb{G}_1$ just as they are in a real experiment. Second, we note that the probability that none of the generators $G_0, H_0, H_1, \ldots, H_L$ equals 0 is $(\frac{p-1}{p})^{L+2}$. Therefore, it follows that the probability that F occurs in the experiment simulated by the discrete logarithm solver is $(\mathsf{Pr}[F \text{ in } \mathsf{Exp}^{\mathsf{BT}}_{8,b}] \cdot (\frac{p-1}{p})^{L+2})$. Third, we note that if F occurs in the simulated experiment and $v \neq 0$ then it holds that $H = hG$. To see this, first recall that the occurrence of event F implies that

$$\begin{aligned}\sigma^* &= \frac{1}{z+e^*}(G_0 + r^* H_0 + \sum_{i\in[L]} x^*_i H_i)\\ &= \frac{1}{z+e'}(G_0 + r' H_0 + \sum_{i\in[L]} x'_i H_i).\end{aligned}$$

Therefore, it holds that

$$\begin{aligned}0 =&(z+e')(G_0 + r^* H_0 + \sum_{i\in[L]} x^*_i H_i)\\ &- (z+e^*)(G_0 + r' H_0 + \sum_{i\in[L]} x'_i H_i)\\ =&((z+e')(\alpha_0 + \alpha_h r^* + \sum_{i\in[L]} \alpha_i x^*_i)\\ &- (z+e^*)(\alpha_0 + \alpha_h r' + \sum_{i\in[L]} \alpha_i x'_i))G\\ &+ ((z+e')(\beta_0 + \beta_h r^* + \sum_{i\in[L]} \beta_i x^*_i)\\ &- (z+e^*)(\beta_0 + \beta_h r' + \sum_{i\in[L]} \beta_i x'_i))H\\ =&uG + vH \;\Leftrightarrow\; H = \frac{-u}{v}G = hG \quad (\text{if } v \neq 0).\end{aligned}$$

Fourth, we observe that the probability that $v = 0$ is $\frac{1}{p}$. Let h be such that $H = hG$, $\gamma_0 = \alpha_g + h\beta_g$ and $\gamma_i = \alpha_i + h\beta_i$ for $i \in [0, L]$. By definition, it holds that $G_0 = \gamma_g G$ and $H_i = \gamma_i G$ for $i \in [0, L]$. Note that the only information about the α and β variables provided to the adversary are the generators $G_0, H_0, H_1, \ldots, H_L$. During the execution of the simulated experiments, the α- and β-variables are no longer used. With the generators, the adversary gets some information on $\gamma_h, \gamma_0, \gamma_1, \ldots \gamma_L$ but the variables $\beta_g, \beta_0, \beta_1, \ldots \beta_L$ are independent from the γ-variables as the α-values act as one-time pads. Now, we re-write v as

$$\begin{aligned}v =&(e' - e^*)\beta_g + ((z+e')r^* - (z+e^*)r')\beta_0\\ &+ \sum_{i\in[L]} ((z+e')x^*_i - (z+e^*)x'_i)\beta_i\end{aligned}$$

From this representation, it becomes evident that, if $r^* \neq r'$ or there exists $i \in [L]$ such that $x^*_i \neq x'_i$ (which is necessary for event F), we can conclude that the distribution of v conditioned on fixed generators is uniform. Therefore, it follows that $\Pr[v = 0 \mid \text{event } F] = \frac{1}{p}$. By combing these

observations, we can conclude that $\mathcal{E}$ indeed solves discrete logarithm problems with probability

$$\mathsf{Pr}[\mathsf{Exp}^{\mathsf{dlog}}_{\mathcal{B}}(1^\lambda)=1] = \mathsf{Pr}[F \text{ in } \mathsf{Exp}^{\mathsf{BT}}_{8,b}(1^\lambda)](\frac{p-1}{p})^{L+3}.$$

Finally, the discrete logarithm assumption states that the advantage of every PPT discrete logarithm solver is bound by $\mathsf{Adv}_{\mathsf{dlog}}(\lambda) \leq \mathsf{negl}(\lambda)$. Therefore, we conclude that

$$\mathsf{Pr}[\mathsf{Exp}^{\mathsf{dlog}}_{\mathcal{B}}(1^\lambda)=1] \leq \mathsf{Adv}_{\mathsf{dlog}}(\lambda) \leq \mathsf{negl}(\lambda)$$
$$\Leftrightarrow \mathsf{Pr}[F \text{ in } \mathsf{Exp}^{\mathsf{BT}}_{8,b}(\lambda)] \leq \mathsf{Adv}_{\mathsf{dlog}}(\lambda) \cdot (\frac{p}{p-1})^{L+3} \leq \mathsf{negl}(\lambda)$$

This allows us to use the difference lemma to conclude that

$$|\Pr[\mathsf{Exp}^{\mathsf{BT}}_{8,b}(\lambda)=1] - \Pr[\mathsf{Exp}^{\mathsf{BT}}_{7,b}(\lambda)=1]| \leq \mathsf{negl}(\lambda) \quad (\text{for } b \in \{0,1\}).$$

Compute x^* based on the witnesses stored in Γ. In the next game, we do not decrypt the ciphertexts but compute x^* based on $\{x_i^*\}_{i\in[L]}$.

Experiment: $\mathsf{Exp}^{\mathsf{BT}}_{9,b}(\lambda)$

...

$\underline{\mathcal{O}^{\mathsf{Rd}}(\mathcal{P}, \mathsf{sid}, \mathsf{m}):}$

19: ...
24: **else**
25: $(k, K, \sigma, \mathsf{pk}) \leftarrow \mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}}$, $(\pi, \sigma', \{A_i, B_i\}_{i\in[L]}) \leftarrow \mathsf{m}$
26: **if** $\mathsf{PK}_{\mathsf{Rd}}.\mathsf{Verify}((\mathsf{pk}, K, \sigma, \sigma', \{A_i, B_i\}_{i\in[L]}), \pi_{\mathsf{Rd}}) = 0 \vee \mathsf{e}(\sigma, \mathsf{pk}_s) \neq \mathsf{e}(\sigma', G_2)$ **then** $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}} \leftarrow 0$, **return** 0
27: **if** $\mathcal{P} = B \wedge \mathsf{sid} \in \mathcal{T} \wedge x^* = \mathbf{null}$ **then**
28: $(y', r', e', \{x_i', r_i'\}_{i\in[L]}) \leftarrow \mathcal{E}^{\mathcal{A}_V(\cdot),\mathcal{A}_B(\cdot)}_{\mathsf{Rd}}(\mathsf{trans})$
29: **if** $e' = \perp$ **then abort** experiment with output 0
30: **if** $\nexists(\sigma^*, \cdot, \cdot, \cdot) \in \Gamma$ **then abort** experiment with output 0
31: Let $(e^*, r^*, \{x_i^*\}_{i\in[L]})$ be such that $(\sigma^*, e^*, r^*, \{x_i^*\}_{i\in[L]}) \in \Gamma$ (first occurrence, if multiple)
32: **if** $(e', r', \{x_i'\}_{i\in[L]}) \neq (e^*, r^*, \{x_i^*\}_{i\in[L]})$ **then abort** experiment with output 0
33: $\mathsf{success} \leftarrow \mathbf{true}$, $x^* \leftarrow \sum_{i\in[L]} 2^{(i-1)\cdot\ell} x_i^*$
34: $\mathsf{st}^{(S,\mathcal{P})}_{\mathsf{sid}} \leftarrow 1$, **return** 1

First, we make the following observations with respect to $\mathsf{Exp}^{\mathsf{BT}}_{8,b}$. For $i \in [L]$ it needs to hold that $D_i = x_i' G_1$ (due to the successful extraction of the redemption proof), $x_i' = x_i^*$ (due to the previous checks), and $x_i^* \in [0, 2^\ell - 1]$ (due to the successful extraction of the request proofs). It follows that $D_i = x_i^* G_1$ (from the first two) and that $\mathsf{dlog}_{G_1}(D_i) \neq \perp$ (from the third). From these, we can conclude that the check $(\nexists i \in [L] : x_i = \perp)$ is redundant, i.e., x^* is defined whenever the decryption is attempted, and that $x_i = x_i^*$ for $i \in [L]$. It follows that x^* is defined whenever $\mathsf{success}$ is set to $\mathbf{true}$ and that x^* is always equal to $\sum_{i\in[L]} 2^{(i-1)\ell} x_i^*$ in $\mathsf{Exp}^{\mathsf{BT}}_{8,b}$. Therefore, we can conclude that

$$\Pr[\mathsf{Exp}^{\mathsf{BT}}_{9,b}(\lambda)=1] = \Pr[\mathsf{Exp}^{\mathsf{BT}}_{8,b}(\lambda)=1] \text{ for } b \in \{0,1\}.$$

Final game. Finally, we modify the success condition of the experiment. In particular, if $b = 0$, we only output 1 if $x^* G_1 \in \mathcal{K}^* \wedge \mathsf{success} = \mathbf{true}$. The former of the two conditions is new. The resulting final experiment is defined as follows:

Experiment: $\mathsf{Exp}^{\mathsf{BT}}_{10}(\lambda)$

$\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda)$, $(\mathsf{sk}_s, \mathsf{pk}_s) \leftarrow \mathsf{S.KeyGen}(\mathsf{pp})$, $b \xleftarrow{\$} \{0,1\}$
$(\mathcal{K}, \mathcal{T}, \mathcal{T}^*, \mathsf{m}_0^V) \leftarrow \emptyset^4$, $\mathsf{st}_1 \leftarrow \{\mathsf{pp}, n, \mathsf{pk}_s\}$
$(\sigma^*, \mathsf{st}^B_{(2,0)}) \leftarrow \tilde{\mathcal{A}}_V^{\mathcal{O}^{\mathsf{Rq}}, \mathcal{O}^{\mathsf{Rd}}(V,\cdot,\cdot)}(1, \mathsf{st}_1)$, $\mathcal{K}^* \leftarrow \mathcal{K}$
$(\mathsf{pk}^{\mathsf{Id}}_*, \mathsf{sk}^{\mathsf{Id}}_*, \mathsf{st}^V_{(2,0)}) \leftarrow \mathcal{A}_B^{\mathcal{O}^{\mathsf{Rq}}, \mathcal{O}^{\mathsf{Rd}}(B,\cdot,\cdot)}(1, \sigma^*, \mathsf{st}_1)$
if $\mathsf{KeyV}(\mathsf{pp}, \mathsf{sk}^{\mathsf{Id}}_*, \mathsf{pk}^{\mathsf{Id}}_*) = 0$ **then return** 0
$\langle \mathcal{A}_B^{\mathcal{O}^{\mathsf{Rq}}, \mathcal{O}^{\mathsf{Rd}}(B,\cdot,\cdot)}(2,\cdot,\cdot,\cdot), \mathcal{A}_V^{\mathcal{O}^{\mathsf{Rq}}, \mathcal{O}^{\mathsf{Rd}}(V,\cdot,\cdot)}(2,\cdot,\cdot,\cdot) \rangle$
if $b = 0 \wedge (x^* G_1) \in \mathcal{K}^* \wedge \mathsf{success} = \mathbf{true}$ **then return** 1
if $b = 1 \wedge (x^* G_1) \in \mathcal{K}^*$ **then return** 1
else return 0

$\underline{\mathcal{O}^{\mathsf{Rq}}(\mathsf{pk}, \mathsf{query} = (T, \pi_{\mathsf{Rq}})):}$

$\mathcal{K} \leftarrow \mathcal{K} \cup \{\mathsf{pk}\}$
if $\mathsf{PK}_{\mathsf{Rq}}.\mathsf{Verify}(\mathsf{rpp}, (\mathsf{pk}, T), \pi_{\mathsf{Rq}}) = 0$ **then return** $\emptyset$
$(\{x_i\}_{i\in[L]}, r, \delta) \leftarrow \mathcal{E}^{\mathcal{A}_V(\cdot),\mathcal{A}_B(\cdot)}_{\mathsf{Rq}}(\mathsf{trans})$
if $r = \perp$ **then abort** experiment with output 0
$e \xleftarrow{\$} \mathbb{Z}_p$, $S \leftarrow \frac{1}{\mathsf{sk}_s + e} T$
$\sigma \leftarrow \frac{1}{\mathsf{sk}_s + e}(G_0 + r H_0 + \sum_{i\in[L]} x_i H_i)$
if $\mathcal{K}^* = \mathbf{null}$ **then** $\Gamma \leftarrow \Gamma \cup \{(\sigma, e, r, \{x_i\}_{i\in L})\}$
if $\sigma^* \neq \mathbf{null} \wedge \sigma = \sigma^*$ **then abort** experiment with output 0
return (e, S)

...

We observe that $x^* G_1 \in \mathcal{K}^*$ holds whenever $\mathsf{success} = \mathbf{true}$. To see this, first observe that for every $(\cdot,\cdot,\cdot,\{x_i\}_{i\in[L]}) \in \Gamma$, it needs to hold that $\mathsf{pk} = (\sum_{i\in[L]} 2^{(i-1)\ell} x_i) G_1$ (due to the successful extraction) and that $\mathsf{pk} \in \mathcal{K}^*$ as for every tuple added to Γ the corresponding public key pk is added to $\mathcal{K}$ before $\mathcal{K}^*$ is defined and, hence, also ends up in $\mathcal{K}^*$. Secondly, observe that if $\mathsf{success}$ is set to $\mathbf{true}$, then x^* is defined to be $x^* = \sum_{i\in[L]} 2^{(i-1)\ell} x_i^*$ for $(\cdot,\cdot,\cdot,\{x_i^*\}_{i\in[L]}) \in \Gamma$. From these two observations, we can conclude that there exists $\mathsf{pk}^* \in \mathcal{K}^*$ such that $\mathsf{pk}^* = x^* G_1$ if $\mathsf{success} = \mathbf{true}$. As $x^* G_1 \in \mathcal{K}^*$ holds whenever $\mathsf{success} = \mathbf{true}$, inserting the additional success condition does not affect the output distribution of the experiment. Therefore, we can conclude that

$$\Pr[\mathsf{Exp}^{\mathsf{BT}}_{10,b}(\lambda)=1] = \Pr[\mathsf{Exp}^{\mathsf{BT}}_{9,b}(\lambda)=1] \text{ for } b \in \{0,1\}.$$

Output distribution in the final game. In the final game, the experiment for $b = 0$ and $b = 1$ only differ in the condition for outputting 1. The condition for outputting 1 if $b = 1$ is contained in the one for outputting 1 if $b = 0$. Therefore, it follows that $\mathsf{Pr}[\mathsf{Exp}^{\mathsf{BT}}_{10,1}(\lambda)=1] \geq \mathsf{Pr}[\mathsf{Exp}^{\mathsf{BT}}_{10,0}(\lambda)=1]$.

Further, we have used a polynomial number of game hops and shown that the difference in the success probability of the adversary in any two adjacent experiments is at most negligible. Therefore, we can conclude that

$$|\Pr[\mathsf{Exp}^{\mathsf{BT}}_{10,b}(\lambda)=1] - \Pr[\mathsf{Exp}^{\mathsf{BT}}_{\mathcal{A}_B,\mathcal{A}_V,\mathcal{E},n,b}(\lambda)=1]| \leq \mathsf{negl}(\lambda) \quad (\text{for } b \in \{0,1\}).$$

By combining the above two, we can conclude that

$$\mathsf{Pr}[\mathsf{Exp}^{\mathsf{BT}}_{\mathcal{A}_B,\mathcal{A}_V,\mathcal{E},n,1}(\lambda)=1] \geq \mathsf{Pr}[\mathsf{Exp}^{\mathsf{BT}}_{\mathcal{A}_B,\mathcal{A}_V,\mathcal{E},n,0}(\lambda)=1] - \mathsf{negl}(\lambda).$$