Non-interactive Anonymous Tokens with Private Metadata Bit

Foteini Baldimtsi[*‡], Lucjan Hanzlik[†], Quan Nguyen[*] and Aayush Yadav[*]
[*]*George Mason University*
Fairfax (USA)
Email: {foteini, qngnuye31, ayadav5}@gmu.edu
[†]*CISPA Helmholtz Center for Information Security*
Saarbrücken (Germany)
Email: hanzlik@cispa.de
[‡]*Mysten Labs*
Palo Alto (USA)

Abstract—**Anonymous tokens with private metadata bit (ATPM) have received increased interest as a method for anonymous client authentication while also embedding trust signals that are only readable by the authority who holds the issuance secret key and nobody else. A drawback of all existing ATPM constructions is that they require client-issuer interaction during the issuance process. In this work, we build the first *non-interactive* anonymous tokens (NIAT) with private metadata bit, inspired by the recent work of Hanzlik (EUROCRYPT '23) on non-interactive blind signatures. We discuss how the non-interaction property during the issuance process allows for more efficient issuance protocols that avoid the need for online signing. We construct an efficient NIAT scheme based on Structure-preserving Signatures on Equivalence Classes (SPS-EQ) and experimentally evaluate its performance. We also present an extension to our NIAT construction that allows the identification of clients who attempt to double-spend (i.e., present the same token twice).**

1. Introduction

Anonymous tokens are a powerful primitive that allow users to access services or resources securely while maintaining their privacy. Typically, an anonymous token system involves three types of parties: a *client*, an *issuer*, and a *redemptee* (or verifier). The client engages in an issuance protocol with the issuer who first verifies the trustworthiness of the client and then issues them a token. Later, the client can present the token to a redemptee, who simply verifies its authenticity to grant the client access to a service or resource, and additionally checks that the same token was not redeemed before (double-spending detection). The basic properties satisfied by an anonymous token system include unforgeability, which ensures that a client cannot issue tokens on its own, and anonymity which prevents issuers and redemptees to link issued tokens to any token later redeemed.

The applications of anonymous tokens are numerous including controlled access to content delivery networks (CDNs) without CAPTCHA solving [1], private web-browsing [2], private contact tracing [3], fraud detection [4] and private click measurement [5] just to name a few. The importance of anonymous tokens is also highlighted by the fact that they have received significant attention from the industry with notable projects including Google's Private State Tokens [6] and Cloudflare's Privacy Pass [7]. Concurrently, there has also been a standardization effort through IETF [8] supported by large industry stakeholders such as Google, Apple, Cloudflare, and Fastly.

Given the increased interest in anonymous tokens, a number of protocols have been proposed in the literature [1, 9, 10, 3, 11, 12] satisfying a variety of properties and offering interesting tradeoffs. A first distinction between existing works is in regards to whether they support private or public token verification, i.e., whether the issuer and the redemptee are the same party (private setting) [1, 9, 11] or any third party can verify the issued tokens (public setting) [10, 3]. Having a single party for both issuance and redemption has the benefit of making double-spending detection much easier but it is restrictive for applications with multiple verifiers or verification of tokens in a public setting (such as a blockchain smart contract). On the other hand, public verification can enhance the application of anonymous tokens in broader systems by outsourcing signing or verification of tokens, but in order to detect double-spending, you still need to rely on centralized mechanisms.

A second distinction amongst existing works is on whether they support the embedding of public [3] or private metadata [9, 10, 11]. Anonymous tokens with public metadata support applications that require public labeling and can facilitate the embedding of geographical tags or expiration dates, while anonymous tokens with private metadata bit (ATPM) can be used to privately convey trust signals. In the most common scenario, the private metadata is a single bit. The idea is that the issuer can encode a hidden bit within the token, in a way that the user cannot distinguish which of the two bits their token encodes. Only the issuer, if it later receives the token, can extract the embedded bit. This private metadata bit support is highly practical as it enables the issuer to flag potentially malicious users without alerting them to their detection. When adversaries remain oblivious to whether they have been detected, the task of developing sophisticated tools in order to avoid being detected gets significantly more challenging. Since anonymous tokens with private metadata bit require the involvement of the issuer in order to extract the hidden bit, they are most often proposed in the private verification setting [9, 11]. However, in certain scenarios one might desire to allow hidden metadata in publicly verifiable tokens [10].

An anonymous token scheme with public verifiability and hidden metadata bit, is suitable for applications where there is a single, central issuing authority that has enough context to derive trust signals for users, but at the same time there exist many possible redemptees[1]. The redemptees can run the public verification algorithm and then submit the already verified tokens to the central authority that will check for double-spending. Having a secret metadata bit embedded in the token, can enable more efficient double-spending detection by potentially only checking the flagged tokens. As another example consider decentralized applications running on a blockchain. There, public verification ensures transparency and accessibility, while private metadata allows for the inclusion of sensitive information or trust signals that can be detected by designated parties (the issuer) without exposing them to the public.

A common drawback among all currently proposed schemes for anonymous tokens is the requirement for an interactive issuance protocol. Typically, the client initiates the protocol by preparing the first message which they then submit to the issuer. When the client receives the issuer's response, they can locally retrieve the final token. However, this interactive nature can lead to various practical challenges. Such challenges include latency issues, especially in real-time applications, and scalability concerns, particularly when the issuer must handle a large volume of requests simultaneously where a relatively expensive computational task needs to be executed (i.e., a signature operation).

Our approach. In this work we propose the first anonymous token protocols with *non-interactive* issuance. Our main observation is that at the core of many of the proposed protocols is a blind signature scheme on a randomly selected message [1, 9, 3, 11]. Typically, by definition blind signatures require at least one round of interaction (2 moves) between the signer/issuer and the client [13, 14, 15]. However, recent works [16, 17] have demonstrated the feasibility of constructing non-interactive blind signatures (NIBS) as long as the message is random and does not come from a specific distribution or has a specific structure. In anonymous tokens, the signed message is effectively a random identifier, thus the constructions of [16, 17] could immediately give rise to an anonymous token scheme with public verifiability. Additionally, the non-interactive blind signature constructions of [16, 17] could also be turned into partially blind signatures and thus also support the embedding of public metadata. Yet, achieving support for a private metadata bit remains a non-trivial extension, as demonstrated previously in the case of interactive blind signatures.

Our Contributions. We summarize our contributions as follows:

- **Defining non-interactive anonymous tokens.** We propose a formal framework for *Non-interactive Anonymous Tokens with Private Metadata Bit* (NIAT). In particular we define one-more unforgeability, unlinkability, metadata bit privacy and reusability (namely the property that an issuer is able to issue multiple tokens under the same client public key[2]. Additionally, we provide formal definition for double-spending identification.
- **Main NIAT construction.** In Section 4, we give our main NIAT construction from Structure-preserving Signatures on Equivalence Classes (SPS-EQ) [18]. Our SPS-EQ construction extends the SPS-EQ NIBS construction of [16] to also support metadata bit hiding and extraction, and is secure under the inverse and strong decisional Diffie-Hellman assumptions as well as the unforgeability of the underlying SPS-EQ scheme. In our evaluations, we leverage the fact that verification of the SPS-EQ signature requires computation of pairing operations, some of which can be aggregated. Our construction can be upgraded to support public metadata, however due to space constraints, we provide details in the full version.
- **NIAT with DS identification.** Our Section 4 construction requires an "online" check during verification in order to avoid double-spending (that the same token was not previously presented). While this is in line with previous works on anonymous tokens with private metadata bit, the requirement for an "online" check during verification can turn out to be pretty cumbersome. For instance, in application scenarios where verification happens by multiple distinct verifiers, it is not easy to keep a synchronized record of all spent tokens across all verifiers.

  We define and present the first anonymous token scheme with a private metadata bit that supports double-spend (DS) *identification*. This feature allows for "offline" verification—meaning tokens can be presented without immediate verification of their uniqueness. If a token is double-spent, it can be identified post-event, enabling detection and potential penalties. In Section 5 we provide an extension of our main NIAT scheme to also support public double-spending identification support.
- **Evaluation.** In Section 6, we experimentally evaluate our main SPS-EQ based construction using a proof-of-concept implementation. In our implementation, token issuance takes about 0.84 ms, token generation takes 1.28 ms (amortized) and token redemption takes 0.8 ms (amortized). The amortized costs are due to the

[1] The setting of private and public metadata, in addition to public verifiability is also considered in the IETF standardization process [8].

[2] We note that this property is not needed for the interactive schemes of prior work

aforementioned ability to aggregate some pairing operations. Thus, the cost of verification can be amortized over the number of presignatures issued (for token generation) and the number of clients serviced (for token redemption). Lastly, for completeness we provide an asymptotic comparison of our scheme with existing ATPM schemes, although we note that a direct comparison is difficult given that most existing schemes either do not support public verification or private metadata bit hiding and all schemes are interactive.

The benefits of non-interaction: access to CDNs with NIAT. The main advantage of our main construction is the fact that it does not require any client–issuer interaction during the issuance protocol. This can allow for offline pre-computations leading to *significant* savings in the issuance process, especially if an issuance server is responsible for a large number of requests.

Let us consider one of the most popular applications of anonymous tokens, that of privacy preserving access to CDNs which was also the inspiration for the seminal Privacy Pass protocol [1]. Currently, in a CDN system, there exists an attesting server which on a client request, examines the request and presents a puzzle challenge, such as a CAPTCHA, if the server believes the incoming request is from an unstruthworthy origin. The client will solve the puzzle and respond with the solution to prove that they are a real human user. The attesting server will check for correctness of the puzzle solution and respond accordingly. It may forward the client's request to the web server that the client is wishing to access to fetch dynamic content, or it may serve the client with a cached version of the static web page. This is a reasonable approach for CDNs to prevent bot requests and DDoS attacks, and has been deployed in practice for a long time. However, it does not offer privacy for users and their history of web browsing can be traced back to the clients. Privacy Pass anonymous tokens were deployed to address such privacy issues [1].

In the setting of Privacy Pass, when a client sends the CAPTCHA solution to the attesting server, they also include a number of blinded tokens for the issuer to sign. If the issuer determines that the client is an honest human user, by checking CAPTCHA solutions, it signs the received blinded tokens *on the fly* and sends them back to the client alongside the content in the response. Later, when the client wants to access some web servers that are protected by the same infrastructure, they can redeem the tokens to bypass CAPTCHAs while maintaining their anonymity due to the tokens' unlinkability properties. To avoid token hoarding, where the clients may be able to request a large amount amount tokens and redistribute them for future attacks, [1] recommended issuing 30 tokens during an issuance session. This number was considered to offer a reasonable trade-off between usability, performance, and the token hoarding issue.

However, it is well known that online signing is a costly operation, which if, possible should be avoided in network protocols where a server serves a large number of clients simultaneously. This concern was pivotal in the development of protocols like DNSSEC. Leveraging our NIAT primitive, we propose an alternative system for token issuance that eliminates the need for online interaction between clients and issuers and circumvents the necessity for online signing.

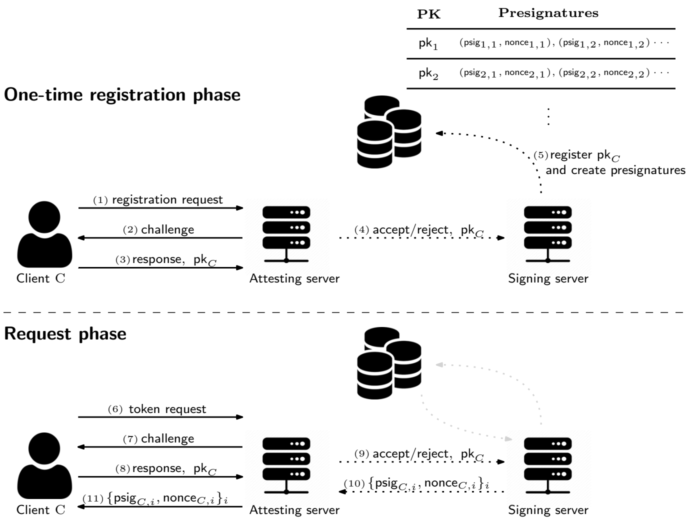
PK
Presignatures
pk_1
$(\mathsf{psig}_{1,1}, \mathsf{nonce}_{1,1}), (\mathsf{psig}_{1,2}, \mathsf{nonce}_{1,2}) \cdots$
pk_2
$(\mathsf{psig}_{2,1}, \mathsf{nonce}_{2,1}), (\mathsf{psig}_{2,2}, \mathsf{nonce}_{2,2}) \cdots$
One-time registration phase
(5) register pk_C and create presignatures
(1) registration request
(2) challenge
(3) response, pk_C
(4) accept/reject, pk_C
Client C
Attesting server
Signing server
Request phase
(6) token request
(7) challenge
(8) response, pk_C
(9) accept/reject, pk_C
(10) $\{\mathsf{psig}_{C,i}, \mathsf{nonce}_{C,i}\}_i$
(11) $\{\mathsf{psig}_{C,i}, \mathsf{nonce}_{C,i}\}_i$
Client C
Attesting server
Signing server

Figure 1. NIAT issuance for CDNs with offline signing.

We describe the idea in Figure 1. This system includes two distinct phases: an one-time registration phase, and a request phase. During the registration phase, clients interact with the issuer to prove their legitimacy by solving some CAPTCHA puzzles. When responding to a challenge, a client will include their public key pk_C alongside the puzzle solution. After verifying a client's solution, the issuer will register the public key pk_C, and then proactively create a batch of presignatures for the newly registered client offline. Notably, an issuer only requires one short message from the client to process a large batch of presignatures. Once a client has registered with the issuer and has been deemed trustworthy, they may start requesting tokens. Since the issuer has the presignatures readily at-hand, the client does not need to wait for online signing processes to occur. This results in a significantly faster turnaround during the request phase. On the issuer's side, they can create new presignatures offline and as long as they always have a new batch of presignatures prepared for all registered clients, the amortized cost for the issuance protocol no longer depends on the number of tokens requested. Rather, only client authentication and network delays incur the most significant cost during the request phase.

Public verifiability and embedding a private metadata bit are directly supported by instantiating the above described system with our NIAT construction. This can allow clients to verify their tokens with multiple servers while also carrying trust signals that can be extracted by the attesting server.

1.1. Related Work

Privacy Pass [1] introduced the idea of one-time use anonymous tokens as a method to prevent DDoS attacks while enhancing privacy and ensuring a seamless user experience of those who access web servers protected by Cloudflare infrastructure. In the simplest setting, Privacy Pass is an interactive, two-move, client-initiated protocol which relies on Verifiable Oblivious Pseudorandom Functions (VOPRFs), and is secure in the random oracle model. The recent partially oblivious PRFs [19] may be used in place of VOPRFs for a more efficient and flexible instantiation of Privacy Pass. One disadvantage of the Privacy Pass protocol is that if a client is deemed malicious, they will not be issued tokens and the request would be dropped. This kind of feedback may inform malicious actors of their detection, which could be leveraged to refine their methods to circumvent bot detection mechanisms.

To address this problem, Kreuter et al. proposed the idea of anonymous tokens with private metadata bit [9] in order to propagate trust signal from the issuers to the verifiers in a private way. They formalized the security properties of this primitive, and proposed a construction called PMBTokens which essentially is an extension of the Privacy Pass protocol to support the private metadata bit. PMBTokens is also based on VOPRFs and secure in the random oracle model similar to Privacy Pass. Similar to Privacy Pass, PMBTokens do not allow any parties other than the the authoritative parties holding the secret issuing keys to verify the validity of tokens, as well as to extract the private metadata bits in the case of PMBTokens. On the other hand, publicly verifiable anonymous tokens with private metadata bit [10] allow for public token verification under the issuer's public key, while the private metadata bit is still hidden from users and only extractable with the issuing secret key. Publicly verifiable anonymous tokens may also be instantiated from RSA blind signatures [20], however they do not support private metadata.

Chase et al. proposed ATHM [11] which is an anonymous token protocol for the setting where the issuer and verifier are the same entity. ATHM is based on a symmetric key primitive - algebraic MACs, and is secure in the generic group model. In addition, they also revisted the definition of an anonymous tokens scheme and merged the two notions of token validity in [9]. Specifically, in the new definition, only tokens from which we can extract the embedded metadata bit successfully are considered valid tokens, and the private verification algorithm $\mathsf{AT.Verify}$ was removed for redundancy. ATHM can also be extended to support public metadata in tokens.

Anonymous Counting Tokens (ACTs) [21] is another variant of anonymous tokens where clients are not able to redeem more than one token per message with the same verifier. This security property can be achieved through two different approaches: either allowing the issuer to detect repeated token issuance requests for the same message from the same client during token issuance, or allowing the verifier to detect two different tokens for the same message were issued to the same client during redemption.

2. Preliminaries

Notation. We use λ to denote the security parameter. We use $\leftarrow\!\!\$$ to denote the output of a randomized algorithm, $\leftarrow$ to denote output of a deterministic algorithm, and $:=$ for assignment. In all that follows, $\mathbb{Z}$ is the ring of integers, and $\mathbb{Z}_p$ denotes the set of integers modulo p. $\mathbb{G}$ is a multiplicative group of prime order p. Lastly, for a vector $\mathbf{x}$, we write x_i as its i^{th} element.

2.1. Bilinear Pairings

Definition 1 (Bilinear pairings)**.** *Given a security parameter λ, a bilinear group generator $\mathsf{BG}(1^\lambda)$ returns a tuple $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \mathsf{e}, p, g_1, g_2)$, where $(\mathbb{G}_1, \mathbb{G}_2$ are groups of the same prime order p with the generators g_1, g_2 respectively. Also, let $\mathbb{Z}_p$ be the field of order p. A bilinear pairing is an efficiently computable map, $\mathsf{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, satisfying the following properties:*

- **Bilinearity:** $\forall P \in \mathbb{G}_1, Q \in \mathbb{G}_2,\ a, b \in \mathbb{Z}_p$,
$$e(P^a, Q)^b = \mathsf{e}(P, Q^b)^a = \mathsf{e}(P, Q)^{ab} \quad .$$
- **Non-degeneracy:** $\mathsf{e}(g_1, g_2) \neq 1_{\mathbb{G}_T}$.

Bilinear pairings can be of a few types depending on whether there is an efficient isomorphism from $\mathbb{G}_1$ to $\mathbb{G}_2$ in both directions (Type 1), only one direction (Type 2), or in neither direction (Type 3) [22]. Type 3 pairings are the most efficient setting for a relevant security parameter and they are commonly deployed.

2.2. Hardness Assumptions

Definition 2 (Inverse decisional Diffie-Hellman)**.** *For all* PPT *adversaries $\mathcal{A}$ given tuple $(\mathbb{G}, g, g^\alpha, g^\beta)$, it is hard to decide whether $\beta = \alpha^{-1} \mod p$ or $\beta \leftarrow\!\!\$\ \mathbb{Z}_p$.*

Definition 3 (k-decisional Diffie-Hellman)**.** *For all* PPT *adversaries $\mathcal{A}$ given tuple $(\mathbb{G}, g, g^\alpha, g^{\alpha^2}, \ldots, g^{\alpha^k}, g^\beta)$, it is hard to decide whether $\beta = \alpha^{-1} \mod p$ or $\beta \leftarrow\!\!\$\ \mathbb{Z}_p$.*

2.3. Zero-knowledge Proofs

We recall the standard definition of a Zero-knowledge (ZK) proof as an interactive protocol between a Prover $\mathcal{P}$ and a Verifier $\mathcal{V}$. The Prover $\mathcal{P}$ must convince the verifier $\mathcal{V}$ that she knows a private witness w for a public instance $x \in \mathscr{L}$ such that $\mathscr{R}(x, w) = 1$, and $\mathcal{V}$ gains no additional information. A ZK proof system consists of the following polynomial time algorithms:

- $\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$. The setup algorithm takes as input the security parameter λ, and outputs a crs.
- $\mathsf{Prove}(\mathsf{crs}, x, w) \to \pi$. The prover algorithm takes as input a crs, an instance $x \in \mathscr{L}$, and a witness w. It outputs a proof π.
- $\mathsf{Verify}(\mathsf{crs}, x, \pi) \to \{0, 1\}$. The verification algorithm takes as input a crs, an instance x, and a proof π. It outputs 0 or 1.

Definition 4 (Zero-knowledge proof)**.** *A zero-knowledge proof between $\mathcal{P}$ and $\mathcal{V}$ for an* NP *relation $\mathscr{R}$ must satisfy the following properties:*

- **Completeness:** If $\mathscr{R}(x, w) = 1$ and both players are honest, $\mathcal{V}$ always accepts.
- **Soundness**: For every malicious and computationally unbounded $\mathcal{P}^*$, there is a negligible function $\epsilon(\cdot)$ such that if x is a false statement (i.e., $\forall w : \mathscr{R}(x, w) = 0$), after $\mathcal{P}^*$ interacts with $\mathcal{V}$, $\Pr[\mathcal{V} \text{ accepts}] \leq \epsilon(|x|)$. Moreover, a proof system is an *argument of knowledge* (AoK) if there exists a PPT extractor $\mathcal{E}$ such that for every stateful PPT attacker $\mathcal{A}$, the following probability is negligible
$$\Pr\left[\begin{array}{l} \mathsf{Verify}(\mathsf{crs}, x, \pi) = 1 \\ \wedge\ \mathscr{R}(x, \mathcal{E}(\mathsf{crs}, x, \pi)) = 0 \end{array} : \begin{array}{r} \mathsf{crs} \leftarrow\$\ \mathsf{Setup}(1^\lambda) \\ (x, \pi) \leftarrow \mathcal{A}(1^\lambda, \mathsf{crs}) \end{array}\right]$$
- **Zero-knowledge**: For every malicious PPT adversary $\mathcal{V}^*$, there exists a PPT simulator $\mathcal{S}$ and a negligible function $\epsilon(\cdot)$ such that for every distinguisher $\mathcal{D}$ and $(x, w) \in R$, the following is negligible in $|x|$
$$|\Pr[\mathcal{D}(\mathsf{View}_{\mathcal{V}^*}(x, w)) = 1] - \Pr[\mathcal{D}(\mathcal{S}) = 1]|$$

Composed statements. ZK proofs can be composed as follows: (1) AND composition $\pi_1 \wedge \pi_2$ can be constructed by sequential or parallel proving of underlying assertions, and (2) OR composition $\pi_1 \vee \pi_2$ can be constructed by proving knowledge for one statement and simulating knowledge for the other, without revealing which of the two is actually proved and which is simulated [23].

Non-interactive zero-knowledge. Public-coin interactive ZK proofs can be made non-interactive in the random oracle model using the Fiat-Shamir heuristic [24].

2.4. Structure-preserving Signatures on Equivalence Classes

Structure-preserving Signatures on Equivalence Classes (SPS-EQ) [18] are used to sign equivalence classes $[M]$ of message vectors $M \in (\mathbb{G}_i^*)^\ell$ for $\ell > 1$, having the equivalence relation $M, N \in \mathbb{G}_i^\ell : M \sim_{\mathscr{R}} N \Leftrightarrow \exists s \in \mathbb{Z}_p : M = N^s$. Let us now recall the definition for SPS-EQ [18], adapting the notation for multiplicative groups as in [16].

Definition 5 (SPS-EQ)**.** *An SPS-EQ scheme consists of the following* PPT *algorithms:*

- $\mathsf{KeyGen}(1^\lambda, \ell) \to (\mathsf{pk_{EQ}}, \mathsf{sk_{EQ}})$. On input the security parameter 1^λ, and the length of message vectors ℓ, it outputs a key pair $(\mathsf{pk_{EQ}}, \mathsf{sk_{EQ}})$.
- $\mathsf{VerKey}(\mathsf{sk_{EQ}}, \mathsf{pk_{EQ}}) \to b \in \{0, 1\}$. On input a public-secret key pair $(\mathsf{pk_{EQ}}, \mathsf{sk_{EQ}})$, it deterministically returns a bit $b \in \{0, 1\}$.
- $\mathsf{Sign}(\mathsf{sk_{EQ}}, M) \to \sigma_{\mathsf{EQ}}$. On input the secret key $\mathsf{sk_{EQ}}$ and a representative $M \in (\mathbb{G}_i^*)^\ell$, it outputs a signature σ_{EQ} for the equivalence class $[M]$.
- $\mathsf{ChRep}(\sigma_{\mathsf{EQ}}, \mu) \to \sigma'_{\mathsf{EQ}}$. On input a signature σ_{EQ} and a scalar μ, it returns an updated message-signature pair $(M', \sigma'_{\mathsf{EQ}})$ where the new representative is $M' = M^\mu$ and σ'_{EQ} is the corresponding updated signature.
- $\mathsf{Verify}(\mathsf{pk_{EQ}}, M, \sigma_{\mathsf{EQ}}) \to b \in \{0, 1\}$. On input a public key $\mathsf{pk_{EQ}}$, a representative $M \in (\mathbb{G}_i^*)^\ell$, and a signature σ_{EQ}, it deterministically outputs a bit $b \in \{0, 1\}$.

```
Game EUnf-CMA_{A,ℓ}(λ)
$Q := \varnothing$
$(\mathsf{pk_{EQ}}, \mathsf{sk_{EQ}}) \leftarrow\$\ \mathsf{EQ.KeyGen}(1^\lambda, \ell)$
$(M^*, \sigma^*_{\mathsf{EQ}}) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(\mathsf{pk_{EQ}})$
return $[M^*] \neq [M]\ \forall M \in Q\ \wedge$
        $\mathsf{Verify}(\mathsf{pk_{EQ}}, M^*, \sigma^*_{\mathsf{EQ}}) = 1$

Oracle $\mathcal{O}_{\mathsf{Sign}}(M)$
$\sigma_{\mathsf{EQ}} \leftarrow\$\ \mathsf{EQ.Sign}(\mathsf{sk_{EQ}}, M)$
$Q := Q \cup \{M\}$
return $\sigma_{\mathsf{EQ}}$
```

Figure 2. The existential unforgeability (under chosen message attack) experiment for SPS-EQ.

Remark 1. We point out that unlike the definition of [18] we have omitted the message M and the public key $\mathsf{pk_{EQ}}$ from the ChRep algorithm. This is without loss of generality, as these inputs are only needed for verifying the signature inside ChRep, and signature verification is a public operation.

An SPS-EQ scheme satisfies correctness, existentially unforgeable under adaptive chosen-message attacks (EUnf-CMA), and perfect signature adaptation under malicious keys. We recall the definitions from [18].

Definition 6 (Correctness)**.** *An SPS-EQ scheme over $\mathbb{G}_i^*$ is correct if for all security parameters $\lambda \in \mathbb{N}$, $\ell > 1$, $(\mathsf{pk_{EQ}}, \mathsf{sk_{EQ}}) \leftarrow\$\ \mathsf{KeyGen}(1^\lambda, \ell)$, $M \in (\mathbb{G}_i^*)^\ell$, such that $\sigma_{\mathsf{EQ}} \leftarrow\$\ \mathsf{EQ.Sign}(\mathsf{sk_{EQ}}, M)$ and for all scalars $\mu \in \mathbb{Z}_p$ we have*

$$\mathsf{EQ.VerKey}(\mathsf{sk_{EQ}}, \mathsf{pk_{EQ}}) = 1\ \wedge$$
$$\Pr[\mathsf{EQ.Verify}(\mathsf{pk_{EQ}}, M, \sigma_{\mathsf{EQ}}) = 1] = 1\ \wedge$$
$$\Pr\left[\mathsf{EQ.Verify}\left(\mathsf{pk_{EQ}}, M^\mu, \mathsf{EQ.ChRep}(\sigma_{\mathsf{EQ}}, \mu)\right) = 1\right] = 1$$

Definition 7 (Existential unforgeability under chosen message attack)**.** *Let $\mathsf{Adv}^{\mathsf{EUnf\text{-}CMA}}_{\mathcal{A},\ell} = \Pr\left[\mathsf{EUnf\text{-}CMA}_{\mathcal{A},\ell}(\lambda) = \mathbf{accept}\right]$ be the advantage of an* PPT *adversary $\mathcal{A}$ in the* EUnf-CMA *experiment defined in Figure 2. An SPS-EQ scheme over $(\mathbb{G}_i^*)^\ell$ is existentially unforgeable under adaptive chosen-message attacks if for all $\ell > 1$, the advantage $\mathsf{Adv}^{\mathsf{EUnf\text{-}CMA}}_{\mathcal{A}} = \mathsf{negl}(\lambda)$.*

Definition 8 (Perfect signature adaptation under malicious keys)**.** *For $\ell > 1$, an SPS-EQ scheme on $(\mathbb{G}_i^*)^\ell$ perfectly adapts signatures under malicious keys if for all tuples $(\mathsf{pk_{EQ}}, M, \sigma_{\mathsf{EQ}}, \mu)$ satisfying*

$$M \in \mathbb{G}_i^*\ \wedge \mathsf{EQ.Verify}(\mathsf{pk_{EQ}}, M, \sigma_{\mathsf{EQ}}) = 1\ \wedge \mu \in \mathbb{Z}_p$$

we have that the output of $\mathsf{EQ.ChRep}(\sigma_{\mathsf{EQ}}, \mu)$ *is a uniformly random element in the space of signatures, conditioned on* $\mathsf{EQ.Verify}(\mathsf{pk}_{\mathsf{EQ}}, M^{\mu}, \sigma'_{\mathsf{EQ}}) = 1$.

In this work, we will use the pairing-based construction of SPS-EQ due to [18].

2.5. Anonymous Tokens with Private Metadata Bit

We recall the anonymous tokens (AT) interface defined by Chase et al. [11]. Formally, it is comprised of three polynomial time algorithms Setup, KeyGen_I and ReadBit, and a probabilistic polynomial time (PPT) interactive token issuance protocol between Client and Issuer.

- $\mathsf{Setup}(1^{\lambda}) \to \mathsf{pp}$. The setup algorithm takes as input the security parameter λ and outputs the protocol's pubic parameters pp.
- $\mathsf{KeyGen}_I(\mathsf{pp}) \to (\mathsf{pk}_I, \mathsf{sk}_I)$. The issuer's key generation algorithm the public parameters pp as input and generates the public key pk_I and secret key sk_I of the issuer.
- $\langle \mathsf{Client}(\mathsf{pk}_I, \tau), \mathsf{Issuer}(\mathsf{sk}_I, b, \tau)\rangle \to (t, \sigma)$ or $\perp$. This is the interactive token issuance protocol between Client and Issuer where the client's input is the issuer's public key pk_I and some public metadata $\tau \in \mathcal{M}$, and the issuer's input is its secret key sk_I, the hidden metadata bit $b \in \{0, 1\}$ and the public metadata τ. It can be broken down into three algorithms.
  - $\mathsf{ClientQuery}(\mathsf{pk}_I, \tau) \to (\mathsf{query}, \mathsf{state})$. Client initiates the protocol by sending query to the Issuer.
  - $\mathsf{IssueToken}(\mathsf{sk}_I, b, \tau, \mathsf{query}) \to \mathsf{resp}$. Issuer replies with resp.
  - $\mathsf{ClientObtain}(\mathsf{state}, \mathsf{resp}) \to (t, \sigma)$. Client locally computes the token from the Issuer's response.
- $\mathsf{ReadBit}(\mathsf{sk}_I, t, \sigma, \tau) \to \{0, 1\}$ or $\perp$. The Issuer's ReadBit algorithm takes as input the issuer's secret key sk_I, a token (t, σ), the public metadata τ and outputs $b \in \{0, 1\}$ or $\perp$.

Correctness of an AT scheme holds if for any given execution of the interactive protocol, the output of the Issuer's ReadBit algorithm is the same as the embedded metadata bit b. In terms of security, an AT scheme must satisfy *one-more unforgeability*, *unlinkability*, and *privacy of the metadata bit*. One-more unforgeability ensures that for an issuer running ℓ times, on some fixed (b, m) pair, an adversary should not be able to produce more than ℓ tokens with pairwise distinct tags. Unlinkability ensures that a malcious issuer can not link tokens to any given client, and more generally to the corresponding issuing sessions. Finally, privacy of the metadata bit ensures that an adversary can do no better than guess the hidden metadata bit with trivial probability. We refer the reader to [11] for the formal security definitions.

3. Non-interactive Anonymous Tokens

Our starting point for defining Non-interactive Anonymous Tokens is the NIBS definition of Baldimtsi et al. [17], which in turn is a slight modification of the original definition given by Hanzlik [16]. Formally, a non-interactive anonymous tokens (NIAT) scheme consists of the following polynomial time algorithms (values in light gray are required only for NIAT with double-spend protection):

- $\mathsf{Setup}(1^{\lambda}) \to (\mathsf{pp}, \mathsf{aux})$. The setup algorithm takes a security parameter λ as input and outputs a set of common public parameters pp (and some auxiliary data aux). All of the remaining algorithms take pp as an input, but for notational clarity, we usually omit it as an explicit input.
- $\mathsf{KeyGen}_I(\mathsf{pp}) \to (\mathsf{pk}_I, \mathsf{sk}_I)$. The issuer's key generation algorithm takes public parameters pp as input, and outputs a public-secret key pair $(\mathsf{pk}_I, \mathsf{sk}_I)$.
- $\mathsf{KeyGen}_C(\mathsf{pp}) \to (\mathsf{pk}_C, \mathsf{sk}_C)$. The client's key generation algorithm takes pp as input, and outputs a public-secret key pair $(\mathsf{pk}_C, \mathsf{sk}_C)$.
- $\mathsf{Issue}(\mathsf{sk}_I, \mathsf{pk}_C, b, \tau, \mathsf{aux}) \to (\mathsf{psig}, \mathsf{nonce}, \mathsf{aux})$. The issuer runs the probabilistic algorithm Issue which takes as input the issuer's secret key sk_I, the client's public key pk_C, a private metadata bit $b \in \{0, 1\}$, and public metadata τ (and some auxiliary data aux). It then outputs a pre-signature psig, and a random nonce (and an updated aux).
- $\mathsf{Obtain}(\mathsf{sk}_C, \mathsf{pk}_I, \mathsf{psig}, \mathsf{nonce}, \tau, \mathsf{id}) \to (t, \sigma)$ or $\perp$. The Obtain algorithm is run by the client to obtain the final token. It takes as input the client's secret key sk_C, the issuer's public key pk_I, a pre-signature psig, and nonce (and a verifier identifier id). It outputs a token (t, σ) if the algorithm runs successfully, or $\perp$ otherwise.
- $\mathsf{Verify}(\mathsf{pk}_I, t, \sigma, \tau) \to b \in \{0, 1\}$ or $\perp$. The public verification algorithm takes as input the issuer's public key pk_I, the token (t, σ) and the public metadata τ, and outputs a bit $b \in \{0, 1\}$ or $\perp$.
- $\mathsf{ReadBit}(\mathsf{sk}_I, t, \sigma, \tau) \to b \in \{0, 1\}$ or $\perp$. On input the issuer's secret key, public metadata τ, a token (t, σ), the issuer's ReadBit algorithm outputs a bit $b \in \{0, 1\}$ or $\perp$.

Furthermore, for a NIAT with double-spend protection, we additionally define:

- $\mathsf{DSIden}(\mathsf{pk}_I, t, \sigma, \sigma', \mathsf{aux}) \to (\mathsf{pk}_C, \Pi)$ or $\perp$. The double-spend identification algorithm takes as input the issuer's public key pk_I, a tag t, two different presentations[3] (σ, σ') with respect to t, and some auxiliary data aux, and outputs the embedded client public key and a proof of guilt (pk_C, Π) or $\perp$.
- $\mathsf{DSVer}(\mathsf{pk}_C, \Pi, \mathsf{aux}) \to b \in \{0, 1\}$. The double-spend verification algorithm takes a client public key pk_C, a proof of guilt Π and the auxiliary data aux, and outputs a bit $b \in \{0, 1\}$.

Comparisons with Tagged NIBS. Essentially, one can view the "tag" in the [17] TNIBS scheme as the public

[3] We consider schemes where the tag t is unique during the execution of the Obtain algorithm (i.e. each tag t is uniquely determined by $(\mathsf{psig}, \mathsf{nonce})$), while the σ part can be prepared during presentation and differ for every possible verifier.

metadata τ, and suitably adapt the definition to fit the NIAT interface in a straightforward way[4]. The main difference between TNIBS and our NIAT definition above, is the input b to the Issue algorithm, and the $\mathsf{ReadBit}$ algorithm to extract the embedded bit from the token. This is a natural extension to meet the goals of an ATPM scheme.

Two notions of validity. Our interface has two algorithms that could signal whether or not a token is valid, which was similar to [9]. However, as pointed out by [11], having two different notions of validity would open up new attack vectors for adversaries. For instance, a malicious client could forge tokens that yield a valid bit, but fail verification or vice-versa. This can be concerning in situations where distinct parties are performing $\mathsf{ReadBit}$ and Verify. To address this problem, we will require the $\mathsf{ReadBit}$ algorithm to execute Verify internally before bit extraction. By doing so, we can keep the notion of validity consistent between the two algorithms, while allowing for public verification, which may make sense in some applications where the cost of token verification is outsourced to a standalone server.

3.1. Security Definitions

Definition 9 (Correctness). *A non-interactive anonymous tokens scheme is correct if for every security parameter $\lambda \in \mathbb{N}$ such that* $\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda), (\mathsf{pk}_I, \mathsf{sk}_I) \leftarrow\!\!\$\ \mathsf{KeyGen}_I(\mathsf{pp}), (\mathsf{pk}_C, \mathsf{sk}_C) \leftarrow\!\!\$\ \mathsf{KeyGen}_C(\mathsf{pp})$*, and for every $b \in \{0,1\}$ and public metadata τ, such that* $(\mathsf{psig}, \mathsf{nonce}) \leftarrow\!\!\$\ \mathsf{Issue}(\mathsf{sk}_I, \mathsf{pk}_C, b, \tau)$ *and* $(t, \sigma) \leftarrow\!\!\$\ \mathsf{Obtain}(\mathsf{sk}_C, \mathsf{pk}_I, \mathsf{psig}, \mathsf{nonce}, \tau)$*, we have*

$$\Pr\left[\mathsf{ReadBit}(\mathsf{sk}_I, t, \sigma, \tau) = b\right] = 1$$

We also formalize the notion of *reusability* for NIAT. This property is not needed in the ATPM definition given by Chase et al. [11] because their scheme is interactive. More precisely, in an ATPM scheme, whenever a client initiates a token issuance session, the client sends to the issuer a query which suffices to uniquely identify that session. This is not possible in the non-interactive setting, as the only information that the issuer knows about a client is its public key pk_C. Reusability allows us to capture the meaningful property that an issuer should be able to issue multiple tokens to the same client's public key pk_C. In other words, an issuer can *reuse* a client's public key pk_C to issue tokens non-interactively.

Definition 10 (Reusability). *A non-interactive anonymous token scheme is reusable if for every security parameter $\lambda \in \mathbb{N}$, metadata bit $b \in \{0,1\}$ and public metadata $\tau \in \mathcal{M}$ the following probability is negligible*

$$\Pr\left[\begin{array}{l} \mathsf{nonce}_0 = \mathsf{nonce}_1 \ \lor\ t_0 = t_1 \ : \\ \hfill \mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda) \\ \hfill (\mathsf{pk}_I, \mathsf{sk}_I) \leftarrow\!\!\$\ \mathsf{KeyGen}_I(\mathsf{pp}) \\ \hfill (\mathsf{pk}_C, \mathsf{sk}_C) \leftarrow\!\!\$\ \mathsf{KeyGen}_C(\mathsf{pp}) \\ \hfill \forall i \in \{0,1\} : (\mathsf{psig}_i, \mathsf{nonce}_i) \leftarrow\!\!\$\ \mathsf{Issue}\begin{pmatrix} \mathsf{sk}_I, \mathsf{pk}_C, \\ b, \tau \end{pmatrix} \\ \hfill \forall i \in \{0,1\} : (t_i, \sigma_i) \leftarrow\!\!\$\ \mathsf{Obtain}\begin{pmatrix} \mathsf{sk}_C, \mathsf{pk}_I, \\ \mathsf{psig}_i, \mathsf{nonce}_i, \tau \end{pmatrix} \end{array}\right]$$

We now define one-more unforgeability for NIAT. A minor point of distinction from the one-more unforgeability definition of Chase et al. [11] is that the Issue oracle $\mathcal{O}_{\mathsf{Issue}}$ (corresponding to their $\mathcal{O}_{\mathsf{Sign}}$) accepts client public keys in place of the client's query message.

Definition 11 (One-more unforgeability). *Let* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{OM\text{-}Unf}} = \Pr\left[\mathsf{OM\text{-}Unf}_{\mathcal{A}}(\lambda) = \mathbf{accept}\right]$ *be the advantage of an adversary $\mathcal{A}$ in the experiment defined in Figure 3. We say a NIAT scheme* NIAT *is one-more unforgeable if for any* PPT *adversary $\mathcal{A}$ this advantage is negligible.*

```
Game OM-Unf_A(λ)

$\mathsf{pp}, \mathsf{aux} \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}_I, \mathsf{sk}_I) \leftarrow\!\!\$\ \mathsf{KeyGen}_I(\mathsf{pp})$
$b^*, \tau^*, \{(t_i, \sigma_i)\}_{i \in [N]} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Issue}}, \mathcal{O}_{\mathsf{Read}}}(\mathsf{pp}, \mathsf{pk}_I)$
Define $\ell$ as the multiplicity of $(b^*, \tau^*)$ in $Q$
if $\#\{t_i\}_{i \in [N]} \leq \ell$ then reject
if $\bigvee_{i \in [N]} \mathsf{ReadBit}(\mathsf{sk}_I, t_i, \sigma_i, \tau^*) \neq b^*$ then reject
if $\bigvee_{i \in [N]} \mathsf{Verify}(\mathsf{pk}_I, t_i, \sigma_i, \tau^*) \neq 1$ then reject
accept

Oracle $\mathcal{O}_{\mathsf{Issue}}(\mathsf{pk}_C, b, \tau)$

if multiset $Q$ is uninitialized then $Q := \varnothing$
$Q := Q \cup \{(b, \tau)\}$
$(\mathsf{psig}, \mathsf{nonce}, \mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Issue}(\mathsf{sk}_I, \mathsf{pk}_C, b, \tau, \mathsf{aux})$
return $(\mathsf{psig}, \mathsf{nonce})$

Oracle $\mathcal{O}_{\mathsf{Read}}(t, \sigma, \tau)$

return $\mathsf{ReadBit}(\mathsf{sk}_I, t, \sigma, \tau)$
```

Figure 3. The one-more unforgeability experiment for NIAT.

Next, we formalize NIAT unlinkability. The overall intuition here remains the same as in [11], except that the adversary is now equipped with a $\mathsf{GenUser}$ oracle $\mathcal{O}_{\mathsf{GenUser}}$ that issues new public keys, and an Obtain oracle $\mathcal{O}_{\mathsf{Obtain}}$ that returns the tokens on chosen $(\mathsf{psig}, \mathsf{nonce}, \tau)$ triples.

Definition 12 (κ-unlinkability). *Let* $\mathsf{Adv}_{\mathcal{A},n}^{\mathsf{UNLINK}} = \Pr\left[\mathsf{UNLINK}_{\mathcal{A},n}(\lambda) = \mathbf{accept}\right]$ *be the advantage of an adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ for parameter $n \in \mathbb{N}$ in the experiment defined in Figure 4. We say a NIAT scheme* NIAT *is κ-unlinkable if for any* PPT *adversary $\mathcal{A}$ and $n \in \mathbb{N}$, this*

[4] Note that one can similarly extend a definition from (un-tagged) NIBS in the absence of public metadata.

```
Game $\mathsf{UNLINK}_{\mathcal{A},n}(\lambda)$

$\mathsf{pp}, \mathsf{aux} \leftarrow\$ \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}_I, \mathsf{state}_1) \leftarrow \mathcal{A}_1(\mathsf{pp}, \mathsf{aux})$
$(Q^*, \tau, \mathsf{id}, \mathsf{state}_2) \leftarrow \mathcal{A}_2^{\mathcal{O}_{\mathsf{GenUser}}, \mathcal{O}_{\mathsf{Obtain}}}(\mathsf{state}_1)$
foreach $(\mathsf{pk}_C, \mathsf{psig}, \mathsf{nonce}) \in Q^*$
    if $(\mathsf{pk}_C, \cdot) \notin Q_{\mathsf{usr}}$
    $\vee\ (\mathsf{pk}_C, \cdot, \mathsf{nonce}) \in Q_{\mathsf{obt}}$ then reject
if $|Q^*| \neq n$ then reject
foreach $i \in [n]$ do
    $\mathsf{out}_i \leftarrow\$ \mathsf{Obtain}(\mathsf{sk}_{C_i}, \mathsf{pk}_I, \mathsf{psig}_i, \mathsf{nonce}_i, \tau, \mathsf{id})$
    if $\mathsf{out}_i = \bot$ then reject
$\hat{i} \leftarrow\$ [n]$
Pick a random permutation $\varphi$ of $[n]$
$i^* \leftarrow \mathcal{A}_3\left(\mathsf{state}_2, \mathsf{out}_{\hat{i}}, (\mathsf{out}_{\varphi(i)})_{i \in [n] \setminus \{\hat{i}\}}\right)$
if $i^* = \hat{i}$ then accept

Oracle $\mathcal{O}_{\mathsf{GenUser}}(\mathsf{pp})$

if $Q_{\mathsf{usr}}$ is uninitialized then $Q_{\mathsf{usr}} := \varnothing$
$(\mathsf{pk}_C, \mathsf{sk}_C) \leftarrow\$ \mathsf{KeyGen}_C(\mathsf{pp})$
$Q_{\mathsf{usr}} := Q_{\mathsf{usr}} \cup \{(\mathsf{pk}_C, \mathsf{sk}_C)\}$
return $\mathsf{pk}_C$

Oracle $\mathcal{O}_{\mathsf{Obtain}}(\mathsf{pk}_C, \mathsf{psig}, \mathsf{nonce}, \tau, \mathsf{id})$

if $Q_{\mathsf{obt}}$ is uninitialized then $Q_{\mathsf{obt}} := \varnothing$
if $(\mathsf{pk}_C, \cdot) \notin Q_{\mathsf{usr}}$ then return $\bot$
Retrieve $\mathsf{sk}_C$ from $Q_{\mathsf{usr}}$
$\mathsf{out} \leftarrow\$ \mathsf{Obtain}(\mathsf{sk}_C, \mathsf{pk}_I, \mathsf{psig}, \mathsf{nonce}, \tau, \mathsf{id})$
if $\mathsf{out} \neq \bot$ then
    $Q_{\mathsf{obt}} := Q_{\mathsf{obt}} \cup \{(\mathsf{pk}_C, \mathsf{psig}, \mathsf{nonce})\}$
return $\mathsf{out}$
```

Figure 4. The unlinkability experiment for NIAT.

advantage at most $\kappa/n + \mathsf{negl}(\lambda)$[5].

Concretely, the parameter n is the number of pairwise distinct $(\mathsf{pk}_C, \mathsf{psig}, \mathsf{nonce})$ triples in the challenge set Q^*. Notice that we do not enforce any other constraints on the values inside Q^*. This allows for cases where some public keys can repeat. In particular, when $n = 2$, $\kappa = 1$, and $\mathsf{pk}_{C_1} \neq \mathsf{pk}_{C_2}$ (resp. $\mathsf{pk}_{C_1} = \mathsf{pk}_{C_2}$), we can view this experiment as being analogous to the stronger versions of (T)NIBS receiver (resp. nonce) blindness [17]. Thus we view our unlinkability definition as a generalization of the (T)NIBS blindness definitions.

Remark 2. An important requirement for unlinkability is that all $(\mathsf{psig}, \mathsf{nonce})$ pairs in Q^* must be under the same public metadata, otherwise the property is trivially broken. Although not stated explicitly, this also holds for the definition given in [11].

```
Game $\mathsf{PMB}_{\mathcal{A},\hat{b}}(\lambda)$

$\mathsf{pp}, \mathsf{aux} \leftarrow\$ \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}_I, \mathsf{sk}_I) \leftarrow\$ \mathsf{KeyGen}_I(\mathsf{pp})$
$\mathsf{flag} := \mathsf{False}$
$b^* \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Issue}}, \mathcal{O}_{\mathsf{Read}}, \mathcal{O}_{\mathsf{Challenge}}}(\mathsf{pp}, \mathsf{pk}_I)$
if $b^* = \hat{b}$ then accept

Oracle $\mathcal{O}_{\mathsf{Issue}}(\mathsf{pk}_C, b, \tau)$

$(\mathsf{psig}, \mathsf{nonce}, \mathsf{aux}) \leftarrow\$ \mathsf{Issue}(\mathsf{sk}_I, \mathsf{pk}_C, b, \tau, \mathsf{aux})$
return $\mathsf{psig}, \mathsf{nonce}$

Oracle $\mathcal{O}_{\mathsf{Read}}(t, \sigma, \tau)$

if $\mathsf{flag} \wedge (\tau = \hat{\tau})$ then return $\bot$
return $\mathsf{ReadBit}(\mathsf{sk}_I, t, \sigma, \tau)$

Oracle $\mathcal{O}_{\mathsf{Valid}}(t, \sigma, \tau)$

$b \leftarrow \mathsf{ReadBit}(\mathsf{sk}_I, t, \sigma, \tau)$
return $(b \neq \bot)$

Oracle $\mathcal{O}_{\mathsf{Challenge}}(\mathsf{pk}_C, \tau)$

if $\mathsf{flag}$ then return $\bot$
$\mathsf{flag} := \mathsf{True}, \hat{\tau} := \tau$
$(\mathsf{psig}, \mathsf{nonce}, \mathsf{aux}) \leftarrow\$ \mathsf{Issue}(\mathsf{sk}_I, \mathsf{pk}_C, \hat{b}, \hat{\tau}, \mathsf{aux})$
return $(\mathsf{psig}, \mathsf{nonce})$
```

Figure 5. The privacy of metadata bit experiment for NIAT.

We then define the equivalent notion of privacy for the metadata bit in a NIAT.

Definition 13 (Privacy of metadata bit)**.** *Let* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{PMB}} = |\Pr[\mathsf{PMB}_{\mathcal{A},1}(\lambda) = \mathbf{accept}] - \Pr[\mathsf{PMB}_{\mathcal{A},0}(\lambda) = \mathbf{accept}]|$ *be the advantage of an adversary $\mathcal{A}$ in the experiment defined in Figure 5. We say a NIAT scheme* NIAT *preserves privacy of the metadata bit if for any* PPT *adversary $\mathcal{A}$, this advantage is negligible.*

NIAT with Double-Spending Identification Security. For a NIAT scheme with double-spending identification security we define two additional properties: double-spending identification and exculpability. Double-spending identification guarantees that no adversary is able to present the same token twice without having their public key revealed. Formally, for oracles defined as above,

Definition 14 (Double-spending identification)**.** *Let* $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{Iden}} = \Pr[\mathsf{Iden}_{\mathcal{A}}(\lambda) = \mathbf{accept}]$ *be the advantage of an adversary $\mathcal{A}$ in the experiment defined in Figure 6. We say a NIAT scheme* NIAT *achieves double-spending identification if for any* PPT *adversary $\mathcal{A}$, this advantage is negligible.*

[5] Intuitively, the probability κ/n quantifies the best strategy for an honest issuer that has κ choices for each private metadata b_i. In the case that the private metadata is a bit, ATPM (and thus also NIAT) schemes generally focus on the case where $\kappa = 2$.

```
Game $\mathsf{Iden}_{\mathcal{A}}(\lambda)$

$(\mathsf{pp}, \mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}_I, \mathsf{sk}_I) \leftarrow\!\!\$\ \mathsf{KeyGen}_I(\mathsf{pp})$
$(t, \sigma, \sigma') \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Issue}}, \mathcal{O}_{\mathsf{Read}}}(\mathsf{pp}, \mathsf{pk}_I)$
$\textbf{if } \mathsf{Verify}(\mathsf{pk}_I, t, \sigma) = 1 \wedge \mathsf{Verify}(\mathsf{pk}_I, t, \sigma') = 1$
    $\wedge\ \mathsf{DSIden}\left(\mathsf{pk}_I, t, \sigma, \sigma', \mathsf{aux}\right) = \bot\ \textbf{then accept}$
```

Figure 6. The double-spending identification experiment for NIAT. Oracles $\mathcal{O}_{\mathsf{Issue}}$ and $\mathcal{O}_{\mathsf{Read}}$ are as defined in Figure 3.

We finally define exculpability which guarantees that an issuer cannot wrongly accuse an honest client of double-spending. Formally,

Definition 15 (Double-spending exculpability)**.** *Let* $\mathsf{Adv}^{\mathsf{Excul}}_{\mathcal{A}} = \Pr\left[\mathsf{Excul}_{\mathcal{A}}(\lambda) = \textbf{accept}\right]$ *be the advantage of an adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *in the experiment defined in Figure 7. We say a NIAT scheme* NIAT *achieves double-spending exculpability if for any* PPT *adversary* $\mathcal{A}$*, this advantage is negligible.*

```
Game $\mathsf{Excul}_{\mathcal{A}}(\lambda)$

$(\mathsf{pp}, \mathsf{aux}) \leftarrow\!\!\$\ \mathsf{Setup}(1^\lambda)$
$(\mathsf{pk}_I, \mathsf{state}_1) \leftarrow \mathcal{A}_1(\mathsf{pp})$
$(\mathsf{pk}_C, \Pi) \leftarrow \mathcal{A}_2^{\mathsf{O}_{\mathsf{GenUser}}, \mathcal{O}_{\mathsf{Obtain}}}(\mathsf{state}_1)$
$\textbf{if } (\mathsf{pk}_C, \cdot) \in Q_{\mathsf{usr}} \wedge \mathsf{DSVer}(\mathsf{pk}_C, \Pi, \mathsf{aux}) = 1$
    $\textbf{then accept}$
```

Figure 7. The double-spend exculpability experiment for NIAT. Oracles $\mathcal{O}_{\mathsf{GenUser}}$ and $\mathcal{O}_{\mathsf{Obtain}}$ are as defined in Figure 4, except $\mathcal{O}_{\mathsf{Obtain}}$ behaves as an honest user that accepts a given $(\mathsf{psig}, \mathsf{nonce})$ pair exactly once.

4. NIAT from Equivalence Class Signatures

We now present our NIAT construction from structure-preserving signatures on equivalence classes. At a high level, we extend the NIBS construction of [16] to additionally embed the private metadata bit into the issuer's presignature. This is a non-trivial exercise as the bit must also be extractable from the final token, under the issuer's secret key. We resolve this by hiding the bit inside the exponent of a random group element and sending the result as part of the presignature. Thus, when a client sends the re-randomized signature, the part corresponding to the hiddent bit can be checked in the exponent using the issuer's secret key.

In all that follows, we will ignore the public metadata τ for simplicity, instead noting that (and as we show in the full version) all our constructions are extendable to include τ in a manner similar to [16].

4.1. Construction

Tools required. Our construction requires a hash function $\mathsf{H} : \{0,1\}^\lambda \to \mathbb{G}_1$ modeled as a random oracle, an SPS-EQ scheme $\mathsf{EQ} = (\mathsf{EQ.Setup}, \mathsf{EQ.Sign}, \mathsf{EQ.ChRep}, \mathsf{EQ.Verify}, \mathsf{EQ.VerKey})$, and a NIZKAoK $\mathsf{NIZK} = (\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove}, \mathsf{NIZK.Verify})$ for the following language:

Language $\mathscr{L}_{\mathsf{iss}}$

Instance: Each instance x is interpreted as a collection of group elements pk_I and elements R, S.
Witness: Witness w consists of an integer vector $\mathsf{sk}_I := (x_1, x_2, y_1, y_2)$ and a bit b.
Membership: w is a valid witness for x if the following are satisfied:

$$b \in \{0,1\}\ \wedge\ S = R^{(1-b)x_1 + bx_2}\ \wedge$$
$$\mathsf{pk}_I = (g_1^{x_1}, g_1^{x_2}, g_2^{y_1}, g_2^{y_2})$$

System setup. This algorithm sets up the generators for the bilinear groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order p. It also runs the NIZKAoK setup to generate the crs for the language $\mathscr{L}_{\mathsf{iss}}$.

```
$\mathsf{Setup}(1^\lambda)$

$(\mathsf{pp}_{\mathsf{EQ}}, g_1, g_2, p) \leftarrow\!\!\$\ \mathsf{EQ.Setup}(1^\lambda)$
$\mathsf{crs} \leftarrow\!\!\$\ \mathsf{NIZK.Setup}(1^\lambda, \mathsf{pp}_{\mathsf{EQ}})$
$\textbf{return } \mathsf{pp} := (\mathsf{crs}, \mathsf{pp}_{\mathsf{EQ}}, g_1, g_2, p, \mathsf{H})$
```

Key generation. The issuer's key generation algorithm samples a random integer vector $\mathbf{x}$ and runs the SPS-EQ setup algorithm. The client's key generation algorithm samples a random value as the secret key, and sets the public key with respect to this secret.

```
$\mathsf{KeyGen}_I(\mathsf{pp})$

$\mathbf{x} \leftarrow\!\!\$\ (\mathbb{Z}_p^*)^2$
$g_1^{\mathbf{y}}, \mathbf{y} \leftarrow\!\!\$\ \mathsf{EQ.Setup}(\mathsf{pp}_{\mathsf{EQ}}, 2)$
$\textbf{return}\ \mathsf{pk}_I := (g_1^{\mathbf{x}}, g_2^{\mathbf{y}}),\ \mathsf{sk}_I := (\mathbf{x}, \mathbf{y})$
```

```
$\mathsf{KeyGen}_C(\mathsf{pp})$

$\alpha \leftarrow\!\!\$\ \mathbb{Z}_p^*$
$\textbf{return}\ \mathsf{pk}_C := g_1^\alpha,\ \mathsf{sk}_C := \alpha$
```

Presignature issuance and token generation. We show the presignature issuance and token generation protocols in detail in Figure 8. To issue a presignature, the issuer creates an equivalence class signature on $(\mathsf{pk}_C, \mathsf{H}(r) \cdot S)$, where group element $S \in \mathbb{G}_1$ embeds the metadata bit b. The presignature psig includes the signature $\overline{\boldsymbol{\sigma}}$ and the bit embedding S, and the nonce is the uniformly chosen hash input r. We must additionally include a NIZK proof π in psig, proving $b \in \{0,1\}$, the the knowledge of secret keys corresponding to the public keys and the correct computation of the bit embedding. In order to obtain a redeemable token, the client first verifies the NIZK proof and the equivalence class signature. It can then transform the representation of $\overline{\boldsymbol{\sigma}}$ to the equivalence class signature on $(g_1, (\mathsf{H}(r) \cdot S)^{\alpha^{-1}})$. The final token is the tag $\mathbf{t} = \left(\mathsf{H}(r)^{\alpha^{-1}}, S^{\alpha^{-1}}\right)$ along with the transformed signature $\boldsymbol{\sigma}$. We discuss the protocol for validation of this token next.

Public verification (token redemption). The public verification algorithm simply verifies the equivalence class signature $\boldsymbol{\sigma}$.

```
Issuer$^{(b)}$                                                    Client
$\underbrace{\mathbf{x} = (x_1, x_2, y_1, y_2) \in (\mathbb{Z}_p^*)^4}_{\mathsf{sk}_I}$ ,                  $\underbrace{\alpha \in \mathbb{Z}_p^*}_{\mathsf{sk}_C}$ , $\underbrace{g_1^{\alpha} \in \mathbb{G}_1}_{\mathsf{pk}_C}$
$\underbrace{g_1^{\mathbf{x}} = (g_1^{x_1}, g_1^{x_2}) \in \mathbb{G}_1^2,\ g_2^{\mathbf{y}} = (g_2^{y_1}, g_2^{y_2}) \in \mathbb{G}_2^2}_{\mathsf{pk}_I}$
.............................. Issue ..............................
$r \leftarrow\!\!\$\ \{0,1\}^{\lambda},\ R \coloneqq \mathsf{H}(r),\ S \coloneqq R^{(1-b)x_1+bx_2}$
$\overline{\boldsymbol{\sigma}} \leftarrow\!\!\$\ \mathsf{EQ.Sign}\left((y_1, y_2), (\mathsf{pk}_C, R \cdot S)\right)$
$\pi \leftarrow\!\!\$\ \mathsf{NIZK.Prove}\begin{pmatrix} \mathsf{crs},\ x \coloneqq (\mathsf{pk}_I, R, S) \\ w \coloneqq (\mathsf{sk}_I, b) \end{pmatrix}$   $\xrightarrow{\mathsf{psig} \coloneqq (\overline{\boldsymbol{\sigma}}, S, \pi),\ \mathsf{nonce} \coloneqq r}$
.............................. Obtain ..............................
                                                    $R \coloneqq \mathsf{H}(r),\ \mathbf{m} \coloneqq (\mathsf{pk}_C, R \cdot S)$
                                                    if $\mathsf{NIZK.Verify}(\mathsf{crs}, x \coloneqq (\mathsf{pk}_I, R, S), \pi) \neq 1$
                                                    $\vee\ \mathsf{EQ.Verify}\left((\mathsf{pk}_I^{(3)}, \mathsf{pk}_I^{(4)}), \mathbf{m}, \overline{\boldsymbol{\sigma}}\right) \neq 1$
                                                      then return $\perp$
                                                    $\mathbf{t} \coloneqq \left(R^{\alpha^{-1}}, S^{\alpha^{-1}}\right)$
                                                    $\boldsymbol{\sigma} \leftarrow\!\!\$\ \mathsf{EQ.ChRep}(\overline{\boldsymbol{\sigma}}, \alpha^{-1})$
```

Figure 8. SPS-EQ construction for NIAT presignature issuance and token generation.

```
$\mathsf{Verify}(\mathsf{pk}_I, \mathbf{t}, \boldsymbol{\sigma})$
return $\mathsf{EQ.Verify}((\mathsf{pk}_I^{(3)}, \mathsf{pk}_I^{(4)}), (g_1, t_1 \cdot t_2), \boldsymbol{\sigma})$
```

Bit extraction. The bit extraction proceeds by first calling the public verification algorithm. Then, the issuer simply checks which $b \in \{0,1\}$ satisfies $t_1^{x_{1+b}} = t_2$ and returns that bit if one is found, otherwise it returns an error.

```
$\mathsf{ReadBit}(\mathsf{sk}_I, \mathbf{t}, \boldsymbol{\sigma})$
if $\mathsf{Verify}(\mathsf{pk}_I, \mathbf{t}, \boldsymbol{\sigma}) \neq 1$ then
  return $\perp$
foreach $b \in \{0,1\}$ do
  if $t_1^{\mathsf{sk}_I^{(1+b)}} = t_2$ then
    return $b$
return $\perp$
```

Correctness. Correctness of Issue follows from correctness of the SPS-EQ scheme EQ, and that of the NIZKAoK scheme NIZK. Thus, $\overline{\boldsymbol{\sigma}}$ is a valid signature on $(\mathsf{pk}_C, \mathsf{H}(r) \cdot S)$, and π is a valid NIZKAoK for $b \in \{0,1\}$ and the well-formedness of psig. Furthermore, correctness of EQ also guarantees the correctness of Obtain. Consequently, $\boldsymbol{\sigma}$ is a valid signature on $(g_1, (\mathsf{H}(r) \cdot S)^{\alpha^{-1}})$. Lastly, we have that

$$t_1^{x_{1+b}} = \mathsf{H}(r)^{\frac{x_{1+b}}{\alpha}} = S^{\frac{1}{\alpha}} = t_2$$

where $\alpha =: \mathsf{sk}_C$. Therefore, NIAT Construction 4 satisfies correctness.

Reusability. Let t_i be the tag created by the client in session i for $i \in \{0,1\}$, and similarly let $(\overline{\boldsymbol{\sigma}}_i, S_i, \pi_i) =: \mathsf{psig}_i$ and $r_i =: \mathsf{nonce}_i$ be the corresponding presignature and nonce. Then,

$$\begin{aligned} &\Pr[r_0 = r_1 \ \vee\ \mathbf{t}_0 = \mathbf{t}_1] \\ =\ &\Pr[r_0 = r_1] + \Pr[\mathbf{t}_0 = \mathbf{t}_1] \\ &\quad - \Pr[\mathbf{t}_0 = \mathbf{t}_1 \mid r_0 = r_1] \cdot \Pr[r_0 = r_1] \end{aligned}$$

Since, according to the definition of reusability, the metadata b is the same for both issuances, this probability is

$$\begin{aligned} &= \Pr[r_0 = r_1] + \Pr[\mathbf{t}_0 = \mathbf{t}_1] - 1 \cdot \Pr[r_0 = r_1] \\ &= \Pr\left[\mathsf{H}(r_0)^{\alpha^{-1}} = \mathsf{H}(r_1)^{\alpha^{-1}} \ \wedge\ S_0^{\alpha^{-1}} = S_1^{\alpha^{-1}}\right] \\ &= \Pr[\mathsf{H}(r_0) = \mathsf{H}(r_1)] \quad . \end{aligned}$$

Since each r_i is chosen uniformly from $\{0,1\}^{\lambda}$, we have by collision resistance of H that the above probability is negligible. So, Construction 4 satisfies reusability.

Security. Below, we present our main theorems for NIAT security of Construction 4. Due to space constraints, we provide the proofs in Appendix A.

Theorem 3 (One-more unforgeability)**.** *Construction 4 is one-more unforgeable (in the random oracle model) assuming* NIZK *satisfies zero knowledge and* EQ *is existentially unforgeable under adaptively chosen-message attacks.*

Theorem 4 (κ-unlinkability)**.** *Construction 4 is κ-unlinkable (in the random oracle model) for $\kappa = 2$, assuming* NIZK *is an argument of knowledge, that inverse DDH assumption holds in $\mathbb{G}_1$ and* EQ *perfectly adapts signatures under a malicious signer.*

Theorem 5 (Privacy of metadata bit)**.** *Construction 4 has private metadata bit (in the random oracle model) assuming* NIZK *satisfies zero-knowledge, that DDH assumption holds and* EQ *is existentially unforgeable under adaptively chosen-message attacks.*

4.2. Instantiation

We use the SPS-EQ scheme given by [18] to instantiate our protocol from Figure 8. We provide the expanded token issuance and generation protocol in Figure 11 of the Appendix.

Issuer's ZK proof. An essential part of the issuing protocol is the disjunctive OR-proof for consistency of the secret values x_1 and x_2 used in $S = R^{(1-b)x_1+bx_2}$ against the public key pk_I. Specifically, the OR-proof proves that $S =$

R^{x_1} (when $b = 0$), or R^{x_2} (when $b = 1$). Disjunctive proofs of two statements can be constructed by proving knowledge for one statement and simulating knowledge for the other, without revealing which of the two is actually proved and which is simulated [23]. We detail the full ZK protocol for the language $\mathscr{L}_{\mathsf{iss}}$ in Appendix B.

A note on batch verification. The main computational bottleneck in our protocol is due to the pairing operations required for signature verification during, both, token generation and redemption (verification). However, we observe that this cost can actually be amortized over the number of presignatures issued (resp., tokens redeemed) as some of the pairing operations are jointly verifiable. We will take advantage of this fact when we discuss our concrete implementation in Section 6. The precise verification check is explained with the concrete instantitation in Appendix B.

5. Double-spend Protection

While one-more unforgeability prevents a client from creating more than one tokens from a single presignature issuance, a NIAT system by itself does not preclude a client from attempting to redeem the same token twice. More importantly, even if an issuer were able to detect such a double-spending attempt by keeping track of all redeemed tokens, unlinkability seems to suggest that it may be hard to identify the offending client. However, we note that this is not the case—unlinkability with respect to a *fixed* client is defined for distinct nonces, and it turns out that a NIAT scheme that allows the public identification of a client attempting to redeem the same token more than once is perfectly within the bounds of Definition 12.

5.1. Construction

We give a modification of Construction 4 that facilitates protection against double-spending by allowing a verifier to identify the offending client's public key. We utilize the fact that the presignature already embeds the client's public key pk_C (a unique property of our NIAT construction compared to interactive constructions). Due to space constraints, we omit algorithms which are identical to (or trivially extendable from) their counterparts in Construction 4.

Tools required. Our modified construction requires hash functions $\mathsf{H}_{\mathbb{G}} : \{0,1\}^\lambda \to \mathbb{G}_1$, $\mathsf{H}_{\mathbb{Z}} : \{0,1\}^\lambda \to \mathbb{Z}_p$, $\mathsf{H}_{\mathbb{G}\to\mathbb{G}} : \mathbb{G}_1 \to \mathbb{G}_1$, an SPS-EQ scheme $\mathsf{EQ} = (\mathsf{EQ.Setup}, \mathsf{EQ.Sign}, \mathsf{EQ.ChRep}, \mathsf{EQ.Verify}, \mathsf{EQ.VerKey})$, $\mathsf{NIZK}_{\mathsf{iss}} = (\mathsf{NIZK}_{\mathsf{iss}}.\mathsf{Setup}, \mathsf{NIZK}_{\mathsf{iss}}.\mathsf{Prove}, \mathsf{NIZK}_{\mathsf{iss}}.\mathsf{Verify})$ and $\mathsf{NIZK}_{\mathsf{obt}} = (\mathsf{NIZK}_{\mathsf{obt}}.\mathsf{Setup}, \mathsf{NIZK}_{\mathsf{obt}}.\mathsf{Prove}, \mathsf{NIZK}_{\mathsf{obt}}.\mathsf{Verify})$ where the former is a NIZKAoK for $\mathscr{L}_{\mathsf{iss}}$ defined previously, and the latter is a NIZK for the following language:

Language $\mathscr{L}_{\mathsf{obt}}$
Instance: Each instance x is interpreted as group elements t_1, ϕ_1 and ϕ_2 and an integer id.
Witness: Witness w consists of integers sk_C and $\hat{r}$.
Membership: w is a valid witness for x if the following are satisfied:

$$\phi_2 = g_1^{\mathsf{id}\cdot(\mathsf{sk}_C+\hat{r})} \cdot \phi_1^{\mathsf{sk}_C+\hat{r}} \ \wedge \ t_1 = g_1^{(\mathsf{sk}_C+\hat{r})^{-1}}$$

Presignature issuance and token generation. In Figure 9, we show the presignature issuance and token generation protocols in detail, and highlight the differences from the previous construction. The main distinction in issuance is the inclusion of the integer $\hat{r}$ computed as $\mathsf{H}_{\mathbb{Z}}(r)$ and the signature, which is now computed over $\mathbf{m} := (g_1, \mathsf{pk}_C \cdot R \cdot g_1^{\hat{r}}, S)$. As we will see, this is the key to our double-spend identification mechanism.

To obtain a token, a client performs the same steps as before, except that instead of $\mathbf{m}^{\alpha^{-1}}$, it now computes the transformed signature over $\mathbf{m}^{(\alpha+\hat{r})^{-1}}$ with the corresponding tag $\mathbf{t} := \left(g_1^{(\alpha+\hat{r})^{-1}}, R^{(\alpha+\hat{r})^{-1}}, S^{(\alpha+\hat{r})^{-1}}\right)$.

Now, if a client intends to redeem its token with some verifier with public identifier id, as we show in Figure 9, the client's Obtain algorithm must additionally compute an ElGamal ciphertext $\boldsymbol{\phi}$ over id along with a proof π_{o} of the well-formedness of the ciphertext. Therefore, the final signature is the equivalence class signature $\mathbf{s}$, the ciphertext $\boldsymbol{\phi}$ and the proof π_{o}.

Public verification. The public verification algorithm verifies the equivalence class signature $\mathbf{s}$ and the ZK proof π_{o}.

Double-spend identification. Suppose that a client attempts to double-spend a token (i.e., one created under the same presignature and nonce pair) with another verifier with public identifier id'. In order to do so, it must create a fresh encryption $\boldsymbol{\phi}'$ over id' and the corresponding proof π'_{o} with the same $\mathbf{s}$ and $\mathbf{t}$. When a system detects that a double-spend of $(\mathbf{t}, \mathbf{s})$ has occurred, it can compute $\left({\phi_2}^{-1} \cdot \phi'_2\right)^{(\mathsf{id}'-\mathsf{id})^{-1}} = g_1^{\alpha+\hat{r}}$ and then obtain the offending client's public key by checking against the available nonces. Note that we have assumed distinct id's for ease of exposition, however, this is just as easily accomplished for the same verifier by hashing a timestamp into the identifier.

```
$\mathsf{DSIden}(\mathsf{pk}_I, \mathbf{t}, \boldsymbol{\sigma}, \boldsymbol{\sigma}', \mathsf{aux})$
if $\mathsf{Verify}(\mathsf{pk}_I, \mathbf{t}, \boldsymbol{\sigma}) \wedge \mathsf{Verify}(\mathsf{pk}_I, \mathbf{t}, \boldsymbol{\sigma}') \neq 1$ then
    return $\bot$
$\boldsymbol{\sigma} =: (\mathbf{s}, \boldsymbol{\phi}, \pi_{\mathsf{o}}, \mathsf{id})$ , $\boldsymbol{\sigma}' =: (\mathbf{s}', \boldsymbol{\phi}', \pi'_{\mathsf{o}}, \mathsf{id}')$
$h := \left({\phi_2}^{-1} \cdot \phi'_2\right)^{(\mathsf{id}'-\mathsf{id})^{-1}}$
foreach $(\mathsf{pk}_C, \mathsf{nonce}) \in \mathsf{aux}$ do
    if $h = \mathsf{pk}_C \cdot g_1^{\mathsf{H}_{\mathbb{Z}}(\mathsf{nonce})}$ then
        return $\mathsf{pk}_C, \Pi := (\mathbf{t}, \boldsymbol{\sigma}, \boldsymbol{\sigma}', \mathsf{nonce})$
return $\bot$
```

Notably, this step can be performed publicly by anyone

Issuer$^{(b)}$ Client$^{(\mathsf{id})}$

$$\underbrace{\mathbf{x} = (x_1, x_2, y_1, y_2, y_3) \in (\mathbb{Z}_p^*)^5}_{\mathsf{sk}_I}\ , \qquad \underbrace{\alpha \in \mathbb{Z}_p^*}_{\mathsf{sk}_C}\ , \underbrace{g_1^{\alpha} \in \mathbb{G}_1}_{\mathsf{pk}_C}$$

$$\underbrace{g_1^{\mathbf{x}} = (g_1^{x_1}, g_1^{x_2}) \in \mathbb{G}_1^2,\ g_2^{\mathbf{y}} = (g_2^{y_1}, g_2^{y_2}, g_2^{y_3}) \in \mathbb{G}_2^3}_{\mathsf{pk}_I}$$

.... Issue ....

$r \leftarrow\!\!\$\ \{0,1\}^{\lambda},\ R := \mathsf{H}_{\mathbb{G}}(r),\ \hat{r} := \mathsf{H}_{\mathbb{Z}}(r)$

$S := R^{(1-b)x_1 + bx_2}$

$\overline{\boldsymbol{\sigma}} \leftarrow\!\!\$\ \mathsf{EQ.Sign}\left((y_1, y_2, y_3), (g_1, \mathsf{pk}_C \cdot g_1^{\hat{r}}, R \cdot S)\right)$

$\pi_{\mathsf{i}} \leftarrow\!\!\$\ \mathsf{NIZK}_{\mathsf{iss}}.\mathsf{Prove}\begin{pmatrix} \mathsf{crs}_{\mathsf{iss}},\ x := (\mathsf{pk}_I, R, S) \\ w := (\mathsf{sk}_I, b) \end{pmatrix} \qquad \xrightarrow{\mathsf{psig} := (\overline{\boldsymbol{\sigma}}, S, \pi), \mathsf{nonce} := r}$

.... Obtain ....

$R := \mathsf{H}_{\mathbb{G}}(r),\ \hat{r} := \mathsf{H}_{\mathbb{Z}}(r),\ \mathbf{m} := (g_1, \mathsf{pk}_C \cdot g_1^{\hat{r}}, R \cdot S)$

if $\mathsf{NIZK}_{\mathsf{iss}}.\mathsf{Verify}(\mathsf{crs}_{\mathsf{iss}}, x := (\mathsf{pk}_I, R, S), \pi_{\mathsf{i}}) \neq 1$

$\vee\ \mathsf{EQ.Verify}\left((\mathsf{pk}_I^{(3)}, \mathsf{pk}_I^{(4)}, \mathsf{pk}_I^{(5)}), \mathbf{m}, \overline{\boldsymbol{\sigma}}\right) \neq 1$

then return $\perp$

$\mathbf{s} \leftarrow\!\!\$\ \mathsf{EQ.ChRep}\left(\overline{\boldsymbol{\sigma}}, \left(\alpha + \hat{r}\right)^{-1}\right)$

$\phi_1 := \mathsf{H}_{\mathbb{G}\to\mathbb{G}}\left(g_1^{(\alpha+\hat{r})^{-1}}\right),\ \phi_2 := g_1^{\mathsf{id}\cdot(\alpha+\hat{r})} \cdot \phi_1^{\alpha+\hat{r}}$

$\pi_{\mathsf{o}} \leftarrow\!\!\$\ \mathsf{NIZK}_{\mathsf{obt}}.\mathsf{Prove}\begin{pmatrix} \mathsf{crs}_{\mathsf{obt}},\ x := \left(g_1^{(\alpha+\hat{r})^{-1}}, \boldsymbol{\phi}, \mathsf{id}\right) \\ w := (\mathsf{sk}_C, \hat{r}) \end{pmatrix}$

$\mathbf{t} := \left(g_1^{(\alpha+\hat{r})^{-1}}, R^{\left(\alpha + \hat{r}\right)^{-1}}, S^{\left(\alpha + \hat{r}\right)^{-1}}\right),\ \boldsymbol{\sigma} := \left(\mathbf{s}, \boldsymbol{\phi}, \pi_{\mathsf{o}}, \mathsf{id}\right)$

Figure 9. Modified SPS-EQ construction for NIAT presignature issuance and token generation.

and leads to a strong incentive against double-spending[6]. A simple optimization that offers significant asymptotic advantage over the linear search on aux is to instead store the value $\mathsf{pk}_C \cdot g_1^{\mathsf{H}_{\mathbb{Z}}(\mathsf{nonce})}$ value itself, which allows the verifier running DSIden to perform efficient lookup over aux.

Double-spend verification. The double-spend verification algorithm must essentially check the identification algorithm's work. It ensures that the accused pk_C and the corresponding nonce are a valid pair in the auxiliary data, and confirms that the DS identification equation holds.

```
$\mathsf{DSVer}(\mathsf{pk}_C, \Pi, \mathsf{aux})$
if $\mathsf{Verify}(\mathsf{pk}_I, \mathbf{t}, \boldsymbol{\sigma}) \wedge \mathsf{Verify}(\mathsf{pk}_I, \mathbf{t}, \boldsymbol{\sigma}') \neq 1$ then
    return $0$
$\boldsymbol{\sigma} =: (\mathbf{s}, \boldsymbol{\phi}, \pi_{\mathsf{o}}, \mathsf{id})\ ,\ \boldsymbol{\sigma}' =: (\mathbf{s}', \boldsymbol{\phi}', \pi_{\mathsf{o}}', \mathsf{id}')$
return $\left({\phi_2}^{-1} \cdot \phi_2'\right)^{(\mathsf{id}-\mathsf{id}')^{-1}} = \mathsf{pk}_C \cdot g_1^{\mathsf{H}_{\mathbb{Z}}(\mathsf{nonce})}$
    $\wedge\ (\mathsf{pk}_C, \mathsf{nonce}) \in \mathsf{aux}$
```

Security. Below, we present our main theorems for NIAT security of Construction 5. Due to space constraints, we provide the proofs in Appendix C.

Theorem 6 (One-more unforgeability)**.** *Construction 5 is one-more unforgeable (in the random oracle model) assuming* $\mathsf{NIZK}_{\mathsf{iss}}$ *satisfies zero knowledge,* EQ *is existentially unforgeable under adaptively chosen-message attacks and* $\mathsf{NIZK}_{\mathsf{obt}}$ *is sound.*

Theorem 7 (κ-unlinkability)**.** *Construction 5 is κ-unlinkable (in the random oracle model) for $\kappa = 2$, assuming* $\mathsf{NIZK}_{\mathsf{iss}}$ *is an argument of knowledge, that DDH, k-DDH assumption holds in $\mathbb{G}_1$ and* EQ *perfectly adapts signatures under a malicious signer* $\mathsf{NIZK}_{\mathsf{obt}}$ *satisfies zero knowledge.*

Theorem 8 (Privacy of metadata bit)**.** *Construction 5 has private metadata bit (in the random oracle model) assuming* NIZK *satisfies zero-knowledge, that DDH assumption holds and* EQ *is existentially unforgeable under adaptively chosen-message attacks.*

Theorem 9 (Double-spend identification)**.** *Construction 5 satisfies double-spend identification (in the random oracle model) assuming* $\mathsf{NIZK}_{\mathsf{obt}}$ *is sound.*

Theorem 10 (Double-spend exculpability)**.** *Construction 5 satisfies double-spend exculpability assuming* $\mathsf{NIZK}_{\mathsf{obt}}$ *is sound and the discrete log assumption holds in $\mathbb{G}_1$.*

6. Implementation and Evaluation

We implemented our main NIAT Construction 4 in C++ using the `mcl` library [25] [7]. For our implementation, we chose the pairing-friendly BLS12-381 curve [26, 27]. All our experiments were done on a laptop equipped with an Apple M3 Pro chip and the reported numbers are the average of 1000 executions.

Storage and communication costs. An element in $\mathbb{G}_1$ occupies 48 bytes in its compressed representation, and similarly, an element in $\mathbb{G}_2$ occupies 96 bytes. The message from the issuer to the client consists of 3 elements in $\mathbb{G}_1$, 1 element in $\mathbb{G}_2$, 6 integers for the proof, and a nonce $r \in \{0,1\}^{\lambda}$; and a token consists of 4 elements in $\mathbb{G}_1$ and 1 element in $\mathbb{G}_2$, we followed the `minSig` approach (signatures in $\mathbb{G}_1$ and signing verification keys in $\mathbb{G}_2$) to

[6] One could potentially alter the protocol to instead reveal the sk_C, but we consider this to be an extremely punitive measure given that DSIden is a public algorithm.

[7] Source code available at: https://anonymous.4open.science/r/NIAT-94D7.

	# Moves	Public Verification	Private Metadata Bit	Issue	Obtain	Readbit	Token size $\lvert t\rvert + \lvert\sigma\rvert$
Const. 4*	1	✓	✓	$14\times$	$1\mathsf{e} + 17\times$	$2\mathsf{e} + 1\times$	$5\mathbb{G}$
[10]	3	✓	✓	$7\times$	$11\times$	$6\times$	$3\mathbb{G} + 4\mathbb{Z}$
[11]	2	✗	✓	$11\times$	$17\times$	$1\times$	$2\mathbb{G} + 1\mathbb{Z}$
[9]	2	✗	✓	$12\times$	$15\times$	$4\times$	$2\mathbb{G} + 1\mathbb{Z}$
[1]	2	✗	✗	$7\times$	$2\times$	$1\times$	$1\mathbb{G}$
[3]	2	✓	✗	$3\times$	$2\mathsf{e} + 1\times$	$2\mathsf{e}$	$1\mathbb{G} + 1\mathbb{Z}$

TABLE 1. Comparison of anonymous token schemes. $\times$ represents group exponentiation and e represents pairing operation. $\mathbb{G}$ and $\mathbb{Z}$ stand for group elements and integers respectively. $*$ indicates the verification cost is amortized.

minimize the token size. With this, we estimate that in terms of communication costs, a presignature issuance (to the client) needs around 464 bytes to be transferred over the wire, and the final token size is around 288 bytes. Therefore, in order to design a system that incorporates the offline signing protocol shown in Figure 1 where the server supports m clients, and each client will be issued n presignatures in a batch, the required total server storage would be calculated as $m \cdot (48 + 464n)$ bytes. We provide an estimation of storage needs for up to 10^8 clients in Figure 10 (left) for batch size $m = 30, 60$ and 100. We further note that in order to prevent double spending, the system must anyway keep track of all redeemed tokens, which will require an additional storage capacity comparable to our initial storage estimation.

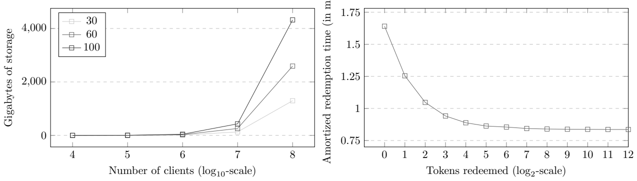

Figure 10. (left) Storage estimates for an offline signing server storing batches of 30, 60 and 100 presignatures per client; and (right) Issuer's amortized redemption cost per token.

Computational cost. Due to the non-interactive nature of our protocol, we are able to cut down the first message from a client to the issuer. Consequently, the issuer does not need to maintain an open connection with a client for token issuance. To issue a token, the issuer performs either 14 exponentiations (1 for bit embedding, 5 for signature generation and 8 for the proof computation) which can be done in an *offline* manner as part of a precomputation. After receiving the issuer's message, the client at the other end computes 4 pairing operations (plus one precomputed pairing of $\mathsf{e}(\mathsf{pk}_C, \mathsf{pk}_I^{(3)})$) to verify the SPS-EQ signature, performs 12 exponentiations to verify the proof, and 5 more to obtain the final token. Finally, at redemption, the verifier performs 4 pairing operations (plus one precomputed pairing of $\mathsf{e}(g_1, \mathsf{pk}_I^{(3)})$) to verify the SPS-EQ signature, and then extracts the embedded bit with 1 exponentiation. We summarize our results (after amortization) in Table 1 and present them as part of a comparison with related works.

Experimental benchmarks. From our experiments we found that an exponentiation in $\mathbb{G}_1$ took approximately 0.047 ms, an exponentiation in $\mathbb{G}_2$ took 0.088 ms, and a pairing operation took 0.445 ms on average. We further found that (i) the issuer took 0.84 ms to issue a token, (ii) the client verified the psig in 1.73 ms and the disjunctive NIZK proof in 0.69 ms, and finalized the token in another 0.34 ms, and (iii) during bit extraction, the issuer took 1.64 ms to verify a token and additional 0.06 ms to extract the bit. It is worth noting that since our scheme is non-interactive, there is, as such, no issuance "session" and these costs may incur asynchronously. Our results are summarized in Table 2. We remark that although the σ verification step is a required check before bit extraction during token redemption, these costs are reported separately because our verification algorithm can be done publicly and independently of the bit extraction computation.

	Operation	**Time** (in ms)
Issuer	Issue	0.84
	Read bit (σ verification)	1.64
	Read bit (extraction)	0.06
Client	Obtain ($\overline{\sigma}$ verification)	1.73
	Obtain (proof verification)	0.69
	Obtain (finalization)	0.34

TABLE 2. Benchmarks for our implementation of Construction 4.

Batched verification. We also ran benchmark tests for a batched version of signature verification as explained in Appendix B. In this batched version the client (resp. the issuer) is able to perform $n + 4$ (resp. $2n + 2$) pairings instead of $4n$, for a batch of n presignatures (resp. tokens). As shown in Table 1, we calculated the amortized cost of our token generation to be approximately equivalent to 1 pairing (and 17 exponentiations).

Batch size	1	30	60	100
Time (in ms)	1.73	0.27	0.25	0.24

TABLE 3. Client's amortized runtimes for batch verification.

In Table 3 we show that actual costs amortized over $30, 60$ and 100 presignatures[8] based on experimental evaluations averaged over 1000 runs. A similar calculation gives the amortized cost of our token redemption (verification) to be approximately equivalent to 2 pairings, (plus 1 exponentiation for bit extraction). In Figure 10 (right) we show the actual amortized costs over up to 2^{12} token redemptions.

Estimates for double-spending protection. Although we did not implement the NIAT with double-spend protection,

[8] The number of presignatures issued to a client in Privacy Pass is between 30 and 100.

we do provide the estimates of the various *additional* costs incurred by the resulting extension over the base scheme. In particular, we estimate the cost of token issuance to be about 0.15 ms over the base scheme as it requires 2 extra exponentiations (1 more for the equivalence class signature and 1 more for the proof), and the overall presignature size should require about 2 additional integers in the proof. For amortized token generation, we estimate the additional cost to be about 0.6 ms over the base scheme as it requires 8 extra exponentiations (2 more for proof verification, 1 more for the tag, 2 for the proof and 3 for the encryption), and the overall token size to be about 3 additional $\mathbb{G}_1$ (1 more in the tag and 2 for the ciphertext) elements and 4 additional integers (3 for the client's proof and 1 for the verifier's identifier). For amortized redemption, we estimate the additional cost to be about 0.4 ms over the base scheme as it requires about 1 additional pairing and 4 exponentiations for the proof verification. Aside from the additional cost of verification, the cost of bit extraction stays the same. Finally, double-spend identification requires 5 pairings and the cost of a lookup over aux.

References

[1] A. Davidson, I. Goldberg, N. Sullivan, G. Tankersley, and F. Valsorda, "Privacy pass: Bypassing internet challenges anonymously," *Proceedings on Privacy Enhancing Technologies*, vol. 2018, no. 3, pp. 164–180, Jul. 2018.

[2] Apple, "icloud private relay overview," https://www.apple.com/icloud/docs/iCloud_Private_Relay_Overview_Dec2021.pdf, 2021, accessed: 2024-04-29.

[3] T. Silde and M. Strand, "Anonymous tokens with public metadata and applications to private contact tracing," in *FC 2022: 26th International Conference on Financial Cryptography and Data Security*, ser. Lecture Notes in Computer Science, I. Eyal and J. A. Garay, Eds., vol. 13411. Grenada: Springer, Cham, Switzerland, May 2–6, 2022, pp. 179–199.

[4] S. Dutton, "Getting started with trust tokens," https://web.dev/articles/trust-tokens, 2020, accessed: 2024-04-29.

[5] J. Wilander, "Introducing private click measurement, pcm," https://webkit.org/blog/11529/introducing-private-click-measurement-pcm/, 2021, accessed: 2024-04-29.

[6] Google, "Private state tokens," https://developers.google.com/privacy-sandbox/protections/private-state-tokens, 2024, accessed: 2024-04-29.

[7] T. Meunier, C. D. Rubin, and A. Faz-Hernández, "Privacy pass: upgrading to the latest protocol version," https://blog.cloudflare.com/privacy-pass-standard, 2024, accessed: 2024-04-29.

[8] IETF, "Privacy pass (privacypass)," https://datatracker.ietf.org/wg/privacypass/about/, 2024, accessed: 2024-04-29.

[9] B. Kreuter, T. Lepoint, M. Orrù, and M. Raykova, "Anonymous tokens with private metadata bit," in *Advances in Cryptology – CRYPTO 2020, Part I*, ser. Lecture Notes in Computer Science, D. Micciancio and T. Ristenpart, Eds., vol. 12170. Santa Barbara, CA, USA: Springer, Cham, Switzerland, Aug. 17–21, 2020, pp. 308–336.

[10] F. Benhamouda, T. Lepoint, M. Orrù, and M. Raykova, "Publicly verifiable anonymous tokens with private metadata bit," Cryptology ePrint Archive, Report 2022/004, 2022. [Online]. Available: https://eprint.iacr.org/2022/004

[11] M. Chase, F. B. Durak, and S. Vaudenay, "Anonymous tokens with stronger metadata bit hiding from algebraic MACs," in *Advances in Cryptology – CRYPTO 2023, Part II*, ser. Lecture Notes in Computer Science, H. Handschuh and A. Lysyanskaya, Eds., vol. 14082. Santa Barbara, CA, USA: Springer, Cham, Switzerland, Aug. 20–24, 2023, pp. 418–449.

[12] M. Orrù, S. Tessaro, G. Zaverucha, and C. Zhu, "Oblivious issuance of proofs," in *Advances in Cryptology – CRYPTO 2024, Part IX*, ser. Lecture Notes in Computer Science, L. Reyzin and D. Stebila, Eds., vol. 14928. Santa Barbara, CA, USA: Springer, Cham, Switzerland, Aug. 18–22, 2024, pp. 254–287.

[13] D. Chaum, "Blind signature system," in *Advances in Cryptology – CRYPTO'83*, D. Chaum, Ed. Santa Barbara, CA, USA: Plenum Press, New York, USA, 1983, p. 153.

[14] D. Pointcheval and J. Stern, "Provably secure blind signature schemes," in *Advances in Cryptology – ASIACRYPT'96*, ser. Lecture Notes in Computer Science, K. Kim and T. Matsumoto, Eds., vol. 1163. Kyongju, Korea: Springer Berlin Heidelberg, Germany, Nov. 3–7, 1996, pp. 252–265.

[15] ——, "Security arguments for digital signatures and blind signatures," *Journal of Cryptology*, vol. 13, no. 3, pp. 361–396, Jun. 2000.

[16] L. Hanzlik, "Non-interactive blind signatures for random messages," in *Advances in Cryptology – EUROCRYPT 2023, Part V*, ser. Lecture Notes in Computer Science, C. Hazay and M. Stam, Eds., vol. 14008. Lyon, France: Springer, Cham, Switzerland, Apr. 23–27, 2023, pp. 722–752.

[17] F. Baldimtsi, J. Cheng, R. Goyal, and A. Yadav, “Non-interactive blind signatures: Post-quantum and stronger security,” Cryptology ePrint Archive, Report 2024/614, 2024. [Online]. Available: https://eprint.iacr.org/2024/614

[18] G. Fuchsbauer, C. Hanser, and D. Slamanig, “Structure-preserving signatures on equivalence classes and constant-size anonymous credentials,” *Journal of Cryptology*, vol. 32, no. 2, pp. 498–546, Apr. 2019.

[19] N. Tyagi, S. Celi, T. Ristenpart, N. Sullivan, S. Tessaro, and C. A. Wood, “A fast and simple partially oblivious prf, with applications,” in *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part II*, ser. Lecture Notes in Computer Science, O. Dunkelman and S. Dziembowski, Eds., vol. 13276. Springer, 2022, pp. 674–705. [Online]. Available: https://doi.org/10.1007/978-3-031-07085-3_23

[20] G. Amjad, K. Yeo, and M. Yung, “Rsa blind signatures with public metadata,” Cryptology ePrint Archive, Paper 2023/1199, 2023, https://eprint.iacr.org/2023/1199. [Online]. Available: https://eprint.iacr.org/2023/1199

[21] F. Benhamouda, M. Raykova, and K. Seth, “Anonymous counting tokens,” in *Advances in Cryptology – ASIACRYPT 2023, Part II*, ser. Lecture Notes in Computer Science, J. Guo and R. Steinfeld, Eds., vol. 14439. Guangzhou, China: Springer, Singapore, Singapore, Dec. 4–8, 2023, pp. 245–278.

[22] S. D. Galbraith, K. G. Paterson, and N. P. Smart, “Pairings for cryptographers,” Cryptology ePrint Archive, Report 2006/165, 2006. [Online]. Available: https://eprint.iacr.org/2006/165

[23] I. Damgård, “On $\sum$-protocols,” *Lecture Notes, University of Aarhus, Department for Computer Science*, vol. 84, 2010. [Online]. Available: https://www.cs.au.dk/~ivan/Sigma.pdf

[24] A. Fiat and A. Shamir, “How to prove yourself: Practical solutions to identification and signature problems,” in *Advances in Cryptology – CRYPTO’86*, ser. Lecture Notes in Computer Science, A. M. Odlyzko, Ed., vol. 263. Santa Barbara, CA, USA: Springer Berlin Heidelberg, Germany, Aug. 1987, pp. 186–194.

[25] S. Mitsunari, “MCL,” https://github.com/herumi/mcl, 2024.

[26] P. S. L. M. Barreto, B. Lynn, and M. Scott, “Constructing elliptic curves with prescribed embedding degrees,” in *SCN 02: 3rd International Conference on Security in Communication Networks*, ser. Lecture Notes in Computer Science, S. Cimato, C. Galdi, and G. Persiano, Eds., vol. 2576. Amalfi, Italy: Springer Berlin Heidelberg, Germany, Sep. 12–13, 2003, pp. 257–267.

[27] S. Bowe, “Bls12-381: New zk-snark elliptic curve construction,” https://electriccoin.co/blog/new-snark-curve/, 2017, accessed: 2024-04-17.

[28] Y. Dodis and A. Yampolskiy, “A verifiable random function with short proofs and keys,” in *PKC 2005: 8th International Workshop on Theory and Practice in Public Key Cryptography*, ser. Lecture Notes in Computer Science, S. Vaudenay, Ed., vol. 3386. Les Diablerets, Switzerland: Springer Berlin Heidelberg, Germany, Jan. 23–26, 2005, pp. 416–431.

[29] D. Boneh, H. W. Montgomery, and A. Raghunathan, “Algebraic pseudorandom functions with improved efficiency from the augmented cascade,” in *ACM CCS 2010: 17th Conference on Computer and Communications Security*, E. Al-Shaer, A. D. Keromytis, and V. Shmatikov, Eds. Chicago, Illinois, USA: ACM Press, Oct. 4–8, 2010, pp. 131–140.

Appendix

1. Security Proofs for Section 4

Theorem 3 (One-more unforgeability). Construction 4 is one-more unforgeable (in the random oracle model) assuming NIZK satisfies zero knowledge and EQ is existentially unforgeable under adaptively chosen-message attacks.

Pf. We proceed through a series of hybrids.

- Hybrid $\mathbf{H}_0$: This is the original NIAT one-more unforgeability experiment $\mathsf{OM\text{-}Unf}_{\mathcal{A}}$ (see Definition 11) defined with respect to Construction 4.
- Hybrid $\mathbf{H}_1$: This is the same as $\mathbf{H}_0$ except, when there is a collision on an adversaries random oracle query to H, the challenger aborts. For a PPT adversary making $q_{\mathsf{H}} = \mathsf{poly}(\lambda)$ random oracle queries, it can be shown by a simple application of the union bound that the adversary’s advantage in $\mathbf{H}_0$ and that in $\mathbf{H}_1$ differs by at most $q_{\mathsf{H}}/2^{\lambda}$.
- Hybrid $\mathbf{H}_2$: This is the same as $\mathbf{H}_1$ except, the challenger simulates the proof π without a witness. If NIZK satisfies zero knowledge, then $\mathbf{H}_1$ and $\mathbf{H}_2$ are indistinguishable to the adversary.

Let $\mathsf{Adv}^j_{\mathcal{A}}$ be the advantage of a PPT adversary $\mathcal{A}$ in hybrid $\mathbf{H}_j$. Then, we claim that $\mathsf{Adv}^2_{\mathcal{A}}$ must be negligible. In order to do so, we give a reduction $\mathcal{B}$ such that if $\mathcal{A}$ wins the game in $\mathbf{H}_2$, $\mathcal{B}$ can break the existentially unforgeability (under adaptively chosen-message attack) of the underlying SPS-EQ scheme EQ. In particular, the reduction does the following:

- It sets $(\mathsf{pk}_I^{(3)}, \mathsf{pk}_I^{(4)}) := \mathsf{pk}_{\mathsf{EQ}}$ given by the $\mathsf{EUnf\text{-}CMA}$ challenger $\mathcal{C}$. It additionally samples x_1 and x_2 from $\mathbb{Z}_p$ and sets the rest of pk_I in the usual way.
- On receiving a Issue query (pk_C, b), it inserts b into Q, samples r, and computes S in the usual way. It then queries $\mathcal{C}$ for the signature $\overline{\boldsymbol{\sigma}}$ on $(\mathsf{pk}_C, \mathsf{H}(r)\cdot S)$ and returns $\mathsf{psig} := (\overline{\boldsymbol{\sigma}}, S, \pi)$ and $\mathsf{nonce} := r$, where π is the simulated NIZK proof.
- On receiving a Read query $(\mathbf{t}, \boldsymbol{\sigma})$ it first checks if $\mathsf{Verify}(\mathsf{pk}_I, \mathbf{t}, \boldsymbol{\sigma}) = 1$ and returns $\perp$ if it is not. Otherwise for each $b \in \{0,1\}$ it checks whether $t_1^{x_{1+b}} = t_2$. If such a b is found, it outputs it. Otherwise it outputs $\perp$.
- Finally when $\mathcal{A}$ outputs its forgery $\{(\mathbf{t}^{(i)}, \boldsymbol{\sigma}^{(i)})\}_{i\in[N]}$, $\mathcal{B}$ uniformly chooses a pair $(\mathbf{t}^*, \boldsymbol{\sigma}^*)$ and outputs it as its forgery.

For a successful adversary $\mathcal{A}$, each $\boldsymbol{\sigma}^{(i)}$ must be a signature on a unique class represented by $(g_1 \cdot t_1^{(i)}, t_2^{(i)})$. However, since $\mathcal{B}$ can not determine which of the N forgeries is with respect to a fresh equivalence class, it uniformly chooses one in the last step. Therefore, $\mathsf{Adv}^2_{\mathcal{A}} = \frac{\ell}{N-\ell} \cdot \mathsf{Adv}^{\mathsf{EUnf\text{-}CMA}}_{\mathcal{B}}$, which is negligible for any PPT adversary $\mathcal{A}$. □

Theorem 4 (κ-unlinkability). Construction 4 is κ-unlinkable (in the random oracle model) for $\kappa = 2$, assuming NIZK is an argument of knowledge, that inverse DDH assumption holds in $\mathbb{G}_1$ and EQ perfectly adapts signatures under a malicious signer.

Pf. We proceed through a series of hybrids.

- Hybrid $\mathbf{H}_0$: This is the original NIAT unlinkability experiment $\mathsf{UNLINK}_{\mathcal{A},n}$ (see Definition 12) defined with respect to Construction 4.

- Hybrid $\mathbf{H}_1$: This is the same as $\mathbf{H}_0$ except, we program the random oracle such that for any *fresh* query $r \in \{0,1\}^\lambda$, we set $H(r) = g_1^{\nu_r}$ for $\nu_r \leftarrow\!\!\$\ \mathbb{Z}_p$. We must also keep track of each (r, ν_r) pair and answer any repeated random oracle queries accordingly. This clearly affects no change from the adversary's point of view, so $\mathbf{H}_0$ and $\mathbf{H}_1$ are indistinguishable.
- Hybrid $\mathbf{H}_2$: This is the same as $\mathbf{H}_1$ except, on every Obtain query $(\mathsf{pk}_C, \mathsf{psig} := (\overline{\boldsymbol{\sigma}}, S, \pi), \mathsf{nonce} := r)$, the challenger runs the NIZKAoK extractor on proof π to extract $\mathsf{sk}_I = (x_1, x_2, y_1, y_2)$. Then if $\mathsf{EQ.Verify}\left((\mathsf{pk}_I^{(3)}, \mathsf{pk}_I^{(4)}), (\mathsf{pk}_C, \mathsf{H}(r) \cdot S)\right) \neq 1$ it sets $\mathsf{out} := (\perp, \perp)$. Otherwise, it sets $\mathbf{t}$ appropriately, and computes $\boldsymbol{\sigma} \leftarrow\!\!\$\ \mathsf{EQ.Sign}\left((\mathsf{sk}_I^{(3)}, \mathsf{sk}_I^{(4)}, (g_1, t_1 \cdot t_2)\right)$. Since EQ perfectly adapts signatures under malicious keys, hybrids $\mathbf{H}_1$ and $\mathbf{H}_2$ are indistinguishable to the adversary.
- Hybrid $\mathbf{H}_3$: This is the same as $\mathbf{H}_2$ except, the challenger parses each $(\mathsf{pk}_C^{(i)}, \mathsf{psig}^{(i)}, \mathsf{nonce}^{(i)})$ triple in Q^* such that for each $i \in [n]$, $\mathsf{psig}^{(i)} = (\overline{\boldsymbol{\sigma}}^{(i)}, S^{(i)}, \pi^{(i)})$. It then runs the NIZKAoK extractor on any proof $\pi^{(i)}$ to extract $\mathsf{sk}_I = (x_1, x_2, x_3, y_1, y_2)$. Then for any i, if $\mathsf{EQ.Verify}\left((\mathsf{pk}_I^{(3)}, \mathsf{pk}_I^{(4)}), (\mathsf{pk}_C^{(i)}, \mathsf{H}(r^{(i)}) \cdot S^{(i)})\right) \neq 1$ it sets $\mathsf{out}_i := (\perp, \perp)$ for all i; otherwise, it sets each $\mathbf{t}^{(i)}$ appropriately, and computes $\boldsymbol{\sigma}^{(i)} \leftarrow\!\!\$\ \mathsf{EQ.Sign}\left((\mathsf{sk}_I^{(3)}, \mathsf{sk}_I^{(4)}), \left(g_1, t_1^{(i)} \cdot t_2^{(i)}\right)\right)$. Since EQ perfectly adapts signatures under malicious keys, hybrids $\mathbf{H}_2$ and $\mathbf{H}_3$ are indistinguishable to the adversary.
- Hybrid $\mathbf{H}_4$: This is the same as $\mathbf{H}_3$ except, on every Obtain query $(\mathsf{pk}_C, \mathsf{psig} := (\overline{\boldsymbol{\sigma}}, S, \pi), \mathsf{nonce} := r)$, the challenger runs the NIZKAoK extractor on proof π to extract $\mathsf{sk}_I = (x_1, x_2, y_1, y_2)$ and b. Then if $\mathsf{EQ.Verify}\left((\mathsf{pk}_I^{(3)}, \mathsf{pk}_I^{(4)}), (\mathsf{pk}_C, \mathsf{H}(r) \cdot S)\right) \neq 1$ it sets $\mathsf{out} := (\perp, \perp)$. Otherwise, it looks up sk_C corresponding to pk_C and sets $t_1 := \mathsf{H}(r)^{\mathsf{sk}_C^{-1}}$ as usual, and $t_2 := t_1^{(1-b)x_1 + bx_2}$ using the extracted values. Clearly, this affects no change from the adversary's perspective. Thus hybrids $\mathbf{H}_3$ and $\mathbf{H}_4$ are indistinguishable to the adversary.
- Hybrid $\mathbf{H}_5$: This is the same as $\mathbf{H}_4$ except, the challenger parses each $(\mathsf{pk}_C^{(i)}, \mathsf{psig}^{(i)}, \mathsf{nonce}^{(i)})$ triple in Q^* such that for each $i \in [n]$, $\mathsf{psig}^{(i)} = (\overline{\boldsymbol{\sigma}}^{(i)}, S^{(i)}, \pi^{(i)})$. It then runs the NIZKAoK extractor on any proof $\pi^{(i)}$ to extract $\mathsf{sk}_I = (x_1, x_2, x_3, y_1, y_2)$ and b. Then for any i, if $\mathsf{EQ.Verify}\left((\mathsf{pk}_I^{(3)}, \mathsf{pk}_I^{(4)}), (pk_C^{(i)}, \mathsf{H}(r^{(i)}) \cdot S^{(i)})\right) \neq 1$ it sets $\mathsf{out}_i := (\perp, \perp)$ for all i; otherwise, it looks up $\mathsf{sk}_C^{(i)}$ corresponding to $\mathsf{pk}_C^{(i)}$ and sets and sets $t_1^{(i)} := \mathsf{H}(r^{(i)})^{1/\mathsf{sk}_C^{(i)}}$ as usual, and $t_2^{(i)} := \left(t_1^{(i)}\right)^{(1-b)x_1 + bx_2}$ using the extracted values. Clearly, this affects no change from the adversary's perspective. Thus hybrids $\mathbf{H}_4$ and $\mathbf{H}_5$ are indistinguishable to the adversary.
- Hybrid $\mathbf{H}_6$: This is the same as $\mathbf{H}_5$ except, on every Obtain query $(\mathsf{pk}_C, \mathsf{psig} := (\overline{\boldsymbol{\sigma}}, S, \pi), \mathsf{nonce} := r)$, instead of setting t_1 as $\mathsf{H}(r)^{sk_C^{-1}}$, it sets it to g_1^ρ for some $\rho \leftarrow\!\!\$\ \mathbb{Z}_p$ of its choice. The signature is now computed with respect to this new t_1. We argue that $\mathbf{H}_6$ is indistinguishable from $\mathbf{H}_5$ if the inverse DDH assumption holds in $\mathbb{G}_1$. In particular, for each user in Q_{usr}, the reduction algorithm $\mathcal{B}$ instantiates a new inverse DDH challenger over $\mathbb{G}_1$ and receives the corresponding challenge (g_1^α, g_1^β). It sets $\mathsf{pk}_C := g_1^\alpha$ and $\mathsf{sk}_C := \perp$. On any valid Obtain query, it looks up g_1^β value corresponding to the pk_C, and ν_r corresponding the nonce r. It then sets $t_1 := (g_1^\beta)^{\nu_r}$. Now observe that if $\beta = \alpha^{-1}$, the reduction simulates $\mathbf{H}_5$ to $\mathcal{A}$ and otherwise simulates $\mathbf{H}_6$. Thus the adversary's advantage in $\mathbf{H}_5$ and that in $\mathbf{H}_6$ differs by at most $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{invDDH}}$.
- Hybrid $\mathbf{H}_7$: This is the same as $\mathbf{H}_6$ except, after parsing each $(\mathsf{pk}_C^{(i)}, \mathsf{psig}^{(i)}, \mathsf{nonce}^{(i)})$ triple in Q^* such that for each $i \in [n]$, $\mathsf{psig}^{(i)} = (\overline{\boldsymbol{\sigma}}^{(i)}, S^{(i)}, \pi^{(i)})$, instead of setting $t_1^{(i)}$ as $\mathsf{H}(r^{(i)})^{sk_C^{(i)-1}}$ for each $i \in [n]$, the challenger sets it to $g_1^{\rho^{(i)}}$ for $\rho^{(i)} \leftarrow\!\!\$\ \mathbb{Z}_p$ of its choice. Each signature is now computed with respect to this new $t_1^{(i)}$. Indistinguishability between hybrids $\mathbf{H}_6$ and $\mathbf{H}_7$ follow similarly to the previous argument.

Finally, observe that the final $(\mathbf{t}^{(i)}, \boldsymbol{\sigma}^{(i)})$ pairs are all independent of their presignatures and nonces. So the best adversarial strategy is to create some n_b responses with bit b for each bit $(n_0 + n_1 = n)$, read the bit $b_{\hat{i}}$ of $\mathsf{out}_{\hat{i}}$ and output a random value i^* from the set of all $n_{b_{\hat{i}}}$ indices where the embedded bit was equal to $b_{\hat{i}}$. The probability that the adversary wins is

$$\sum_{b \in \{0,1\}} \Pr\left[b_{i^*} = b_{\hat{i}}\right] \cdot \Pr\left[i^* = \hat{i}\right] = \sum_{b \in \{0,1\}} \frac{n_b}{n} \cdot \frac{1}{n_b} = \frac{2}{n} \quad .$$

Let $\mathsf{Adv}_{\mathcal{A}}^j$ be the advantage of a PPT adversary $\mathcal{A}$ in hybrid $\mathbf{H}_j$. Then, $\mathsf{Adv}_{\mathcal{A}}^7 \leq \frac{2}{n}$.

□

Theorem 5 (Privacy of metadata bit). Construction 4 has private metadata bit (in the random oracle model) assuming NIZK satisfies zero-knowledge, that DDH assumption holds and EQ is existentially unforgeable under adaptively chosen-message attacks.

Pf. We proceed through a series of hybrids.

- Hybrid $\mathbf{H}_0$: This is the original NIAT metadata bit privacy experiment $\mathsf{PMB}_{\mathcal{A},\hat{b}}$ for $\hat{b} \in \{0,1\}$ (see Definition 13) defined with respect to Construction 4.
- Hybrid $\mathbf{H}_1$: This is the same as $\mathbf{H}_0$ except, we program the random oracle such that for any *fresh* query $r \in \{0,1\}^\lambda$, we set $H(r) = g_1^{\nu_r}$ for $\nu_r \leftarrow\!\!\$\ \mathbb{Z}_p$. We must also keep track of each (r, ν_r) pair and answer any repeated random oracle queries accordingly. This clearly affects no change from the adversary's point of view, so $\mathbf{H}_0$ and $\mathbf{H}_1$ are indistinguishable.
- Hybrid $\mathbf{H}_2$: This is the same as $\mathbf{H}_1$ except, the challenger simulates the proof π without a witness. If

NIZK satisfies zero knowledge, then $\mathbf{H}_1$ and $\mathbf{H}_2$ are indistinguishable to the adversary.

- Hybrid $\mathbf{H}_3$: This is the same as $\mathbf{H}_2$ except, on receiving the challenge query, the challenger runs the issue algorithm with $\rho \leftarrow\$ \mathbb{Z}_p$ instead of $\hat{b}$. In particular, it computes $S := g_1^{\rho}$. We argue that $\mathbf{H}_3$ is indistinguishable from $\mathbf{H}_2$ if the DDH assumption holds in $\mathbb{G}_1$. In particular, the reduction algorithm $\mathcal{B}$ instantiates a DDH challenger over $\mathbb{G}_1$ and receives the corresponding challenge $(g_1^{\alpha}, g_1^{\beta}, g_1^{\gamma})$. If $\hat{b} = 0$, it sets $\mathsf{pk}_I^{(1)} := g_1^{\alpha}$ and $\mathsf{sk}_I^{(1)} := \perp$ otherwise if $\hat{b} = 1$ it sets $\mathsf{pk}_I^{(2)} := g_1^{\alpha}$ and $\mathsf{sk}_I^{(2)} := \perp$. On any valid Issue query (pk_C, b), it answers as before if $b \neq \hat{b}$. Otherwise, $\mathcal{B}$ sets R as before, sets $S := (g_1^{\alpha})^{\nu_r}$ and computes the rest of the presignature using this S. On receiving a valid Read query, the reduction first verifies the signature under pk_I, and outputs $\perp$ if it fails. Otherwise, it checks if $t_1^{\mathsf{sk}_I^{(2-\hat{b})}} = t_2$ (note that $\mathcal{B}$ knows $\mathsf{sk}_I^{(2-\hat{b})}$) and outputs $1 - \hat{b}$ if the check passes, and $\hat{b}$ otherwise. Finally, on receiving the Challenge query for pk_C, $\mathcal{B}$ samples the nonce r as usual, and then programs $\mathsf{H}(r) := g_1^{\beta}$, sets $S := g_1^{\gamma}$ and performs rest of the computation as before. Now, notice that if an adversary is able to distinguish the two hybrids, then either $\mathcal{B}$ breaks DDH or $\mathcal{A}$ managed to forge a SPS-EQ signature $\boldsymbol{\sigma}$ on some tag $\mathbf{t}$ such that $\mathcal{B}$ answers the Read query incorrectly on $(\mathbf{t}, \boldsymbol{\sigma})$. This happens if the underlying "bit" is invalid, but $\mathcal{B}$ answered with $\hat{b}$. It follows that the adversary's advantage in distinguishing hybrids $\mathbf{H}_2$ and $\mathbf{H}_3$ is equal to $\mathsf{Adv}_{\mathcal{B}}^{\mathsf{DDH}} + \mathsf{Adv}_{\mathcal{B}}^{\mathsf{EUnf\text{-}CMA}}$.

Let $\mathsf{Adv}_{\mathcal{A},\hat{b}}^{j}$ be the advantage of a PPT adversary $\mathcal{A}$ in hybrid $\mathbf{H}_j$ with respect to $\hat{b}$. Then,

$$\left|\Pr\left[\mathbf{H}_3^{\mathcal{A},1}(\lambda) = \mathbf{accept}\right] - \Pr\left[\mathbf{H}_3^{\mathcal{A},0}(\lambda) = \mathbf{accept}\right]\right|$$
$$= \left|\mathsf{Adv}_{\mathcal{A},1}^{3} - \mathsf{Adv}_{\mathcal{A},0}^{3}\right|$$

which is zero. This proves the theorem. □

2. Full Instantiation of Construction 4

In this section, we describe the expanded version of our NIAT protocol along with the full zero knowledge proof for the language $\mathscr{L}_{\mathsf{iss}}$. In particular, we instantiate our NIAT protocol with the SPS-EQ scheme from [18].

2.1. The Concrete SPS-EQ Component. In Figure 11, we expand the token issuance and generation algorithms by instantiating with the aforementioned SPS-EQ of [18].

Batch verification. Let us now explain the joint verification check that reduces the overall number of pairing computations required for batch verifications. Firstly notice that for both, the client and the verifier, the pairing computation for $\mathsf{e}(\mathsf{pk}_C, \mathsf{pk}_I^{(3)})$ and $\mathsf{e}(g_1, \mathsf{pk}_I^{(3)})$ respectively, can be precomputed. This already reduces the number of pairing computations per presignature/token by one. For an issuing authority with key pair $(\mathsf{sk}_I, \mathsf{pk}_I)$, let n be the number of presignatures issued to some client with public key pk_C, and n' be the number of tokens redeemed. Then the batch verification check for the client is given by

$$\prod_{i=1}^{n} \mathsf{e}\left(Z_i, Y_{2,i}\right) \overset{?}{=} \mathsf{e}\left(\mathsf{pk}_C, \mathsf{pk}_I^{(3)}\right)^n \cdot \mathsf{e}\left(\prod_{i=1}^{n}(R_i \cdot S_i), \mathsf{pk}_I^{(4)}\right) \tag{1}$$

$$\wedge\ \mathsf{e}\left(\prod_{i=1}^{n} Y_{1,i}, g_2\right) \overset{?}{=} \mathsf{e}\left(g_1, \prod_{i=1}^{n} Y_{2,i}\right) \tag{2}$$

Similarly, the batch verification check for the verifier is given by replacing n by n' in (2)[9] and linearly computing the other pairing check. So that instead of $4n$ (resp. $4n'$) pairings, the client (resp. the verifier) performs $n + 4$ (resp. $2n' + 2$) pairings.

2.2. The Disjunctive ZK Proof. Recall that the issuer's ZK proof consists of a proof of knowledge of discrete log of the elements of pk_I corresponding to the secret key, along with a disjunctive OR-proof that $S = R^{(1-b)x_1 + bx_2}$ for x_1, x_2 in the issuer's secret key. As previously mentioned, we can apply the Fiat-Shamir heuristic [24] to transform the proof into a NIZK for our construction.

- **Prove.** The NIZK prover algorithm takes as input a common reference string crs, the statement $x = (S, U_0, U_1, V, W)$, where $U_0 = \mathsf{pk}_I^{(1)} = g_1^{x_1}, U_1 = \mathsf{pk}_I^{(2)} = g_1^{x_2}, V = \mathsf{pk}_I^{(3)} = g_2^{y_1}$ and $W = \mathsf{pk}_I^{(4)} = g_2^{y_2}$ and the witness $w = (\mathsf{sk}_I := (x_1, x_2, y_1, y_2), b)$. It does the following depending on the bit b:
  1) If $b = 0$, the issuer (prover) samples integers z_0, z_v, z_w, c_1, a_1, and computes commitments $\tilde{S}_0 := R^{z_0}, \tilde{S}_1 := R^{a_1} \cdot S^{-c_1}, \tilde{U}_0 := g_1^{z_0}, \tilde{U}_1 := g_1^{a_1} \cdot U_1^{-c_1}, \tilde{V} := g_2^{z_v}, \tilde{W} := g_2^{z_w}$. It then computes the challenge $c := \mathsf{H}(g_1, g_2, \mathsf{pk}_C, S, U_0, U_1, V, W, \tilde{S}_0, \tilde{S}_1, \tilde{U}_0, \tilde{U}_1, \tilde{V}, \tilde{W})$, followed by $a_v := z_v + cy_1, a_w := z_w + cy_2, c_0 := c - c_1, a_0 := z_0 + c_0 x_1$.
  2) If $b = 1$, the issuer (prover) samples integers z_1, z_v, z_w, c_0, a_0, and computes commitments $\tilde{S}_0 := R^{a_0} \cdot S^{-c_0}, \tilde{S}_1 := R^{z_1}, \tilde{U}_0 := g_1^{a_0} \cdot U_0^{-c_0}, \tilde{U}_1 := g_1^{z_1}, \tilde{V} := g_2^{z_v}, \tilde{W} := g_2^{z_w}$. It then computes the challenge $c := \mathsf{H}(g_1, g_2, \mathsf{pk}_C, S, U_0, U_1, V, W, \tilde{S}_0, \tilde{S}_1, \tilde{U}_0, \tilde{U}_1, \tilde{V}, \tilde{W})$, followed by $a_v := z_v + cy_1, a_w := z_w + cy_2, c_1 := c - c_0, a_1 := z_1 + c_1 x_2$.

  Finally, it outputs the proof $\pi = (c_0, c_1, a_v, a_w, a_0, a_1)$.
- **Verify.** The NIZK verification algorithm takes the crs, the statement $x = (S, U_0, U_1, V, W)$ and the proof π. We use a small modification to the Fiat-Shamir transformation similar to [11] for efficient verification. Specifically, the client (verifier) parses $\pi =: (c_0, c_1, a_v, a_w, a_0, a_1)$ and computes $c = c_0 + c_1, \tilde{S}_0 := R^{a_0} \cdot S^{-c_0}, \tilde{S}_1 := R^{a_1} \cdot S^{-c_1}, \tilde{U}_0 := g_1^{a_0} \cdot U_0^{-c_0}, \tilde{U}_1 := g_1^{a_1} \cdot U_1^{-c_1}, \tilde{V} := g_2^{a_v} \cdot V^{-c}, \tilde{W} := g_2^{a_w} \cdot W^{-c}$. Finally,

[9] At verification, we are cautious to not batch first part of the pairing check similar to (1) as the client does not produce any ZK proof for its computations.

```
Issuer$^{(b)}$                                                          Client
$\underbrace{\mathbf{x} = (x_1, x_2, y_1, y_2) \in (\mathbb{Z}_p^*)^4}_{\mathsf{sk}_I}$ ,      $\underbrace{\alpha \in \mathbb{Z}_p^*}_{\mathsf{sk}_C}$ , $\underbrace{g_1^\alpha \in \mathbb{G}_1}_{\mathsf{pk}_C}$
$\underbrace{g_1^{\mathbf{x}} = (g_1^{x_1}, g_1^{x_2}) \in \mathbb{G}_1^2,\ g_2^{\mathbf{y}} = (g_2^{y_1}, g_2^{y_2}) \in \mathbb{G}_2^2}_{\mathsf{pk}_I}$
........................................ Issue ........................................
$r \leftarrow\!\$\ \{0,1\}^\lambda,\ R \coloneqq \mathsf{H}(r),\ S \coloneqq R^{(1-b)x_1+bx_2}$
  EQ.Sign $((y_1, y_2), (\mathsf{pk}_C, R \cdot S))$
  ----------------------------------------
  $\upsilon \leftarrow\!\$\ \mathbb{Z}_p^*,\ Y_1 \coloneqq g_1^{\frac{1}{\upsilon}},\ Y_2 \coloneqq g_2^{\frac{1}{\upsilon}}$
  $Z \coloneqq \left(\mathsf{pk}_C^{y_1} \cdot (R \cdot S)^{y_2}\right)^\upsilon$
  $\overline{\boldsymbol{\sigma}} \coloneqq (Z, Y_1, Y_2)$
$\pi \leftarrow\!\$$ NIZK.Prove$\left(\begin{array}{c}\mathsf{crs},\ x \coloneqq (\mathsf{pk}_I, R, S)\\ w \coloneqq (\mathsf{sk}_I, b)\end{array}\right)$   $\xrightarrow{\mathsf{psig} \coloneqq (\overline{\boldsymbol{\sigma}}, S, \pi), \mathsf{nonce} \coloneqq r}$
........................................ Obtain ........................................
                                                          $R \coloneqq \mathsf{H}(r),\ \mathbf{m} \coloneqq (\mathsf{pk}_C, R \cdot S)$
                                                          if NIZK.Verify$(\mathsf{crs}, x \coloneqq (\mathsf{pk}_I, R, S), \pi) \neq 1$
                                                          $\vee$ EQ.Verify $\left((\mathsf{pk}_I^{(3)}, \mathsf{pk}_I^{(4)}), \mathbf{m}, \overline{\boldsymbol{\sigma}}\right) \neq 1$
                                                              then return $\bot$
                                                            EQ.Verify $\left((\mathsf{pk}_I^{(3)}, \mathsf{pk}_I^{(4)}), \mathbf{m}, \overline{\boldsymbol{\sigma}}\right)$
                                                            ----------------------------------------
                                                            $\mathsf{e}(Z, Y_2) \overset{?}{=} \mathsf{e}(\mathsf{pk}_C, \mathsf{pk}_I^{(3)}) \cdot \mathsf{e}(R \cdot S, \mathsf{pk}_I^{(4)})$
                                                            $\wedge\ \mathsf{e}(Y_1, g_2) \overset{?}{=} \mathsf{e}(g_1, Y_2)$
                                                          $\mathbf{t} \coloneqq \left(R^{\alpha^{-1}}, S^{\alpha^{-1}}\right)$
                                                          $\boldsymbol{\sigma} \leftarrow\!\$$ EQ.ChRep$(\overline{\boldsymbol{\sigma}}, \alpha^{-1})$
                                                            EQ.ChRep$(\overline{\boldsymbol{\sigma}}, \alpha^{-1})$
                                                            ----------------------------------------
                                                            $\psi \leftarrow \mathbb{Z}_p^*,\ \boldsymbol{\sigma} \coloneqq \left(Z^{\psi\alpha^{-1}}, Y_1^{\frac{1}{\psi}}, Y_2^{\frac{1}{\psi}}\right)$
                              $\xleftarrow{\mathbf{t},\ \boldsymbol{\sigma}}$
........................................ ReadBit ........................................
$\mathbf{m} \coloneqq (g_1, t_1 \cdot t_2)$
if EQ.Verify $\left((\mathsf{pk}_I^{(3)}, \mathsf{pk}_I^{(4)}), \mathbf{m}, \overline{\boldsymbol{\sigma}}\right) \neq 1$
    then return $\bot$
  EQ.Verify $\left((\mathsf{pk}_I^{(3)}, \mathsf{pk}_I^{(4)}), \mathbf{m}, \overline{\boldsymbol{\sigma}}\right)$
  ----------------------------------------
  $\mathsf{e}(Z, Y_2) \overset{?}{=} \mathsf{e}(m_1, \mathsf{pk}_I^{(3)}) \cdot \mathsf{e}(m_2, \mathsf{pk}_I^{(4)})$
  $\wedge\ \mathsf{e}(Y_1, g_2) \overset{?}{=} \mathsf{e}(g_1, Y_2)$
foreach $b \in \{0,1\}$ do
  if $t_1^{\mathsf{sk}_I^{(1+b)}} = t_2$ then
    return $b$
return $\bot$
```

Figure 11. Expanded SPS-EQ construction for NIAT presignature issuance and token generation using [18].

the client checks that $c \coloneqq \mathsf{H}(g_1, g_2, \mathsf{pk}_C, S, U_0, U_1, V, W, \tilde{S}_0, \tilde{S}_1, \tilde{U}_0, \tilde{U}_1, \tilde{V}, \tilde{W})$.

3. Security Proofs for Section 5

Correctness, reusability of Construction 5 are easily extendable from that of the previous protocol (see Appendix A); and privacy of metadatbit is also proven identically. One-more unforgeability of Construction 5 can similarly be extended from that of the previous protocol with the additional requirement of soundness of $\mathsf{NIZK}_{\mathsf{obt}}$. Due to space constraints, these proofs are thus omitted. We now provide sketches for the unlinkability, double-spend identification and exculpability of our construction, and defer the proofs to the full version of this article.

Theorem 7 (κ-unlinkability). Construction 5 is κ-unlinkable (in the random oracle model) for $\kappa = 2$, assuming $\mathsf{NIZK}_{\mathsf{iss}}$ is an argument of knowledge, that DDH, k-DDH assumption holds in $\mathbb{G}_1$ and EQ perfectly adapts signatures under a malicious signer $\mathsf{NIZK}_{\mathsf{obt}}$ satisfies zero knowledge.

Pf sketch. Recall the proof of Theorem 4 (Appendix A). The idea there was to make the final token effectively independent of $\mathsf{sk}_C =: \alpha$, $\mathsf{psig} =: (\overline{\boldsymbol{\sigma}}, R, S, \pi)$ and $\mathsf{nonce} =: r$, and then use the perfect adaptation of the SPS-EQ signature scheme in order to simulate a valid token. This was facilitated by our use of the random oracle model along with assumptions from the DDH family that allowed us to replace $\mathsf{H}(r)^{\alpha^{-1}}$ in the tag with a random group element in $\mathbb{G}_1$. The approach here remains the same, however we now need to deal with several additional elements in the final token.

Notice that due to zero-knowledge of $\mathsf{NIZK}_{\mathsf{obt}}$, the proof π_{o} is simulatable. Next, we can replace $\mathbf{t} = \left(g_1^{(\alpha+\hat{r})^{-1}}, R^{(\alpha+\hat{r})^{-1}}, S^{(\alpha+\hat{r})^{-1}}\right)$ by $\left(g_1^\rho, (g_1^\varepsilon)^\rho, \left(g_1^{\varepsilon \cdot ((1-b)x_1+bx_2)}\right)^\rho\right)$ where we can might hope to again use inverse DDH argue indistinguishability, but this does not fully work here. Instead, we make the observation that each $g_1^{(\alpha+\hat{r})^{-1}}$ is the output of the Dodis-Yampolskiy PRF [28] which is adaptively secure under the k-DDH assumption [29] for $\mathsf{poly}(\lambda)$ sized domains. This, in particular means that we must ensure that the range of $\mathsf{H}_{\mathbb{Z}}(r)$ is of size $\mathsf{poly}(\lambda)$. However, we remark that this does not influence security of our scheme in any way, although resuability will now hold only with high (and not all but negligible) probability.

Lastly, viewing $\boldsymbol{\phi}$ as ElGamal encryption of $g_1^{\mathsf{id}\cdot(\alpha+\hat{r})}$ (times some extra randomness), we replace it with the encryption of a random value and argue indistinguishability by IND-CPA security of the encryption scheme (which, in turn, follows from DDH). At a high level, we first set each $\mathsf{H}_{\mathbb{G}\to\mathbb{G}}(r) := g_1^{\nu_r}$. Then, for a query of the form $(\cdot, r, \mathsf{id})$, we create the encryption $\boldsymbol{\phi}$ as $\phi_1 := g_1^{\nu_r}$, and $\phi_2 := \mu \cdot \phi_1^{\alpha}$ for message $\mu := \mathsf{pk}_C^{\mathsf{id}} \cdot g_1^{(\mathsf{id}+\nu_r)\cdot\hat{r}}$. Next, we can replace $\phi_1^{\alpha} = g_1^{\alpha\cdot\nu_r}$ with g_1^{γ} for $\gamma \leftarrow\$ \mathbb{Z}_p$, so that $\phi_2 = \mu \cdot g_1^{\gamma}$ is indistinguishable from uniform. Consequently, the final token is now independent of $\mathsf{pk}_C, \mathsf{psig}$ and nonce as desired. □

Theorem 9 (Double-spend identification). Construction 5 satisfies double-spend identification (in the random oracle model) assuming $\mathsf{NIZK}_{\mathsf{obt}}$ is sound.

Pf sketch. If $\mathsf{NIZK}_{\mathsf{obt}}$ is sound, then it follows that the adversary knows $(\mathsf{sk}_C, \hat{r})$ (resp. $(\mathsf{sk}'_C, \hat{r}')$) such that $t_1 = g_1^{(\mathsf{sk}_C+\hat{r})^{-1}}$ (resp. $t_1 = g_1^{\left(\mathsf{sk}'_C+\hat{r}'\right)^{-1}}$) and $\phi_2 = g_1^{\mathsf{id}\cdot(\mathsf{sk}_C+\hat{r})} \cdot \mathsf{H}_{\mathbb{G}\to\mathbb{G}}(t_1)^{\mathsf{sk}_C+\hat{r}}$ (resp. $\phi'_2 = g_1^{\mathsf{id}'\cdot\left(\mathsf{sk}'_C+\hat{r}'\right)} \cdot \mathsf{H}_{\mathbb{G}\to\mathbb{G}}(t_1)^{\mathsf{sk}'_C+\hat{r}}$) with respect to $(\mathbf{t}, \boldsymbol{\sigma})$ (resp. $(\mathbf{t}, \boldsymbol{\sigma}')$). Now since $\mathbf{t}$ is common between both tokens, we have $(\mathsf{sk}_C + \hat{r}) = \left(\mathsf{sk}'_C + \hat{r}'\right)$ (all computations are $\mod p$) so that $\left(\phi_2{}^{-1} \cdot \phi'_2\right)^{\left(\mathsf{id}-\mathsf{id}'\right)^{-1}}$ simplifies to $g_1^{\mathsf{sk}_C+\hat{r}} = g_1^{\mathsf{sk}'_C+\hat{r}'}$, such that at least one of $(g_1^{\mathsf{sk}_C}, \mathsf{nonce})$ and $(g_1^{\mathsf{sk}'_C}, \mathsf{nonce}')$ is in aux with $\hat{r} =: \mathsf{H}_{\mathbb{Z}}(\mathsf{nonce})$ (similarly, $\hat{r}'$). Thus, with all but negligible probability, $\mathsf{DSIden}(\mathsf{pk}_I, \mathbf{t}, \boldsymbol{\sigma}, \boldsymbol{\sigma}') \neq \perp$. □

Theorem 10 (Double-spend exculpability). Construction 5 satisfies double-spend exculpability assuming $\mathsf{NIZK}_{\mathsf{obt}}$ is sound and the discrete log assumption holds in $\mathbb{G}_1$.

Pf sketch. Suppose an adversary accuses some (honest) client pk_C and provides a proof of guilt $\Pi := (\mathbf{t}, \boldsymbol{\sigma}, \boldsymbol{\sigma}', \mathsf{nonce})$ where $\boldsymbol{\sigma} := (\mathsf{s}, \boldsymbol{\phi}, \pi_{\mathsf{o}}, \mathsf{id})$ (similarly $\boldsymbol{\sigma}'$) such that $\left(\phi_2^{-1} \cdot \phi'_2\right)^{\left(\mathsf{id}-\mathsf{id}'\right)^{-1}} = \mathsf{pk}_C \cdot g_1^{\mathsf{H}_{\mathbb{Z}}(\mathsf{nonce})}$ and, both, the proof and signature verify. Having queried $\mathcal{O}_{\mathsf{Issue}}$ at most once per $(\mathsf{psig}, \mathsf{nonce})$ pair (w.l.o.g. suppose the query corresponds to the token $(\mathbf{t}, \boldsymbol{\sigma})$) then, such an adversary must have to create a satisfying proof π'_{o}. However, in order to do so, it either learns sk_C—in which case, we can reduce to the hardness of discrete log—or it is able to create a proof without sk_C, and we can reduce to the soundness of $\mathsf{NIZK}_{\mathsf{obt}}$. □