2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)

SMART Credentials in the Multi-queue of Slackness
(or Secure Management of Anonymous Reputation Traits without Global Halting)

Jack P. K. Ma
Department of Information Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong

Sherman S. M. Chow
Department of Information Engineering
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong

***Abstract*—Anonymous credentials encourage online communication without fear of surveillance, but may invite misbehavior like hate speech. Previous updatable anonymous credentials keep a chronological queue of authenticated sessions and a global pointer to the last chunk of judged sessions. This design allows efficient authentication for proving over only a subset of sessions. However, complications in subjective evaluation often introduce hard-to-judge sessions, which halt all users since sessions that come after the global pointer cannot be redeemed, eventually exceeding the queue size that limits the creation of bad sessions. Such a global-halting loophole may also make judgments overly harsh and hasty.**

We propose SMART (slack management of anonymous reputation traits), maintaining multiple queues so the server could issue interim judgments many times before finalization. Such slackness removes the binary judgment of old methods and mitigates the global-halting issue. Prior schemes only allow score upgrades (WPES '14) or require proving against a global session list since the last checkpoint (S&P '22). Our authentication time is linear in the number of queues or sessions in a designated queue for immediate revocation.

1. Introduction

Securing anonymous access to internet services such as social media and collaborative platforms fosters participation via privacy protection, among other benefits. For example, users may risk censorship and legal/social persecution for posting "sensitive" content online. Unfortunately, anonymous access can also facilitate harmful behavior by allowing malicious users to evade detection and carry out evil deeds such as spreading misinformation or cyberbullying. Service providers (SPs) like Wikipedia [2] restrict what users connecting via Tor [23] can do since its anonymous connection makes it hard to prevent vandals.

Some systems address the tension between anonymity and accountability with trusted third parties (TTPs). TTP-based revocation (*e.g.*, group signatures [15], [21], [38], traceable signatures [17], [28] or "revocable anonymous credentials" [1], [14]) often assumes that the misbehavior has been deanonymized by external means. A TTP could use the identifying information to recover some tracing tokens for revocation. Some systems define misbehavior with objectively evaluable criteria, *e.g.*, double-voting [19] or exceeding the limit of k-times authentication [9]. It takes no external mechanism to judge. Such misbehavior can often be detected quickly.

Sherman Chow (corresponding) is supported by General Research Fund (CUHK 14210621 and 14209918) and is grateful for the Early Career Award on "Accountable Privacy in Online Communication" (439713) from RGC, Hong Kong, We thank reviewers for their helpful comments.

Many systems allow diverse forms of user interactions. Subjective assessments are needed to identify misbehavior. The SP may resort to "external" evaluation, say, voting among the clients, user reports, judgment of moderators, or a professional review by domain experts such as lawyers. This process often takes time, especially when there are controversies or complexities like translation, *e.g.*, "Zhemao" hoaxes on simplified Chinese Wikipedia.

Completely blocking users may not always be desirable. It is important to incentivize positive contributions by maintaining reputation scores. It also helps alleviate sybil attacks. Users who have yet to build up a reputation might be deemed less trustworthy than those who have. Similar to judging bad deeds, (finalizing the) subjective evaluation can be time-consuming. Systems with slow/sporadic reviews, *e.g.*, Wikipedia, might employ a "time-dependent model" (*cf.*, blockchain consensus) – if a contribution has not been flagged inappropriate for, say, six months, it is deemed solid, and a score will be assigned.

1.1. Two Major Paradigms

Judging since Genesis. Blacklistable anonymous credentials (BLAC) [33] feature TTP-free subjective judgment, enforced by zero-knowledge proof (ZKP) against a global blocklist $\mathcal{L}$. A user generates a ticket $\mathsf{tag} = \mathsf{PRF}_x(\mathsf{sID})$, a pseudorandom function (PRF) output of its credential secret x over an assigned session identifier sID. If a session is judged to be bad, the SP puts sID and its tag into $\mathcal{L}$. BLAC is readily extensible to BLACR [8] with reputation via proving membership (claiming positive scores) on top of non-membership proof. Its authentication costs $O(|\mathcal{L}|)$.

SNARKBlock [30] improves the non-revoked proof in BLAC with zero-knowledge succinct non-interactive arguments [27] (zkSNARK) for $O(\log |\mathcal{L}|)$-time verification. Since the user proving complexity remains linear, hidden common input aggregate proofs [30] is proposed, which allows for the aggregation of existing proof for different chunks of the blocklist. However, if we update $\mathcal{L}$, say, using it for reputation [8] and allowing updatable scores, it invalidates the corresponding (offline) precomputation. As an amortized approach, infrequent users need to catch up with linear proof for all sessions that have happened since they walked away. New users cannot benefit either unless they reveal their joining time to skip older chunks. Their authentication is set apart from other users and degrades their anonymity. BLACR-express [8] faces similar shortcomings. To mitigate the scalability bottleneck due to the ever-growing global list, we study alternative designs.

A Fixed-Size Queue Design. PEREA [34] proposes to store only the K most recent session identifiers as *tickets* in a credential and use K (non-)membership proofs for claiming positive scores or showing they are guilt-free. Reputation can only be transient. PERM [7] keeps tickets in a user credential[1]. So, (infrequent/new) users only prove w.r.t. their own fixed-size credentials but not the ever-growing list of global sessions. Unfortunately, this design comes with a dealbreaker. Adversaries are motivated to authenticate rapidly to *wash out* problematic tickets from the fixed-sized queue. A possible remedy is to tightly couple another *anonymous* rate-limiting mechanism with the credential, which is non-trivial.[2] Rate-limiting also unnecessarily curbs the enthusiasm of honest users [37].

1.2. The Unspoken Global Halting Issue

To support persistent reputation scores besides blocklisting, users must absorb judgments into their credentials upon each interaction, even if judgment comes later. Later designs [7], [35] speed up authentication by skipping sessions that have been absorbed or proven irrelevant.

PERM [7] extends PEREA with a global *judgment pointer* pointing to the last one in a *continuous* chronological sequence of judged sessions, enabling an easy proof of judged/unjudged status of tickets in a credential instead of proving w.r.t. $\mathcal{L}$. Meanwhile, the last unjudged ticket in a credential cannot be too far away from such a pointer. Without this rule as a part of authentication semantics, malicious users can inject many "bad" sessions before being caught, leading to a *denial-of-service (DoS)* vulnerability of overloading the session evaluation operation.

FARB [35] explicitly makes the sequential order assumption on session identifiers (higher IDs for later sessions). Every ticket after the global pointer must be unjudged. A single range proof can replace PERM's linear disjunctive proofs (judged/unjudged) of each ticket.

Cryptographic enforcement of such authentication semantics using the global pointer is incompatible with the time-consuming evaluation based on subjective criteria. A *single* controversial session that takes long to finalize stalls the moving of *global* judgment pointer and halts *all users*, not just the one who originated the session, no matter how many later sessions have been judged. The technical reason for such *global halting* is that no users can redeem (and remove) any sessions beyond the global judgment pointer (pointing to the *earliest* unjudged session) since their status cannot be inferred without direct explicit proof.

Once the global pointer moves a bit slower than the authentication rate (and hence the birth rate of sessions), tickets accumulate in the credential and reach the queue size limit for DoS resilience. Such a global-halting vulnerability also affects the operational aspect of judgments, forcing the subjective evaluation operates like a single-thread process on a first-come-first-served basis.

Notable Exceptions. PE(AR)^2 proposed by Yu *et al.* [37] is an interesting attempt to achieve the best of both worlds, namely, maintaining K pending sessions in the credential but without mandating sequential judging. It uses a blocklist to revoke [33] instead of assigning a very negative score [35]. Apart from a faster non-membership proof for a set of elements, it supports "soft" rate-limiting by storing a variable number of unjudged sessions and proving that it is less than some given threshold, albeit at a degraded level of anonymity PE(AR)^2 does not support negative/downgradable scores. ARBRA [36] supports negative scores, but the user authentication cost is again linear in $|\mathcal{L}|$ plus the number of the user's unredeemed tickets. We defer our concurrent work [20] to concluding remarks.

1.3. Revocation with Mercy/Elasticity or Urgency

Anonymous blocklisting/reputation systems should allow "revocation with mercy." When accused, anonymous blocklisting suspends users instead of permanently revoking access. If cleared, users can be removed from blocklists. It is exactly the existence of "difficult-to-judge" sessions that motivates the need for anonymous blocklisting.

The sequential requirement of PERM/FARB mandates the SP to judge sequentially and only once. The one-time judgment forces SPs to catch misbehaving users in time, or otherwise, they may escape from punishment forever. These system constraints severely limit the applicability. Consider contact tracing; a user can be "forgiven" if the place s/he checked in is indeed safe (and vice versa). Given the system constraints, overly cautious and harsh judgments for preemptive measures seem logical.

Enabling multi-judging, specifically downgrading, can be challenging since users have no incentive to voluntarily redeem worse ratings. Naïve attempt probably involves inefficient ZKP enumerating all sessions ever associated with a credential, not to mention the wash-away problem.

PERM/FARB also does not readily allow immediate revocation. The SP must rapidly *finalize* the judgment of all tickets prior to the one needing immediate attention.

1.4. Motivating the Design Choices of SMART

We design SMART for "secure management of anonymous reputation traits" with a few distinctive features.

Sublinear Authentication. SMART mandates users to redeem the (temporary) score of the earliest session across all queues (independent of their number of transient judgments). The authentication cost scales linearly with the number of queues, or sublinear in the number of user sessions NK when each queue has K of them, as unjudged tickets of a queue can be shown via range proofs.

Multi-queue of Slackness. A credential maintains multiple queues, each for sessions with the same number of transient judgments made but with different transient (*i.e.*, not finalized) scores. Making one more transient judgment

1. Non-membership proof via an accumulator takes constant verification costs. However, creating a witness is linear and requires recomputation when the accumulated values change. PERM replaces PEREA's K such proofs w.r.t. the public accumulator storing all blocked tickets to K disjunctive proofs, allowing faster authentication with $K > 10$ [34].

2. The rate-limiting of k-time anonymous authentication [9] is periodic. A burst of $2k$ authentications can be run across the epoch boundary.

on a session moves it to the next queue. When the score is finalized, the user redeems it out of the credential.

Supporting multiple queues brings us many benefits. Scores of different sessions can keep updating (up to $(N-1)$ times) and possibly at different paces. They could also be finalized at different times. The SP now has more flexibility when finalizing the session scores. Our design thus fits with different operational characteristics.

Maintaining multiple queues seems an intuitive and arguably "trivial" solution to avoid one single queue from getting stuck due to global halting. Note that the global halting problem happens at the head of a queue but not the end. Enlarging the buffer alone does not help much.

Custom Rate-Limiting. More importantly, we argue that the apparent quantitative difference from a single queue actually contributes to qualitative differences in two ways.

FARB has built-in rate-limiting from range proof. In single-queue FARB, rate-limiting must be enforced; otherwise, it is vulnerable to DoS attacks of rapidly inducing many bad sessions before the blocklisting becomes effective, *i.e.*, when the earliest such session is put into the blocklist and the authenticating user is forced to redeem it.

Note that these newly spawned sessions only appear at the end of a queue. In our multi-queue design, they all appear in $\mathcal{Q}_0$. In other words, SMART only needs to rate-limit $\mathcal{Q}_0$. Note also that the SP knows at which queues after $\mathcal{Q}_0$ those difficult-to-judge sessions are. The SP can set a different rate limit accordingly or even remove the limit altogether. This ensures the correctness of SMART's cryptographic enforcement. Namely, transient judgments move the tickets to the next queue, which will never be full, in contrast to the (only) base queue $\mathcal{Q}_0$ (in FARB).

Immediate Revocation. FARB features constant-time authentication due to its clever use of range proof instead of proving about each ticket in the single queue $\mathcal{Q}_0$. With multiple queues, SMART can assign different "responsibilities" to them. Of particular interest is an emergency queue for immediate revocation. Specifically, any severely serious session will be sent there. This queue does not need to be long. Authentication now runs a special linear scan of the emergency queue. Here, we can do reputation-based blocklisting via assigning a very negative score, or a direct blocklisting that merely a single ticket there will block the credential from further authentication. Our easiest implementation is to make this queue the last one.

Rendering this idea with FARB runs into a dilemma. This queue cannot be long for efficiency with the forced linear scan (which spoils the constant-time authentication time of FARB). This queue cannot be short either for being the only storage for unjudged sessions. In short, prior schemes with sequential judging require the SP to judge once-and-for-all. An immediate revocation of a new session can only be performed by removing all previously unjudged sessions. The SP either blocks or forgives all of them. SMART allows transient judgments to keep the affected sessions and extend to support microscopic queue management, *e.g.*, designated queues to handle these bans.

Mitigating Global-Halting. Sequential judging still exists to a certain degree since we move a sequential chunk of tickets to the next queue. However, the backlog of non-finalized sessions is split across different queues.

There does exist a pathological case that our system will reach the "limit." Still, the possibility is dramatically trimmed down along three dimensions – individual transient judgment, enlarging the buffer, and customizable queue-specific rate-limiting. The SP can make informed choices in "slacking off" along these dimensions. In short, the SP can better manage the operations for subjective evaluation with all these additional features of SMART.

Revocations with Mercy and Elasticity. Transient scoring naturally supports revocation with mercy/elasticity.

Versatile Applicabilities. Consider contact tracing; a user checked in a place can have an initial (neutral) score indicating the risk level. For visits later identified as posing a high risk, the health department can downgrade the score (*e.g.*, to the minimum). After the negative results of some antigen tests, it can be upgraded. With more negative results from regular tests, the risk of a visit will eventually vanish. The number of changes before finalization is limited, fitting the time-dependent model.

Consider content reviewing (*e.g.*, videos or photos); an uploaded post may go through an initial review and is put aside. The SP may wish to temporarily reward/punish the uploader by issuing transient judgments. The multi-queue design allows a multi-layered (periodic) review process, *e.g.*, the post can be judged monthly, and finalizing and blocking can be done when needed. This also fits some (anonymous) point reward systems that take a few days to review each point collection/transaction.

Cryptographic Enforcement. Ensuring the efficiency of authentication requires careful formulations of how users are identified and how their sessions are scored. SMART uses multiple queues to ease the tensions between operational characteristics of judgment and its corresponding cryptographic enforcement. It boils down to moving sessions across credential queues (rather than simply adding them to a single queue). This aligns with how judgments are updated across the SP-side global queues. We extensively use zero-knowledge proof of knowledge (PoK) of credential signatures (signatures with efficient protocols) and homomorphic commitments for credential updates. To prevent attackers from reusing signatures and rolling back credentials, we rely on nonces and data-structure invariants, namely, monotonic increasing session identifiers and the head/tail pointers of the queues.

2. Preliminaries

Let λ be the security parameter and $\mathbb{G}$ be a cyclic group of order p, where p is a λ-bit prime. PPT means probabilistic polynomial time (in λ). The set of integers $\{0,\ldots,N-1\}$ is denoted by $[N]$. If X is a finite set, $x \leftarrow\!\$\ X$ denotes sampling an element x uniformly at random from the set X. An empty output is denoted by $\perp$.

Let $\mathcal{R} : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ be a binary relation which takes a statement x and witness w. Let $\mathcal{L}_{\mathcal{R}} = \{x | \exists w \text{ s.t. } \mathcal{R}(x,w) = 1\}$ be a language associated with the relation $\mathcal{R}$, which can also include a common reference string crs as a part of the statement x.

We use $\mathsf{P}(\mathcal{U}(in_{\mathcal{U}}), \mathcal{S}(in_{\mathcal{S}})) \to (out_{\mathcal{U}}, out_{\mathcal{S}})$ to denote parties $\mathcal{U}$ and $\mathcal{S}$ interact via the protocol P taking in_i and outputting out_i for $i \in \{\mathcal{U}, \mathcal{S}\}$, respectively.

2.1. Homomorphic (Vector) Commitments

In a commitment scheme Π^{com}, the sender can commit to a message m via a commitment $C_m = \mathsf{Com}(m; r)$ for some randomness r. It can later be opened by providing (m, r) (possibly in ZKP). For an ordered tuples of message $M = (m_1, \dots, m_n)$, we denote the set of committed messages by $C_M = \{\mathsf{Com}(m_i; r_i)\}_{i=1}^n$. Note that this set of commitments does not preserve the message-index relation (m_i, i) in the vector, *i.e.*, it is not position binding.

Definition 1. *A commitment scheme is defined as follows:*

$\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$*: It takes as input the security parameter* 1^λ*, and outputs a public parameter* pp*.*

$\mathsf{Com}_{\mathsf{pp}}(m; r) \to (C, o)$*: Given the public parameter* pp *(as an implicit input), the message* m*, and randomness* r*, it outputs a commitment* C *and opening* $o = (m, r)$*.*

$\mathsf{Vf}(\mathsf{pp}, C, m, o) \to b$*: The verification algorithm outputs* 1 *if* o *is a valid opening for the commitment* C*,* 0 *otherwise.*

It should satisfy the hiding and binding properties.

(Perfect) Hiding. Given pp, any (computationally unbounded) adversary who chooses (m_0, m_1) cannot distinguish b when given $c = \mathsf{Com}(m_b; r_b)$ for random b.

Binding. No PPT adversary can produce valid openings o_0, o_1 that open a commitment c to distinct m_0, m_1.

In a vector commitment scheme, one can commit to a vector of message $\vec{m} = (m_1, \dots, m_n)$. The commitment should be hiding and (position)-binding.

The (generalized) Pedersen commitment [29] allows m to be a vector of field elements. Let $\{g_i\}_{i=1}^{n+1}$ be independent generators of $\mathbb{G}$. A commitment of $\vec{m} \in (\mathbb{Z}_p)^n$, denoted by $C_{\vec{m}} = \mathsf{Com}(\{m_i\}_{i=1}^n; r)$, is $g_{n+1}^r \prod_{i=1}^n g_i^{m_i}$, where r is randomly picked from $\mathbb{Z}_p^*$. It also features homomorphism, *i.e.*, $\mathsf{Com}(\vec{a}) \cdot \mathsf{Com}(\vec{b}) = \mathsf{Com}(\vec{a} + \vec{b})$.

One can prove that the set of committed messages $\{C_{m_i} = \mathsf{Com}(m_i)\}_{i=1}^n$ commits to the entries of a generalized Pedersen commitment $C_{\vec{m}}$ on vector $\vec{m}$. We use Pedersen commitments for our credential attributes.

2.2. Zero-Knowledge Proof or Argument

A prover can convince a verifier via a zero-knowledge proof of knowledge (ZKPoK) protocol that a statement x is true without revealing w and any other information. A (non-interactive) proof system Π is defined as follows.

Definition 2 (Proof System)**.** *A proof system for a language* $\mathcal{L}$ *consists of three algorithms* $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$*:*

$\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$*: It outputs the common reference string.*

$\mathsf{P}(\mathsf{crs}, x, w) \to \pi$*: It outputs a proof for the statement* x *and witness* w*, such that* $(x, w) \in R$ *given* crs*.*

$\mathsf{V}(\mathsf{crs}, x, \pi) \to b$*: It outputs* 1 *if* π *is a valid proof for* x*.*

A proof system must be complete and sound.

Completeneess. $\forall x \in \mathcal{L}$ and w where $(x, w) \in R$, the prover can always output π such that the verifier accepts.

Soundness. $\forall x \notin \mathcal{L}$, no prover can output π to make a verifier accepts.

A zero-knowledge proof (argument) of knowledge is a proof system (denoted by ZKP) that satisfies knowledge soundness (against computationally bounded provers) and zero-knowledgeness (defined in Appendix A.1).

Knowledge Soundness. No cheating provers can convince an honest verifier that for some false statement $x \notin \mathcal{L}$. It requires the existence of an extractor Ext that can extract the witness w from any proof π passing the verification.

Zero Knowledge. For all statements $x \in \mathcal{L}$, there exists a simulator Sim that can simulate a proof indistinguishable from the real proof, *i.e.*, the proof does not leak additional information about the witness except the statement is true.

ZKP Notation. We use the standard notation introduced by Camenisch and Stadler [15]. For example, $\mathsf{ZKP}\{(x) : y = g^x\}$ denotes ZKPoK for proving x such that $y = g^x$. The values on the left of the colon are not known by any verifier. Symbols on the right are public values.

2.2.1. Non-Interactive Proof and Disjunctive Proof.

Σ-protocols [31] are three-move ZKP, which can be made non-interactive with the Fiat-Shamir transform.

Disjunctive ZKP of two or more statements can be realized by the secret-sharing technique [5], [22]. For $R_1 \vee R_2$, the prover can prove R_1 with the witness under the challenge $e - e_1$, where e is picked by the verifier and e_1 is self-picked randomness, then simulate the ZKP for R_2.

2.2.2. zkSNARK for Range & Circuits.

A succinct non-interactive argument of knowledge (SNARK) features succinct proofs sublinear in the statement size. The arithmetic circuit over secret (witness) inputs and public inputs and outputs can be translated into rank-1 constraint systems (R1CS) and proved via (zk)SNARK. Notable zkSNARK examples include Groth16 [27] and Bulletproofs [11].

Bulletproofs support succinct range proof using inner product arguments. It is special honest-verifier zero-knowledge for the relation $\mathcal{R} : ((\mathsf{pp}, C_v, A, B), (v, r) : C_v = \mathsf{Com}(v; r) \wedge A \le v < B)$ where $A = 0, B = 2^n$ and Com is a Pedersen commitment $(g^v h^r)$ with $\mathsf{pp} = (g, h)$. It takes a logarithmic number of rounds in the bit-length n and can be made non-interactive with a logarithmic proof size via Fiat-Shamir transform. With only a $\log(m)$ overhead, m range proof can be aggregated.

2.2.3. Commit-and-Prove (CP) ZKP and zkSNARK.

LegoSNARK [16], using a trusted setup, enables composing different zkSNARK for modular designs. Let $\Pi^{\mathsf{com}} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Vf})$ be a commitment scheme. A CP-SNARK $(\mathsf{KGen}, \mathsf{P}, \mathsf{V})$ is a zkSNARK for the relation:

$$\mathcal{R}_{\mathsf{CP}} : ((x, (c_1, \dots, c_\ell)), ((m_j, o_j)_{j \in [\ell]}, w) :$$
$$\bigwedge_{j \in [\ell]} \Pi^{\mathsf{com}}.\mathsf{Vf}(\mathsf{pp}, c_j, m_j, o_j) = 1 \wedge R(x, (m_j)_{j \in [\ell]}, w) = 1)$$

where $\mathsf{pp} \leftarrow \Pi^{\mathsf{com}}.\mathsf{Setup}(1^\lambda)$, $c_j = \Pi^{\mathsf{com}}.\mathsf{Com}(m_j; o_j)$ and R is a relation over statement x on committed inputs (m_j) and witness w.

2.3. Credential Signatures with Protocols

Definition 3. *A credential signature scheme* $\mathsf{Cred} = (\mathsf{KGen}, \mathsf{Sig}, \mathsf{Vf})$ *for message vectors, associated with a vector commitment scheme* $\Pi^{\mathsf{com}} = (\mathsf{Setup}, \mathsf{Com}, \mathsf{Vf}')$ *with public parameter* pp*, is defined by the algorithms/protocols below, all taking* pp *as implicit input.*

$\mathsf{KGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$*: The probabilistic key generation algorithm takes the security parameter* λ *and outputs a public key* pk *and a secret key* sk.

$\mathsf{Sig}(\mathcal{U}(\vec{m};\rho), \mathcal{S}(\mathsf{aux}, \mathsf{sk})) \to (\sigma, \mathsf{aux}')$*: In the signature-issuing protocol, the user* $\mathcal{U}$ *hides the message* $\vec{m}$ *with randomness* ρ *using* Π^{com}. *The signer* $\mathcal{S}$ *inputs* sk *and* aux. *The user receives a signature* σ. *The signer receives* aux'.

Both aux *and* aux' *represent auxiliary information, one for input and one for output. For example,* aux *can be commonly known messages or ZKPs involving previously issued signatures, and* aux' *can be commitments of the messages* $\mathsf{Com}(\vec{m}), \{\mathsf{Com}(m_i)\}$ *and proofs of opening.*

$\mathsf{Vf}(\mathsf{pk}, \vec{m}, \sigma) \to b$*: The deterministic verification algorithm takes as input the public key* pk*, the message vector* $\vec{m}$*, and a signature* σ*. It outputs a result bit* b*. For brevity,* $(\vec{m}, \sigma)$ *might be collapsed into a single input.*

It also supports efficient (non-interactive) proof for the knowledge of a signature over a vector commitment of $\vec{m}$ or over a subset of committed messages, *e.g.*, $\mathsf{ZKP}\{(\vec{m}, \rho, \sigma) : \mathsf{Vf}(\mathsf{pk}, \vec{m}, \sigma) = 1 \wedge C_{\vec{m}} = \mathsf{Com}(\vec{m}; \rho)\}$.

It should be existentially unforgeable against adaptive chosen-message attacks. No PPT adversary can output a valid signature on a message not in the queried message set, given that it can run the signing protocol with the signer for queried messages polynomially many times. The detailed definition is presented in Appendix A.2.

We use the BBS+ signature of Au, Susilo, Mu, and Chow [9]. Camenisch, Drijvers and Lehmann [13] proposed a PoK for signature on partially disclosed messages without target-group operations in the (Type-III) pairing setting, by also publishing signer secrets in the exponent.

In SMART, the user commits the attribute vector $\mathbf{attr}$ (and its update $\mathbf{attr}'$) and sends it to the SP. For flexibility (*e.g.*, using the most efficient ZKP for a specific relation), the user can also send a set of individual commitments on C_{v_i} for $v_i \in \mathbf{attr}$ and proves in ZK that each C_{v_i} corresponds to the i-th attribute committed in $C_{\mathbf{attr}}$.

3. Anonymous Reputation

Anonymous credentials with reputation revoke users by assigning a low score to their previous sessions, barring them from fulfilling authentication policies in the future. A secure design should mandate that users redeem all redeemable sessions before authentication. Redemption can be done one by one or in a batch. It can be a standalone protocol or integrated with the authentication.

Our system allows unblocking by supporting upgrading the score of non-finalized sessions. Meanwhile, the score can also be downgraded. Not all systems offer all these features. For example, PEREA/PERM does not support score downgrading or separate credential updates from authentication; each authentication removes one ticket from the credential, which could even be unjudged!

3.1. Syntax

Definition 4. *An anonymous credential system with reputation consists of the PPT algorithms/protocols below.*

$\mathsf{Setup}(1^\lambda) \to (\mathsf{pp}, \mathsf{sk}, \mathsf{st})$ *is the SP setup algorithm. It inputs a security parameter* 1^λ *and outputs the public parameter* pp*, a secret key* sk*, and public state* st.

In all protocols below, both the SP and users take the public state st *as implicit input. If it fails to run, both the SP and users get* $\perp$*; otherwise, the SP updates* st *into* st'*. Policies* $\mathtt{AP}$*,* $\mathtt{UP}$*, and update function* f *also take* $(\mathsf{pp}, \mathsf{st})$.

$\mathsf{Reg}(\mathcal{U}(\mathsf{pp}), \mathcal{S}(\mathsf{pp}, \mathsf{sk})) \to (\mathsf{cred}, \mathsf{st}')$*: The SP* $\mathcal{S}$ *uses its secret key* sk *to register the user* $\mathcal{U}$ *by creating a credential* cred *with initial attributes defined by* pp.

$\mathsf{Auth}(\mathcal{U}(\mathsf{pp}, \mathsf{cred}, \mathtt{AP}, f), \mathcal{S}(\mathsf{pp}, \mathsf{sk}, \mathtt{AP}, f)) \to (\mathsf{cred}', \mathsf{st}')$*: A user* $\mathcal{U}$ *with a credential* cred *authenticates to the SP* $\mathcal{S}$*. If it passes authentication policy* $\mathtt{AP}$*, i.e.,* $\mathtt{AP}(\mathsf{cred}) = 1$*,* $\mathcal{U}$ *obtains an updated credential* cred' *on updated attributes* $\mathbf{attr}' \leftarrow f(\mathsf{cred})$ *as generated by* $\mathcal{S}$ *using its secret key* sk*. The created authenticated session is captured by* st'.

$\mathsf{StMgn}(\mathsf{pp}, \mathsf{sk}, \mathsf{st}) \to \mathsf{st}'$*: The SP runs the state management algorithm to render the subjective judgment (not explicitly listed as an input) into a redeemable state. It uses the secret key* sk *to update current state* st *into* st'.

$\mathsf{Redeem}(\mathcal{U}(\mathsf{pp}, \mathsf{cred}, \mathtt{UP}, f), \mathcal{S}(\mathsf{pp}, \mathsf{sk}, \mathtt{UP}, f)) \to (\mathsf{cred}', \mathsf{st}')$*: Any user* $\mathcal{U}$ *runs this protocol to update* cred *to redeem what is pending to redeem, i.e.,* $\mathtt{UP}(\mathsf{cred}) = 1$*, and barring* cred *from authentication, i.e.,* $\mathtt{AP}(\mathsf{cred}) = 0$*. The update does not generate new authenticated sessions.*

Throughout this work, the attributes $\mathbf{attr}$ are the main credential data (*e.g.*, the data structure storing tickets), while cred contains a credential signature on $\mathbf{attr}$ (and $\mathbf{attr}$ itself) and possibly other (certified) auxiliary data. In particular, update function f in SMART also returns *"enqueuing data"* for the SP to issue enqueuing signatures as a part of the user credential.

3.2. Security Requirements

An anonymous credential system with reputation should provide the following security properties:

Completeness. An honest user can authenticate (redeem) with cred if it satisfies the access (update) policy, *i.e.*, $\mathtt{AP}(\mathsf{cred}) = 1$ ($\mathtt{UP}(\mathsf{cred}) = 1$), when talking to an honest SP. Upon completion, cred updates according to $f(\mathsf{cred})$.

Anonymity. A corrupted SP cannot distinguish any two users who request to authenticate or redeem with credential cred_0 and cred_1, respectively, where $\mathtt{AP}(\mathsf{cred}_0) = \mathtt{AP}(\mathsf{cred}_1)$ or $\mathtt{UP}(\mathsf{cred}_0) = \mathtt{UP}(\mathsf{cred}_1)$, *i.e.*, they hide among the anonymity set indexed by the policy $\mathtt{AP}$ or $\mathtt{UP}$.

Soundness. Users with an invalid cred cannot authenticate or redeem with SP, *i.e.*, Auth (Redeem) returns $\perp$ for $\mathtt{AP}(\mathsf{cred}) = 0$ ($\mathtt{UP}(\mathsf{cred}) = 0$).

Security games are deferred to the full version.

It remains to specify soundness regarding $\mathtt{AP}, \mathtt{UP}$, and credential update function f specifically used by SMART.

(i) **Redeemable.** Authentication updates a credential to include a new, unique session. Redeeming updates the credential to incorporate the finalized score of the session.

In more detail, every successful authentication Auth creates a unique session identifier (ID), which is added into $\mathbf{attr}'$ of cred' as updated by f. A ticket of the same ID with an unjudged status will be added into st' too, which allows later SP judgment via StMgn. User with cred can run Redeem, where $\mathtt{UP}(\mathsf{cred}) = 1$, to accumulate the judgment from st' into cred' via f. The uniqueness of tickets prevents users from "sharing" a session.

(ii) **Non-escaping.** The authentication policy AP disallows further authentication of credential cred, *i.e.*, $\mathtt{AP}(\mathsf{cred}) = 0$, until cred is free from any unredeemed but judged session as specified by st. Meanwhile, Redeem only removes finalized tickets from cred according to UP and incorporates its score into cred' via f, where the SP specifies the score in st' via StMgn.

(iii) **Transient Judging.** A ticket t can be temporarily judged with some score via StMgn until it is finalized. In more detail, when cred that originated t is used in Auth, its evaluation of AP will take into account the score of t as reflected by st on top of the current score of cred. Note that Redeem does not remove such non-finalized tickets.

4. Technical Overview of SMART

4.1. Data Structures for Credential & Judgments

SMART extends the single implicit queue design of FARB [35] to multiple queues while carefully ensuring that no user can escape punishment even when they are anonymous. In our "baseline" scheme below for illustration, we consider users specify the queue to redeem. Still, the identifier of the ticket being redeemed remains private. Table 1 lists some major notations of the whole paper.

4.1.1. Our Baseline Design from FARB. A main credential is $(x, \nu, \mathsf{s}, \mathsf{head}, \mathsf{tail}, \sigma)$, with s as its current score and σ is an SP's signature signing everything else. To prevent reuse, each authentication/redemption reveals the nonce ν. The user then chooses a fresh value for the SP to blindly issue an updated credential.

After each authentication, the credential will be equipped with a new ticket, meaning that the user also gets *enqueuing signatures* $\{\mathsf{enQ}_x^{(i)} = (x, t_i, i, \sigma_x^i)\}_{i=1}^{|\mathcal{Q}|}$. The ticket number t_i comes from a global counter $\mathfrak{tc}$ that increments by 1 for each authentication. The secret x links to enqueuing signatures in an implicit ticket queue $\mathcal{Q} = (\{\mathsf{enQ}_x^{(\mathsf{head}+1)}, \ldots, \mathsf{enQ}_x^{(\mathsf{tail})}\}, \mathsf{head}, \mathsf{tail})$ with $k = (\mathsf{tail} - \mathsf{head})$ tuples. Due to the *ticket order invariant*, *i.e.*, the increasing order of t_i's, the user only needs to prove the first ticket $t_{\mathsf{head}+1}$ is unjudged via proving $\mathfrak{jp} < t_{\mathsf{head}+1}$, where $\mathfrak{jp}$ is the *global judgment pointer* pointing to the last judged authentication. FARB assumes a continuous order of judgment, *i.e.*, all tickets smaller than $\mathfrak{jp}$ are judged, and all bigger than $\mathfrak{jp}$ are unjudged.

A judgment is $(t, \mathsf{s}_t, \sigma')$ with an SP signature σ' signing on a ticket t with score s_t. Redeeming a judgment on $t_{\mathsf{head}+1}$ with $\mathsf{enQ}_x^{(\mathsf{head}+1)}$ performs dequeue on $\mathcal{Q}$, which updates $\mathcal{Q}$ to $(\{\mathsf{enQ}_x^{(\mathsf{head}+2)}, \ldots, \mathsf{enQ}_x^{(\mathsf{tail})}\}, \mathsf{head}+1, \mathsf{tail})$, and s in cred absorbs its score $\mathsf{s}_{t_{\mathsf{head}+1}}$. Authentication increments tail into $\mathsf{tail}' = \mathsf{tail} + 1$, and the SP blindly issues an enqueuing signature on $(x, t_{\mathsf{tail}'}, \mathsf{tail}')$ to enqueue t'_{tail}. The queue of signatures becomes $\mathcal{Q}' = (\{\mathsf{enQ}_x^{(\mathsf{head}+1)}, \ldots, \mathsf{enQ}_x^{(\mathsf{tail})}\} \cup \{\mathsf{enQ}_x^{(\mathsf{tail}')}\}, \mathsf{head}, \mathsf{tail}')$.

Figure 1 illustrates the global-halting issue of FARB and its failure to support immediate revocation. Suppose each of the two users authenticated five times; session $t = 2$ is hard to judge, blocking the advancement of the global pointer $\mathfrak{jp}$. Alice is then halted if the queue size limit is 5. Also, for the SP to immediately revoke the user who spawned $t = 8$, it needs to rush through $t \in [2..7]$.

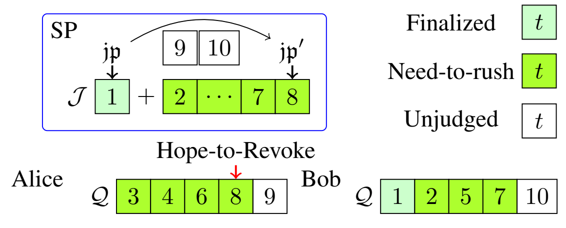


Figure 1. FARB's global halting and delayed revocation: i) Suppose only $\mathfrak{jp} = 1$ is judged and $t = 2$ is "hard-to-judge": Alice cannot authenticate if the queue size limit has been reached, even though she did not originate session $t = 2$. ii) Immediately revoking ticket 8 requires the SP to rush in finalizing judgments on $[2..7]$ to advance $\mathfrak{jp}$.

4.1.2. SMART's Three Types of Signatures. We extensively use signatures on commitments to maintain the data structure in credentials. This part focuses on the data they sign, and so do many parts in Section 5. We sometimes use an abused notation that omits the signature for brevity.

i) Main credential stores N pair of pointers, *i.e.*, $\mathsf{cred} = (x, \nu, \mathsf{s}, \{\mathcal{Q}_j.\mathsf{head}, \mathcal{Q}_j.\mathsf{tail}\}_{j \in [0, N-1]})$, *e.g.*, σ in Figure 4.

ii) Enqueuing signatures on $\{\mathsf{enQ}_x^{(i,j)} = (x, t_i, i, j)\}$ associate ticket t_i to the i-th position in the j-th (implicit) queue, *e.g.*, the columns at $\mathcal{Q}_0, \mathcal{Q}_1$ in Figures 4-6.

iii) Judgment is a signature on the state (elaborated below) of a ticket possibly with a score, *e.g.*, list $\mathcal{J}_0$ in Figures 5-6. Unlike the above two, the SP maintains this in public. We also call lists $\mathcal{J}_j, \forall j \in [0, N-1]$, by *score lists*.

4.1.3. Multiple Queues for Transient Judgment.

$$\mathcal{Q}_j = (\{\mathsf{enQ}_x^{(\mathsf{head}+1,j)}, \ldots, \mathsf{enQ}_x^{(\mathsf{tail},j)}, \mathsf{head}, \mathsf{tail})\}, j \in [N]$$

are queues implicitly maintained by $(\mathcal{Q}_j.\mathsf{head}, \mathcal{Q}_j.\mathsf{tail})$ in the credential. $t \in \mathcal{Q}_j$ refers to ticket t that is encoded in some $\mathsf{enQ}_x^{(i,j)} \in \mathcal{Q}_j$, with $i \in [\mathcal{Q}_j.\mathsf{head} + 1, \mathcal{Q}_j.\mathsf{tail}]$.

Each judgment is a tuple $(t, j, b_{\mathsf{next}}, \mathsf{s}_{(t,j)})$ for ticket t residing at $\mathcal{Q}_j$ with score $\mathsf{s}_{(t,j)}$ (or simply s_t). It can be:

i) not judged (the initial status, or judged $j = 0$ times),

ii) skipped ($b_{\mathsf{next}} = 1$, as an interim decision), or

iii) finalized ($b_{\mathsf{next}} = 0$, to redeem out of a credential). A signature (by the SP) on the judgment tuple changes the state of ticket t, moving it from the j-th queue to the $(j+1)$-th queue (if $b_{\mathsf{next}} = 1$) of a credential or out of the credential (if $b_{\mathsf{next}} = 0$) when the user redeems it.

For system parameter N (the total number of queues), a session can be "skipped" at most $(N - 1)$ times, each corresponds to an interim decision or *transient judgment*. Correspondingly, SMART maintains N judgment pointers $\mathfrak{jp}_j, \forall j \in [0, N-1]$, which is simply the largest ticket (ID) in the j-th score list $\mathcal{J}_j$, not like the single $\mathfrak{jp}$ in FARB.

4.1.4. Intra-Queue and Inter-Queue Invariants. Sequential judging *within each queue* is still needed to maintain the *intra-queue invariant.* A judgment on t can only be done after a (transient) judgment on each of the previous tickets in the *same* queue has been made.

For simplicity, we enforce an *inter-queue invariant*

$$t \in \mathcal{Q}_i \implies t > t' \ \forall \ t' \in \mathcal{Q}_{i+1} \ (\forall i \in [N]).$$

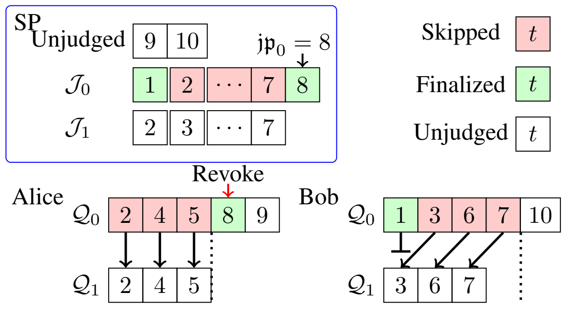


Figure 2. The SP decides to make a transient judgment for each $t \in [2..7]$, which moves them up from $\mathcal{Q}_j$ to $\mathcal{Q}_{j+1}$, as denoted by the arrows (or $\dashv$ for a finalized judgment). Tickets $t \in [2..7]$ are then pending judgment on $\mathcal{J}_1$. (The dotted lines depict $\mathcal{Q}_j.\mathsf{head} \geq \mathcal{Q}_{j+1}.\mathsf{tail}$.)

It means only tickets with smaller identifiers can go to the next (upper) queue. Since they will be larger than those in the upper queue, they will be appended to the end to maintain the intra-queue invariant.

4.2. Multiple Queues in Action

Authentication (enqueuing t to $\mathcal{Q}_0$) works like FARB but across multi-queues, which requires $t_{\mathcal{Q}_j.\mathsf{head}+1} > \mathfrak{jp}_j$, $\forall j \in [N]$. All newly spawned tickets are enqueued to $\mathcal{Q}_0$.

Redeeming in SMART depends on the judgment type: i) redeeming a finalized ticket at $\mathcal{Q}_j$, which removes it from the j-th queue and hence the credential;

ii) redeeming a skipped ticket at $\mathcal{Q}_j$, which dequeues it from $\mathcal{Q}_j$ and enqueues to $\mathcal{Q}_{j+1}$, when $j < N - 1$.

We illustrate how the tickets are moved in credentials, corresponding to the judgments made by the SP as reflected by the multiple global score lists $\mathcal{J}_0, \ldots \mathcal{J}_{N-1}$. Figure 2 focuses on the credential of Alice and Bob, respectively holding $(2, 4, 5, 8, 9)$ and $(1, 3, 6, 7, 10)$ in $\mathcal{Q}_0$.

First, note that when $\mathfrak{jp}_0 = 1$, Alice can authenticate as long as $\mathcal{Q}_0$ is not yet full since $t_{\mathcal{Q}_0.\mathsf{head}+1} = 2 > \mathfrak{jp}_0$.

Suppose the SP skips all tickets $t \in [2..7]$. This can be responding to any of the plausible scenarios below.

i) Those sessions now receive their first transient judgment, *e.g.*, in the time-dependent model, one month has passed, and no negative report has been filed against them.
ii) Those sessions are difficult to judge, but the SP notices that the global list $\mathcal{J}_0$ has been accumulating. To avoid halting all users like in FARB, the SP moves them to $\mathcal{J}_1$.
iii) Similar to the plausible scenario above, the SP further notices that session $t = 8$ is "especially" bad and would like to revoke the user behind as soon as possible.

The SP makes the first transient judgments on $t \in [2..7]$, which moves the global judgment pointer of $\mathcal{J}_0$ from $\mathfrak{jp}_0 = 1$ to $\mathfrak{jp}_0 = 8$. Figure 2 depicts the (projected) movements to $\mathcal{Q}_1$ of the affected tickets. Alice is now forced to redeem $t \in [2..7]$. This is to make $t_{\mathcal{Q}_0.\mathsf{head}+1} > \mathfrak{jp}_0$ or Alice cannot authenticate anymore. (The conditions will be made more precise in Section 4.3.) Likewise, Bob redeems $t \in \{1, 3, 6, 7\}$, moving them from $\mathcal{Q}_0$ to $\mathcal{Q}_1$, which makes $t_{\mathcal{Q}_0.\mathsf{head}+1} = 10 > \mathfrak{jp}_0$. Subjective evaluation can now have flexibility, either judging (the first one in) $\{2, \ldots, 7\} \in \mathcal{J}_1$ or the unjudged ones in $\{9, 10\}$.

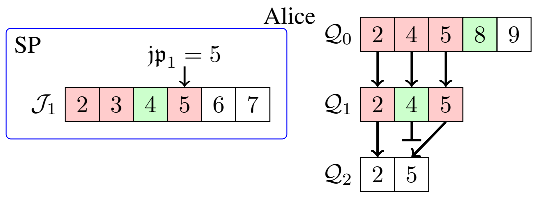


Figure 3. The set of skipped tickets $\{2, 3, 5\}$ continues to be judged one more time, which moves those in user credentials to the next queue $\mathcal{Q}_2$. If $N = 3$, their next judgment must be final for being in the last queue.

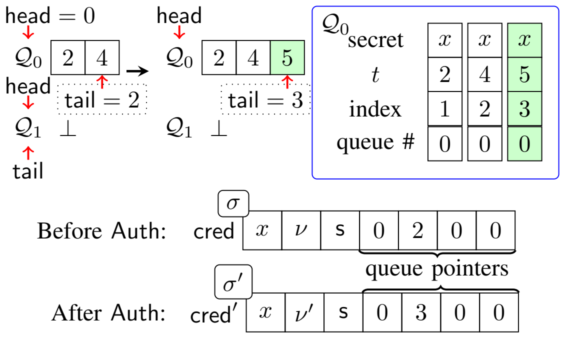


Figure 4. Alice authenticates when $\mathfrak{tc} = 4$. The new ticket $t = 5$ is enqueued to $\mathcal{Q}_0$ at index 3. She obtains cred' with $\mathcal{Q}_0.\mathsf{tail}' = 3$ and a new enqueuing signature (in green).

Figure 3 illustrates subsequent actions after some sessions have been "skipped" to the next global score list $\mathcal{J}_1$, and the corresponding changes to Alice's session queues. Suppose the SP further skips $2, 3, 5$ and finalizes 4 in $\mathcal{J}_1$. Alice's last queue will store tickets $2, 5$ after redemptions.

4.3. Illustrations involving Proof of Signatures

We further illustrate the involvement of signatures in our cryptographic protocols in a simpler example. Suppose Alice has authenticated at $\mathfrak{tc} = 2, 4$, a global ticket counter maintained by the SP. Tickets $\{1, 3\}$ do not belong to her.

4.3.1. Authentication.

`AP` checks for each queue:
i) $\mathcal{Q}_j$ is empty ($\mathcal{Q}_j.\mathsf{head} = \mathcal{Q}_j.\mathsf{tail}$) or
ii) the first ticket at $\mathcal{Q}_j$ is unjudged ($\mathfrak{jp}_j < t_{\mathcal{Q}_j.\mathsf{head}+1}$) and $\mathcal{Q}_j$ is not full ($\mathcal{Q}_j.\mathsf{tail} - \mathcal{Q}_j.\mathsf{head} < K_j$).

In Figure 4, without any judged tickets, Alice proves the credential and the enqueuing signatures $\mathsf{enQ}_x^{\mathcal{Q}_j}$ for the first ticket (if it exists) in each queue j for authentication. The SP then increments $\mathfrak{tc}$ (to 5) and binds a new ticket $t' = \mathfrak{tc} = 5$ to the $(\mathcal{Q}_0.\mathsf{tail} + 1)$-th position (*i.e.*, 3) of $\mathcal{Q}_0$ of her credential via issuing an enqueuing signature on $(x, 5, 3, 0)$, where the commitment of $\mathcal{Q}_0.\mathsf{tail}$ is sent by Alice during the proof of `AP`. Her cred is also updated.

4.3.2. SP Judging.

SP sequentially judges unjudged sessions to appear in $\mathcal{J}_j$, *e.g.*, $t \in [1..5]$ for $\mathcal{J}_0$ in Figure 5. Suppose the SP wants to finalize ticket 1 with score s_1 and revoke 3 with a very negative score s_3 as a non-final judgment. Any unjudged tickets <3 need to be moved to the next queue or finalized. In this case, it is only ticket 2, with a transient score, say, $\mathsf{s}_2 = 0$.

TABLE 1. NOTATIONS

Notation	Description
N, K_i	# of queues, the maximum size of the i-th queue
t	Session/Ticket (identifier)
x, ν, s	Credential secret, nonce, and score
$\mathsf{enQ}_x^{(i,j)} =$	Enqueuing signature of t_i at i in the j-th queue $\mathcal{Q}_j$,
(x, t_i, i, j, σ)	($\mathsf{enQ}_x^{\mathcal{Q}_j}$ denotes t is at the head ($i = \mathcal{Q}_j.\mathsf{head} + 1$))
$\mathcal{Q}_j$	$(\{\mathsf{enQ}_x^{(\mathsf{head}+1,j)}, \ldots, \mathsf{enQ}_x^{(\mathsf{tail},j)}\}, \mathsf{head}, \mathsf{tail})$
$\mathcal{J}_j[t]$	A judged ticket $(t, j, b_{\mathsf{next}}, \mathsf{s}_t, \sigma')$ in score list $\mathcal{J}_j$
$\mathsf{s}^*_{(t,0)}$	Score for a finalized judgment on ticket $(t, 0)$
$\mathbb{T}^{\mathsf{S}}_0/\mathbb{T}^{\mathsf{S}}_j/\mathbb{T}^{\mathsf{F}}$	Global 0-/j-time judged/finalized ticket list
$\mathcal{V}$	The set of used credential nonces
$\mathfrak{tc}$	A counter for the new session/ticket identifier
$\mathfrak{jp}_j$	Judgment pointer for $\mathcal{J}_j$

The SP gives judgment signatures on:
i) $(1, 0, b_{\mathsf{next}} = 0, \mathsf{s}_1)$,
ii) $(2, 0, b_{\mathsf{next}} = 1, \mathsf{s}_2)$, and
iii) $(3, 0, b_{\mathsf{next}} = 1, \mathsf{s}_3)$, respectively.
They are put into the public score list $\mathcal{J}_0$.

The SP also updates the global judgment pointer $\mathfrak{jp}_0$ to 3, meaning that the third and its prior sessions (in $\mathcal{J}_0$) have been judged and must be redeemed (from $\mathcal{Q}_0$ of any credential). This is how SP ensures a user will redeem any judged sessions or the credential cannot pass `AP`.

4.3.3. Redeeming. Alice checks her credential against the score list to see if it has judged tickets ($\mathfrak{jp}_j \geq t_{\mathcal{Q}_j.\mathsf{head}+1}$). If so, she fetches those (by downloading everything) and runs the Redeem protocol on $\mathcal{Q}_j$ with signatures below:
i) $\mathsf{cred} = (x, \nu, \mathsf{s}, \{\mathcal{Q}_j.\mathsf{head}, \mathcal{Q}_j.\mathsf{tail}\}_{j=0}^{N-1})$,
ii) $\mathsf{enQ}_x^{\mathcal{Q}_j} = (x, t, \mathcal{Q}_j.\mathsf{head} + 1, j)$, and
iii) a judgment $(t, j, b_{\mathsf{next}}, \mathsf{s}_t)$.
Upon completion, cred will be updated with:
i) a fresh nonce ν' and a new score $(\mathsf{s} + \mathsf{s}_t)$,
ii) $\mathcal{Q}_j.\mathsf{head}{+}{+}$ (dequeue), and
iii) $\mathcal{Q}_{j+1}.\mathsf{tail} := \mathcal{Q}_{j+1}.\mathsf{tail} {+} b_{\mathsf{next}}$ (enqueue if $b_{\mathsf{next}} = 1$).
An enqueuing signature $(x, t, \mathcal{Q}_{j+1}.\mathsf{tail} + b_{\mathsf{next}}, j + b_{\mathsf{next}})$ will be blindly issued[3] upon redemption on (t, j) too.

In Figure 5, Alice's $\mathcal{Q}_0$ has a transient judgment $(2, 0, b_{\mathsf{next}} = 1, \mathsf{s}_2)$. She thus redeems ticket 2 in $\mathcal{Q}_0$, which dequeues 2 from $\mathcal{Q}_0$ and enqueues it to $\mathcal{Q}_1$, resulting in $\mathcal{Q}_0 = (4, 5)$; $\mathcal{Q}_1 = (2)$. $\mathcal{Q}_0.\mathsf{head}$ advances which voids the old $\mathsf{enQ}_x^{\mathcal{Q}_0} = (x, 2, 1, 0)$ since $\mathcal{Q}_0.\mathsf{head} + 1 > 1$. For $\mathcal{Q}_1$, an enqueuing signature on $(x, 2, 1, 1)$ is issued; $\mathcal{Q}_1.\mathsf{tail}$ is also incremented since $b_{\mathsf{next}} = 1$.

4.3.4. Finalizing a Transient Judgment. The SP now needs to judge $4, 5$, and 2 that will be posted on $\mathcal{J}_0, \mathcal{J}_1$. Suppose the SP finalizes ticket 2 (Figure 6), it posts:
i) a final judgment $(2, 1, b_{\mathsf{next}} = 0, \mathsf{s}_{(2,1)})$ on $\mathcal{J}_1$ and
ii) [optional] a final judgment[4] $(2, 0, 0, \mathsf{s}^*_{(2,0)})$ on $\mathcal{J}_0$.
For example, if an initial score $\mathsf{s}_{(2,0)} = \mathsf{s}_2 = 3$ has been given and a finalized score of $\mathsf{s}^*_{(2,0)} = 10$ is to be given; the SP finalizes it by setting $\mathsf{s}_{(2,1)} = 10 - 3 = 7$. If Alice

3. This makes redeeming a final/transient judgment indistinguishable, except for the last $\mathcal{J}_{N-1}$ having only final judgments.

4. $\mathsf{s}^*_{(t,j)}/\mathsf{s}_{(t,j)}$ distinguish scores for finalized/skipped judgments in the same score list, *e.g.*, ticket 2 in $\mathcal{J}_0$ in Figure 6. It reduces the number of redeeming requests when Alice waits until $(2, 0)$ is finalized.

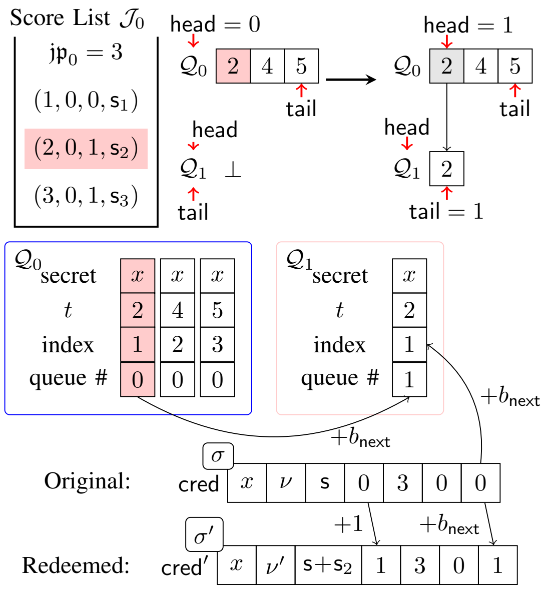
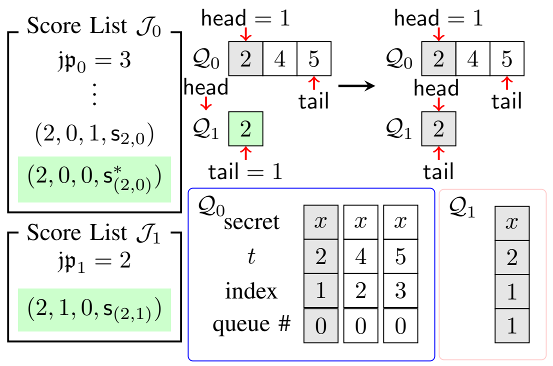


Figure 5. Alice redeems a judgment (red) on $t = 2$ (a transient one with score s_2) via PoK of i) judgment signature on $(2, 0, b_{\mathsf{next}} = 1, \mathsf{s}_2)$ and ii) enqueuing signature $(x, 2, 1, 0)$ (stored in the $\mathcal{Q}_0$ box). $\mathcal{Q}_0.\mathsf{head}$ advances to remove ticket 2 at $\mathcal{Q}_0$ and $\mathcal{Q}_1.\mathsf{tail}$ advances by $b_{\mathsf{next}} = 1$ for accommodating the moved ticket. s becomes $\mathsf{s} + \mathsf{s}_2$.



Score List $\mathcal{J}_0$
$\mathfrak{jp}_0 = 3$
$(2, 0, 1, \mathsf{s}_{2,0})$
$(2, 0, 0, \mathsf{s}^*_{(2,0)})$
Score List $\mathcal{J}_1$
$\mathfrak{jp}_1 = 2$
$(2, 1, 0, \mathsf{s}_{(2,1)})$
head = 1
$\mathcal{Q}_0$
head
tail
$\mathcal{Q}_1$
tail = 1
secret
t
index
queue #

Figure 6. Alice redeems a finalized ticket 2 at $\mathcal{Q}_1$ with $(2, 1, 0, s_{(2,1)})$. $\mathcal{Q}_1.\mathsf{tail}$ advances by $b_{\mathsf{next}} = 0$, *i.e.*, unchanged. $\mathcal{Q}_1.\mathsf{head}$ advances to $\mathcal{Q}_1.\mathsf{tail}$, so $\mathcal{Q}_1$ is empty.

absorbs $s_{(2,0)}, s_{(2,1)}$ via two updates in Figures 5 and 6, they sum up to 10. The SP also updates $\mathfrak{jp}_1 = 2$ to indicate all tickets $t \in \mathcal{J}_1$, $t \leq 2$ are judged (ticket 1 is finalized at $\mathcal{J}_0$ and thus will not appear in subsequent score lists).

4.3.5. Redeeming a Finalized Judgment. With new $\mathfrak{jp}_1$, Alice needs to redeem judgment $(2, 1, 0, \mathsf{s}_{(2,1)})$ with $\mathsf{enQ}_x^{\mathcal{Q}_1} = (x, 2, 1, 1)$ since her $\mathcal{Q}_1$ contains a judged ticket 2. (Alternatively, if she waits until ticket 2 is finalized, *cf.* Figure 5, she uses $(2, 0, 0, s^*_{(2,0)})$ to redeem and no ticket will be enqueued to $\mathcal{Q}_1$.) After redeeming, $\mathcal{Q}_0 = (4, 5)$; $\mathcal{Q}_1 = (\perp)$, with $\mathcal{Q}_1.\mathsf{head} = \mathcal{Q}_1.\mathsf{tail} = 1$ in cred'. Thus, Alice can prove tickets in $\mathcal{Q}_0$ are unjudged ($t_{\mathcal{Q}_0.\mathsf{head}+1} > \mathfrak{jp}_0$) and $\mathcal{Q}_1$ is empty ($\mathcal{Q}_1.\mathsf{head} = \mathcal{Q}_1.\mathsf{tail}$). Figure 6 illustrates the resulting changes.

5. SMART Credential Construction

5.1. Overview of Actions taken by Users/SP

5.1.1. User's Actions. A user registers with the SP to obtain a signature encoding a starting credential. It has all empty queues, so no enqueuing signature is issued.

To authenticate, the user first checks its credential with the score lists for judged tickets. If there are judged tickets, the user redeems them prior to actual authentication.

The user redeems with a valid credential, an enqueuing signature, and a judgment on (t, j), where t is the first ticket at $\mathcal{Q}_j$. We assume the user redeems one ticket at each time and j is revealed in the baseline construction.

When $j < N$, the redeemed ticket would be moved to the next queue, so the user should receive a (valid) new enqueuing signature on $(x, t, \mathcal{Q}_{j+1}.\mathsf{tail} + 1, j + 1)$.

5.1.2. SP's Actions. The SP setups the system parameters, *e.g.*, the number of queues N and their bounds $\vec{K}$. During authentication and redemption, the revealed nonce is stored in $\mathcal{V}$, and a new ticket (ID) generated in Auth is added to the public state st. The SP can issue transient or finalized judgments on tickets via StMgn to update st (which consists of score lists and pointers $\{(\mathcal{J}_j, \mathfrak{jp}_j)\}_{j=0}^{N-1}$).

5.2. Setup

The SP calls $\mathsf{Setup}(1^\lambda)$ to run the following steps.

1. Determine the following system parameters appropriate for the application scenario:
   - N: the number of queues;
   - $\vec{K} \in \mathbb{N}^N$ (or $\{K_j\}_{j=0}^{N-1}$): the maximum number of sessions in each of the N queues in a credential;
   - $\mathsf{s}^{\mathsf{min}}$: the minimum score for the authentication policy $\mathtt{AP}$. (SMART could be easily extended to cover ℓ score categories.)
2. Initialize the following sets to be an empty set $\emptyset$:
   - $\mathcal{V}$: the set of credential nonces that the SP has seen in user authentication and redeeming;
   - $\{\mathcal{J}_i\}_{i\in[N]}$: the set of judgments that have been skipped/(temporarily-)judged for i times.
   - $\{\mathbb{T}_j^{\mathsf{S}}\}_{j\in[N]}, \mathbb{T}^{\mathsf{F}}$: the queues of tickets that are judged j times, and the queue of finalized tickets.
3. Initialize the following values to be 0:
   - $\{\mathfrak{jp}_i\}$: global judgment pointers indicating the last judged tickets of the i-th score list $\mathcal{J}_i$;
   - $\mathfrak{tc}$: a ticket counter indicating the last session ID assigned to an authenticated session.
4. Setup the cryptographic systems:
   - $\mathsf{crs} \leftarrow \mathsf{ZKP.Setup}(1^\lambda)$: the common reference string for the zero-knowledge proof/argument;
   - $\{(\mathsf{pk}_k, \mathsf{sk}_k) \leftarrow \mathsf{Cred.KGen}(1^\lambda)\}_{k\in\{\mathsf{c,t,j}\}}$: the key pair(s) of the credential signature scheme for signing tuples of credentials, tickts, and judgments.
5. Output:
   - $\mathsf{pp} := (\mathsf{crs}, \{\mathsf{pk}_k\}_{k\in\{\mathsf{c,t,j}\}}, N, \mathsf{s}^{\mathsf{min}})$;
   - $\mathsf{st} := (\mathfrak{tc}, \mathbb{T}_0^{\mathsf{S}}, \{\mathbb{T}_j^{\mathsf{S}}\}, \mathbb{T}^{\mathsf{F}}, \mathcal{V}, \{K_i, \mathcal{J}_i, \mathfrak{jp}_i\})$, where $j \in [N]$ and $i \in [N]$.
   - Return $(\mathsf{pp}, (\mathsf{sk_c}, \mathsf{sk_t}, \mathsf{sk_j}), \mathsf{st})$.

```
$\mathsf{Reg}(\mathcal{U}(\mathsf{pp}), \mathcal{S}(\mathsf{pp}, \mathsf{sk}))$
$\mathcal{U} : x, \nu \leftarrow\$ \mathbb{Z}_p, \mathsf{s} := 0$
    $\mathbf{attr} := (x, \nu, \mathsf{s}, \{\mathcal{Q}_i.\mathsf{head}, \mathcal{Q}_i.\mathsf{tail}\}_{i\in[N]})$
$\mathcal{U} \leftrightarrow \mathcal{S}$ : run $\mathsf{Cred.Sig}(\mathcal{U}(\mathbf{attr}; \rho), \mathcal{S}(\mathsf{pp}, \mathsf{sk_c})) \to (\sigma, \perp)$
  / the user-chosen secret and nonce are hidden in a commitment
$\mathcal{U}$ : return $(\mathsf{cred} := (\mathbf{attr}, \sigma))$
$\mathcal{S}$ : return st
```

Figure 7. Registration Protocol

```
$\mathtt{AP}(\mathsf{cred})$
parse $\mathsf{pk_c}, \mathsf{pk_t}, \mathsf{pk_j}, \{\mathfrak{jp}_i, K_i\}_{i\in[N]}$ from pp, st
parse $(x, \nu, \mathsf{s}, \{\mathcal{Q}_i.\mathsf{head}, \mathcal{Q}_i.\mathsf{tail}\}_{i\in[N]}, \sigma)$ from cred
$\mathbf{attr} := (x, \nu, \mathsf{s}, \{\mathcal{Q}_i.\mathsf{head}, \mathcal{Q}_i.\mathsf{tail}\}_{i\in[N]})$
parse $\{\mathsf{enQ}_x^{\mathcal{Q}_i} = (x, t_{\mathcal{Q}_i.\mathsf{head}+1}, \mathcal{Q}_i.\mathsf{head} + 1, i, \sigma_x^i)\}_{i\in[N]}$
if $(\mathsf{Cred.Vf}(\mathsf{pk_c}, \mathbf{attr}, \sigma) = 0)$ then return 0
if $(\mathsf{s} < \mathsf{s}^{\mathsf{min}})$ then return 0   / minimum score
for $i$ in $[N]$
  if $(\mathcal{Q}_i.\mathsf{tail} = \mathcal{Q}_i.\mathsf{head})$ then continue   / $\mathcal{Q}_i$ is empty
  if $(t_{\mathcal{Q}_i.\mathsf{head}+1} \le \mathfrak{jp}_i)$ then return 0   / unjudged first ticket
  if $(\mathcal{Q}_i.\mathsf{tail} - \mathcal{Q}_i.\mathsf{head} \ge K_i)$ then return 0
  if $(\mathsf{Cred.Vf}(\mathsf{pk_t}, \mathsf{enQ}_x^{\mathcal{Q}_i}) = 0)$ then return 0
endfor
return 1
```

Figure 8. Authentication Policy Checking

5.3. Registration

Figure 7 presents the registration protocol. The initial attributes of the user credential are set to $\mathbf{attr} = (x, \nu, \mathsf{s}, \{(\mathcal{Q}_j.\mathsf{head}, \mathcal{Q}_j.\mathsf{tail})\}_{j\in[N]})$, with initial score s and N empty queues $(\mathcal{Q}_j.\mathsf{head}, \mathcal{Q}_j.\mathsf{tail}) = (0, 0)$.

The user engages in the signature issuance protocol of Cred with the SP, proving the commitment of attributes (using some random ρ) is well-formed, *i.e.*, everything in $\mathbf{attr}$ other than x and ν is 0. The secret x can be jointly sampled by the user and SP, or related to a long-term user public key, *e.g.*, g^x. The user then gets a credential cred.

5.4. Authentication

To authenticate, the user credential must pass the authentication policy $\mathtt{AP}$ in Figure 8 on the following:

1) Main credential cred is valid with a fresh nonce ν.
2) The scores satisfy[5] $\mathsf{s}^{\mathsf{min}}$ announced by the SP.
3) Each queue i has no unredeemed tickets, *i.e.*, $\mathcal{Q}_i$ is empty (via a simple equality check), **or** the first ticket at $\mathcal{Q}_i$ is unjudged (via the enqueuing signature and a range proof against $\mathfrak{jp}_i$).
4) Each queue i has not reached its full capacity K_i.

If it passes, the parties run the protocol in Figure 9, *i.e.*, $\mathsf{Auth}(\mathcal{U}(\mathsf{pp}, \mathsf{cred}, \mathtt{AP}, f), \mathcal{S}(\mathsf{pp}, \mathsf{sk}, \mathtt{AP}, f))$, as follows.

5. We use $\mathsf{s} \ge \mathsf{s}^{\mathsf{min}}$ to denote score satisfaction. The policy can be all categories' scores or weighted some of them being above the threshold.

```
$\mathsf{Auth}(\mathcal{U}(\mathsf{pp}, \mathsf{cred}, \mathtt{AP}, f), \mathcal{S}(\mathsf{pp}, \mathsf{sk}, \mathtt{AP}, f))$
$\mathcal{U}$ : parse $\mathsf{cred} := (\mathbf{attr}, \sigma, \{\mathsf{enQ}_x^{\mathcal{Q}_i}\})$
    parse $\mathbf{attr} := (x, \nu, \mathsf{s}, \{\mathcal{Q}_i.\mathsf{head}, \mathcal{Q}_i.\mathsf{tail}\}_{i \in [N]})$
    if $!\mathtt{AP}(\mathsf{cred})$ then return $\bot$
    $\pi \leftarrow \mathsf{ZKP.P}(\mathsf{pp}, (\mathtt{AP}, \mathsf{st}), \mathsf{cred})$
    $\mathbf{attr}' \leftarrow f(\mathsf{cred})$
$\mathcal{U} \rightarrow \mathcal{S} : (\pi, \nu)$
$\mathcal{S}$ : if $!(\mathsf{ZKP.V}(\mathsf{pp}, (\mathtt{AP}, \mathsf{st}), \pi) \wedge (\nu \notin \mathcal{V}))$ return $\bot$
    $\mathcal{V} := \mathcal{V} \cup \{\nu\}, \mathfrak{tc} := \mathfrak{tc} + 1, \mathbb{T}_0^{\mathsf{S}} := \mathbb{T}_0^{\mathsf{S}} \cup \{\mathfrak{tc}\}$
$\mathcal{U} \leftrightarrow \mathcal{S}$ :   / Run in parallel
  run $\mathsf{Cred.Sig}(\mathcal{U}(\mathbf{attr}'; \rho), \mathcal{S}((\pi, f), \mathsf{sk_c})) \rightarrow (\sigma, \bot)$
  run $\mathsf{Cred.Sig}(\mathcal{U}((x, \mathfrak{tc}, \mathcal{Q}_0.\mathsf{tail} + 1, 0); \rho'),$
                    $\mathcal{S}((\pi, f), \mathsf{sk_t})) \rightarrow (\sigma_1, \bot)$
$\mathcal{S}$ : return $\mathsf{st}'$   / Updated $\mathcal{V}$, $\mathfrak{tc}$, $\mathbb{T}_0^{\mathsf{S}}$
$\mathcal{U}$ : $\mathsf{enQ}'_x := (x, \mathfrak{tc}, \mathcal{Q}_0.\mathsf{tail} + 1, 0, \sigma_1)$
$\mathcal{U}$ : return $\mathsf{cred}' := (\mathbf{attr}', \sigma; \mathsf{enQ}'_x)$
```

Figure 9. Authentication Protocol

```
$f(\mathsf{cred})$ for Auth
parse $\mathbf{attr} := (x, \nu, \mathsf{s}, \{\mathcal{Q}_i.\mathsf{head}, \mathcal{Q}_i.\mathsf{tail}\}_{i \in [N]})$
$\nu' \leftarrow\!\!\$\ \mathbb{Z}_p$
$\mathbf{attr}' := (x, \nu', \mathsf{s}, \mathcal{Q}_0.\mathsf{head}, \mathcal{Q}_0.\mathsf{tail} + 1, \ldots, \mathcal{Q}_{N-1}.\mathsf{tail})$
$\mathsf{enQ}' := (x, \mathfrak{tc}, \mathcal{Q}_0.\mathsf{tail} + 1, 0)$   / where $\mathfrak{tc}$ is from st
return $\mathsf{cred}' := (\mathbf{attr}'; \mathsf{enQ}')$
```

Figure 10. Credential Updating in Auth

1. The user generates ZKP π for satisfying $\mathtt{AP}$:

$$\mathsf{ZKP}\left\{\begin{array}{l} (\mathsf{cred}, \{\mathsf{enQ}_x^{\mathcal{Q}_i}\}_{i=1}^{N}, \nu') : \\ (C_x = \mathsf{Com}(x)) \ \wedge \ (C_{\mathsf{s}} = \mathsf{Com}(\mathsf{s})) \\ \ \wedge \ (C_{\nu'} = \mathsf{Com}(\nu')) \\ \bigwedge\limits_{i \in [N]} \left\{\begin{array}{rl} C_{\mathcal{Q}_i.\mathsf{head}} & = \mathsf{Com}(\mathcal{Q}_i.\mathsf{head}) \\ \wedge\, C_{\mathcal{Q}_i.\mathsf{tail}} & = \mathsf{Com}(\mathcal{Q}_i.\mathsf{tail}) \\ \wedge\, C_{t_{\mathcal{Q}_i.\mathsf{head}}} & = \mathsf{Com}(t_{\mathcal{Q}_i.\mathsf{head}}) \end{array}\right\} \\ \wedge \ (\mathsf{Cred.Vf}(\mathsf{pk_c}, \mathsf{cred}) = 1) \ \wedge \ (\mathsf{s} \geq \mathsf{s}^{\mathsf{min}}) \\ \bigwedge\limits_{i \in [N]} \left\{\begin{array}{l} (\mathcal{Q}_i.\mathsf{head} = \mathcal{Q}_i.\mathsf{tail}) \vee \\ \left(\begin{array}{l} (\mathcal{Q}_i.\mathsf{tail} - \mathcal{Q}_i.\mathsf{head} \in [0, K_i - 1]) \\ \wedge\, (t_{\mathcal{Q}_i.\mathsf{head}+1} > \mathfrak{jp}_i) \\ \wedge\, \mathsf{Cred.Vf}(\mathsf{pk_t}, \mathsf{enQ}_x^{\mathcal{Q}_i}) = 1 \end{array}\right) \end{array}\right\} \end{array}\right\}.$$

The statement involves the opening of (individually) committed attributes in cred, the new nonce ν', the enqueuing signatures $\mathsf{enQ}_x^{\mathcal{Q}_i}$, and the score passes the threshold $\mathsf{s}^{\mathsf{min}}$. The implicit queue design and the intra-queue order invariant allow checking only the first ticket (knowledge of $\mathsf{enQ}_x^{\mathcal{Q}_i}$) in $\mathcal{Q}_i$ with $\mathfrak{jp}_i$ or $\mathcal{Q}_i.\mathsf{head} = \mathcal{Q}_i.\mathsf{tail}$ (for an empty $\mathcal{Q}_i$) via an OR-composition proof. The user also proves the queue size is within the public bound K_i.

2. The user sends the nonce ν and above ZKP π to the SP, which proves that its credential satisfies the policy.

3. The SP confirms that $\nu \notin \mathcal{V}$ and the validity of π. Then it updates $\mathcal{V} := \mathcal{V} \cup \{\nu\}$, the ticket counter $\mathfrak{tc} := \mathfrak{tc} + 1$, and the unjudged ticket set $\mathbb{T}_0^{\mathsf{S}} := \mathbb{T}_0^{\mathsf{S}} \cup \{\mathfrak{tc}\}$.

4. The user prepares the update as $f(\mathsf{cred})$ in Figure 10:

```
Commitments for Signing $f(\mathsf{cred})$, $C_1 \leftarrow \mathsf{Com}(1)$
$C_{\mathbf{attr}'} := (C_x, C_{\nu'}, C_{\mathsf{s}}, C_{\mathcal{Q}_0.\mathsf{head}}, C_{\mathcal{Q}_0.\mathsf{tail}} \cdot C_1, \ldots, C_{\mathcal{Q}_{N-1}.\mathsf{tail}})$
$C_{\mathsf{enQ}} := (C_x, C_{\mathfrak{tc}}, C_{\mathcal{Q}_0.\mathsf{tail}} \cdot C_1, C_0)$   / $C_0 \leftarrow \mathsf{Com}(0)$
```

Figure 11. SP Computation of $f(\mathsf{cred})$ in Auth

- randomly picks a new nonce $\nu' \leftarrow\!\!\$\ \mathbb{Z}_p$;
- increments $\mathcal{Q}_0.\mathsf{tail}$ in $\mathbf{attr}'$ for adding a slot for the new ticket at the end of the 0-th queue;
- prepares the enqueuing data for ticket ID $\mathfrak{tc}$ at the i-th slot, $i = \mathcal{Q}_0.\mathsf{tail} + 1$, of the 0-th queue.

5. The parties reuse commitments from ZKP π (namely, $(C_x, C_\nu, C_{\mathsf{s}}, \{C_{\mathcal{Q}_i.\mathsf{head}}, C_{\mathcal{Q}_i.\mathsf{tail}}\}_{i \in [N]})$ for the old credential and $C_{\nu'}$ for the new nonce) to invoke in parallel 2 credential-signature issuance instances over commitments $C_{\mathbf{attr}'}$ and C_{enQ} for SP's signatures on $\mathbf{attr}'$ and enQ'. $C_{\mathbf{attr}'}$ and C_{enQ} can be created as in Figure 11.

The user stores the two obtained signatures: the updated credential signature σ on new attributes $\mathbf{attr}'$ and the enqueuing signature for enQ'_x, which enqueues $t = \mathfrak{tc}$.

6. The SP updates state st to st$'$ with new $\mathfrak{tc}$, $\mathbb{T}_0^{\mathsf{S}}$, and $\mathcal{V}$.

5.5. Session Judging and Queue Management

A successful authentication appends a new ticket to $\mathbb{T}_0^{\mathsf{S}}$, the unjudged ticket set. In StMgn, the SP puts judgments into $\{\mathcal{J}_j\}_{j \in [N]}$ for $\mathbb{T}_0^{\mathsf{S}}$, j-time judged sets $\{\mathbb{T}_j^{\mathsf{S}}\}_{j \in [1..N-1]}$, and the finalized ticket list $\mathbb{T}^{\mathsf{F}}$, all in st. (In Figure 2, $\mathbb{T}_0^{\mathsf{S}} = (9, 10), \mathbb{T}_1^{\mathsf{S}} = \emptyset, \mathbb{T}_2^{\mathsf{S}} = (2, \ldots, 7)$.)

With the sequential judgment of each queue, the SP should judge tickets at the heads of $\{\mathbb{T}_j^{\mathsf{S}}\}$ (*e.g.*, $2, 9$ in Figure 2). For a transient (or final) judgment of a ticket t, the SP posts it to score list $\mathcal{J}_j$, moves t to $\mathbb{T}_{j+1}^{\mathsf{S}}$ (or $\mathbb{T}^{\mathsf{F}}$), and advances judgment pointer $\mathfrak{jp}_j$ to t. Specifically,

1. Transient judgment moves t from $\mathbb{T}_j^{\mathsf{S}}$ to $\mathbb{T}_{j+1}^{\mathsf{S}}$ and puts a signature on $(t, j, b_{\mathsf{next}} = 1, \mathsf{s}_{(t,j)})$ to $\mathcal{J}_j$.

2. Final judgment moves t from $\mathbb{T}_j^{\mathsf{S}}$ to $\mathbb{T}^{\mathsf{F}}$ and puts a signature on $(t, j, b_{\mathsf{next}} = 0, \mathsf{s}_{(t,j)})$ to $\mathcal{J}_j$ and a signature on $(t, 0, 0, \mathsf{s}^*_{(t,0)})$ to $\mathcal{J}_0$, where $\mathsf{s}_{(t,j)}$ is computed using previous transient scores, *e.g.*, $\mathsf{s}_{(t,j)} = \mathsf{s}^*_{(t,0)} - \sum_{i=0}^{j-1} \mathsf{s}_{(t,j)}$.

As an example, a ticket t that is skipped twice goes through $\mathbb{T}_0^{\mathsf{S}}$ (unjudged), $\mathbb{T}_1^{\mathsf{S}}$ and $\mathbb{T}_2^{\mathsf{S}}$ (2 transient judgments in $\mathcal{J}_0, \mathcal{J}_1$). To finalize t, $(t, 2)$ is added to $\mathbb{T}^{\mathsf{F}}$ where the SP judges $(t, 0), (t, 2)$ with $b_{\mathsf{next}} = 0$ and adjusted scores.

5.6. Session Redeeming or Credential Updating

When a valid credential has $t \in \mathbb{T}_j^{\mathsf{S}}$ in its $\mathcal{Q}_j$. the user needs to redeem judged tickets in every queue j before the next authentication. Redeeming a ticket updates the credential. Like enqueuing a new ticket by Auth, the credential (and other signatures) is updated by obtaining a signature on the committed multi-block message.

The user can redeem $t \in \mathcal{Q}_j$ by revealing j or keeping j private. They have different update functions f.

5.6.1. Redeeming a Ticket in a Known Queue.

Redeem in Figure 13 proceeds mostly like the steps in Auth.

1. When j is public, the user checks if the credential satisfies the specific redeeming policy $\mathtt{UP}^j$ parameterized

by j in Figure 12, *i.e.*, eligible to redeem $t_{\mathcal{Q}_j.\mathsf{head}+1} \in \mathcal{Q}_j$. If so, the user proves the ZKP below by proving the knowledge of signatures on each of the following:
i) $\mathsf{cred} = ((x, \nu, \mathsf{s}, \{\mathcal{Q}_i.\mathsf{head}, \mathcal{Q}_i.\mathsf{tail}\}_{i\in[N]}), \sigma)$,
ii) $\mathsf{enQ}_x^{\mathcal{Q}_j} = ((x, t_{\mathcal{Q}_j.\mathsf{head}+1}, \mathcal{Q}_j.\mathsf{head}+1, j), \sigma_x^j)$, and
iii) $\mathcal{J}_j[t_{\mathcal{Q}_j.\mathsf{head}+1}] = ((t_{\mathcal{Q}_j.\mathsf{head}+1}, j, b_{\mathsf{next}}, \mathsf{s}_t), \sigma'')$ for:

$$\mathsf{ZKP}\left\{\begin{array}{l}(\mathsf{cred}, \mathsf{enQ}_x^{\mathcal{Q}_j}, \mathcal{J}_j[t], \nu') : \\ C_x = \mathsf{Com}(x) \ \wedge \ C_{\nu'} = \mathsf{Com}(\nu') \wedge \\ C_j = \mathsf{Com}(j) \ \wedge C_{\mathsf{s}} = \mathsf{Com}(\mathsf{s}) \wedge \\ \left\{\begin{array}{l} C_{\mathcal{Q}_i.\mathsf{head}} = \mathsf{Com}(\mathcal{Q}_i.\mathsf{head}) \ \wedge \\ C_{\mathcal{Q}_i.\mathsf{tail}} = \mathsf{Com}(\mathcal{Q}_i.\mathsf{tail}) \end{array}\right\}_{i\in[N]} \wedge \\ C_{\mathsf{s}_t} = \mathsf{Com}(\mathsf{s}_t) \ \wedge \ C_{b_{\mathsf{next}}} = \mathsf{Com}(b_{\mathsf{next}}) \ \wedge \\ \mathsf{Cred.Vf}(\mathsf{pk_c}, \mathsf{cred}) = 1 \wedge \\ \mathsf{Cred.Vf}(\mathsf{pk_t}, \mathsf{enQ}_x^{\mathcal{Q}_j}) = 1 \wedge \\ \mathsf{Cred.Vf}(\mathsf{pk_j}, (t, j, b_{\mathsf{next}}, \mathsf{s}_t), \sigma'') = 1 \end{array}\right\}.$$

Secret x and $\mathcal{Q}_j.\mathsf{head}$ in cred link to those in $\mathsf{enQ}_x^{\mathcal{Q}_j}$, which contains the ticket ID $t = t_{\mathcal{Q}_j.\mathsf{head}+1}$ and queue index j that judgment $\mathcal{J}_j[t]$ is for.

2. The user sends the proof π and current credential nonce ν to the SP, whom confirms $\nu \notin \mathcal{V}$ and π is valid. If so, the SP updates the nonce set $\mathcal{V} \cup \{\nu\}$.

3. The parties also invoke in parallel 2 credential-signature issuance instances with the individual homomorphic commitments in ZKP π as auxiliary inputs to compute updated $\mathbf{attr}' \leftarrow f(\mathsf{cred})$ as in Figure 14.

- samples a new nonce $\nu' \leftarrow \mathbb{Z}_p$;
- absorbs s_t by updating s as $\mathsf{s} + \mathsf{s}_t$;
- increments $\mathcal{Q}_j.\mathsf{head}$ which dequeues $\mathcal{Q}_j$;
- adds b_{next} to $\mathcal{Q}_{j+1}.\mathsf{tail}$ in $\mathbf{attr}'$ (*i.e.*, adding a slot if $b_{\mathsf{next}} = 1$);
- prepares the data to enqueue $t_{\mathcal{Q}_j.\mathsf{head}+1}$ to the $(i = \mathcal{Q}_{j+1}.\mathsf{tail}+b_{\mathsf{next}})$-th slot of the $(j+b_{\mathsf{next}})$-th queue.

All the involved terms, including $\mathcal{Q}_{j+1}.\mathsf{tail}$, are from the credential ($b_{\mathsf{next}}, \mathsf{s}_t$ are from the judgment signature) and have been proven by ZKP already. The SP computes the commitment of $\mathbf{attr}'$ as in Figure 15 (which also lists the individual commitments from π) for blind signature issuance. Here, the SP can pick the $\mathcal{Q}_j$ head and $\mathcal{Q}_{j+1}$ tail from $C_{\mathbf{attr}}$ and update accordingly by adding $C_1 \leftarrow \mathsf{Com}(1)$ and $C_{b_{\mathsf{next}}}$ since j is revealed. The user receives:

- an updated credential cred' on $\mathbf{attr}' \leftarrow f(\mathsf{cred})$: $(x, \nu', \mathsf{s} + \mathsf{s}_t, \ldots, \mathcal{Q}_j.\mathsf{head} + 1, \ldots, \mathcal{Q}_{j+1}.\mathsf{tail} + b_{\mathsf{next}}, \ldots, \mathcal{Q}_{N-1}.\mathsf{tail})$;
- an enqueuing signature on $(x, t_{\mathcal{Q}_j.\mathsf{head}+1}, \mathcal{Q}_{j+1}.\mathsf{tail} + b_{\mathsf{next}}, j + b_{\mathsf{next}})$.

A transient judgment has $b_{\mathsf{next}} = 1$; or 0 if final. A user redeeming a transient judgment obtains a new enqueueing signature on the redeemed ticket binding to $\mathcal{Q}_{j+1}.\mathsf{tail}+1$ and index $j+1$. (In Figure 5, after redeeming a transient judgment on ticket $t = 2$ at $\mathcal{Q}_0$, Alice gets a enqueuing signature on $(x, 2, 1, 1)$ for $t = 2$ at $(\mathcal{Q}_1.\mathsf{tail} + 1)$.)

On the other hand, the enqueuing signature obtained for redeeming a finalized judgment will be "outdated" for any queues since it binds t to the current queue j with index $\mathcal{Q}_{j+1}.\mathsf{tail} < \mathcal{Q}_j.\mathsf{head}+1$ (the inter-queue invariant), *e.g.*, the (invalid) enqueuing signature Alice obtains in

```
$\mathtt{UP}^j(\mathsf{cred})$
parse $(\mathsf{pk_c}, \mathsf{pk_t}, \mathsf{pk_j}), (\mathfrak{jp}_j, \mathcal{J}_j)$ from $\mathsf{pp}, \mathsf{st}$
parse $(\{\mathcal{Q}_j.\mathsf{head}, \mathcal{Q}_j.\mathsf{tail}\}_{j\in[N]})$ from cred
parse $(x, t_{\mathcal{Q}_j.\mathsf{head}+1}, \mathcal{Q}_j.\mathsf{head}+1, j, \sigma_x^j)$ from cred
$\mathsf{enQ}_x^{\mathcal{Q}_j} := (x, t_{\mathcal{Q}_j.\mathsf{head}+1}, \mathcal{Q}_j.\mathsf{head}+1, j, \sigma_x^j)$
parse $\mathcal{J}_j[t_{\mathcal{Q}_j.\mathsf{head}+1}]$ from $\mathcal{J}_j$   / returns $\perp$ if not exist
if $(\mathsf{Cred.Vf}(\mathsf{pk_c}, \mathsf{cred}) = 0)$ then return 0
if $(\mathsf{Cred.Vf}(\mathsf{pk_t}, \mathsf{enQ}_x^j) = 0)$ then return 0
if $(\mathsf{Cred.Vf}(\mathsf{pk_j}, \mathcal{J}_j[t_{\mathcal{Q}_j.\mathsf{head}+1}]) = 0)$ then return 0
return 1
```

Figure 12. Redeeming/Update Policy Checking

```
$\mathsf{Redeem}(\mathcal{U}(\mathsf{pp}, \mathsf{cred}, \mathtt{UP}^j, f), \mathcal{S}(\mathsf{pp}, \mathsf{sk}, \mathtt{UP}^j, f))$
$\mathcal{U}$ : parse $\mathsf{cred} := (\mathbf{attr}, \sigma, \{\mathsf{enQ}_x^{\mathcal{Q}_j}\})$
    parse $\mathbf{attr} := (x, \nu, \mathsf{s}, \{\mathcal{Q}_i.\mathsf{head}, \mathcal{Q}_i.\mathsf{tail}\}_{i\in[N]})$
    parse $\mathcal{J}_j[t_{\mathcal{Q}_j.\mathsf{head}+1}] := (t, j, b_{\mathsf{next}}, \mathsf{s}_t, \sigma'')$ from $\mathcal{J}_j$
    if $!\mathtt{UP}^j(\mathsf{cred})$ then return $\perp$
    $\pi \leftarrow \mathsf{ZKP.P}(\mathsf{pp}, (\mathtt{UP}^j, \mathsf{st}), \mathsf{cred})$
    $\mathbf{attr}' \leftarrow f(\mathsf{cred})$
$\mathcal{U} \rightarrow \mathcal{S} : (\pi, \nu)$
$\mathcal{S}$ : if $!(\mathsf{ZKP.V}(\mathsf{pp}, (\mathtt{UP}^j, \mathsf{st}), \pi) \wedge (\nu \notin \mathcal{V}))$ then
     return $\perp$; else $\mathcal{V} := \mathcal{V} \cup \{\nu\}$
$\mathcal{U} \leftrightarrow \mathcal{S}$ :   / Run in parallel, using $(\mathsf{Com}(\mathcal{Q}_{j+1}.\mathsf{tail}), \mathsf{Com}(b_{\mathsf{next}}))$ in $\pi$
   run $\mathsf{Cred.Sig}(\mathcal{U}(\mathbf{attr}'; \rho), \mathcal{S}((\pi, f), \mathsf{sk_c})) \rightarrow (\sigma, \perp)$
   run $\mathsf{Cred.Sig}(\mathcal{U}((x, t_{\mathcal{Q}_j.\mathsf{head}+1}, \mathcal{Q}_{j+1}.\mathsf{tail} + b_{\mathsf{next}},$
         $j + b_{\mathsf{next}}); \rho'), \mathcal{S}((\pi, f), \mathsf{sk_t})) \rightarrow (\sigma', \perp)$
$\mathcal{S}$ : return $\mathsf{st}'$   / Updated $\mathcal{V}$
$\mathcal{U}$ : $\mathsf{enQ}'_x := (x, t_{\mathcal{Q}_j.\mathsf{head}+1}, \mathcal{Q}_{j+1}.\mathsf{tail}+b_{\mathsf{next}}, j+b_{\mathsf{next}}, \sigma')$
    return $\mathsf{cred}' := (\mathbf{attr}', \sigma; \mathsf{enQ}'_x)$
```

Figure 13. Redeeming Protocol

```
$f(\mathsf{cred})$ for Redeem
parse $\mathbf{attr} := (x, \nu, \mathsf{s}, \{\mathcal{Q}_i.\mathsf{head}, \mathcal{Q}_i.\mathsf{tail}\}_{i\in[N]})$
parse $\mathsf{enQ}_x^{\mathcal{Q}_j} = (x, t_{\mathcal{Q}_j.\mathsf{head}+1}, \mathcal{Q}_j.\mathsf{head}+1, j)$
parse $\mathcal{J}_j[t_{\mathcal{Q}_j.\mathsf{head}+1}] := (t_{\mathcal{Q}_j.\mathsf{head}+1}, j, b_{\mathsf{next}}, \mathsf{s}_t)$
$\nu' \leftarrow\$ \mathbb{Z}_p$
$\mathbf{attr}' := (x, \nu', \mathsf{s} + \mathsf{s}_t, \ldots, \mathcal{Q}_j.\mathsf{head} + 1,$
       $\ldots, \mathcal{Q}_{j+1}.\mathsf{tail} + b_{\mathsf{next}}, \ldots \mathcal{Q}_{N-1}.\mathsf{tail})$
$\mathsf{enQ}' := (x, t_{\mathcal{Q}_j.\mathsf{head}+1}, \mathcal{Q}_{j+1}.\mathsf{tail} + b_{\mathsf{next}}, j + b_{\mathsf{next}})$
return $\mathsf{cred}' := (\mathbf{attr}'; \mathsf{enQ}')$
```

Figure 14. Credential Updating in Redeem

Figure 6 will be $(x, 2, 0, 1)$, where $i < \mathcal{Q}_1.\mathsf{head} + 1 = 1$, assuming $\mathcal{Q}_2$ is empty. Knowing j is the last queue, the signature issuance protocol needs not to be run.

4. The user stores the updated credential signature σ on the new attributes $\mathbf{attr}'$ and the enqueuing signature on $t_{\mathcal{Q}_j.\mathsf{head}+1}$ (if $b_{\mathsf{next}} = 0$, the user can discard the signature).

5. The SP updates its state st with the new nonce set $\mathcal{V}$.

5.6.2. Hiding the Queue Index.

If j is also hidden in a commitment, the *user*, instead of the SP, will compute the

```
Commitments for Signing $f(\mathsf{cred})$
$\textbf{parse from } \pi : C_{\nu'},$
    $(C_x, C_\nu, C_\mathsf{s}, \{C_{\mathcal{Q}_i.\mathsf{head}}, C_{\mathcal{Q}_i.\mathsf{tail}}\}_{i\in[N]}),$    / $C_{\mathbf{attr}}$
    $(C_x, C_{t_{\mathcal{Q}_j.\mathsf{head}+1}}, C_{\mathcal{Q}_j.\mathsf{head}+1}, C_j),$    / $C_{\mathsf{enQ}_x^{\mathcal{Q}_j}}$
    $(C_{t_{\mathcal{Q}_j.\mathsf{head}+1}}, C_j, C_{b_\mathsf{next}}, C_{\mathsf{s}_t})$    / $C_{\mathcal{J}_j[t_{\mathcal{Q}_j.\mathsf{head}+1}]}$
$C_{\mathbf{attr}'} := (C_x, C_{\nu'}, C_\mathsf{s}\cdot C_{\mathsf{s}_t}, C_{\mathcal{Q}_0.\mathsf{head}}, \ldots, C_{\mathcal{Q}_j.\mathsf{head}}\cdot C_1,$
        $\ldots, C_{\mathcal{Q}_{j+1}.\mathsf{tail}}\cdot C_{b_\mathsf{next}}, \ldots C_{\mathcal{Q}_{N-1}.\mathsf{tail}})$
$C_\mathsf{enQ} := (C_x, C_{t_{\mathcal{Q}_j.\mathsf{head}+1}}, C_{\mathcal{Q}_{j+1}.\mathsf{tail}}\cdot C_{b_\mathsf{next}}, C_j\cdot C_{b_\mathsf{next}})$
```

Figure 15. SP Computation of $f(\mathsf{cred})$ in Redeem

TABLE 2. Possible Ticket Judgments for the j-th Queue

Type	b_next	Next Queue Index j'
Transient	1	$(j+1) \neq (N-1)$
Final	0	j
Exigent	1	$N-1$

commitment(s) of $\mathbf{attr}'$ and enQ, and proves to the SP that they are updated according to update function f The user prepares the update $f(\mathsf{cred})$ similar to Section 5.6.1 (with differences highlighted in blue):

- $\nu' \leftarrow \mathbb{Z}_p$; $s_t := \mathsf{s} + \mathsf{s}_t$;
- increments $\mathcal{Q}_j.\mathsf{head}$ to dequeue $\mathcal{Q}_j$;
- adds b_next to $\mathcal{Q}_{j+b_\mathsf{next}}.\mathsf{tail}$ in $\mathbf{attr}'$;
- prepares the data to enqueue $t_{\mathcal{Q}_j.\mathsf{head}+1}$ to the i'-th slot of the $(j + b_\mathsf{next})$-th queue, where $i' = b_\mathsf{next}\cdot(\mathcal{Q}_{j+b_\mathsf{next}}.\mathsf{tail} + b_\mathsf{next})$.

Unlike when j is known, now the SP does not know which commitment to apply additive homomorphism to compute $f(\mathsf{cred})$, and hence it relies on the user. Besides, j could be the last queue where $\mathcal{Q}_{j+1}$ is not well-defined. This explains the formula of i'. For a final judgment where $b_\mathsf{next} = 0$, $i' = 0$. "Enqueuing" to the 0-th slot of any slot means nothing, as it is an invalid slot for any credential.

The user proves that $C_{\mathbf{attr}'}$ updates $\mathcal{Q}_j.\mathsf{head}$ and $\mathcal{Q}_{j+b_\mathsf{next}}.\mathsf{tail}$ in Figure 16 with two ZKPs on $\mathcal{R}_\mathsf{sel} = ((C_{\vec{m}}, C_{\vec{m}'}, C_{j'}, C_b), (\vec{m}, \vec{m}', j', b) : \vec{m}'[j'] = \vec{m}[j'] + b)$, which proves the j'-th slot of both $\vec{m}'$ and $\vec{m}$ differ by b:

- ($\mathcal{Q}_j.\mathsf{head}$) $\mathcal{R}_\mathsf{sel}$ holds for $\vec{m} = \mathbf{attr}$, $\vec{m}' = \mathbf{attr}'$, $j' = 3 + 2j + 1$, $b = 1$ (public);
- ($\mathcal{Q}_{j+b_\mathsf{next}}.\mathsf{tail}$) $\mathcal{R}_\mathsf{sel}$ holds for $\vec{m} = \mathbf{attr}$, $\vec{m}' = \mathbf{attr}'$, $j' = 3 + 2(j + 1 + b_\mathsf{next})$, $b = b_\mathsf{next}$.

For C_enQ, $\mathcal{R}_{\mathsf{sel}'} = ((C_{\vec{m}}, C_{j'}, C_{i'}, C_b), (\vec{m}, j', i', b) : i' = b\cdot(\vec{m}[j']))$ is used to prove $C_{i'}$ (in C_enQ) commits the product of $b = b_\mathsf{next}$ and the j'-th element of $\vec{m} = \mathbf{attr}'$.

If all three ZKPs above are valid, the SP runs $\mathsf{Cred.Sig}()$ on the commitments $(C_{\mathbf{attr}'}, C_\mathsf{enQ})$, and the user will obtain an updated credential over $(\mathbf{attr}', \mathsf{enQ})$.

5.7. Emergency Revocation

In the baseline design, tickets in $\mathcal{Q}_j$ can only be moved to $\mathcal{Q}_{j+1}$ (or out of the credential). When a judgment can easily be deemed to be final, slowly advancing one queue at a time is not needed. Allow moving a ticket from $\mathcal{Q}_j$ directly to the finalized queue $\mathcal{Q}_{N-1}$ also enables emergency revocation. *i.e.*, the SP wants to revoke the user

```
Proof of $\mathbf{attr}' \leftarrow f(\mathsf{cred})$ when $j$ is hidden
$\textbf{parse } C_{\mathbf{attr}}, C_{\nu'} \textbf{ from } \pi$
$\textbf{parse } (C_x, C_{t_{\mathcal{Q}_j.\mathsf{head}+1}}, C_{\mathcal{Q}_j.\mathsf{head}+1}, C_j) \textbf{ from } \pi$    / enQ
$\textbf{parse } (C_{t_{\mathcal{Q}_j.\mathsf{head}+1}}, C_j, C_{b_\mathsf{next}}, C_{\mathsf{s}_t}) \textbf{ from } \pi$    / Judgment
$C_{j'} := C_{3+2j+1}$    / computes homomorphically
$C_{j''} := C_{3+2(j+1+b_\mathsf{next})}$    / index of $\mathcal{Q}_{j+b_\mathsf{next}}$'s tail
$\mathcal{U} : C_{\mathbf{attr}'} := (C_x, C_{\nu'}, C_\mathsf{s}\cdot C_{\mathsf{s}_t}, C_{\mathcal{Q}_0.\mathsf{head}}, \ldots,$
    $C_{\mathcal{Q}_j.\mathsf{head}}\cdot C_1, \ldots, C_{\mathcal{Q}_{j+b_\mathsf{next}}.\mathsf{tail}}\cdot C_{b_\mathsf{next}}, \ldots C_{\mathcal{Q}_{N-1}.\mathsf{tail}})$
    $C_\mathsf{enQ} := (C_x, C_{t_{\mathcal{Q}_j.\mathsf{head}+1}}, C_{i'}, C_j\cdot C_{b_\mathsf{next}})$
    $\pi_1 \leftarrow \mathsf{ZKP.P}(\mathsf{pp}, \mathcal{R}_\mathsf{sel}, (\mathbf{attr}, \mathbf{attr}', j', 1))$
    $\pi_2 \leftarrow \mathsf{ZKP.P}(\mathsf{pp}, \mathcal{R}_\mathsf{sel}, (\mathbf{attr}, \mathbf{attr}', j'', b_\mathsf{next}))$
    $\pi_3 \leftarrow \mathsf{ZKP.P}(\mathsf{pp}, \mathcal{R}_{\mathsf{sel}'}, (\mathbf{attr}', j'', i', b_\mathsf{next}))$
$\mathcal{U} \rightarrow \mathcal{S} : (C_{\mathbf{attr}'}, C_\mathsf{enQ}, \pi_1, \pi_2, \pi_3)$
```

Figure 16. Proving $f(\mathsf{cred})$ in Redeem for Hidden j

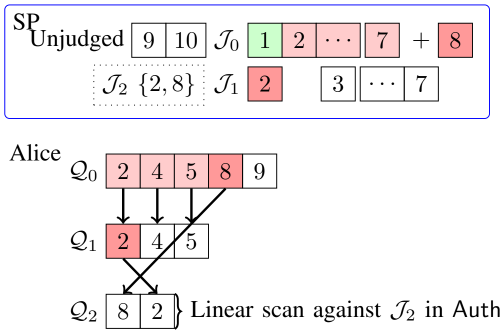


Figure 17. Let the emergency queue be $\mathcal{Q}_2$. Dark-red boxes (8 in $\mathcal{Q}_0$, 2 in $\mathcal{Q}_1$) are exigently-judged tickets. Redeeming them moves them directly to $\mathcal{Q}_2$. Authentication will prove each ticket in $\mathcal{Q}_2$ w.r.t. $\mathcal{J}_2$.

concerned immediately, without any delay when the concerned ticket is not in the first slot of each queue. We thus designate the last queue as the "emergency" queue. The tickets there can only be redeemed and incorporated into the credential score, instead of moving to other queues.

To realize this technically, we start by enabling an SP to move a ticket to the last queue from previous queues (via exigent judgments); then, we describe the linear scan in authentication for proving each ticket in this queue.

5.7.1. Augmented Judgment and Queue Advancement. During redemption, j, b_next in the judgment $(t, j, b_\mathsf{next}, \mathsf{s}_t)$ determines the destination queue $(j + b_\mathsf{next})$. We extend the judgment from $(t, j, b_\mathsf{next}, \mathsf{s}_t)$ to $(t, j, b_\mathsf{next}, j', \mathsf{s}_t)$, which adds the next queue index j' and is set according to Table 2. The update function f is the same as f in Section 5.6.2, except the new enqueuing signature's queue index $(j + b_\mathsf{next})$ is replaced by j' (highlighted in blue):

- prepares the data to enqueue $t_{\mathcal{Q}_j.\mathsf{head}+1}$ to the $(i = \mathcal{Q}_{j'}.\mathsf{tail} + b_\mathsf{next})$-th slot of the j'-th queue.

Indices-hiding can still be achieved as in Section 5.6.2.

5.7.2. Emergency "Queue" with a Linear Scan. Tickets in the emergency queue are likely out of order. For example (*cf.*, Figure 17, where $N = 3$), the SP decides to move ticket 8 and ticket 2 at $\mathcal{Q}_0$ and $\mathcal{Q}_1$ to $\mathcal{Q}_2$. Thus, it posts two exigent judgments $(8, 0, b_{\mathsf{next}}=1, j'=2, \mathsf{s}_8)$, $(2, 1, 1, 2, \mathsf{s}_2)$ on $\mathcal{J}_0, \mathcal{J}_1$. Alice must redeem them to authenticate again. Whether she redeems from $\mathcal{Q}_0$ or $\mathcal{Q}_1$ first is not enforced, so the intra-queue invariant does not hold for $\mathcal{Q}_2$. Nonetheless, it is no longer needed when authentication checks each ticket in $\mathcal{Q}_2$ against the score list $\mathcal{J}_2$ instead of a single range proof against $\mathfrak{jp}_2$.

To authenticate, the user proves for the last queue $\mathcal{Q}_2$ in AP that $|\mathcal{Q}_2| \leq K_2$ where $|\mathcal{Q}_2| = \mathcal{Q}_2.\mathsf{tail}$, and each ticket t_i in $\mathcal{Q}_2$ satisfies some predicate $\mathtt{P}(\mathcal{J}_2, t_i)$. If $|\mathcal{Q}_2|$ shall be revealed, the user proves the knowledge of $|\mathcal{Q}_2|$ enqueuing signatures and $\mathtt{P}(\mathcal{J}_2, t_i) = 1 \ \forall \ t_i \in \mathcal{Q}_2$.

The range $0 \leq |\mathcal{Q}_2| \leq K_2$, is where $|\mathcal{Q}_2|$ can be hidden within. For each i in $[1, K_2]$, we can use OR-composition proof to prove either the possession of (message-)signature (pair) $\mathsf{enQ}_x^{(i,2)}$ for $i \leq |\mathcal{Q}_2|$, or $\mathcal{Q}_2.\mathsf{tail} < i$ (exceeding the user's $\mathcal{Q}_2$) for $|\mathcal{Q}_2| < i \leq K_2$.

Specifically, for known $(\mathcal{J}_2, \mathsf{pk_t}, C_{\mathcal{Q}_2.\mathsf{head}}, C_{\mathcal{Q}_2.\mathsf{tail}})$, we prove $\mathcal{R}_{\mathsf{scan}} = ((\{\mathsf{enQ}_x^{(i,2)}\}_{i=1}^{|\mathcal{Q}_2|}, \mathcal{Q}_2.\mathsf{head}, \mathcal{Q}_2.\mathsf{tail}) : \bigwedge_{i=1}^{K_2}[(\mathsf{Cred.Vf}(\mathsf{pk_t}, \mathsf{enQ}_x^{(i,2)}) \wedge (\mathtt{P}(\mathcal{J}_2, t_i) = 1)) \vee (\mathcal{Q}_2.\mathsf{tail} < i)] \wedge (\mathcal{Q}_2.\mathsf{tail} \leq K_2))$ holds.

Note that the emergency queue size would be small for its special purpose.

5.7.3. Blocklist/Reputation Check.

$\mathtt{P}(\mathcal{J}_2, t_i)$ can be:

i) (If $\mathcal{J}_2$ stores blocked tickets :) t_i is a non-member of the revoked ticket set $\mathcal{J}_2$, *e.g.*, via an accumulator [1], *i.e.*, one is revoked for holding any ticket in this queue; or

ii) There is an up to date judgment on t_i in $\mathcal{J}_2 = \{(t, \mathsf{s}_t, \tau)\}$ where τ is the current timestamp.

The latter reputation-based approach can still incorporate the transient/finalized judgment using timestamp τ. Each exigent judgment can first be judged with a (very negative) initial score $\mathsf{s}_{(t,\tau)}$, and can be updated to $\mathsf{s}_{(t,\tau')}$ afterward; until a special τ^* is marked, denoting the finalized judgment. The user can redeem finalized ones out of the emergency queue with a more complex operation, such as using verifiable shuffle [20]. This also requires a loosely-synchronized clock for the timestamps across all judgments or a signature expiry mechanism [20]. One can see this as adopting the idea in our concurrent work [20] for a specific queue instead of the entire credential.

6. Instantiation and Performance

6.1. Related Zero-Knowledge Proofs

An authenticating user needs to compute ZKP that each ticket queue $\mathcal{Q}_j$ is either empty (equality check over commitments) or $t_{\mathcal{Q}_j.\mathsf{head}+1}$ is unjudged. Given a Σ-protocol for the equality check, signature verification, and range proof over committed values, the OR-proof technique [22] could be applied directly. We use the following ZKP for different relations:

- $\mathsf{ZKP}_{\mathsf{equal}}\{(a, b, r, s) : C_a = \mathsf{Com}(a; r) \wedge C_b = \mathsf{Com}(b; s) \wedge a = b\}$
- $\mathsf{ZKP}_{\mathsf{score}}\{(a, r) : C_a = \mathsf{Com}(a; r) \wedge A \leq a < B\}$
- $\mathsf{ZKP}_{\mathsf{diff}}\{(a, b, r, s) : C_a = \mathsf{Com}(a; r) \wedge C_b = \mathsf{Com}(b; s) \wedge a - b \in [0, K-1]\}$
- $\mathsf{ZKP}_{\mathsf{inner}}\{(\vec{a}, \vec{b}, c, r, s, t) : C_{\vec{a}} = \mathsf{Com}(\vec{a}; r) \wedge C_{\vec{b}} = \mathsf{Com}(\vec{b}; s) \wedge C_c = \mathsf{Com}(c; t) \wedge \langle \vec{a}, \vec{b} \rangle = c\}$
- ZKP for knowing BBS+ signature over individual commitments of $\vec{m}$ [7], [13].

The range proof can be instantiated using the signature-based set membership proof [12] or Bulletproofs [11], [25] which directly works with Pedersen commitment. One can also utilize the compressed-Σ protocol theory [4]–[6], which applies Bulletproofs to Σ-protocols for opening linear forms on a committed vector.

A range proof of $v \in [0, 2^n - 1]$ (supported natively by Bulletproofs) can be converted to an arbitrary range $[A, B]$ using two range proofs [12] on subtracted values to A, B. Assume $2^{n-1} < B < 2^n$, we can prove that $v - B + 2^n \in [0, 2^n - 1] \wedge v - A \in [0, 2^n - 1]$. Thus, we set a soft upper bound on the maximum value of the tickets and score to be $2^{64} - 1$.

For our combined use with BBS+ signature, we work on a pairing-friendly curve for the pairing operations, *e.g.*, BLS12-381. The proof of message equality in two commitments is straightforward. A redeeming user needs to prove the satisfaction of redeeming policy by ZKP of three valid signatures. The details on ZKP of the BBS+ signature can be found in Appendix B.

6.2. Using CP-SNARK

We also utilize the commit-and-prove paradigm. Most of the efficiently-implemented zkSNARKs focus on proving general circuits or rank-1 constraint systems. ZKP of $\mathtt{AP}(\mathsf{cred}) = 1$ on committed attributes of $\mathsf{enQ}_x^{\mathcal{Q}_i}$ and cred, and public parameters $\mathfrak{jp}_i$ and K_i, can be separated into

$$\bigwedge_{i\in[N]} \left\{ \begin{array}{l} (\mathcal{Q}_i.\mathsf{head} = \mathcal{Q}_i.\mathsf{tail}) \vee \\ \begin{pmatrix} (\mathcal{Q}_i.\mathsf{tail} - \mathcal{Q}_i.\mathsf{head} \in [0, K_i - 1]) \wedge \\ (t_{\mathcal{Q}_i.\mathsf{head}+1} > \mathfrak{jp}_i) \end{pmatrix} \end{array} \right\} \text{ and}$$

$$\bigwedge_{i\in[N]} ((\mathcal{Q}_i.\mathsf{head} = \mathcal{Q}_i.\mathsf{tail}) \vee \mathsf{Cred.Vf}(\mathsf{pk}, \mathsf{enQ}_x^{\mathcal{Q}_i}) = 1),$$

where the first and second nested relations can be proved by CP-SNARK (*e.g.*, LegoGro16 [16] on circuits written with circom library[6]) and proof of partial knowledge respectively. The committed values in the two relations are linked with the commit-and-prove paradigm.

Instantiating ZKP for $\mathcal{R}_{\mathsf{sel}}$ and $\mathcal{R}_{\mathsf{sel'}}$. In Section 5.6.2, ZKP on $\mathcal{R}_{\mathsf{sel}}$ and $\mathcal{R}_{\mathsf{sel'}}$ are used: $\mathcal{R}_{\mathsf{sel}} = ((C_{\vec{m}}, C_{\vec{m}'}, C_{j'}, C_b), (\vec{m}, \vec{m}', j', b) : \vec{m}'[j'] = \vec{m}[j'] + b)$ is used to prove correct update of queue pointers in $f(\mathsf{cred})$. Using zkSNARK, $\vec{m} = \mathbf{attr}, \vec{m}' = \mathbf{attr}'$, the committed index j' and bit $b = b_{\mathsf{next}}$ are the inputs to a circuit that checks if $\mathcal{R}_{\mathsf{sel}}$ is satisfied, where the commitments are used for the proof of UP. One can utilize the one-hot encoding $\vec{e}$ of j' (as an implicit input), where the circuit checks:
i) $\langle \vec{e}, \vec{1} \rangle = 1 \ \wedge \langle \vec{e}, \vec{1} - \vec{e} \rangle = 0$ ($\vec{e}$ is a weight-1 bit vector);
ii) $\langle \vec{e}, (0, \ldots, 3 + 2N) \rangle = j'$;
iii) $\langle \vec{e}, \vec{m}' - \vec{m} \rangle = b$.

ZKP on $\mathcal{R}_{\mathsf{sel'}} = ((C_{\vec{m}}, C_{j'}, C_{i'}, C_b), (\vec{m}, j', i', b) : i' = b \cdot (\vec{m}[j']))$ is used to prove the index i' in the enqueuing data, where $\vec{m} = \mathbf{attr}'$. Utilizing the one-hot encoding $\vec{e}$, ZKP on $\mathcal{R}_{\mathsf{sel'}}$ can be instantiated by proving the first and second conditions with two more conditions: $\langle \vec{e}, \vec{m} \rangle = \vec{m}[j']$ and $b \cdot (\vec{m}[j']) = i'$. The above relations are readily supported by zkSNARKs.

6. https://github.com/iden3/circom

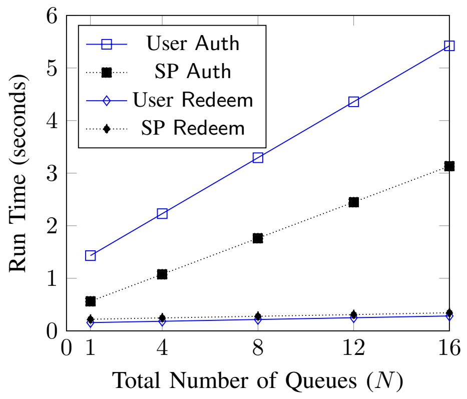


Figure 18. Computation times for Auth, Redeem (using Bulletproofs)

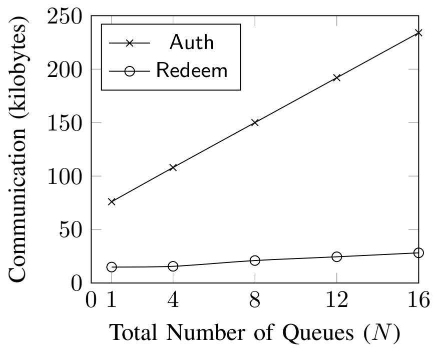


Figure 19. Total communication costs for Auth, Redeem

6.3. Experimental Evaluation

We use the Rust library from docknetwork[7] which is based on the arkwork library [3]. We also use the modified version of the Bulletproofs[8] for the pairing-friendly curve BLS12-381. We test on a desktop PC (localhost setting) with an 8-core AMD Ryzen CPU running at 3.7GHz. We use Windows Linux Subsystem to emulate Ubuntu 20.04.

We benchmark our authentication and redeeming protocols against numbers of queues $N = \{1, 4, 8, 12, 16\}$ with 8 scoring categories ($\ell = 8$). We compare with our re-implementation of FARB [35], the state-of-the-art sequential-judging system. The runtimes in Figure 18 are taken as the average timing of multiple runs. The bit-length of the score, tickets, and indexes is 64. Note that the improved ZKP for signature verification [13] is used. We also plot the total communication costs of authentication and redeeming in kilobytes versus the number of queues.

Compared to FARB ($N = 1$) with no slackness, SMART's user authentication time increases by about $1.35\times$ when the SP can slack for $N - 1 = 7$ times. We deem such kind of overhead acceptable. For example, the time for $N = 16$ is still in the order of 5 seconds. More fundamentally, this can be sped up by using better computing platforms alone; while the subjective judging complexity and hence its associated global-halting issue are inherent no matter how fast the future computing platform will be (unless the subjective assessment can be fully automated, say, by fancy AI technologies).

Our protocol scales linearly in the number of queues in both computation and communication costs (Figure 19). The increased communication cost in redeeming mainly comes from the increased credential proof size. The bottleneck of the authentication is the computation (and verification) of the N disjunctive proof.

Our authentication and redeeming costs are independent of the number of sessions in each queue and the global score lists. Efficiency-wise, one could roughly consider our N takes the role of prevalent parameter K in prior sequential-judging designs. Nonetheless, as explained in the introduction, the implications brought by tuning these two parameters are different. SMART gains flexibility and resilience against global halting.

7. Concluding Remarks

Securing the online world and protecting user privacy are becoming increasingly important, asking for anonymous credential systems free from trusted third parties. Meanwhile, incorporating reputation mechanisms based on subjective evaluation can add an additional layer of trust and security to these anonymous credential systems.

We unveil the unspoken global-halting vulnerability of the prevailing first-in-first-out sequential judging (PEREA, PERM, BLACR-express, and FARB [35]). Adversaries can launch denial-of-service attacks by introducing many hard-to-judge sessions. As long as one session remains unjudged, the entire system becomes paralyzed, affecting honest users as well. They also fail to support transient judgments. Another design (BLAC(R), SNARKBlock) processes all judgments ever made. Precomputations are only reusable when the judgments never change.

We propose SMART, with an extended multi-queue data structure in the credential. Besides alleviating the global-halting issue, SMART allows the SP to give adjustable judgment to authenticated sessions, avoiding the dilemma of being too lenient or overly cautious. SMART needs to carefully maintain the user sessions across multiple queues, in particular, hiding the source queue and the transient/finalized nature while redeeming, which are non-issues in single-queue designs like FARB.

We end with introducing two concurrent works. SAC [20] supports transient judgments without sequential judging and is also free from the global-halting issue. The credential uses an unordered data structure of a set to store the ticket. Redeeming (and hence removing) finalized tickets requires (signature-based) membership proof against the finalized ticket set. Verifiable shuffle is employed to hide the indices of any added or removed tickets. Unlike SMART, SAC requires signature expiration to invalidate previous judgments, *i.e.*, all active tickets need to be re-signed whenever the SP upgrades a score.

In a regular system, judgments on the tickets cannot be casually deleted since the privacy guarantee makes it unclear whether they have been redeemed or not. Decentralized anonymous reputation with sustainability [18] is recently formalized. A subset of surely-redeemed tickets can be identified and hence deleted from time to time.

7. https://github.com/docknetwork/crypto
8. https://github.com/dalek-cryptography/bulletproofs

References

[1] Tolga Acar, Sherman S. M. Chow, and Lan Nguyen. Accumulators and U-Prove revocation. In Ahmad-Reza Sadeghi, editor, *Financial Cryptography and Data Security - 17th International Conference, FC 2013, Okinawa, Japan, April 1-5, 2013, Revised Selected Papers*, volume 7859 of *Lecture Notes in Computer Science*, pages 189–196. Springer, 2013.

[2] antonela. The value of Tor and anonymous contributions to wikipedia. https://blog.torproject.org/the-value-of-anonymous-contributions-wikipedia.

[3] arkworks contributors. `arkworks` zkSNARK ecosystem, 2022. https://arkworks.rs.

[4] Thomas Attema and Ronald Cramer. Compressed Σ-protocol theory and practical application to plug & play secure algorithmics. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 513–543. Springer, 2020.

[5] Thomas Attema, Ronald Cramer, and Serge Fehr. Compressing proofs of k-out-of-n partial knowledge. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 65–91. Springer, 2021.

[6] Thomas Attema, Ronald Cramer, and Matthieu Rambaud. Compressed Σ-protocols for bilinear group arithmetic circuits and application to logarithmic transparent threshold signatures. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 526–556. Springer, 2021.

[7] Man Ho Au and Apu Kapadia. PERM: Practical reputation-based blacklisting without TTPs. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *the ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012*, pages 929–940, 2012.

[8] Man Ho Au, Apu Kapadia, and Willy Susilo. BLACR: TTP-free blacklistable anonymous credentials with reputation. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012*. The Internet Society, 2012.

[9] Man Ho Au, Willy Susilo, Yi Mu, and Sherman S. M. Chow. Constant-size dynamic k-times anonymous authentication. *IEEE Systems Journal*, 7(2):249–261, 2013.

[10] Jan Bobolz, Fabian Eidens, Stephan Krenn, Sebastian Ramacher, and Kai Samelin. Issuer-hiding attribute-based credentials. In Mauro Conti, Marc Stevens, and Stephan Krenn, editors, *Cryptology and Network Security - 20th International Conference, CANS 2021, Vienna, Austria, December 13-15, 2021, Proceedings*, volume 13099 of *Lecture Notes in Computer Science*, pages 158–178. Springer, 2021.

[11] Benedikt Bünz, Jonathan Bootle, Dan Boneh, Andrew Poelstra, Pieter Wuille, and Gregory Maxwell. Bulletproofs: Short proofs for confidential transactions and more. In *2018 IEEE Symposium on Security and Privacy, SP 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 315–334. IEEE Computer Society, 2018.

[12] Jan Camenisch, Rafik Chaabouni, and abhi shelat. Efficient protocols for set membership and range proofs. In Josef Pieprzyk, editor, *Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, December 7-11, 2008. Proceedings*, pages 234–252, 2008.

[13] Jan Camenisch, Manu Drijvers, and Anja Lehmann. Anonymous attestation using the strong Diffie Hellman assumption revisited. In Michael Franz and Panos Papadimitratos, editors, *Trust and Trustworthy Computing - 9th International Conference, TRUST 2016, Vienna, Austria, August 29-30, 2016, Proceedings*, volume 9824 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2016.

[14] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew K. Franklin, editor, *Advances in Cryptology - CRYPTO 2004, 24th Annual International CryptologyConference, Santa Barbara, California, USA, August 15-19, 2004, Proceedings*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, 2004.

[15] Jan Camenisch and Markus Stadler. Efficient group signature schemes for large groups (extended abstract). In Burton S. Kaliski Jr., editor, *Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17-21, 1997, Proceedings*, volume 1294 of *Lecture Notes in Computer Science*, pages 410–424. Springer, 1997.

[16] Matteo Campanelli, Dario Fiore, and Anaïs Querol. LegoSNARK: Modular design and composition of succinct zero-knowledge proofs. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 2075–2092. ACM, 2019.

[17] Sherman S. M. Chow. Real traceable signatures. In Michael J. Jacobson Jr., Vincent Rijmen, and Reihaneh Safavi-Naini, editors, *Selected Areas in Cryptography, 16th Annual International Workshop, SAC 2009, Calgary, Alberta, Canada, August 13-14, 2009, Revised Selected Papers*, volume 5867 of *Lecture Notes in Computer Science*, pages 92–107. Springer, 2009.

[18] Sherman S. M. Chow, Christoph Egger, Russell W. F. Lai, Viktoria Ronge, and Ivy K. Y. Woo. On sustainable ring-based anonymous systems. In *36th IEEE Computer Security Foundations Symposium, CSF 2023, Dubrovnik, Croatia, July 9 - 13, 2023*, 2023. To appear.

[19] Sherman S. M. Chow, Joseph K. Liu, and Duncan S. Wong. Robust receipt-free election system with ballot secrecy and verifiability. In *Proceedings of the Network and Distributed System Security Symposium, NDSS 2008, San Diego, California, USA, 10th February - 13th February 2008*. The Internet Society, 2008.

[20] Sherman S. M. Chow, Jack P. K. Ma, and Tsz Hon Yuen. Scored anonymous credentials. In Mehdi Tibouchi and Xiaofeng Wang, editors, *Applied Cryptography and Network Security - 21st International Conference, ACNS 2023, Kyoto, Japan, June 19-22, 2023, Proceedings, Part II*, volume 13906 of *Lecture Notes in Computer Science*. Springer, 2023. To appear.

[21] Sherman S. M. Chow, Haibin Zhang, and Tao Zhang. Real hidden identity-based signatures. In Aggelos Kiayias, editor, *Financial Cryptography and Data Security - 21st International Conference, FC 2017, Sliema, Malta, April 3-7, 2017, Revised Selected Papers*, volume 10322 of *Lecture Notes in Computer Science*, pages 21–38. Springer, 2017.

[22] Ronald Cramer, Ivan Damgård, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In Yvo Desmedt, editor, *Advances in Cryptology - CRYPTO '94, 14th Annual International Cryptology Conference, Santa Barbara, California, USA, August 21-25, 1994, Proceedings*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer, 1994.

[23] Roger Dingledine, Nick Mathewson, and Paul F. Syverson. Tor: The second-generation onion router. In Matt Blaze, editor, *Proceedings of the 13th USENIX Security Symposium, August 9-13, 2004, San Diego, CA, USA*, pages 303–320. USENIX, 2004.

[24] Jack Doerner, Yashvanth Kondi, Eysa Lee, abhi shelat, and LaKyah Tyner. Threshold BBS+ signatures for distributed anonymous credential issuance. In *44th IEEE Symposium on Security and Privacy, SP 2023, San Francisco, CA, USA, 22-25 May 2023*, pages 2095–2111. IEEE Computer Society, 2023.

[25] Chaya Ganesh, Claudio Orlandi, Mahak Pancholi, Akira Takahashi, and Daniel Tschudi. Fiat-Shamir Bulletproofs are non-malleable (in the algebraic group model). In Orr Dunkelman and Stefan Dziembowski, editors, *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*, volume 13276 of *Lecture Notes in Computer Science*, pages 397–426. Springer, 2022.

[26] Christina Garman, Matthew Green, and Ian Miers. Decentralized anonymous credentials. In *21st Annual Network and Distributed System Security Symposium, NDSS 2014, San Diego, California, USA, February 23-26, 2014*. The Internet Society, 2014.

[27] Jens Groth. On the size of pairing-based non-interactive arguments. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 305–326. Springer, 2016.

[28] Aggelos Kiayias, Yiannis Tsiounis, and Moti Yung. Traceable signatures. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, volume 3027 of *Lecture Notes in Computer Science*, pages 571–589. Springer, 2004.

[29] Torben P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In Joan Feigenbaum, editor, *Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings*, volume 576 of *Lecture Notes in Computer Science*, pages 129–140. Springer, 1991.

[30] Michael Rosenberg, Mary Maller, and Ian Miers. SNARKBlock: Federated anonymous blocklisting from hidden common input aggregate proofs. In *43rd IEEE Symposium on Security and Privacy, SP 2022, San Francisco, CA, USA, May 22-26, 2022*, pages 948–965. IEEE, 2022.

[31] Claus-Peter Schnorr. Efficient signature generation by smart cards. *J. Cryptol.*, 4(3):161–174, 1991.

[32] Stefano Tessaro and Chenzhi Zhu. Revisiting BBS signatures. In Carmit Hazay and Martijn Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 691–721. Springer, 2023.

[33] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. Blacklistable anonymous credentials: Blocking misbehaving users without TTPs. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007*, pages 72–81. ACM, 2007.

[34] Patrick P. Tsang, Man Ho Au, Apu Kapadia, and Sean W. Smith. PEREA: towards practical ttp-free revocation in anonymous authentication. In Peng Ning, Paul F. Syverson, and Somesh Jha, editors, *Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27-31, 2008*, pages 333–344. ACM, 2008.

[35] Li Xi and Dengguo Feng. FARB: Fast anonymous reputation-based blacklisting without TTPs. In Gail-Joon Ahn and Anupam Datta, editors, *Proceedings of the 13th Workshop on Privacy in the Electronic Society, WPES 2014, Scottsdale, AZ, USA, November 3, 2014*, pages 139–148. ACM, 2014.

[36] Li Xi, Jianxiong Shao, Kang Yang, and Dengguo Feng. ARBRA: Anonymous reputation-based revocation with efficient authentication. In Sherman S. M. Chow, Jan Camenisch, Lucas Chi Kwong Hui, and Siu-Ming Yiu, editors, *Information Security - 17th International Conference, ISC 2014, Hong Kong, October 12-14, 2014. Proceedings*, volume 8783 of *Lecture Notes in Computer Science*, pages 33–53. Springer, 2014.

[37] Kin Ying Yu, Tsz Hon Yuen, Sherman S. M. Chow, Siu-Ming Yiu, and Lucas Chi Kwong Hui. PE(AR)^2: Privacy-enhanced anonymous authentication with reputation and revocation. In Sara Foresti, Moti Yung, and Fabio Martinelli, editors, *Computer Security - ESORICS 2012 - 17th European Symposium on Research in Computer Security, Pisa, Italy, September 10-12, 2012. Proceedings*, volume 7459 of *Lecture Notes in Computer Science*, pages 679–696. Springer, 2012.

[38] Tao Zhang, Huangting Wu, and Sherman S. M. Chow. Structure-preserving certificateless encryption and its application. In Mitsuru Matsui, editor, *Topics in Cryptology - CT-RSA 2019 - The Cryptographers' Track at the RSA Conference 2019, San Francisco, CA, USA, March 4-8, 2019, Proceedings*, volume 11405 of *Lecture Notes in Computer Science*, pages 1–22. Springer, 2019.

A. Security Definition of Building Blocks

A function $f : \mathbb{N} \to \mathbb{R}$ is negligible, denoted by $\mathsf{negl}(\lambda)$, if $\forall c > 0$, $\exists\ x_0$ s.t. $f(x) < 1/x^c,\ \forall x > x_0$.

A.1. Zero-Knowledge Proof or Argument System

A (non-interactive) proof system is an argument of knowledge if it satisfies the two properties below.

Definition 5 (Completeness)**.** *A non-interactive proof system* $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ *for language* $\mathcal{L}_R$ *achieves perfect completeness if for all* $(x, w) \in R$*:* $\Pr[1 \leftarrow \mathsf{V}(\mathsf{crs}, x, \pi) : \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \pi \leftarrow \mathsf{P}(\mathsf{crs}, x, w)] = 1$

Definition 6 (Knowledge-Soundness)**.** *A (non-interactive) proof system* $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ *satisfies knowledge-soundness if there exists a* PPT *simulator* Ext *such that for all* PPT *adversaries* $\mathcal{A}$ *and prover* P^**:*

$$\left| \begin{array}{l} \Pr\left[\mathsf{V}(\mathsf{crs}, x, \pi) = 1 \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \\ (x, w) \leftarrow \mathcal{A}(\mathsf{crs}), \\ \pi \leftarrow \mathsf{P}^*(\mathsf{crs}, x, w) \end{array} \right] \\ - \Pr\left[(x, w) \in R \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \\ (x, w) \leftarrow \mathcal{A}(\mathsf{crs}), \\ w \leftarrow \mathsf{Ext}^{\mathsf{P}^*}(\mathsf{crs}, x) \end{array} \right] \end{array} \right| \leq \mathsf{negl}(\lambda).$$

Definition 7 (Perfect Special Honest-Verifier Zero-knowledge (SHVZK))**.** *A non-interactive proof system* $\Pi = (\mathsf{Setup}, \mathsf{P}, \mathsf{V})$ *is said to be Perfect SHVZK if there exists a* PPT *simulator* $\mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2)$ *such that for all pairs of* PPT *adversaries* $(\mathcal{A}_1, \mathcal{A}_2)$*:*

$$\begin{aligned} &\Pr\left[(x, w) \in R, \mathcal{A}_1(\pi) = 1 \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \\ (x, w) \leftarrow \mathcal{A}_2(\mathsf{crs}), \\ \pi \leftarrow \mathsf{P}(\mathsf{crs}, x, w) \end{array} \right] \\ &= \Pr\left[(x, w) \in R, \mathcal{A}_1(\pi) = 1 \;\middle|\; \begin{array}{l} \mathsf{crs} \leftarrow \mathsf{Sim}_1(1^\lambda), \\ (x, w) \leftarrow \mathcal{A}_2(\mathsf{crs}) \\ \pi \leftarrow \mathsf{Sim}_2(\mathsf{crs}, x) \end{array} \right]. \end{aligned}$$

A.2. Credential Signature Scheme

Definition 8 (Unforgeability)**.** *A credential signature scheme* $\mathsf{Cred} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *is said to be unforgeable if for all* PPT *adversaries* $\mathcal{A}$*, the probability of winning the following game is less than* $\mathsf{negl}(\lambda)$*:*

- *The challenger runs* $\mathsf{KGen}(1^\lambda)$ *to obtain* $(\mathsf{pk}, \mathsf{sk})$*,* pk *is given to the adversary* $\mathcal{A}$*.*
- *The adversary can query the signing protocol inputting* $\vec{m}, r$*. The protocol (with the challenger playing as the signer) returns a signature* σ*.*
- *The adversary outputs a message-signature tuple* $(\vec{m}^*, \sigma^*)$ *that is valid and distinct, i.e.,* $\mathsf{Cred.Vf}(\mathsf{pk}, \vec{m}^*, \sigma^*) = 1$ *and* $\vec{m}^*$ *has not been queried before.*

Definition 9 (Blindness)**.** *A credential signature scheme* $\mathsf{Cred} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *satisfies the blindness property if for all* PPT *adversaries* $\mathcal{A}$*, the probability of winning the following game is less than* $1/2 + \mathsf{negl}(\lambda)$*:*

- *The challenger runs* $\mathsf{KGen}(1^\lambda)$ *to obtain* $(\mathsf{pk}, \mathsf{sk})$*,* $(\mathsf{pk}, \mathsf{sk})$ *is given to the adversary* $\mathcal{A}$*.*
- *The adversary on input* $(\mathsf{pk}, \mathsf{sk})$*, outputs two messages* $\vec{m}_0, \vec{m}_1$*.*
- *The challenger samples a bit* b *and randomness* r *and interacts with the adversary in the* $\mathsf{Cred.Sig}$ *protocol with input* $(\vec{m}_b; r)$*.*
- *The adversary outputs a bit* b' *and wins if* $b = b'$*.*

Blindness follows from the (perfectly) hiding property of the Pedersen commitment scheme and zero-knowledgeness of the ZKP.

B. BBS+ Signature

Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be cyclic groups of λ-bit prime order p. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a non-degenerate bilinear map.

The setup algorithm outputs the bilinear group context $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g, h)$ with g, h being a generator of $\mathbb{G}_1$, $\mathbb{G}_2$, respectively, and $e(g, h)$ is a generator of the target group $\mathbb{G}_T$. For Type-III pairing, $\mathbb{G}_1 \neq \mathbb{G}_2$, with no isomorphism between $\mathbb{G}_1$ and $\mathbb{G}_2$.

The public parameter pp contains generators $(g, g_0, \ldots, g_n, h) \leftarrow\!\$\; \mathbb{G}_1^{n+1} \times \mathbb{G}_2$ and the description of the bilinear group. The BBS+ signature (over a bilinear group defined by pp) consists of three algorithms:

- $\mathsf{BBS{+}.KGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: The signer samples $\gamma \in \mathbb{Z}_p^*$, sets $\mathsf{sk} = \gamma$, and sets $\mathsf{pk} = w = h^\gamma$.
- $\mathsf{BBS{+}.Sign}(\vec{m}, \mathsf{sk}) \to \sigma$: To sign $\vec{m} = (m_1, \ldots, m_n) \in \mathbb{Z}_p^n$, the signer samples $e, s \leftarrow\!\$\; \mathbb{Z}_p$ and computes $A \leftarrow (g g_0^s \prod_{i=1}^n g_i^{m_i})^{\frac{1}{e+\gamma}}$. The signature on $\vec{m}$ is $\sigma \leftarrow (A, e, s)$.
- $\mathsf{BBS{+}.Vf}(\mathsf{pk}, \vec{m}, \sigma) \to b$: To verify σ on $\vec{m}$ under $\mathsf{pp} = (g, g_0, \ldots, g_n, h)$ and $\mathsf{pk} = h^\gamma$, the verifier returns $e(A, w h^e) \stackrel{?}{=} e(g g_0^s \prod_{i=1}^n g_i^{m_i}, h)$.

The BBS+ signature supports signing on committed multi-block message $\vec{m}$. We describe the blind signature issuance protocol run between the signer and user:

1) The user computes the commitment of $\vec{m}$ as $C_{\vec{m}} = g_0^{s'} \prod_{i=1}^n g_i^{m_i}$ for a randomly sampled $s' \leftarrow\!\$\; \mathbb{Z}_p^*$.
2) The user sends the commitment $C_{\vec{m}}$ to the signer with ZKPoK of the commitment opening, *e.g.*, $\mathsf{ZKP}\{(\vec{m}; \rho) : C_{\vec{m}} = \mathsf{Com}(\vec{m}; \rho)\}$.
3) The signer computes $A = (g g_0^s C_m)^{\frac{1}{e+\gamma}}$, where $e, s \leftarrow\!\$\; \mathbb{Z}_p^*$, and sends $\sigma' = (A, e, s)$ to the user.
4) The user parses the signature as $\sigma = (A, e, s + s')$ and returns σ, if $\mathsf{BBS{+}.Vf}(\mathsf{pk}, \vec{m}, \sigma) = 1$.

The server stores the auxiliary output aux' that contains the commitment(s) and ZKP. The user can further prove relations about individual commitments of $m_i \in \vec{m}$, *e.g.*, $\mathsf{ZKP}\{(\{m_i; r_i\}_{i=1}^{\ell}; \rho) : C_{\vec{m}} = \mathsf{Com}(\vec{m}; \rho)\ \bigwedge_{i=1}^{\ell} (C_{m_i} = \mathsf{Com}(m_i; r_i))\}$, where C_{m_i} would also be proved about in other ZKPs (commit-and-prove).

To prove the knowledge of the signature on the committed multi-block message, the prover can prepare the commitment and prove in zero-knowledge that the pairing verification equation holds [9] (which requires ZKP in the target group $\mathbb{G}_T$). Camenisch *et al.* [13] proposed a more efficient variant of ZKP that only involves operations over $\mathbb{G}_1$ and supports selective message disclosure, *i.e.*, $\{m_i\}_{i \in D}$ are revealed, where $D \subseteq \{1, n\}$ is a subset of indexes of $\vec{m}$. The prover with $\sigma = (A, e, s)$ on $\vec{m}$ randomizes the signature with $r_1 \leftarrow\!\$\; \mathbb{Z}_p^*$ by setting $A' \leftarrow A^{r_1}$. It computes $r_3 \leftarrow \frac{1}{r_1}$ and set $\bar{A} \leftarrow A'^{-e} \cdot B^{r_1}$, where $B = (g g_0^s \prod_{i=1}^n g_i^{m_i} = A^{\gamma+e})$. It also picks $r_2 \leftarrow\!\$\; \mathbb{Z}_p$ and sets $d \leftarrow b^{r_1} \cdot g_0^{-r_2}$ and $s' \leftarrow s - r_2 \cdot r_3$. Finally, it proves:

$$\pi \leftarrow \mathsf{ZKP}\{(\{m_i\}_{i \notin D}, e, r_1, r_2, r_3, s') : \bar{A}/d = A'^{-e} \cdot g_0^{r_2} \wedge g \prod_{i \in D} g_i^{m_i} = d^{r_3} g_0^{-s'} \prod_{i \notin D} g_i^{-m_i}\}.$$

The proof consists of $(A', \bar{A}, d, \pi)$. Its verification requires checking $A' \neq 1_{\mathbb{G}_1}$ and $e(A', w) = e(\bar{A}, h)$. The signer needs to publish $(\bar{g_1}, \bar{g_1}^\gamma)$ for $\bar{g_1} \neq 1_{\mathbb{G}_1}$ and ensures $A \neq 1_{\mathbb{G}_1}$. The nonce is revealed to the SP in our protocol. Recently, a variant of the BBS+ signatures with a shorter signature size has been proposed [32].

The BBS+ scheme is existentially unforgeable against adaptive chosen message attacks for Type-III pairings under the q-strong Diffie-Hellmen (q-SDH) assumption [13].

Definition 10 (q-SDH Assumption)**.** *Given the tuple* $(g_1, g_1^x, g_1^{(x^2)}, \ldots, g_1^{(x^q)}, g_2, g_2^x) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2$*, no PPT algorithm can output* $(c, g_1^{1/(x+c)})$*, where* $c \in \mathbb{Z}_p \setminus \{-x\}$*.*

C. Extensions to Multiple Service Providers

The SP holds the credential issuance key for ticket management in our setting. As SMART uses a credential signature scheme in a black-box manner, we can distribute the credential-issuing power across multiple SPs by replacing it with its different extensions for different "trust" models. For example, for the n'-out-of-n access structure, *i.e.*, any subgroup of n' SPs among a group of n SPs can issue credentials, we can use 1-out-of-n ZKP [22] or threshold BBS+ [24]. In issuer-hiding attribute-based credentials, Bobolz *et al.* [10] replaced the OR-proof technique with a signature-based set membership proof [12], where the verifier specify the permitted issuer public keys by signing a possibly different set of them according to the expected verification policy (regarding the SPs) in a given scenario.

Different SPs can also judge tickets as they wish, and multiple SPs can judge the same ticket. The redemption policy and the credential update function could take in the same judgment signed by at least n' SPs (with n'-out-of-n proof [5], [22]). With the above in mind, our signature-based design makes it compatible with a possible extension to the decentralized setting. The core idea is that the credentials, policies, and score lists can be put into a trusted public ledger [26] where succinct ZKPs accompany the credential updates.