Post-Quantum Privacy Pass via Post-Quantum Anonymous Credentials

Guru-Vamsi Policharla
UC Berkeley

Bas Westerbaan
Cloudflare, Inc

Armando Faz-Hernández
Cloudflare, Inc

Christopher A. Wood
Cloudflare, Inc

Abstract

It is known that one can generically construct a post-quantum anonymous credential scheme, supporting the showing of arbitrary predicates on its attributes using general-purpose zero-knowledge proofs secure against quantum adversaries [Fischlin, CRYPTO 2006]. Traditionally, such a generic instantiation is thought to come with impractical sizes and performance. We show that with careful choices and optimizations, such a scheme can perform surprisingly well. In fact, it performs competitively against state-of-the-art post-quantum blind signatures, for the simpler problem of post-quantum unlinkable tokens, required for a post-quantum version of *Privacy Pass*.

To wit, a post-quantum Privacy Pass constructed in this way using zkDilithium, our proposal for a STARK-friendly variation on Dilithium2, allows for a trade-off between token size (85–175 KB) and generation time (0.3–5 s) with a proof security level of 115 bits. Verification of these tokens can be done in 20–30 ms. We argue that these tokens are reasonably practical, adding less than a second upload time over traditional tokens, supported by a measurement study.

Finally, we point out a clear advantage of our approach: the flexibility afforded by the general purpose zero-knowledge proofs. We demonstrate this by showing how we can construct a rate-limited variant of Privacy Pass that doesn't not rely on non-collusion for privacy.

1 Introduction

Traditional goals of cryptography include the protection of confidentiality, integrity and authenticity of communication, which are achieved using encryption, key agreement and signature schemes. Recently, more advanced cryptography such as verifiable oblivious pseudorandom functions (vOPRFs), blind signatures, and anonymous credentials are seeing wider adoption to attain more difficult privacy goals, such as preventing the tracking of users. These primitives are used in widely deployed (or to be deployed) protocols such as Cloudflare's *Privacy Pass*. Other examples include Apple's *Private Access Tokens* [3], Google's *Private State Tokens* [31], Brave's *Basic Attention Tokens* [48], private click measurement [21], the Dutch Corona app (CoronaCheck) [43], Facebook's fraud prevention [35] and DIT system for private telemetry on Whatsapp [34], effectively impacting billions of users. Although we focus on Privacy Pass in this work, our results are nonetheless relevant to all other protocols mentioned above, and those using more general anonymous credentials, eg. IRMA [45].

Privacy Pass. In Privacy Pass [22, 23], clients interact with an attester and issuer to *issue* a token that can later be *redeemed* to a website, or origin, upon being challenged. This basic interaction is shown in Fig. 1. In the original deployment of Privacy Pass, the attester and issuer roles were combined into a single entity, and clients demonstrated validity to issue tokens by solving a CAPTCHA. In effect, this let clients use tokens in place of solving a CAPTCHA, resulting in a better end-user experience, especially when accessing content anonymously through VPNs, Tor, or I2P. Since its inception, the Privacy Pass protocol has transitioned to the IETF for standardization *ietf-privacypass-protocol-08* and has seen new deployment in technologies today, including Apple's Private Access Tokens [3].

Given the widespread use of Privacy Pass and other unlinkable tokens, constructing a practical, post-quantum version should be considered an important milestone in future-proofing user privacy [37]. We view this as an opportunity to not only secure the Privacy Pass protocol against quantum adversaries but also to explore alternatives to prior approaches that can a) improve the privacy guarantees for a rate-limited version of Privacy Pass and b) have the flexibility to handle more complicated rate-limiting policies if needed in the future. Unfortunately, there are no known post-quantum solutions

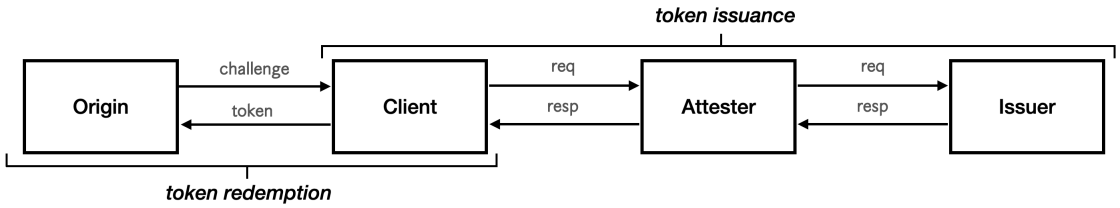


Figure 1: Privacy Pass token issuance and redemption interaction overview

for vOPRFs/blind signatures, the building blocks of Privacy Pass, that perform competitively against their *classical* counterparts based on assumptions that do not hold against quantum adversaries.

Reviving an old approach. We generically construct a very flexible many-show anonymous credential scheme (and hence anonymous tokens for Privacy Pass), supporting the showing of arbitrary predicates evaluated on its attributes using general-purpose zero-knowledge Non-Interactive Arguments of Knowledge (zkNIAoKs) via the Camenisch-Lysyankaya framework [16]. One can also obtain blind signatures through an adaptation of Fischlin's scheme [28]. Here, a client commits to the attributes $(\mathsf{attr}_1, \ldots, \mathsf{attr}_k)$ and sends the commitment com to the issuer who responds with a signature σ on the commitment. The client then provides a zkNIAoK of a signature σ on a commitment com, an opening of this commitment r to a set of k attributes $(\mathsf{attr}_1, \ldots, \mathsf{attr}_k)$ and reveals the output of predicates evaluated on the attributes. This construction is plausibly post-quantum if the commitment scheme and argument system are secure against quantum adversaries. Traditionally, such a generic instantiation is thought to come with impractical sizes and performances but recent advances in succinct arguments warrants a reconsideration. This is by no means straightforward as evidenced by the work of Agrawal et al. [1], who estimated that even with modern proof systems, such a proof is at least 100 KB in size with a prover complexity of *at least one hour*.

A subtlety in the security reduction demands that the proof system is not only a proof of knowledge but also satisfies Multi-Proof Online Extractability [9], where the extractor does not need to rewind a malicious prover and instead, only needs a list of Random Oracle queries made by the adversary and the proof to extract a satisfying witness. Fortunately, zkSTARKs [7, 50], the argument system we use already satisfy these requirements

Rate-Limited Privacy Pass. The flexibility of this new approach has applications beyond making Privacy Pass post-quantum friendly. It can also be used to add new functionality to the Privacy Pass protocol. One such functionality is rate limiting support, wherein token issuance to clients is rate limited without compromising on client privacy. This capability has a number of applications, including rate limiting via "metered paywalls," as well as limiting abusive client behavior.

The IETF is currently standardizing a version of rate-limited Privacy Pass [33], but the proposed design is limited in several respects. Our new approach can be used to construct a conceptually simpler rate-limited version of Privacy Pass. We refer the reader to Section 5 for a detailed discussion of the limitations with the existing design, as well as an overview of our proposed design.

Our Contributions. We show that when instantiated with the right proof system and with careful optimizations, such a scheme can perform surprisingly well. As such, the main focus of our work is to efficiently verify a post-quantum signature using a post-quantum proof system. Blind Signatures and Anonymous Credentials follow almost for free.

- We design and implement a plausibly post-quantum anonymous credential scheme as described above using zkDilithium, our protocol for a STARK-friendly variant of Dilithium2 [26]. To the best of our knowledge, this is the first implementation of a post-quantum anonymous credential scheme and we believe that this will serve as a valuable baseline for future research. We use zkSTARKs for the proof system which allows for a trade-off between proof size (85–175 KB) and prover time (0.3–5 s) for a conjectured proof security level of 115 bits. The proof can be verified in 20–30 ms and the size is comparable to the best known post-quantum blind signature schemes (22–100 KB) [1, 10, 24], which is a weaker primitive implied by anonymous credentials [4]. (Section 3)

- We show how anonymous credentials together can be used to improve Privacy Pass. First, we provide a template for a rate-limiting protocol that does

not require an Attester and guarantees better privacy for clients. Our credentials can also be limited to $k-$show which implies a Privacy Pass protocol where an issuer can give a client k tokens with $O(1)$ work and transcript size. (Section 5)

- We present the findings of a measurement study in collaboration with a large content delivery network (CDN) to determine the impact of larger token sizes on client latency during redemption in the Privacy Pass protocol. We conclude that a vast majority of users have an *effective* upload speed of at least 1 Mbit/s and hence these tokens add roughly 0.5–1.5 s to the upload time over traditional tokens (based on RSA). (Section 6)

1.1 Overview

We begin with a brief introduction to zkSTARKs and the Arithmetic Intermediate Representation (AIR) which defines the corresponding language for which arguments are produced (Section 2). Next we first recall the Dilithium post-quantum signature scheme (Section 3) and present our STARK friendly modification zkDilithium in Section 3.4. We evaluate the performance of our anonymous credential scheme in Section 4 and demonstrate the flexibility of the scheme by considering a rate-limited variant Section 5. Finally, we discuss results of a measurement study to investigate the impact of larger token sizes in Section 6.

2 zkSTARKs and AIR

A Scalable Transparent ARgument of Knowledge (STARK) is a non-interactive argument system, which creates proofs for log-space computations requiring $T(n)$ time and $O(\log T(n))$ space in $O(T(n) \cdot \log T(n))$ time, which can be verified in $O(\log(T(n)))$ time, for an instance of size n. These log-space computations are formally modeled by the language of AIRs. In this section we give a brief introduction to AIR to highlight its relevant features and refer the reader to [39] for more high level details, but leave a proper introduction to [7, 50].

Arithmetic Intermediate Representation (AIR). In its basic form, in AIR, we model the computation as a matrix of field elements, called the *execution trace*. The name makes sense, if one thinks of the columns as registers and rows as time-steps, but it'll become clear later that this is not a perfect analogue. We describe the computation using two kind of constraints. The *boundary constraints* fix the value of a particular cell to a given public value. The heavy lifting is done in the *transition constraints*, where each constraint is a polynomial equation that describes how two consecutive rows relate. In fact, STARKs use a uniform computation model which requires *all* transition constraints to hold for *every* row.

As an example, suppose want to show that $F_{15} = 610$, where F_n are the Fibonacci numbers. As execution trace, we use a 8×2 matrix filled as follows.

a	b	Boundary
0	1	$a_1 = 0$
1	2	$b_1 = 1$
3	5	$b_8 = 610$
8	13	
21	34	Transition
55	89	$a_{i+1} = a_i + b_i$
144	233	$b_{i+1} = a_{i+1} + b_i$
377	610	

We use three boundary constraints. The first two start the sequence with initial values and the third one asserts the desired conclusion that $F_{15} = 610$. The transition constraints ensure that a and b are updated according to the sequence formula.

Uniform computation. It is easy to specify constraints when the same computation is repeated every time (as in the Fibonacci sequence). However, performing different calculations complicates the design of transition constraints as they must be applied to every row.

One way to handle multiple calculations under a uniform computation is using selectors or multiplexers which publicly known values. For example, suppose we want to additionally show that ${F_{15}}^{27} = 610^{27}$. To keep transition degrees low, we compute the power as three consecutive cubes. The selector m allows switching between the computation of the Fibonacci sequence and the calculation of cubes.

a	b	m
0	1	0
1	2	0
⋮	⋮	⋮
377	610	0
	610^3	1
⋮	610^9	1
	610^{27}	1

Boundary constraints
$a_1 = 0 \quad b_1 = 1 \quad b_{11} = 610^{27}$
$m_i = \begin{cases} 0 & i \leq 8 \\ 1 & i > 8 \end{cases}$

Transition constraints
$(a_i + b_i - a_{i+1})(1-m) = 0$
$b_{i+1} = (a_{i+1} + b_i)(1-m) + {b_i}^{3} m$

Time vs Space. A common strategy that can be employed in the design of AIRs is to use more registers to reduce the multiplicative degree of transition constraints. In the above example, one could have used a transition constraint of the form $b_{i+1} = (a_{i+1} + b_i)(1-m) + {b_i}^{27} m$ with just one extra row instead of three but this would mean the transition constraint has a multiplicative degree of 27 instead of 3 which in turn affects prover time[1].

The finite field. STARKs are easiest to implement efficiently in a finite field that has a primitive root of unity (PROU) of order 2^k, for large enough k. Also, to ensure soundness of the proof, field extensions may have to be introduced to make the field sufficiently large. Looking ahead, we use the finite field $\mathbb{F}_{q^6}$, where $q = 2^{23} - 2^{13} + 1$ is the prime used in Dilithium. We construct it as a quadratic extension on top of a cubic extension.

Non-native arithmetic. Emulating arithmetic modulo M (as we will need it later on) when working in $\mathbb{F}_q$ takes some care. For instance, suppose we want to compute $x = x' \bmod M$. We add a new register Q to store $\lfloor \frac{x}{M} \rfloor$ and use the constraint $x + M \cdot Q = x'$. This is not sufficient, as we can now set x to any value we like, by setting Q to $(x' - x)M^{-1}$, where the inverse of M is computed in $\mathbb{F}_q$. To prevent these shenanigans, we need to ensure $0 \leq Q \leq \lfloor \frac{q}{M} \rfloor$. So, how do we do range proofs?

An approach we followed is to show the binary representation of Q. We add $m = \lceil \log_2 \frac{q}{M} \rceil$ new registers Q_0, $\ldots$, Q_{m-1} with constraints $Q_i(Q_i - 1) = 0$, which ensure $Q_i \in \{0,1\}$; and the constraint $Q = \sum_i 2^i Q_i$. This ensures $0 \leq Q \leq 2^m - 1$ and thus ticks off $0 \leq Q$. To ensure $Q \leq \lfloor \frac{q}{M} \rfloor$, we repeat for $Q' = Q + 2^m - \lfloor \frac{q}{M} \rfloor - 1$.

Verifier Injected Randomness. Verifying Dilithium signatures involves showing polynomial equality. This can be done efficiently, by evaluating on a challenge point. Hence, we would like random coins supplied by the verifier as part of the execution trace. This is possible, but needs to happen in multiple stages: the prover first sends a main execution trace, the verifier then sends a challenge and the prover responds with another *auxiliary* execution trace. Using the Fiat–Shamir transform [27] this can be made non-interactive. We refer to these traces as Randomized AIRs with Preprocessing (RAPs).

For example, a prover wants to convince a verifier that they know two polynomials $f(X)$ and $g(X)$ such that $f(X)g(X) = h(X)$. The naive strategy is to implement schoolbook polynomial multiplication in the AIR which involves $O(d^2)$ operations if f and g are degree-d polynomials. A natural improvement is to use NTT to compute the polynomial multiplication in $O(d \log d)$ operations. However, with the added power of verifier injected randomness the prover can convince a verifier with overwhelming probability in just $O(d)$ operations using polynomial identity testing. The prover first commits to the coefficients of f, g and h in the main trace and evaluates these polynomials in the auxiliary trace to show that $f(\xi)g(\xi) = h(\xi)$, where ξ is a random point chosen by the verifier. Note that a malicious prover can fool a verifier with probability at most $2d/|\mathbb{F}|$.

3 Dilithium

Dilithium [26] is a lattice-based post-quantum signature scheme, selected for standardization by NIST [44]. The full scheme contains many optimizations and a full exposition would be too involved. Instead, here, we discuss the key aspects of the scheme [49].

3.1 Parameters

Dilithium uses a polynomial ring $\mathscr{R} = \mathbb{F}_q[x]/(x^n + 1)$, which is the set of polynomials of degree less than n with coefficients in $\mathbb{F}_q$, where $n = 256$ and $q = 2^{23} - 2^{13} + 1$.

The "size" of polynomials plays a crucial role, which is precisely defined as follows. For any $a \in \mathbb{F}_q$, define $a' = a \bmod^{\pm} q$ to be the unique integer a' in the range $-\frac{q-1}{2} \leq a' \leq \frac{q-1}{2}$ such that $a \equiv a' \pmod q$. For any polynomial $p(x) = \sum_i p_i x^i \in \mathscr{R}$, the norm of $p(x)$ is defined as $\|p(x)\|_\infty = \max_i |p_i \bmod^{\pm} q|$. Similarly, for a vector v over $\mathscr{R}$, we define $\|v\|_\infty = \max_i \|v_i\|_\infty$.

A Dilithium's private key consists of two vectors $(s_1, s_2) \in \mathscr{R}^\ell \times \mathscr{R}^k$ sampled uniformly at random such that $\|s_1\|_\infty, \|s_2\|_\infty \leq \eta$, where η, k and ℓ are constants depending on the security level. The corresponding public key is (A, t), where A is a random $k \times \ell$ matrix over $\mathscr{R}$, and $t = As_1 + s_2$. The hardness of the *module learning-with-errors* (M-LWE) problem [15] ensures that it is difficult to recover s_1 and s_2 from A and t.

Let v be a vector over $\mathscr{R}$, the functions $\mathsf{HighBits}(v)$ and $\mathsf{LowBits}(v)$ decompose v uniquely as

$$v = \mathsf{HighBits}(v) \cdot 2\gamma_2 \; + \; \mathsf{LowBits}(v)\,,$$

such that $-\gamma_2 < \mathsf{LowBits}(v) \leq \gamma_2$, for a parameter γ_2.

[1] Lower transition degrees allow for a more efficient Prover.

In this paper we focus on the Level 2 of security set by NIST [44], which specifies the following constant values: $\eta = 2$, $k = \ell = 4$, $\gamma_1 = 2^{17}$, $\gamma_2 = \frac{q-1}{88}$, and $\tau = 39$, and $\beta = \eta\tau = 78$.

3.2 Underlying identification scheme

Dilithium is based on the following *identification scheme* where a prover, having access to the private key, demonstrates this fact to a *verifier* that knows the public key, without leaking any information. The protocol, shown in Fig. 2, runs as follows.

1. The prover samples at random a secret nonce[2] $y \in \mathscr{R}^\ell$ of norm $\leq \gamma_1$, and sends the *commitment* $w_1 = \mathsf{HighBits}(Ay)$ to the verifier. Note that it's crucial that the prover only sends the higher bits of Ay, as it would otherwise leaks y since A is likely to be invertible.

2. After receiving w_1, the verifier returns a random *challenge* $c \in \mathscr{R}$ with exactly τ non-zero coefficients, all either -1 or 1.

3. The prover now computes the *response* $z = y + cs_1$. Note that cs_1 is not so large as $\|cs_1\|_\infty \leq \beta$. Before returning z, the prover performs two checks on the sizes of z and $r_0 = \mathsf{LowBits}(Ay - cs_2)$, whose importance becomes clear later on.

$$\|r_0\|_\infty \leq \gamma_2 - \beta \qquad (r_0\text{-check})$$
$$\|z\|_\infty \leq \gamma_1 - \beta \qquad (z\text{-check})$$

   If any of these checks fails, the prover aborts and starts again from the beginning, generating a new nonce y.

4. Finally, when eventually receiving a response (after, on average, three attempts), the verifier calculates $w_1' = \mathsf{HighBits}(Az - ct)$ and accepts whenever $w_1' = w_1$, and $\|z\|_\infty \leq \gamma_1 - \beta$.

Without the checks, the scheme would not always work. In general $w_1' \equiv \mathsf{HighBits}(Az - ct) = \mathsf{HighBits}(Ay - cs_2) \neq \mathsf{HighBits}(Ay) \equiv w_1$ as even though cs_2 has small coefficients (also $\leq \beta$), they might still carry into the higher bits. The problem is solved by ensuring y does not overflow, which is the purpose of (r_0-check) (see [26, Eq. 3]). A different issue is that z might leak information on y and s_1 if it has large coefficients. The (z-check) prevents this. Indeed, this scheme is perfectly non-abort zero-knowledge [49, §2.1].

[2]Nonce, as in 'number only used once' is misleading: y is neither a number nor is its single use the only requirement that it should satisfy.

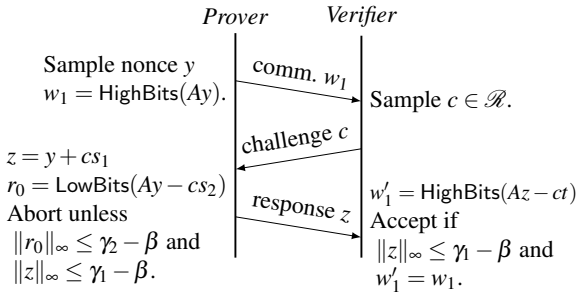


Figure 2: Dilithium's identification scheme.

3.3 Dilithium signatures

This identification scheme can be turned into a signature scheme using the well-known Fiat–Shamir transform [27]. A signature on M is given by a pair (c, z) of a challenge c and response z of a successful interaction of the identification scheme, where the challenge is computed as $c = H(A \parallel t \parallel M \parallel w_1)$ for some hash function H that ranges over the challenge space. To check an alleged signature (c, z), a verifier makes sure that $\|z\|_\infty \leq \gamma_1 - \beta$, computes $w_1' = Az - ct$ and checks whether $c = H(A \parallel t \parallel M \parallel w_1')$.

Some optimizations. The full scheme is rather more complex, as it contains many modifications to improve efficiency. We point out some of these, that are relevant to this work, and leave the full details to Dilithium's specification [26].

For starters, the size of A is reduced by deriving it deterministically from a seed ρ. Furthermore, only the higher bits of t are included in the public key. To be able to correctly verify signatures without the lower bits of t, a hint h is included in signatures. The ring $\mathscr{R}$ is isomorphic to $\mathbb{F}_q^{256}$ via the number-theoretic transform (NTT), which allows for faster multiplications. To speed up operations, A is sampled in the NTT domain.

Finally, to decrease the size of the coefficients of w_1 ever so slightly, the exception $\mathsf{LowBits}(-\gamma_2) = -\gamma_2$ is admitted, so that we can set $\mathsf{HighBits}(-\gamma_2) = 0$ instead of $\frac{q-1}{\gamma_2}$.

3.4 zkDilithium: a zkSTARK-friendly alternative

It is challenging to verify *unmodified* Dilithium signatures in zero-knowledge with zkSTARKs. Ideally one would use a zkSTARK defined over Dilithium's base field $\mathbb{F}_q$, to avoid expensive non-native modular arithmetic. However, the maximum execution trace depth in zkSTARKs is limited by the order of the PROU in the base field. With Dilithium's $q = 2^{23} - 2^{13} + 1$, the largest

PROU has order 2^{13}, which severely restricts the size of the AIR. Its extensions do not change the situation by much. This budget is so tight, that it's difficult to evaluate the SHA3 hash used to derive the challenge c.

Instead, we propose *zkDilithium*, a modified version of Dilithium2 that trades efficiency against STARK-friendliness, while not affecting security as compared to unmodified Dilithium (barring one concession.) At a high level, we make the following changes.

1. Instead of SHA3, we use the STARK-friendly hash Poseidon [32] with $\alpha = -1, t = 35, R_f = 21, R_p = 0$ and rate 24. Algebraic hashes, such as Poseidon, have not received the same scrutiny as SHA3, and this is the only concession to the security we make.

2. We do not use public key compression and distribute the full t in the public key. This removes the need for the hint h in the signature.

3. We change the *sample in ball* subroutine used to sample the challenge c, so it is easier to do in zero-knowledge. We describe later in this section.

These changes are sufficient to verify *zkDilithium* signatures using zkSTARKs with the original q.

Sample In Ball. Sample in ball, is the deterministic algorithm used to generate the challenge c from $\tilde{c} = H(M \| w_1)$. Recall that the challenge is a polynomial with $n = 256$ coefficients, of which $\tau = 39$ non-zero, all either -1 or 1. For security it is important that c is sampled uniformly.

This is done in essence by initializing $c = 0$, sampling the last τ coefficients of c from $\{1, -1\}$ and then applying a random permutation. Because the majority of c is zero, this can be done efficiently all at once using the truncated inside-out version of the Fisher–Yates shuffle algorithm shown in Figure 1.

Algorithm 1 Sample In Ball [26]

Initialize $\mathbf{c} = 0^{256}$
for $i = 256 - \tau$ **to** 255 **do**
  $j \leftarrow \{0, 1, \ldots, i\}$
  $s \leftarrow \{0, 1\}$
  $c_i = c_j$
  $c_j = (-1)^s$
end for

In plain Dilithium, the random numbers required in this algorithm are taken from XOF($\tilde{c}$) using rejection sampling. For instance, to get $j \in [0, i]$, a uniformly random $j \in [0, 255]$ is taken from the XOF output and rejected if $j \notin [0, i]$. Proving statements in zero-knowledge about rejection sampling is difficult and inefficient as the computation has variable length depending on the input. One solution is for the signer to try create signatures until it finds one where no rejection occurs in the derivation of $\tilde{c}$. This requires to many tries to be practical.

Instead, we change the method of sampling of $\tilde{c}$ so that rejections are rare. Recall that we use Poseidon as XOF and hence we have a stream of 23-bit field elements at our disposal. We sample a field element h and then compute $j = h \bmod (i+1)$. If the modulus q of the field is divisible by $i+1$, then j has the desired distribution. Of course, since q is prime, this is not the case and we're more likely to sample low values of j as there exists a remainder of q when dividing by $i+1$.

Thus to correct this, consider $q' = q - (q \bmod (i+1))$. If we reject h that are smaller than q', then j has the correct distribution.

We sample eight signs at the same time from a single field element in a similar way. The probability of a rejection occurring in the full derivation of c is less than 0.06%.

For uniformity it is convenient to bump τ from 39 to 40, so that we can process swaps in batches of eight.

3.4.1 zk^{20}Dilithium

Although verification of zkDilithium fits comfortably in the limits posed by the order-2^{13} PROU, the final applications might be more demanding. For those, we propose *zk^{20}Dilithium*, a variant of zkDilithium with $q = 2^{23} - 2^{20} + 1$, which allows for PROU of order 2^{20}. To accommodate this new q, we only need to change $\gamma_2 = 2^{16}$. Just like zkDilithium, we use $\tau = 40$. These changes are conservative: using the same analysis [25] as used for Dilithium, we reach a slightly higher security level, see Fig. 3. The trade-off is that the (r_0-check) fails more often and signing is roughly twice as slow for zk^{20}Dilithium as it is for zkDilithium.

3.5 Signature verification RAP

We now discuss the implementation of a RAP for the zk^{20}Dilithium verification circuit. Recall from Section 3 that verification of a signature $(z, \tilde{c})$ on a message M consists of the following steps:

Compute:	Check:
$c = \mathsf{SampleInBall}(\tilde{c})$.	$\tilde{c} \stackrel{?}{=} H(\mu \,\|\, w_1)$.
$w = Az - ct$.	$\|z\|_\infty \stackrel{?}{<} \gamma_1 - \beta$
$w_1 = \mathsf{HighBits}(w)$.	
$\mu = H(H(\rho \,\|\, t) \,\|\, M)$.	

We first describe a high-level overview of the organization of the signature verification RAP and then describe in detail how each of the above checks is carried out.

Cost	Dilithium2	zkDilithium	zk^{20}Dilithium
BKZ block-size b to break SIS	417	417	430
Best Known Classical bit-cost	121	121	125
Best Known Quantum bit-cost	110	110	114
Best Plausible bit-cost	86	86	89
BKZ block-size b to break LWE	422	422	427
Best Known Classical bit-cost	123	123	124
Best Known Quantum bit-cost	111	111	113
Best Plausible bit-cost	87	87	88

Figure 3: Security estimates generated using [25].

RAP Organization. In Fig. 4, we pictorially represent the execution trace and show areas in which various components are executed. The figure is approximately to scale where size of each component denotes the number of registers used in the trace. We require 408 rows to implement the constraints but pad this to 512 rows as the Winterfell library requires all execution traces to be padded to a power of two. The main trace contains 701 columns and the auxiliary trace contains 14 columns. The main trace is part of the first message from the prover to the verifier who responds with uniformly random coins and the auxiliary trace is part of the second message from the prover which can make use of verifier injected randomness. The protocol is made non-interactive using the Fiat–Shamir transformation.

We divide the trace into two parts a) sample in ball and b) polynomial multiplication separated by a green line in Fig. 4. Throughout the trace we have a dedicated set of registers for storing the challenge polynomial c and a set of registers where the Poseidon hash is being continuously computed. All areas shaded by red lines are *unused* registers.

During the sample in ball phase, we use a set of n registers denoted by `SWAP` to aid in the computation by lowering the maximum degree of transition constraints. Finally, a set of registers are used for range proofs to emulate non-native modular arithmetic. In the polynomial evaluation phase, we evaluate polynomials at a random point in the extension field using the auxiliary trace after the coefficients have been *committed* to in the main trace. Simultaneously we carry out range proofs on the coefficients of z and hash the coefficients of w.

Poseidon Hash. We use a Poseidon hash function over the appropriate field for zkDilithium and zk^{20}Dilithium, with a state width of 35 registers, out of which 11 registers are for the capacity and 24 are for the rate. The shuffle boxes are implemented using the inverse operation as it minimizes the total number of rounds and the constraint relations have multiplicative degree 3. Indeed our state width is quite wide but this allows us to reduce the total depth of our computation and hence improve the prover time at the cost of slightly larger proofs. Using the security estimates from [32] and the scripts provided in [47], we determine that 14 full rounds and 6 partial rounds is the minimum required to protect against all known attacks in the literature. The authors in [32] state that "*... the same number of full rounds can be used instead of the partial rounds without decreasing the security, but this leads to substantially higher costs in our target applications*". Since we are using zkSTARKs which enforce uniform computation, switching between partial and full rounds is actually *more* expensive than using full rounds throughout as in every round both the full and partial rounds are computed, except a selector wire is used to project out the rounds of interest. Thus, we use 21 full rounds which is one more than the minimum required. We compute a full hash cycle in 7 execution steps of the AIR by computing three rounds in each step[3]. To prevent the degree from blowing up we also triple the state width to ensure the maximum degree of constraints remains the same. The execution trace is divided into two phases: sampling in ball and polynomial multiplication, which are discussed next.

Sample in Ball: The goal is to prove the correctness of the modified sampling algorithm described in Section 3.4. We allocate $n = 256$ resisters in every row to store the coefficients of the challenge polynomial c. We start with the all-zeros string in the first row and use Poseidon as an extendable-output function (XOF) with $\tilde{c}$ as the seed to compute random field elements that determine the positions to swap and signs in Algorithm 1. Recall, that the XOF outputs field elements on every 8-th step at the end of 7 hash cycles. We use one field element each to determine the positions to swap for the next 8 iterations by computing the modulus with respect

[3] We remark that our implementation can support 24 rounds without affecting prover performance where a full hash cycle can be computed in 8 steps of the AIR.

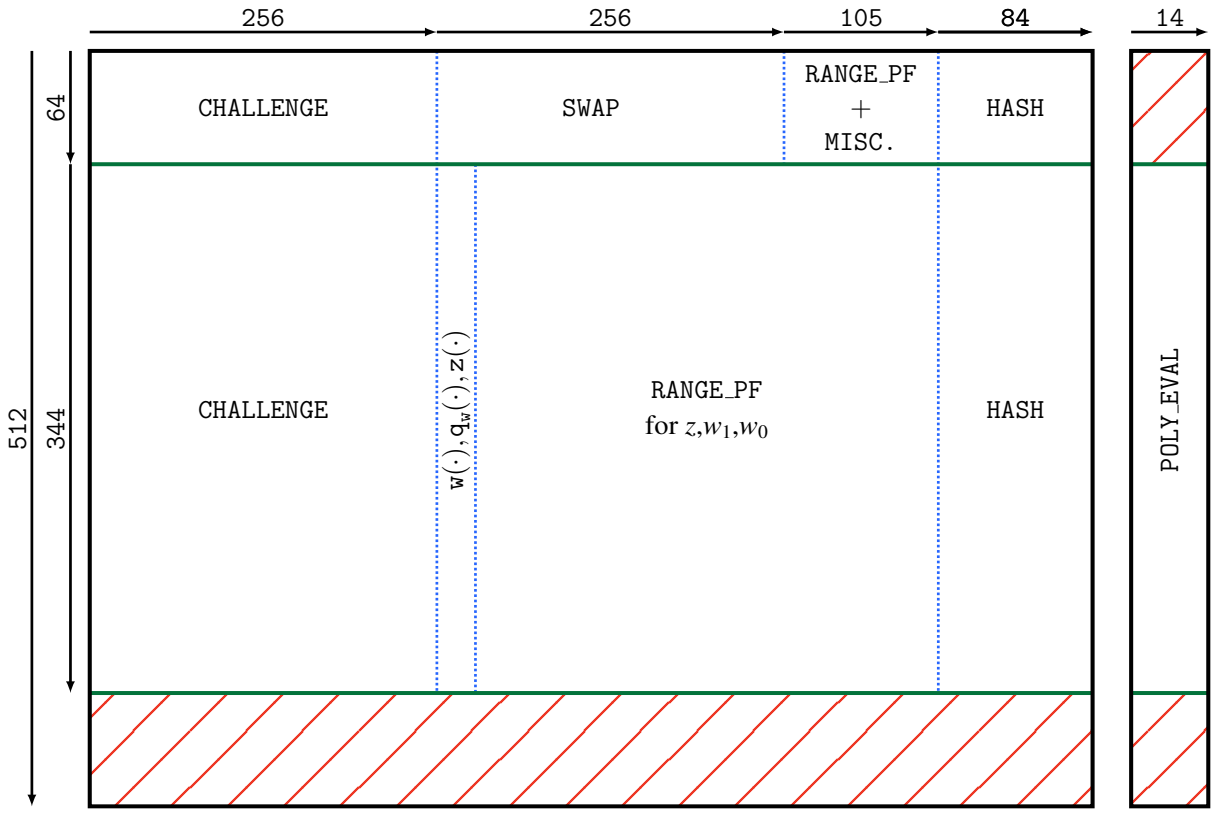


Figure 4: Execution trace for the signature verification RAP. Sizes of components are roughly to scale.

to the appropriate base and then using another field element to sample 8 bits for the signs. Unfortunately, this uses $O(n.T)$ space in the execution trace, for T execution steps and reducing this would indeed help reduce the overall proof size. Ideally, we would store the challenge in *column* format occupying only one register per row. But this makes it difficult to assert correctness of the computation $c = \mathsf{SampleInBall}(\tilde{c})$ as one would now need to write constraints with low multiplicative degree only involving adjacent coefficients of the challenge polynomial.

Polynomial multiplication: At the end of the Sample in Ball stage we have the challenge polynomial stored in n registers. We now need to prove correctness of the computation $\mathbf{w} = A\mathbf{z} - c\mathbf{t}$. One can compute the product of two polynomials which are elements of the ring $\mathscr{R} = \mathbb{F}_q[x]/(x^n+1)$ in $O(n\log n)$ time using the Number Theoretic Transform (NTT). However, implementing NTT in zkSTARKs is non-trivial due to the requirement of uniform computation. We sidestep this issue by observing that the prover can compute $\mathbf{w} = A\mathbf{z} - c\mathbf{t}$ in the clear using NTT and then use polynomial identity testing to prove the correctness in $O(n)$ time which is actually asymptotically optimal. The prover first computes the quotient polynomials $\mathbf{q}_w$ such that $\mathbf{q}_w(x^n+1) + \mathbf{w} = A\mathbf{z} - c\mathbf{t}$ and places the coefficients of each of the polynomials $\mathbf{q}_w, \mathbf{w}, \mathbf{z}$ in a column of the execution trace using one register per row per polynomial. In the auxiliary table, the verifier evaluates these polynomials at a random point in the extension field chosen by the verifier. At the end of n rows, we assert that both sides of the following equation are equal

$$A(r).\mathbf{z}(r) - c(r).\mathbf{t}(r) \;=\; \mathbf{w}(r) + \mathbf{q}_w(r)(r^{256}+1).$$

Note that the maximum number of *bad* challenge points for which the prover can cheat is the degree of the polynomial. Applying a union bound we can see that the probability with a cheating prover will fail in an interactive proof[4] is $1 - 1020/|\mathbb{F}_{q^6}|$ since we use a sextic extension field.

Computing $\tilde{c}$ and z-check While the values of $\mathbf{w}$ and $\mathbf{z}$ are available to us in the main trace, we immediately take

[4]The interaction in the IOP is removed through the Fiat-Shamir transformation but this allows a cheating prover to locally make multiple attempts at cheating with different randomness. Indeed this allows the prover to try multiple times but they are still bound to run in polynomial time.

care of $\|z\|_\infty \leq \gamma_1 - \beta$ and the computation of $\tilde{c}$. The z-check is performed by a straight-forward range proof. Recall that $\tilde{c} = H(\mu \| w_1)$, where $w_1 = \mathsf{HighBits}(w)$. We reuse the hashing registers used earlier for the computation of c. We process four coefficients of w per row, so that we have 24 coefficients of w_1 ready to absorb per Poseidon cycle. The computation of HighBits is somewhat tricky because of the corner case $-\gamma_2$, where $\mathsf{HighBits}(-\gamma_2) = 0$ and $\mathsf{LowBits}(-\gamma_2) = -\gamma_2$. To account for it, we use the following constraints:

$$
\begin{aligned}
w_0 + w_1 2\gamma_2 - w_2\gamma_2 &= w & w_2(w_2 - 1) &= 0 \\
0 \leq w_1 &< \frac{q-1}{2\gamma_2} & w_1 w_2 &= 0 \\
-\gamma_2 < w_0 &\leq \gamma_2 & w_0 w_2 &= 0
\end{aligned}
$$

where w_1 contains $\mathsf{HighBits}(w)$, w_0 contains $\mathsf{LowBits}(w)$, and w_2 is binary a flag that is true when in the corner-case $w = -\gamma_2$. A proof that these constraints do their job can be found in appendix A.

4 Implementation and Evaluation

We implemented our anonymous credential scheme using the Winterfell library [40] for STARKs and can be found at https://github.com/guruvamsi-policharla/pq-anon-creds. We chose zkSTARKs as the post-quantum proof system, mainly because the availability of an open-source, actively maintained library called Winterfell [40] for STARK proofs, which we extend and modify to fit our needs. Other proof systems exist such as plonky2 [51], which combines PLONK and FRI, however we found it less accessible as it lacks documentation and also did not have an open source license at the time of development of this work. Exploring alternate proof systems, such as [11, 20, 30], is an interesting direction for future work.

Although there are tool chains for a more *natural* programming model instead of AIRs such as Cairo[5] in combination with Giza[6] they come with high overheads. Our primary goal is efficiency and hence we take up the tedious task of manually optimizing the AIR for our desired circuit.

Currently, the Winterfell library does not support computing proofs with zero-knowledge but they have plans to add this feature in the future. The performance of a zero-knowledge STARK version will be very close to that of plain STARK as the missing operations add very little overhead in comparison to the rest of the proof generation[7]. All experiments related to proof performance target 115 bits of conjectured proof security [50] and were run on a 2019 MacBook Pro with a 2.4 GHz Intel Core i9 processor and 16 GB of DDR4 RAM in single threaded mode[8]. All constraints are written in the prime field with $q = 2^{23} - 2^{20} + 1$ to verify zk^{20}Dilithium as it allows for a larger depth of the trace table and hence more flexibility for showing arbitrary predicates on anonymous credentials. For simplicity and ease of comparison with related work, we focus on the simpler case of blind signatures but emphasize that our framework is much more powerful and is therefore not entirely a fair comparison, yet performance is comparable. We refer the reader to Section 7 for more details on a comparison.

Scheme	Signature (KB)	Transcript (KB)
[1]	45	1.5
[24]	100	850
[10]	22	200†
Better size	85.6	
Balanced	112.3	2.4
Better time	173.3	

Table 1: Comparison between blind signature schemes and our work, separated by a gap. † denotes a conservative estimate.

Scheme	Prover (ms)	Verifier (ms)	Prover RAM (MB)
Better size	4822	19.8	235
Balanced	660	22.0	40
Better time	304	31.4	19

Table 2: Run times and peak memory usage for our blind signature scheme. Peak verifier RAM usage is under 5MB. Prior work does not have an implementation for comparison.

STARK proofs can be tuned for a trade-off between the size of proof and the time taken to create proofs. Given that there is no one-size (time) that fits all applications we provide three different variants of our Anonymous Credential scheme which are a) Size optimized b) Time optimized and c) Balanced between time and size. The size optimized variant is useful in low bandwidth situations and the time optimized variant is useful in weak device situations. Table 1 contains a comparison with prior work on blind signature schemes where Transcript size denotes the communication during the interactive protocol for obtaining a blind signature and Signature

[5] https://github.com/starkware-libs/cairo
[6] https://github.com/maxgillett/giza
[7] See https://github.com/facebook/winterfell/issues/9.

[8] Although Winterfell supports multi-threading execution, the performance does not seem to be consistent with the expectation that zk-STARKs are massively parallelizable. An optimized implementation of a multi-threaded prover is left as a future work.

size denotes the size of the final signature. We also discuss performance metrics in Table 2, where Prover is the work done by a client who requested a blind signature and Verifier is the server who finally verifies the signature at the end of a blind signature scheme. RAM usage for the verifier is low (< 5 MB) which is conducive for deployment on server-less architectures such as Cloudflare Workers, AWS Lambda or Google Cloud Functions. In Fig. 5 we plot the trade-off between prover time and proof size by increasing the `blowup-factor` parameter but decreasing `number-of-queries` for a fixed target conjectured security level of 115 bits[9].

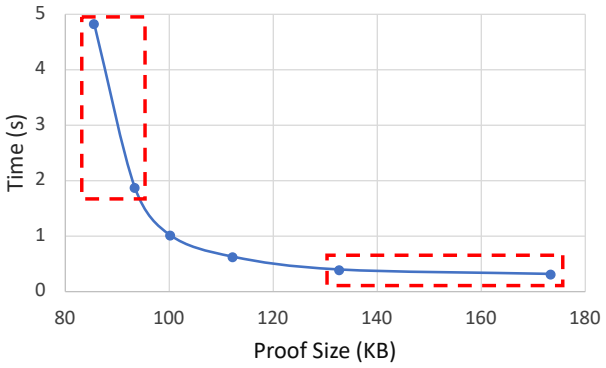


Figure 5: Trade-off between prover time and proof size. Towards the periphery, marks by dotted red boxes, the trade-off is unfavorable and we start to pay too much in proof size/time for too little return in proof time/size.

5 Improved Rate-Limiting for *Privacy Pass*

One of the main applications of the Privacy Pass protocol is to provide users a means for avoiding CAPTCHAs. A Client obtains anonymous tokens from an Issuer after establishing trust and redeems these tokens in place of solving CAPTCHAs. However, the basic Privacy Pass protocol [23] is stateless as there are no persistent client identifiers. This makes it difficult to enforce limits on the number of tokens that any individual client receives. In practice, such limits are quite useful. They can be used to implement application rate limits such as those found in "metered paywalls." They can also limit abusive behavior by individual clients, such as fraudulent account creation or token hoarding[10].

[9] See https://github.com/facebook/winterfell/tree/main/air for details on STARK parameters.

[10] Token hoarding is an attack wherein the adversary hoards tokens over a prolonged period of time and later uses them all at once to overwhelm an Origin server.

5.1 Existing work

To support per-client rate limits, a rate-limited version of Privacy Pass has been proposed [33]. This is a three party protocol involving an Attester, Issuer(s), and Clients and offers the capability to limit the number of tokens a particular Client can obtain for a particular Origin within a given window of time. We briefly summarize the protocol below, omitting some details but refer the reader to [33] for a detailed specification.

Similarly to the basic version, Clients interact with Issuers to obtain tokens. However, the Attester now acts as an intermediary party between the Client and Issuer, and performs an active role in token issuance. In particular, a) the Attester maintains a persistent identifier (unknown to the Issuer and Origin) for each client which enables rate-limiting[11] and b) the Attester serves as a proxy to hide the Origins for which a Client obtained tokens for while also not learning any information about Origins.

Clients initiate the protocol by sending a request for a token to the Attester who validates this request against an expected per-Client public key before forwarding the request to the Issuer. The Issuer processes and validates the request and, if valid, produces a token response for the Client. Composed with this token request and response transaction is a secure computation protocol to evaluate $y = \mathsf{PRF}(x,k)$, where x (Client's input) is a per-Client secret known only to the Client that corresponds to the Client's public key, k (Issuer's input) is a per-Origin secret known only to the Issuer. The Attester receives the output y without learning anything about x or k. Every time a Client visits the same origin, the Attester receives the same y and hence can measure the number of times a Client requested a token for an Origin in a given time frame. The Attester can now enforce rate-limiting policies on Clients by not forwarding the response from the Issuer if a Client exceeds its quotas.

The above solution requires additional trust assumptions beyond the basic version of Privacy Pass – non-collusion between the Attester and Issuers/Origins. Recall that in the basic Privacy Pass protocol the Issuer and Origin can collude and yet cannot identify which Client made a particular redemption request. In contrast, if the Attester colludes with the Issuer in the rate-limited protocol they can actually learn the Origins visited by a particular Client. If the Attester and Origin collude, they can correlate a Client's identity and Origin access patterns through timestamp correlation. Even without collusion, it is possible for a malicious Attester to combine the access pattern information it learns through the protocol with additional auxiliary information to try and learn the Origins associated with Client requests.

Moreover, the rate-limiting mechanism is simple and

[11] The persistent identifier must be resistant to Sybil attacks.

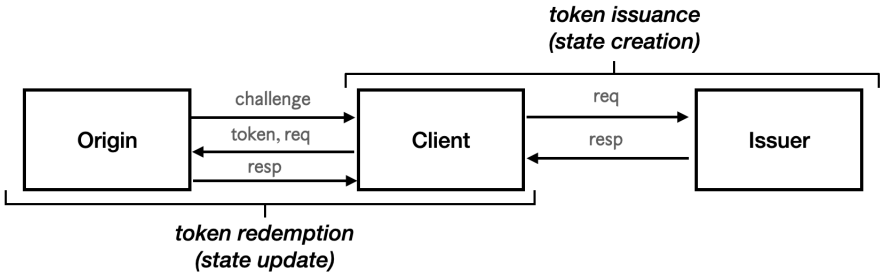


Figure 6: Protocol flow for rate-limited Privacy Pass with anonymous credentials.

inflexible in nature. From a functional perspective, rate limits are expressed in terms of integer quotas for a period of time, and thus cannot be used to support *dynamic* policies, such as those that actively adapt to changing traffic patterns. From a privacy perspective, the rate limit for a specific Origin cannot be unique as this would leak information about Client browsing behavior to the Attester. Finally, there is an inherent centralization problem as the number of issuers must be small in order to maintain a large anonymity set for Origins visited by a client [42].

5.2 Our proposal

Recall that in Privacy Pass (without rate-limiting), every time a client wishes to authenticate they must use a fresh token obtained from an issuer. In our proposal, the client is given one *credential* per origin by the issuer, which has hidden attributes containing details of user access patterns that will be used for rate-limiting. When the client visits an origin, the client must *spend* this credential and is in return issued a new credential with updated rate-limiting information. In effect, the rate limiting state of the client is stored in the tokens themselves rather than in a trusted Attester.

We now work through a concrete example to highlight the utility of anonymous credentials over blind signatures. Clients receive credentials with the following attributes:

1. Nonce. To prevent double spends.

2. Timestamp this token was last (re)issued.

3. Number of times the client used a token in the 5-minute window[12] when the token was last (re)issued.

4. Number of times the client used a token in the one hour window, but not 5-minute window, when the token was last (re)issued.

5. Number of times the client used a token on the day, but not the one-hour window, when the token was last (re)issued.

To get a completely new token, the client first establishes trust with an issuer, e.g., by solving a CAPTCHA. The client then sends a commitment with a new nonce, the current time, and zero counters, all as state, to the issuer, along with a proof that the counters are indeed all zero and that reveals the timestamp to the issuer. After verifying the proof, the issuer returns a signature on the commitment.

To redeem the token with an origin, the client first creates a new state, with a new nonce, the current time, and appropriately updated counters. Then it sends a commitment to the new state and the nonce of the old state to the origin together with a proof that

1. It knows a signature of either the origin or issuer on a commitment on the old state, revealing only the nonce therein.

2. That the counters under the new commitment are updated appropriately with respect to the old state, revealing only the new timestamp.

3. That the issuance satisfies a prescribed rate-limiting policy, for instance:

   (a) That the token was last used within three days;

   (b) that the number of uses per 5 minutes doesn't exceed 300;

   (c) that the number of uses per hour doesn't exceed 1000 and

   (d) that the number of uses in the last day doesn't exceed 5000.

[12] 5-minute windows start at round times, such as 17:05 and 17:10 (or more elegantly when the UNIX timestamps are zero mod 300), so that the state can be updated properly.

The origin checks the proof; that it has not seen the revealed old nonce before and that the revealed new timestamp is close to the current time. If all checks pass, it allows the request and returns a signature on the new commitment. The client then uses this new commitment and signature in subsequent redemption requests to the origin. The example policy, in the Privacy Pass case, still allows hoarding within a three day period and requires sybil attack prevention which can be achieved by having the Issuer uniquely identify the client.

6 Practical proof sizes for *Privacy Pass*

As we have seen, there is a trade-off between proof size and creation/verification time. To pick the right one for *Privacy Pass*, we executed the following experiment.

6.1 Setup

We modified Cloudflare challenge pages, so that on a small fraction of them, an experimental background request is sent, that uploads a dummy token of between 1 and 40 kilobytes. We time the upload and stored those for which the challenge was successful. This resembles the Privacy Pass flow. There are two notable limitations. First, we aborted a request if it took longer than one second, so that users are not unduly affected by the experiment. Secondly, we sent the request to a separate domain[13] that describes the experiment. The latter adds a significant amount of noise to the timing, as it is very likely the request also involves a DNS query and extra TLS handshake.

6.2 Results

During October 2022, 750,000 experimental requests were initiated. Of those 190,000 were aborted, because they didn't finish within a second. Fig. 7a shows the median duration by token size. Aborted requests are included as having a duration of a single second.

The jump at around 14kB is due to an extra round trip required when the client congestion window fills up. The lines are fit using ordinary linear regression, with a dummy factor to account for the congestion window jump. From this, we can estimate that the median effective upload speed is roughly 2.6Mbit/s 95% CI [2.4,3.0]. If we do the same for the other percentiles, we get Fig. 7b, with 95% CI shaded.

Above the 60$^{\text{th}}$ percentile, the fraction aborted requests is too high for a meaningful analysis.

Extrapolating by eye, it seems generous to assume the vast majority of users has an effective upload speed of more than 1Mbit/s, which would limit proof sizes to 100kB if we give ourselves a time budget of one second.

[13] https://privacy-pass-size-experiment.cloudflareresearch.com

7 Related Work

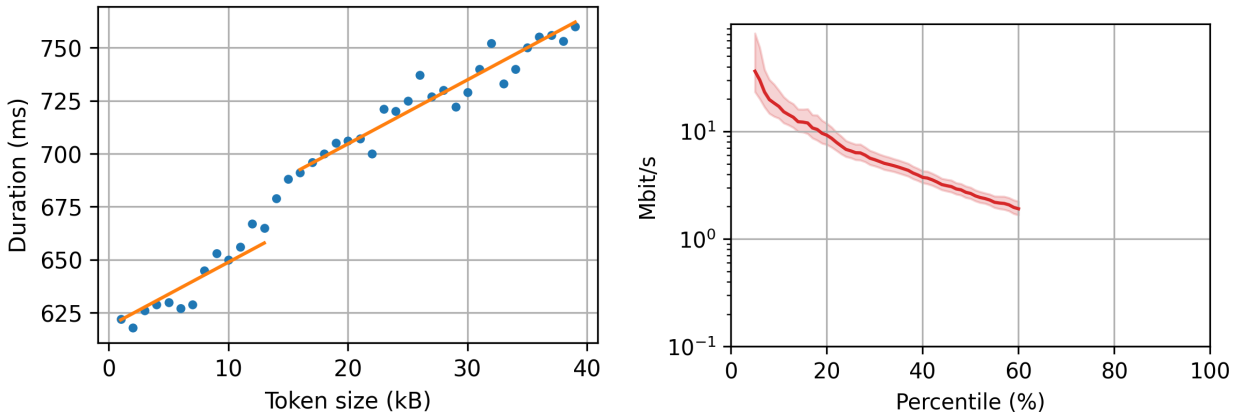
Post-quantum key-agreement [14] and signatures [8, 26, 29] have received a lot of attention in response to the NIST PQC standardization efforts which resulted in very efficient constructions, albeit still worse than *classical* counterparts based on DDH/RSA [52]. The situation for advanced cryptography such as vOPRFs and blind signatures is more complicated and more dire. Even if constructions being currently used are not *future-proof*, at the very least, capturing communication today should not allow an adversary, who later gains access to a quantum computer, to violate *privacy* guarantees. Fortunately, properties such as *unlinkability* of blind signatures or *obliviousness* of evaluation points in OPRFs are typically information-theoretically guaranteed in existing constructions [19, 36]. However, post-quantum schemes with performance comparable to counterparts based on DDH/RSA do not exist.

There has been a long line of theoretical work but we restrict our comparison to constructions which aim to be practical as they are most relevant to our work. We refer the reader to [1] for a more detailed survey. Recently, the problem has gained renewed attention with three *somewhat* practical proposals for lattice-based blind signatures [1, 10, 24] and one based on isogenies [5]. Unfortunately, none of the proposed solutions meet the requirements for a post-quantum variant of privacy-pass.

Given that one of the primary applications of anonymous tokens is to provide users with a seamless experience without interruptions in the form of CAPTCHAs [22], a post-quantum solution must ensure that the signer (server) workload is as low as possible to prevent denial of service attacks. Unfortunately, the construction with the smallest signature size ($\approx$ 22 KB) [10] requires the signature requester to produce a quantum-safe proof of a large circuit involving cryptographic hashes (SHA) and the authors estimate the proof to take under 20 seconds to create and under 10 seconds to verify using Ligero++ [12] with several hundreds of kilobytes in communication. The authors do not focus on this aspect as they are primarily concerned with signature size.

The scheme of [1] is based on a novel cryptographic assumption titled one-more-ISIS, where signature sizes are $\approx$ 45 KB with transcripts of a few kilobytes. Although it is interesting to explore new cryptographic assumptions in order to obtain more efficient constructions, they require more scrutiny before they can be considered for deployment in practice.

Finally, [24] introduced a blind signature based on standard lattice assumptions (SIS + LWE) with signa-



(a) Median request duration by token size.

(b) Estimate of effective upload speed.

Figure 7: Results of experimental measurements.

ture size $\approx$ 100 KB and total communication under a megabyte. Indeed this is only construction which a) does not prove statements involving a Random Oracle and hence does not resort to heuristic security and b) has a proof in the Quantum Random Oracle Model. However, the communication is still quite high and the authors do not give a performance estimate.

Baretto et al. [5] propose a blind signature scheme based on isogenies, but their signature size is roughly half a megabyte and both signing and verification require over 1000 group actions, each of which takes close to 40 ms [18], making the scheme quite expensive.

An alternate path to anonymous tokens (without public verifiability) is via vOPRFs and the only known construction of post-quantum vOPRFs from lattices [2] provides a crude estimate of $> 2^{40}$ bits of communication rendering the construction theoretical. Another construction [13] relies on the hardness of SIDH which was recently shown to be broken [17, 41, 46]. This has since been revamped by incorporating countermeasures against such attacks in [6] but the bandwidth requirements are still quite large at 8.7 MB per query. We remark here that another promising candidate for somewhat practical post-quantum vOPRFs is via secure two party computation of a PRF such as AES and prior work has shown that this can be achieved with $\approx$ 5 MB of communication [38] but this requires more than two rounds.

8 Outlook

We have proposed the first practical post-quantum anonymous credentials system, which performs competitively against best known constructions for the simpler case of blind signatures. We emphasize that carefully combining *off-the-shelf* constructions with small modifications can perform surprisingly well. However, we note that this approach is far from optimal. To name but a few lines of future research.

1. For this work, we committed to a variant of Dilithium as the underlying post-quantum signature scheme. Exploring other signature schemes such as Falcon [29] is an interesting direction for future work.

2. Exploring the performance of alternate proof systems such as [11, 20, 30] or even designing proof systems for the task of verifying post-quantum signatures is also very promising.

3. Formal verification techniques can be used to guarantee soundness of the translation from code to the AIR.

9 Acknowledgments

Acknowledgements We would like to thank Maria Eichlseder and Markus Schofnegger for their advise on STARK friendly hashes; Nikita Borozov for insightful discussions on rate-limiting; Chris Patton, Avani Wildani for their reviews; and finally Wouter Geraedts, for Rust advice.

References

[1] Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav. Practical, round-optimal lattice-based blind signatures. In *CCS*, pages 39–53. ACM, 2022. https://doi.org/10.1145/3548606.3560650.

[2] Martin R. Albrecht, Alex Davidson, Amit Deo, and Nigel P. Smart. Round-optimal verifiable oblivious pseudorandom functions from ideal lattices. In *Public Key Cryptography (2)*, volume 12711 of *Lecture Notes in Computer Science*, pages 261–289. Springer, 2021. https://doi.org/10.1007/978-3-030-75248-4_10.

[3] Apple. Challenge: Private access tokens. In *Apple Developer – Discover*, Jun 2022. https://developer.apple.com/news/?id=huqjyh7k.

[4] Foteini Baldimtsi and Anna Lysyanskaya. Anonymous credentials light. In *CCS*, pages 1087–1098. ACM, 2013. https://doi.org/10.1145/2508859.2516687.

[5] Paulo L. Barreto and Gustavo H. M. Zanon. Blind signatures from zero-knowledge arguments. Cryptology ePrint Archive, Paper 2023/067, 2023. https://eprint.iacr.org/2023/067.

[6] Andrea Basso. A post-quantum round-optimal oblivious prf from isogenies. Cryptology ePrint Archive, Paper 2023/225, 2023. https://eprint.iacr.org/2023/225.

[7] Eli Ben-Sasson, Iddo Bentov, Yinon Horesh, and Michael Riabzev. Scalable, transparent, and post-quantum secure computational integrity. Cryptology ePrint Archive, Paper 2018/046, 2018. https://eprint.iacr.org/2018/046.

[8] Daniel J Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The SPHINCS+ signature framework. In *Proceedings of the 2019 ACM SIGSAC conference on computer and communications security*, pages 2129–2146, 2019. https://doi.org/10.1145/3319535.3363229.

[9] Ward Beullens, Samuel Dobson, Shuichi Katsumata, Yi-Fu Lai, and Federico Pintore. Group signatures and more from isogenies and lattices: Generic, simple, and efficient. In *EUROCRYPT (2)*, volume 13276 of *Lecture Notes in Computer Science*, pages 95–126. Springer, 2022. https://doi.org/10.1007/978-3-031-07085-3_4.

[10] Ward Beullens, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Lattice-based blind signatures: Short, efficient, and round-optimal. Cryptology ePrint Archive, Paper 2023/077, 2023. https://eprint.iacr.org/2023/077.

[11] Ward Beullens and Gregor Seiler. LaBRADOR: Compact Proofs for R1CS from Module-SIS. Cryptology ePrint Archive, Paper 2022/1341, 2022. https://eprint.iacr.org/2022/1341.

[12] Rishabh Bhadauria, Zhiyong Fang, Carmit Hazay, Muthuramakrishnan Venkitasubramaniam, Tiancheng Xie, and Yupeng Zhang. Ligero++: A new optimized sublinear IOP. In *CCS*, pages 2025–2038. ACM, 2020. https://doi.org/10.1145/3372297.3417893.

[13] Dan Boneh, Dmitry Kogan, and Katharine Woo. Oblivious pseudorandom functions from isogenies. In *ASIACRYPT (2)*, volume 12492 of *Lecture Notes in Computer Science*, pages 520–550. Springer, 2020. https://doi.org/10.1007/978-3-030-64834-3_18.

[14] Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Kyber: a CCA-secure module-lattice-based KEM. In *2018 IEEE European Symposium on Security and Privacy (EuroS&P)*, pages 353–367. IEEE, 2018. https://doi.org/10.1109/EuroSP.2018.00032.

[15] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) Fully Homomorphic Encryption without Bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12, page 309–325, New York, NY, USA, 2012. Association for Computing Machinery. https://doi.org/10.1145/2090236.2090262.

[16] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 56–72. Springer, 2004.

[17] Wouter Castryck and Thomas Decru. An efficient key recovery attack on sidh (preliminary version). Cryptology ePrint Archive, Paper 2022/975, 2022. https://eprint.iacr.org/2022/975.

[18] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: an efficient post-quantum commutative group action. In *ASIACRYPT (3)*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018.

[19] David Chaum. Blind signatures for untraceable payments. In *CRYPTO*, pages 199–203. Springer, Boston, MA, 1982. https://doi.org/10.1007/978-1-4757-0602-4_18.

[20] Binyi Chen, Benedikt Bünz, Dan Boneh, and Zhenfei Zhang. Hyperplonk: Plonk with linear-time prover and high-degree custom gates. Cryptology ePrint Archive, Paper 2022/1355, 2022. https://eprint.iacr.org/2022/1355.

[21] World Wide Web Consortium. Private click measurement. https://privacycg.github.io/private-click-measurement/, April 2021.

[22] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: A privacy-enhancing protocol and browser extension. https://privacypass.github.io.

[23] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *Proc. Priv. Enhancing Technol.*, 2018(3):164–180, 2018. https://doi.org/10.1515/popets-2018-0026.

[24] Rafael del Pino and Shuichi Katsumata. A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. In *CRYPTO (2)*, volume 13508 of *Lecture Notes in Computer Science*, pages 306–336. Springer, 2022. https://doi.org/10.1007/978-3-031-15979-4_11.

[25] L. Ducas and J. Shanck. PQ-CRYSTALS: Security estimates. https://github.com/pq-crystals/security-estimates.

[26] Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. CRYSTALS-Dilithium: A lattice-based digital signature scheme. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pages 238–268, 2018. https://doi.org/10.13154/tches.v2018.i1.238-268.

[27] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *Advances in Cryptology — CRYPTO' 86*, page 186–194, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-47721-7_12.

[28] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. In *CRYPTO*, volume 4117 of *Lecture Notes in Computer Science*, pages 60–77. Springer, 2006. https://doi.org/10.1007/11818175_4.

[29] Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Prest, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon: Fast-fourier lattice-based compact signatures over ntru. *Submission to the NIST's post-quantum cryptography standardization process*, 2018. https://falcon-sign.info/.

[30] Alexander Golovnev, Jonathan Lee, Srinath Setty, Justin Thaler, and Riad S. Wahby. Brakedown: Linear-time and post-quantum SNARKs for R1CS. Cryptology ePrint Archive, Paper 2021/1043, 2021. https://eprint.iacr.org/2021/1043.

[31] Google. Private state tokens. In *Chrome Developers*, May 2021. https://developer.chrome.com/en/docs/privacy-sandbox/trust-tokens/.

[32] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. Cryptology ePrint Archive, Paper 2019/458, 2019. https://eprint.iacr.org/2019/458.

[33] Scott Hendrickson, Jana Iyengar, Tommy Pauly, Steven Valdez, and Christopher A. Wood. Rate-Limited Token Issuance Protocol. Internet-Draft draft-ietf-privacypass-rate-limit-tokens-00, Internet Engineering Task Force, September 2022. Work in Progress.

[34] Sharon Huang, Subodh Iyengar, Sundar Jeyaraman, Shiv Kushwah, Chen-Kuei Lee, Zutian Luo, Payman Mohassel, Ananth Raghunathan, Shaahid Shaikh, Yen-Chieh Sung, and Albert Zhang. Dit: De-identified authenticated telemetry at scale. Technical report, Facebook Inc., 2021. White Paper.

[35] Subodh Iyengar and Erik Taubeneck. Fraud resistant, privacy preserving reporting using blind signatures, 2021. Accessed 01-December-2021. https://github.com/siyengar/private-fraud-prevention.

[36] Stanislaw Jarecki, Aggelos Kiayias, and Hugo Krawczyk. Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014*, page 233–253, Berlin, Heidelberg, 2014. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-45608-8_13.

[37] Panos Kampanakis and Tancrède Lepoint. Do we need to change some things? open questions posed by the upcoming post-quantum migration to existing standards and deployments. Cryptology ePrint Archive, Paper 2023/266, 2023. https://eprint.iacr.org/2023/266.

[38] Jonathan Katz, Samuel Ranellucci, Mike Rosulek, and Xiao Wang. Optimizing authenticated garbling for faster secure two-party computation. In *CRYPTO (3)*, volume 10993 of *Lecture Notes in Computer Science*, pages 365–391. Springer, 2018. https://doi.org/10.1007/978-3-319-96878-0_13.

[39] Irakliy Khaburzaniya, Konstantinos Chalkias, Kevin Lewi, and Harjasleen Malvai. Aggregating and thresholdizing hash-based signatures using starks. In *AsiaCCS*, pages 393–407. ACM, 2022. https://doi.org/10.1145/3488932.3524128.

[40] Irakliy Khaburzaniya, François Garillot, Kevin Lewi, Konstantinos Chalkias, and Jasleen Malvai. Winterfell. https://github.com/novifinancial/winterfell, 2022.

[41] Luciano Maino and Chloe Martindale. An attack on sidh with arbitrary starting curve. Cryptology ePrint Archive, Paper 2022/1026, 2022. https://eprint.iacr.org/2022/1026.

[42] Mark McFadden. Privacy Pass: Centralization Problem Statement. Internet-Draft draft-mcfadden-pp-centralization-problem-03, Internet Engineering Task Force, March 2022. Work in Progress.

[43] Ministry of Public Health, Welfare and Sport. CoronaCheck. https://coronacheck.nl/.

[44] National Institute of Standards and Technology. NIST announces first four quantum resistant cryptographic algorithms, jul 2022. https://www.nist.gov/news-events/news/2022/07/nist-announces-first-four-quantum-resistant-cryptographic-algorithms.

[45] Privacy by Design Foundation. IRMA. https://privacybydesign.foundation/irma-en/.

[46] Damien Robert. Breaking SIDH in polynomial time. Cryptology ePrint Archive, Paper 2022/1038, 2022. https://eprint.iacr.org/2022/1038.

[47] Markus Schofnegger. Hadeshash: Reference implementations for various versions of Starkad and Poseidon. https://extgit.iaik.tugraz.at/krypto/hadeshash, 2021.

[48] Brave Software. Basic Attention Token (BAT) Blockchain Based Digital Advertising, 2021. https://basicattentiontoken.org/.

[49] Amber Sprenkels and Bas Westerbaan. Don't throw your nonces out with the bathwater: Speeding up dilithium by reusing the tail of y. Cryptology ePrint Archive, Paper 2020/1158, 2020. https://eprint.iacr.org/2020/1158.

[50] StarkWare. ethSTARK Documentation. Cryptology ePrint Archive, Paper 2021/582, 2021. https://eprint.iacr.org/2021/582.

[51] Polygon Zero Team. Plonky2: Fast Recursive Arguments with PLONK and FRI. https://github.com/mir-protocol/plonky2, 2022.

[52] Bas Westerbaan. Sizing up post-quantum signatures, Nov 2021. https://blog.cloudflare.com/sizing-up-post-quantum-signatures/.

A Constraints for HighBits

To prove the constraints used to compute w_1 from w listed at the end of Section 3.5 do their job, let's start with the corner case. Assume $w_2 = 1$. Then $w_1 = w_0 = 0$ by the $w_1 w_2 = 0 = w_0 w_2$ constraints. Hence $w = -\gamma_2$ by the first constraint. As mentioned $\mathsf{HighBits}(-\gamma_2) = 0 = w_1$, so that is correct. Conversely, if $w = -\gamma_2$, then we must have $w_2 = 1$, for if $w_2 = 0$, then there is not a w_0 in bounds for if $w_1 = 0$ then $w_0 = -\gamma 2$ and if $w_1 = \frac{q-1}{2\gamma_2} - 1$ then $w_0 = 1 + \gamma_2$.

Now, for the other case, assume $w_2 = 0$. Let some $w \neq -\gamma_2$ be given. Consider $w'_0 := w \bmod^+ 2\gamma_2$, where the modulus in computed on the standard representative of w in $\mathbb{Z}$. Set $w''_0 = w'_0$ if $w'_0 \leq \gamma_2$ and $w'_0 - 2\gamma_2$ otherwise. Note w''_0 satisfies the bounds of w_0. Also modulo $2\gamma_2$, we have $w''_0 \equiv w'_0 \equiv w$. Thus (the standard representative of) $w - w''_0$ is divisible by $2\gamma_2$. Set $w'_1 = \frac{w - w''_0}{2\gamma_2}$. Note $0 \leq w'_1 \leq \frac{q-1}{2\gamma_2}$. Set $w_1 = w'_1$, $w_0 = w''_0$ if $w'_1 \neq \frac{q-1}{2\gamma_2}$ and $w_1 = 0$, $w_0 = w''_0 - 1$ otherwise. Clearly w_0 and w_1 satisfy the constraints if $w'_1 = w_1$. In the other case $\frac{q-1}{2\gamma_2} = w'_1 \neq w_1 = 0$, we have $q - \gamma_2 \leq w < q$. Note $w'_1 2\gamma_2 = q - 1$ and so $w = w'_0 + w'_1 2\gamma_2 = w'_0 - 1 + q = w''_0 = w_0 + w_1 2\gamma_2$, where w_0 is in bounds as $w \neq -\gamma_2$.

We have shown that the intended w_0 for any w satisfies the constraints. Now we need to show it's the only that does. Note that there are $2\gamma_2$ possible values of w_0 by the range constraint. Similarly, there are $\frac{q-1}{2\gamma_2}$ possible values for w_1. Thus, there are $\frac{q-1}{2\gamma_2} \cdot 2\gamma_2 = q - 1$ possible pairs (w_0, w_1). That's exactly the number of possible values of w, except for the corner case. Thus the representation is unique.