

Block-A-Mole: The Sustainability Frontier of Moving-Target Censorship Resistance

Anindya Maiti
am@ou.edu
University of Oklahoma
Norman, Oklahoma, USA

Scott Seidenberger
scotty@cornell.edu
Cornell University
Ithaca, New York, USA

Abstract

Many censorship circumvention systems keep moving their servers to new cloud addresses, so that whatever the censor discovers soon goes stale. But clients often reach these servers through a name the censor can see and block, such as a hostname or a front domain, and moving a server to a new address does not give it a new name. Existing models treat each proxy as a single unit and leave open when such a system keeps working. We model addresses and names as two coupled layers and ask which one limits availability. Enough servers, rotated fast enough relative to how quickly the censor finds them one at a time, keep some address reachable, and we compute how fast is fast enough. Past that point, faster rotation buys almost nothing, and the names become the limit. If the censor burns names, blocking them for good, β times as fast as the system mints new ones, clients can connect less than a $1/\beta$ share of the time, however fast the servers move. For an illustrative system with eight servers and room for eight names, sessions lasting five mean mint intervals stay connected with probability at least 0.95 only while $\beta \leq 0.66$, the system’s sustainability frontier, which a censor that times its blocks can cut about tenfold. Public data show that China and Iran block platform names one tenant at a time or a whole suffix at once, that Russia orders such blocks, and that even Snowflake clients need names to obtain proxies. Our solver, simulator and data pipelines are openly available. Once rotation is fast enough, the race that matters is between minting and burning names, not between rotation and discovery.

Keywords

Censorship Circumvention, Moving-Target Defense, Timing Games, Name Blocking, Collateral Freedom

1 Introduction

Static circumvention endpoints can be enumerated. Tor bridges have been harvested at scale from their distribution channels [29], and China’s Great Firewall (GFW) blocked new Tor Browser bridges 2 to 36 days after release [12]. Moving-target systems instead move endpoints across commodity clouds so that the censor’s discovery goes stale. SpotProxy [24] keeps respawning proxies at new IP addresses. NinjaDoH [39] rotates a DNS-over-HTTPS (DoH) resolver’s addresses without domain names. CensorLess [22] regularly replaces its serverless-function bridges to avoid DNS and TLS Server Name Indication (SNI) blocking. These designs rely on collateral freedom [37], since blocking a cloud’s address ranges would also cut off its legitimate tenants.

Many deployments also depend on names, which censors have long blocked in DNS, HTTP and TLS traffic [17, 18], and Russia

and then China extended this to QUIC [16, 56]. We call a deployment *name-dependent* if a client cannot reach an endpoint without presenting a name the censor can observe and block persistently, such as a CensorLess function’s hostname or a front domain for the rendezvous of Snowflake [4] (Section 3). For such a deployment, rotation renews the address but not the name.

Proxy distribution when some users are censor agents has been studied with algorithms and game theory [9, 30, 33, 54], and simulations find that the rate of new proxies is key to availability [9, 33]. These models treat each proxy as one unit. Three questions therefore remain open for name-dependent deployments, namely when rotation wins, which resource binds, and whether the right lever is faster rotation or replacing burned names fast enough.

To answer them, we model such a deployment as two layers coupled through reachability (Section 4). On the address layer, the defender rotates endpoints to fresh addresses while the censor finds and blocks them, which per endpoint is FlipIt [48] with exponential strategies. On the name layer, the defender keeps a buffer of up to $k_{\max}$ live names and mints new ones at rate λ_{intro} , and the censor *burns* live names, that is, blocks them persistently, at an effective rate λ_{burn} that discounts what it fails to see. A client can connect while some endpoint has an unblocked address and some name is live. The *burn ratio* is $\beta = \lambda_{\text{burn}}/\lambda_{\text{intro}}$, and time is measured in mean mint intervals $1/\lambda_{\text{intro}}$.

Redundancy and rotation fast enough relative to discovery win the address layer, provided the censor discovers endpoints independently. The required speed, which we compute, rises with session length measured in mean address-discovery delays (Section 5.3). The name layer binds, since the fraction of time a client can connect stays below $1/\beta$ however fast endpoints rotate (Theorem 5.2). Hence no configuration keeps sessions of length T connected throughout with probability at least α once $\beta \geq 1/\alpha$ (Definition 4.2).

The largest burn ratio a deployment can sustain while meeting this target is its *sustainability frontier* $\beta^*(\alpha, T)$. In an illustrative configuration with eight endpoints, each with a mean discovery delay of one mint interval and rotated three times per mint interval on average, $\beta^*(0.95, 5)$ is 0.66 for a buffer of $k_{\max} = 8$ names and exceeds 1 only for $k_{\max} \geq 63$ (Section 5.2). These values hold against a censor that burns names at a constant rate. A censor that strikes only after finding every name in use, at the same discovery rate, lowers $\beta^*(0.95, 5)$ about tenfold, to 0.064, and a small hidden reserve of names recovers part of the loss (Section 5.4). Names are cheap to mint [35], but if the censor burns names on exposure, for example in public Certificate Transparency (CT) logs, minting faster helps only with names that stay unseen until use.

Public data ground these ingredients (Section 6). The unit a censor burns varies by censor and platform. China has blocked

workers .dev suffix-wide and Iran has likewise blocked pages .dev, yet both block github.io tenants one name at a time [18, 34, 41]. When Fastly ended support in March 2024 for the domain fronting used for Snowflake’s rendezvous [44], estimated Snowflake users worldwide fell 32.5% within a week while the number of proxy addresses barely changed [43, 46].

Our contributions are as follows.

- A two-layer timing model extending FlipIt to a second, burnable resource (Sections 3 and 4).
- The separation of the layers, with a rotation speed that suffices for the address layer and a $1/\beta$ cap on the name layer at any speed (Theorem 5.2 and Section 5.3).
- Closed-form availability, the cap, the exact frontier, and extensions for exposure at mint, strategic block timing and correlated takedowns (Section 5).
- Grounding of the model in public data on which names censors burn and on Snowflake’s rendezvous names (Section 6).
- An operating recipe that sizes rotation and sets the mint rate to keep β below β^* , with a consistent estimator of β (Section 7).
- An exact solver and an event-driven simulator, available with our data pipelines as an open testbed for variants beyond the closed forms (Appendix E).

Beyond the required speed, the operative control of a name-dependent deployment is the rate at which fresh names are minted relative to the rate at which the censor burns them, not the rotation interval. Operators can measure β and keep it below β^* .

2 Background and Related Work

Collateral freedom and its limits. Robinson, Yu and An coined the term *collateral freedom* for circumvention that works because blocking it would cost the censor too much [37]. In their leading example, the GFW poisoned DNS for the Google App Engine names under appspot.com but left the Google addresses that served GoAgent unblocked except for one day, because the same addresses served Google services that businesses needed [37]. Collateral protected the shared addresses, not the names on them, so name dependence is often the price of collateral protection. Domain fronting applies collateral freedom to names, since blocking a fronted connection means blocking the front domain with all its legitimate traffic [11]. Censors also accept large collateral at times. Iran has blocked entire top-level domains such as .il [41], and when Russia moved to block Telegram in April 2018, it blocked millions of cloud addresses and made unrelated services on Google and Amazon unreachable [36].

Timing games and moving-target defense. Moving-target defense (MTD) changes a system’s configuration over time to deny attackers a static target [5, 40], for example by mutating hosts’ IP addresses so that learned addresses go stale [1, 6, 21]. FlipIt models such contests as a timing game in which either player seizes a shared resource by moving and learns who held it only by moving [48], and Li and Zheng model MTD as a leader–follower game over when and where to migrate [28]. Per endpoint, our address layer is FlipIt with exponential strategies and no move costs (Lemma 4.3). FlipThem extends FlipIt to several resources, of which the attacker must hold all (AND) or any one (OR) to control the system [25]. Our address layer, which fails only when all n endpoints are blocked, is the AND case, whether the censor discovers endpoints independently (Lemma 4.3)

or all at once (Remark 4.4). In these games the defender wins every resource back by moving. The name is a second resource, which no move wins back once burned. Only minting replenishes the pool of live names, so availability depends on a birth–death process for that pool, coupled to the address layer through reachability.

Game-theoretic proxy distribution. Nasr et al. model proxy assignment as a game in which the censor decides at each step whether to block a discovered proxy or leave it open to identify its users, and in their simulations a system is eventually blocked below a critical arrival rate of new proxies [33]. SpotProxy and CensorLess are evaluated in this framework [22, 24], and CensorLess’s simulation pairs each bridge URL with a single address [22]. Fares et al. extend the framework to ephemeral proxies such as Snowflake’s, to censors that find proxies by traffic analysis and to multiple censors, and find that the arrival rate of proxies relative to users matters more than the censor’s strategy. They identify blocking-resistant control channels, which deliver proxy assignments to users, as a bottleneck their simulator does not capture [9]. Snowflake’s rendezvous is such a channel, and it depends on names (Section 6.2).

These models treat a proxy as one unit that the censor blocks as a whole. Ours complements them by taking discovery as an effective rate and separating the address, which rotation renews, from the name, which it does not. This separation yields a cap of $1/\beta$ on time-average availability that no rotation speed escapes (Theorem 5.2), and we obtain availability in closed form and the frontier β^* exactly rather than by simulation. The resulting minting condition $\lambda_{\text{intro}} \geq \lambda_{\text{burn}}/\beta^*$ (Section 7) is the name-layer counterpart of Nasr et al.’s critical arrival rate, tied to the burn rate rather than to client arrivals.

3 Threat Model and Scope

We consider a national censor with the capabilities documented for China’s GFW, Iran’s national filter and Russia’s TSPU, the deep packet inspection (DPI) devices that Russia’s regulator distributes to Internet service providers (ISPs) [3, 8, 41, 53, 56]. The defender’s moving-target deployment rotates cloud endpoints between addresses and, if clients reach them by name, replaces blocked names from a pool.

Position and capabilities. The censor sits at international gateways and inside ISP networks, where it can observe, drop and inject packets. Its DPI reads names from DNS queries, the HTTP Host header and the TLS SNI [17, 18, 51]. The TSPU switched to SNI-based censorship of QUIC between May 2022 and July 2023 [16], and the GFW has filtered QUIC by SNI since 7 April 2024 [56]. The censor finds endpoints by classifying traffic [50], probing suspected servers [7, 8] and harvesting distribution channels [12, 29], and it may learn a name before any client uses it, for example from CT logs (Section 6.3). It can block a discovered endpoint’s address, its name or both.

Blocks persist. A burned name stays burned. The GFW’s national name blocks last a median of at least 256 days, and lapses, as at Henan’s regional firewall, only help the defender [51]. An address block need only last while the endpoint keeps the address. The GFW renews it during use and lifts it about 12 hours after the service stops [7], so persistence is conservative for the GFW. Rotations are

assumed to land on unblocked addresses, which nearly holds while blocked addresses are a small share of the provider’s space.

Discovery is an effective rate. The censor need not see every connection or block everything it sees. The model summarizes discovery by two effective rates, λ_a for each endpoint’s address and λ_{burn} for the defender’s names (Section 4). Each is a rate of attempts times the probability that one finds and blocks its target. Attempts include probes, traffic classification, channel enumeration and agents posing as users [33]. Both factors are limited in practice. The GFW applies its filter for fully encrypted traffic only to about a quarter of the connections toward certain data-center ranges [50]. All model results are functions of these rates.

Collateral and provider actions. The collateral damage a censor accepts differs between normal operation and crisis, and collateral protects shared suffixes and front domains only partially (Section 6). The model lets the censor block any discovered address or name at no cost, which is conservative for the defender, and collateral enters through the granularity of burns (Section 6.1). Blocking whole address ranges in a crisis, as Russia did in 2018 (Section 2), is a regime change outside the model. Providers can also remove many names at once, through account suspensions, abuse takedowns or policy changes. We model such events as correlated burns and study spreading names over providers (Section 5.4). The model omits compute suspensions, which remove endpoints and names together.

Scope. The analysis applies to *name-dependent* deployments, whose clients must present a name the censor can observe and block persistently. They include serverless and platform tenants addressed by hostname, as in CensorLess [22], and proxies behind a content delivery network (CDN), reached through a tenant hostname or a front domain [11]. WebTunnel bridges by default use the operator’s domain name with a publicly trusted certificate [47], and stock DoH clients reach a resolver by the hostname in its URL. *Nameless* designs present no such name. Examples are REALITY, which presents an unrelated site’s name [52], peer-to-peer data paths, and a pinned certificate with no SNI or, as WebTunnel supports, an arbitrary one [31, 42]. Nameless designs have no name layer, and their availability is set by the address layer (Lemma 4.3, under independent discovery) and by protocol fingerprinting. NinjaDoH is nameless, since clients reach its resolver by address and learn new addresses through the InterPlanetary Name System (IPNS) rather than DNS [39]. CensorLess is name-dependent. SpotProxy and Snowflake are nameless on their data paths but depend on names to obtain proxies. A SpotProxy client joins, and must reregister whenever its proxy is blocked, through a distribution service such as a domain-fronted broker [24], and a Snowflake client needs its rendezvous for every new proxy (Section 6.2).

Out of scope. The model excludes protocol fingerprinting, which lets a censor block a transport whatever its address or name, as Russia did with Snowflake’s Datagram TLS (DTLS) handshake [4]. Residual blocks, lasting seconds to minutes after a detected connection [3, 50, 53, 56], are neither address nor name events. We assume that clients learn the current endpoints and names through a bootstrap channel, and we exclude national shutdowns, which cut off every provider at once.

4 Model

We model a name-dependent deployment as two resources that the censor depletes and the defender replenishes by rotating endpoint addresses and minting names. A *name* is the unit that the censor blocks in one action. Depending on the censor and the platform, it is a registrable domain with all its subdomains, a tenant hostname on a shared platform, such as `<tenant>.github.io`, or an exact hostname (Section 6.1). *Minting* makes a new name usable, and a name is *burned* when the censor blocks it persistently. Rotation changes an endpoint’s address, not its name. Unless stated otherwise, time is measured in mean mint intervals $1/\lambda_{\text{intro}}$, so a window length T counts mint intervals.

Definition 4.1 (Two-layer moving-target model). A configuration consists of the defender’s choices n , μ , k_{max} and λ_{intro} and the censor’s rates λ_a and λ_{burn} , all positive. Events occur at the points of independent Poisson processes with the stated rates.

- (1) *Address layer.* The defender runs n endpoints. Each endpoint rotates at rate μ to a fresh address that the censor has not blocked. The censor discovers and blocks each unblocked endpoint at rate λ_a . A blocked endpoint stays blocked until its next rotation. Discovery and rotation are independent across endpoints.
- (2) *Name layer.* The defender keeps a pool of $K(t) \in \{0, \dots, k_{\text{max}}\}$ live (unblocked) names. It mints a new name at rate λ_{intro} while $K < k_{\text{max}}$. The censor burns one live name at rate λ_{burn} while $K \geq 1$. Endpoints use any live name and move to another one when theirs is burned.
- (3) *Censor class.* The rates λ_a and λ_{burn} are constant, so the censor does not time its blocks to the defender’s state. The name process is independent of the address processes.
- (4) *Reachability.* The deployment is reachable at time t if $K(t) \geq 1$ and at least one endpoint is unblocked.

The rates are the effective rates of Section 3. The burn rate λ_{burn} counts persistent blocks, not sightings, and is an aggregate rate that does not grow with K (Appendix C relaxes this). The product form of Theorem 5.1 assumes that any live name can reach any unblocked endpoint, as when every endpoint accepts every live name, and that the layers are independent, although one observation can reveal an endpoint’s address and name together. Pairing names with particular endpoints never raises availability and leaves the cap of Theorem 5.2 intact. Section 5.4 goes beyond the constant-rate censor to treat names exposed at mint, blocks timed to the defender’s state and burns in bursts.

Definition 4.2 (Availability). Fix a configuration and let the model of Definition 4.1 run in stationarity. The *time-average availability* A is the probability that the deployment is reachable at a given instant. For $T > 0$, the *interval availability* A_T is the probability that the deployment is reachable at every instant of a window $[t, t + T]$. By stationarity it does not depend on t . The configuration is (α, T) -available if $A_T \geq \alpha$.

The time-average availability A is also the long-run fraction of time the deployment is reachable, and $A_T \leq A$ for every T . The interval metric counts any outage in the window as a failure, however brief. It describes a session that survives a change of endpoint or name, which requires a transport that can move sessions, such as Turbo Tunnel [10].

Address layer. Each endpoint alternates between an unblocked and a blocked state.

LEMMA 4.3 (ADDRESS LAYER). *Under Definition 4.1, each endpoint is blocked with stationary probability*

$$p = \frac{\lambda_a}{\lambda_a + \mu}.$$

Because discovery and rotation are independent across endpoints, all n are blocked with probability p^n . The address factor, the stationary probability that at least one endpoint is unblocked, is $c_a = 1 - p^n$.

Appendix A proves this lemma and Proposition 4.5 below. The lemma restates FlipIt's Theorem 2 and FlipThem's AND model for exponential strategies [25, 48] and describes one instant. Two assumptions behind the lemma need care. Under Poisson rotation and an independent discovery delay D with any law, an endpoint is blocked with probability $p = \mathbb{E}[e^{-\mu D}]$, which neither the mean nor the tail of D fixes. The exponential model is not conservative, so an operator should size n from the measured blocked fraction p (Remark B.1 in Appendix B). Discovery must also be independent across endpoints.

Remark 4.4 (Synchronized discovery). The geometric decay of p^n needs discovery, not only rotation, to be independent across endpoints. Suppose instead that discovery comes in sweeps that each block every endpoint at once, as when the censor reads a directory listing them all. With sweeps at rate λ_a , each endpoint is still blocked with probability p , but an outage lasts until any endpoint rotates, so all n are blocked with probability $\lambda_a/(\lambda_a + n\mu)$ [25], which decays like $1/n$ rather than geometrically. At $\mu = 3\lambda_a$ and $n = 8$ this is 0.040, against $p^n = 1.5 \times 10^{-5}$ under independent discovery. If also $\lambda_a = \lambda_{\text{intro}}$, the address layer stays up throughout a window of $T = 5$ with probability 0.998 under independent discovery but only 0.949 when sweeps make up 1% of each endpoint's discovery rate. With a sweep share s , it stays up throughout a window of length T with probability at most $e^{-s\lambda_a T}$, whatever n and μ , because one sweep inside the window causes an outage. Independence must therefore be engineered. Sweeps arise from any channel that shows the censor all current endpoints, such as a public directory, the DNS records of a name that resolves to them or a list sent to every client, which a censor posing as a user also receives.

In summary, extra endpoints help a lot only if the censor has to find them one at a time.

Name layer. We call

$$\beta = \frac{\lambda_{\text{burn}}}{\lambda_{\text{intro}}} \quad (1)$$

the *burn ratio*, the mean number of names that the censor burns per mint interval while the pool is nonempty.

PROPOSITION 4.5 (NAME POOL). *Under Definition 4.1, $K(t)$ is a birth-death chain on $\{0, \dots, k_{\text{max}}\}$ with birth rate λ_{intro} in states $k < k_{\text{max}}$ and death rate λ_{burn} in states $k \geq 1$. Its stationary law is $\pi_k \propto \beta^{-k}$, and the pool is empty with probability*

$$\pi_0 = \begin{cases} \frac{1 - \beta^{-1}}{1 - \beta^{-(k_{\text{max}}+1)}}, & \beta \neq 1, \\ \frac{1}{k_{\text{max}} + 1}, & \beta = 1. \end{cases} \quad (2)$$

In each state $0 < k < k_{\text{max}}$ the mean drift of K is $\lambda_{\text{intro}}(1 - \beta)$. The pool fills when $\beta < 1$, and then $\pi_0 \leq \beta^{k_{\text{max}}}$. It drains when $\beta > 1$.

In summary, when names are minted faster than they are burned ($\beta < 1$), the chance that no name is live shrinks geometrically as the buffer grows.

5 Analysis

Unless stated otherwise, numbers in this section refer to the canonical configuration, with $n = 8$ endpoints, $\lambda_a = \lambda_{\text{intro}}$, $\mu = 3\lambda_a$, $k_{\text{max}} = 8$, $\alpha = 0.95$ and $T = 5$. This illustrative configuration assumes that addresses are discovered on the timescale of minting, and Section 5.3 examines faster discovery.

5.1 Closed-Form Availability and the Burn Cap

THEOREM 5.1 (CLOSED-FORM AVAILABILITY). *Under Definition 4.1, with p and the address factor $c_a = 1 - p^n$ as in Lemma 4.3,*

$$A = (1 - \pi_0(\beta, k_{\text{max}})) c_a = (1 - \pi_0(\beta, k_{\text{max}}))(1 - p^n), \quad (3)$$

where $\pi_0(\beta, k_{\text{max}})$ is the stationary probability that no live name exists (Proposition 4.5).

Appendix A proves it through the joint chain of the two layers. For a nameless deployment, $A = c_a$. The cap below is an impossibility result. Its bound $1/\beta$ needs neither independent layers nor Poisson events (Proposition A.3).

THEOREM 5.2 (BURN CAP). *Let the name pool follow Proposition 4.5. For every $\beta > 0$ and every n, μ, λ_a and k_{max} ,*

$$A_T \leq A \leq 1 - \pi_0(\beta, k_{\text{max}}) < \frac{1}{\beta},$$

and $1 - \pi_0(\beta, k_{\text{max}})$ increases to $1/\beta$ as $k_{\text{max}} \rightarrow \infty$ if $\beta > 1$. Hence no configuration with $\beta \geq 1/\alpha$ is (α, T) -available, for any $T > 0$, whatever its rotation speed and number of endpoints. If the two layers are independent, as in Definition 4.1, then also $A = c_a(1 - \pi_0) < c_a/\beta$, so the same holds for every $\beta \geq c_a/\alpha$. For $\beta > 1$ the cap is tight, since $A \rightarrow 1/\beta$ as $k_{\text{max}} \rightarrow \infty$ and $c_a \rightarrow 1$.

PROOF. A window that is reachable throughout is reachable at its start, so $A_T \leq A$. A reachable deployment needs a live name, so $A \leq \Pr[K \geq 1] = 1 - \pi_0$, whatever the address layer does. Lemma A.1(c) in Appendix A gives $1 - \pi_0 < 1/\beta$, and Lemma A.1(d) the limit. If $\beta \geq 1/\alpha$, then $A_T < 1/\beta \leq \alpha$. No step so far uses independence. With independent layers, Theorem 5.1 gives $A = c_a(1 - \pi_0) < c_a/\beta$, and the limits of c_a and $1 - \pi_0$ give tightness. $\square$

In summary, if names burn twice as fast as they are minted, clients can connect under half the time, whatever the endpoints do.

What binding means. Theorem 5.2 is the sense in which the name layer binds. Rotation speed enters (3) only through c_a , and under the interval metric also through the address window (Proposition 5.4). Once both are close to 1, the name pool alone decides whether the target is met, however fast the endpoints rotate. Binding does not mean that names are expensive. Cheap minting (Section 6.3) keeps β small but is no way around the cap, and a stock of names bought in advance amounts to minting at its release rate or to a larger buffer, which never lifts β^* beyond c_a/α (Theorem 5.3).

5.2 The Sustainability Frontier

For a target $\alpha \in (0, 1)$, the time-average frontier β_{avg}^* is the largest β with $A \geq \alpha$, and the interval frontier $\beta^* = \beta^*(\alpha, T)$ is the largest β with $A_T \geq \alpha$. Each is 0 if no $\beta > 0$ meets the target. We call $\beta^*(\alpha, T)$ the *sustainability frontier*, the largest burn ratio that a deployment can sustain while meeting its availability target (Figure 1). On common clocks a higher burn rate never leaves more live names (Lemma A.2), so A and A_T are nonincreasing in β . A configuration therefore meets its target exactly when β does not exceed the frontier.

THEOREM 5.3 (FRONTIERS). Fix $\alpha \in (0, 1)$, $n, \mu, \lambda_a > 0$ and k_{max} , under Definition 4.1.

- (i) If $c_a > \alpha$, then β_{avg}^* is the unique $\beta > 0$ with $1 - \pi_0(\beta, k_{\text{max}}) = \alpha/c_a$. If $c_a \leq \alpha$, no $\beta > 0$ gives $A \geq \alpha$.
- (ii) If $c_a > \alpha$, then β_{avg}^* is strictly increasing in k_{max} , n and μ .
- (iii) $\beta_{\text{avg}}^* < c_a/\alpha$, and if $c_a > \alpha$, then $\beta_{\text{avg}}^* \rightarrow c_a/\alpha$ as $k_{\text{max}} \rightarrow \infty$.
- (iv) $\beta^*(\alpha, T)$ is nondecreasing in k_{max} , n and μ , and

$$\beta^*(\alpha, T) \leq \beta_{\text{avg}}^* < c_a/\alpha \leq 1/\alpha.$$

Appendix A gives the proof. Without independence of the layers, $\beta^* \leq \beta_{\text{avg}}^* < 1/\alpha$ still holds by Theorem 5.2.

PROPOSITION 5.4 (EXACT INTERVAL AVAILABILITY). Under Definition 4.1,

$$A_T = W_T^{\text{addr}} W_T^{\text{name}}, \quad W_T = \pi_{\mathcal{U}} e^{Q_{\mathcal{U}\mathcal{U}} T} \mathbf{1}. \quad (4)$$

Here the address window W_T^{addr} is the probability that the stationary number c of unblocked endpoints stays up ($c \geq 1$) throughout a window of length T , and the name window W_T^{name} is the same for the stationary name pool K (up when $K \geq 1$). For a finite chain with generator Q , stationary law π and up set $\mathcal{U}$, $\pi_{\mathcal{U}}$ and $Q_{\mathcal{U}\mathcal{U}}$ are π and Q restricted to $\mathcal{U}$, and $\mathbf{1}$ is the all-ones vector.

Appendix B gives the proof. The address chain moves from c to $c + 1$ at rate $\mu(n - c)$ and to $c - 1$ at rate $\lambda_a c$. Both factors thus need exponentials of matrices with at most $\max\{n, k_{\text{max}}\}$ rows, computed to about 10^{-12} , and a root β^* takes milliseconds. Every frontier in this paper is an exact value of the model, not an estimate from noisy simulation runs. An independent event-driven simulator of the same Markov model agrees with the exact values within Monte Carlo error where standard errors are available, and to within 0.009 elsewhere (Appendix B). Both assume independent layers, so their agreement checks the code, not the model.

Exact frontiers. Both frontiers rise with the buffer (Figure 1). At $k_{\text{max}} = 8$ the time-average frontier is $\beta_{\text{avg}}^* = 0.840$ and the interval frontier is lower, $\beta^*(0.95, 5) = 0.661$. The address window is $W_5^{\text{addr}} = 0.998$ here, so the name pool sets β^* . The interval frontier exceeds 1 from $k_{\text{max}} = 63$ on and tends to 1.016. Longer windows lower the frontier gently, so the smallest buffer with $\beta^* > 1$ grows with the window length, as it does with the target.

For finite buffers A_T falls continuously as β grows (Figure 2), so $\beta^*(\alpha, T)$ is a threshold for a chosen target and window, not a phase transition. Only as $k_{\text{max}} \rightarrow \infty$ does $\beta = 1$ become a sharp point, where the limit $c_a \min\{1, 1/\beta\}$ of the time-average availability has a kink. The interval metric is much stricter, since at $k_{\text{max}} = 8$ and $\beta = 1$ the time-average availability is 0.889 but $A_5 = 0.659$.

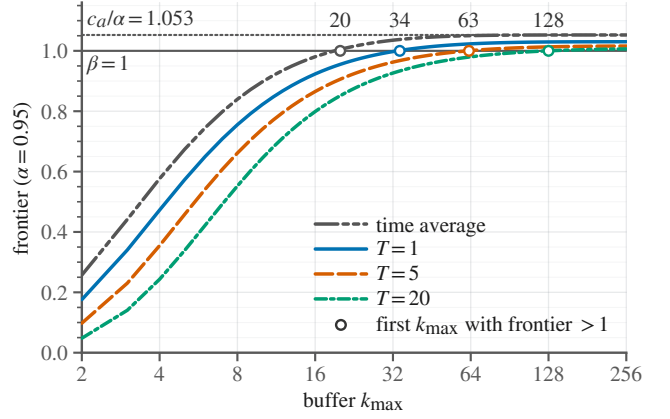


Figure 1: The time-average frontier β_{avg}^* and the interval frontier $\beta^*(0.95, T)$ for $T = 1, 5$ and 20 against the buffer k_{max} (log scale), at $\alpha = 0.95$ ($n = 8, \mu/\lambda_a = 3, \lambda_a = \lambda_{\text{intro}}$). Circles mark the smallest buffer at which each frontier exceeds 1 ($k_{\text{max}} = 20, 34, 63$ and 128). The dotted line is the bound $c_a/\alpha = 1.053$.

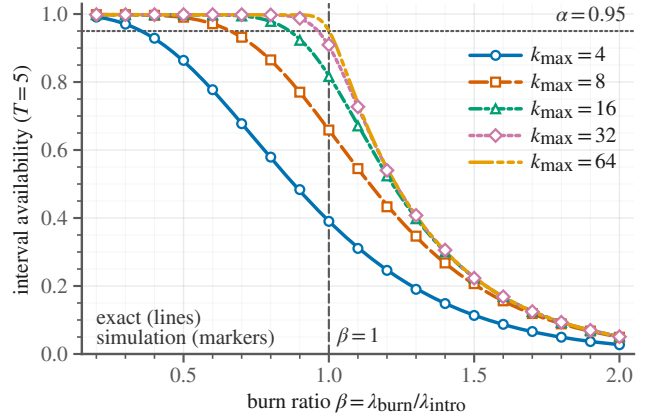


Figure 2: Interval availability A_T ($T = 5$) against the burn ratio β for buffers $k_{\text{max}} \in \{4, 8, 16, 32, 64\}$ ($n = 8, \mu/\lambda_a = 3, \lambda_a = \lambda_{\text{intro}}$). Lines are exact values and markers event-driven simulations ($k_{\text{max}} \leq 32$), with two-standard-error bars smaller than the markers. The horizontal dotted line marks the target $\alpha = 0.95$ and the vertical dashed line $\beta = 1$.

In summary, a larger buffer raises the frontier, but never above $1/\alpha$, so no buffer withstands burning much faster than minting.

5.3 When Rotation Speed Matters

We now relax $\lambda_a = \lambda_{\text{intro}}$. Under the time-average metric the address layer enters only through $c_a = 1 - p^n$ (Lemma 4.3), which depends on n and μ/λ_a alone. The address window W_T^{addr} also depends on $\lambda_a T$, the session length in mean discovery delays, because a session fails if every endpoint is blocked at any moment within it. At $n = 8$ and $\mu/\lambda_a = 1$, $c_a = 0.996$, yet W_T^{addr} is 0.872 for $\lambda_a T = 5$, 0.669 for $\lambda_a T = 15$ and 0.204 for $\lambda_a T = 60$.

Rotation speed therefore matters up to a threshold that rises with $\lambda_a T$. The interval frontier $\beta^*(0.95, 5)$ stays within 0.01 of its value for a perfect address layer (0.665) once $\mu/\lambda_a \geq 2.52, 4.00$ and 5.66 for $\lambda_a T = 5, 50$ and 300 ($\lambda_a = \lambda_{\text{intro}}, 10\lambda_{\text{intro}}$ and $60\lambda_{\text{intro}}$). For the time-average frontier the threshold is about 1.12 for any λ_a . These thresholds are grid values, at most 7% above the true ones. Once rotation reaches the threshold, the name economy sets the frontier. Faster rotation then moves it by less than 0.01, and the deployment meets its target by minting fast enough to keep β below the frontier. Below the threshold, slower rotation lowers the frontier, and once W_T^{addr} falls below α no β meets the target. At the canonical $\mu/\lambda_a = 3$ this happens once $\lambda_a > 29.5\lambda_{\text{intro}}$.

Wall-clock timescales. Published censor behavior is reported in wall-clock time. With D the time from putting an address in use to its block and τ the wall-clock mean mint interval, exponential discovery gives $\lambda_a = \tau/\mathbb{E}[D]$, and a session of wall-clock length T_{wall} has $\lambda_a T = T_{\text{wall}}/\mathbb{E}[D]$, independent of τ . For Tor bridges that the GFW confirms by active probing, blocks follow on the order of minutes, although no source reports a mean [7, 8, 49], and relays listed in the public Tor consensus were blocked after about 10 minutes [7]. Taking a mean time to block of one minute as an order of magnitude, a one-hour session has $\lambda_a T = 60$. With $n = 8$ endpoints, W_T^{addr} then reaches 0.95 only if $\mu/\lambda_a \geq 2.49$, a rotation about every 24 s per endpoint. A one-day session ($\lambda_a T = 1,440$) needs $\mu/\lambda_a \geq 4.63$, a rotation about every 13 s. Both are stress tests that provider quotas may rule out. These bare minimums leave the name layer no margin but suffice for nameless designs. They assume independent, exponential discovery. Delays that are often much shorter than their mean require more, and sweeps must be avoided by engineering independent discovery (Remarks 4.4 and B.1).

In summary, with independent discovery, rotation only has to be fast enough, and how fast is enough depends on session length relative to how quickly the censor finds an address.

5.4 Beyond the Constant-Rate Censor

We now leave Definition 4.1 one assumption at a time (Appendix C). Unless stated otherwise, the numbers are exact frontiers $\beta^*(0.95, 5)$ at the canonical configuration, where the constant-rate frontier is 0.661.

Mint-time exposure. A name can become visible at mint, before any client uses it, for example when its certificate is logged in CT (Section 6.3). Let each mint be exposed independently with probability ρ , and let the censor burn exposed names preemptively at rate θ per name, on top of usage burns at rate λ_{burn} .

PROPOSITION 5.5 (MINT-TIME EXPOSURE). *If exposed names are burned at once ($\theta \rightarrow \infty$), the usable names form the pool of Proposition 4.5 with mint rate $(1 - \rho)\lambda_{\text{intro}}$. Theorems 5.1 to 5.3 and Proposition 5.4 then hold with β replaced by the effective burn ratio $\beta_{\text{eff}} = \lambda_{\text{burn}}/((1 - \rho)\lambda_{\text{intro}})$ and time measured in effective mint intervals $1/((1 - \rho)\lambda_{\text{intro}})$, in which a window of T mint intervals has length $(1 - \rho)T$. At $\rho = 1$ no name is ever usable. In terms of $\beta = \lambda_{\text{burn}}/\lambda_{\text{intro}}$, the time-average frontier is $(1 - \rho)$ times its value at $\rho = 0$, and the interval frontier is at least $(1 - \rho)$ times its value at $\rho = 0$.*

Appendix C gives the proof. At $\theta = \infty$, $\beta^* = 0.510, 0.352, 0.185$ and 0.078 for $\rho = 0.25, 0.5, 0.75$ and 0.9 , and no β suffices at $\rho = 1$. Faster minting lowers β_{eff} only through names not exposed at mint.

Strategic block timing. Censors can delay and batch blocks instead of blocking on discovery, as the GFW apparently did with the Tor Browser default bridges of each release in 2015–16 [12]. A censor that discovers live names at the same aggregate rate λ_{burn} but withholds every block until all names in use are discovered, and then blocks them at once (strike-all), lowers β^* from 0.661 to 0.064 and β_{avg}^* from 0.840 to 0.313. Fixed batches of two or four names barely move β^* (0.671 and 0.647), so the damage comes from emptying the pool at once. A defense is a reserve of r of the live names, unexposed at mint and served only once all names in use are blocked. It has no effect on the constant-rate censor, but against strike-all it raises β^* to 0.237, 0.367 and 0.522 for $r = 1, 2$ and 4 . An average-reward Markov decision process (MDP) gives strike-all as the exact time-average best response at all 17 points solved, for a censor that knows how many names are in use and a defender with no reserve. We do not compute the interval-optimal policy, so 0.064 is an upper bound on the interval frontier against the worst timing. The frontiers of Sections 5.2 and 5.3 therefore hold against constant-rate censors. They are not worst-case guarantees.

Correlated takedowns and diversification. Names can fail together, for example when a provider suspends an account. Let takedowns at rate λ_{burn}/b each burn b names, so the nominal burn rate stays λ_{burn} . Bursts lower the frontier to $\beta^* = 0.523, 0.314$ and 0.066 for $b = 2, 4$ and 8 . Spreading names over P providers confines a takedown to one provider. With pooled minting, let takedowns at total rate λ_{burn} each empty a random nonempty provider, so that β here counts takedowns, not burned names. Splitting the $k_{\text{max}} = 8$ names evenly over $P = 1, 2, 4$ and 8 providers gives $\beta^* = 0.008, 0.092, 0.311$ and 0.661 . Diversification recovers the single pool with unit burns when each provider holds one name, and never exceeds it, because on common clocks each takedown removes at least one name from the diversified pool but exactly one from the single pool (Lemma C.1). It gives no help against a censor that strikes all providers at once.

In summary, the frontiers assume a censor that blocks each name on discovery. Exposure at mint, timed blocks and correlated takedowns can lower them. Reserves soften timed blocks, and spreading names over providers softens takedowns.

6 Grounding the Model in Public Data

We make no new measurements and draw on the literature and on public data, such as the measurements of the Open Observatory of Network Interference (OONI) [13], each cited where it is used. Appendix D documents each dataset and how we handled the dumps of Russia’s blocking registry.

6.1 Which Name Each Censor Burns

Hosting platforms give each customer (a *tenant*) a name under a shared suffix, such as `<tenant>.github.io`. A censor can block one tenant name or the whole suffix, and under a suffix-wide rule, names minted on the platform are blocked on arrival and add nothing to λ_{intro} . Table 1 classifies 19 platform suffixes for China, Iran

Table 1: Observed or ordered granularity of name blocks under 19 platform suffixes, from GFWatch and OONI data for China, IRBlock scans for Iran (block onsets unknown), and registry dumps for Russia (Appendix D). Registry entries are legal orders, not proof of enforcement.

Platform suffix	China	Iran	Russia
github.io	○	○	○
pages.dev	○	●	○→●'24 [†]
workers.dev	○→●'22	?	○
r2.dev	?	?	∅
trycloudflare.com	?	●	∅
vercel.app	○→●'22	?	○
netlify.app	○	○	○
herokuapp.com	○→●'22 [‡]	●	○
onrender.com	○	?	○
fly.dev	○	?	∅
deno.dev	○	?	○
appspot.com	●≤'20	?	○
web.app	○	?	○
firebaseapp.com	○	?	○
run.app	?	?	○
cloudfront.net	○	○	○
lambda-url.*.on.aws	?	?	∅
azurewebsites.net	○	○	○
azurefd.net	○△	?	○

○ Per-name (tenant) blocks only. For China, the cells of pages.dev, netlify.app, onrender.com, fly.dev, deno.dev, firebaseapp.com and azurewebsites.net rest on GFWatch data to September 2024 alone, as OONI tested fewer than five tenant names per suffix in 2026. ● Suffix-wide rule (in Russia, a registry entry). ○→● Per-name blocks, then a suffix-wide rule. Years give the rule's onset (for China, in GFWatch), and ≤'20 means in place when those data begin in March 2020. △ Temporary impairment of tested names, trigger undetermined. ? Too few tenant names known to be tested, and no block seen. ∅ No registry entry. Blocking outside the registry is not covered. [†]Wildcard entry, enforcement unconfirmed by OONI. [‡]In 2026 the bare suffix and Heroku's routing names are largely not blocked.

and Russia. Russia's registry entries are legal orders, and OONI measures too few platform entries to confirm their enforcement.

Some platforms are blocked one tenant at a time. In China, GFWatch recorded 259 blocked github.io tenants but no suffix rule, and in 2026 OONI found 6 of 980 tested names blocked [18, 34]. Iran blocked 27 github.io tenants by name [41], and Russia's registry listed 30 cloudfront.net tenants at the end of 2021 and 10,224 in October 2025 [55]. Whole platforms have also been blocked. China's DNS rules covered workers.dev, vercel.app and herokuapp.com in 2022, at least 405, 42 and 771 days after their first tenant blocks [18], and in 2026 OONI still found 984 of 1,100, 1,008 of 1,008 and 357 of 376 tested names under them blocked [34]. The blockable unit thus depends on the censor and the platform, and an escalation from tenant names to a suffix rule burns every live name on the platform at once, a burst in the sense of Section 5.4. A deployment must therefore count λ_{intro} and λ_{burn} in the units its censor blocks, platform by platform, and expect that granularity to change.

6.2 Names in a Peer-to-Peer System

Snowflake [4] shows that even a peer-to-peer design whose proxy addresses churn depends on names at rendezvous. Tor's broker counted a median of 148,719 unique proxy addresses per day from 28 September to 3 October 2026 [43], and half the 2023 proxy pool turned over in about 20 hours [4]. To reach a proxy, however, a

client must first poll Tor's broker, and every method the broker counts uses a censor-visible name, namely a CDN front or the broker's own domain, Google's Accelerated Mobile Pages (AMP) cache, or Amazon's Simple Queue Service (SQS) [4, 43].

When Turkmenistan blocked the default broker front for months from October 2021, estimated Snowflake users there fell from 28 to 2 [4, 46]. When Fastly ended domain fronting on 1 March 2024 [44], the Fastly fronts in use failed together. Within a week, HTTP rendezvous polls fell 85.8% and estimated users 32.5%, while proxy addresses fell only 2.1% [43, 46] (Appendix D defines the baselines). Tor configured other fronts the same day [45], yet HTTP polls regained half their earlier volume only after 36 days [43]. Tor thus runs a small name economy for rendezvous. Its Connection Assist settings introduced 19 rendezvous names and retired 11 from 2022 to 2026 [45]. In the model's terms, k_{max} is small, minting means changing a configuration or, for built-in defaults, shipping a release, and the Fastly event is a correlated takedown (Section 5.4). A client needs a live name only for new proxies, so session survival lies between A_T and A .

6.3 Minting Economics and Exposure

On 3 October 2026, one registrar listed 36 top-level domains (TLDs) under \$2 for the first year, the cheapest at \$1.54 [35]. Let's Encrypt issues free certificates and allows 2,400 orders a day per account [27]. Labels under an owned domain, and tenant names within a platform's per-account cap, count as free new names if the censor blocks exact names (Section 6.1).

Minting can, however, expose names. Publicly trusted certificates must be logged in CT for Chrome and Apple platforms to accept them [2, 15], the logs commit to adding entries within 60 s or 24 hours [14, 26], and in 2018 third parties looked up new names 73 to 197 s after logging [38]. New names in generic TLDs also appear in daily zone files [19]. Of the unit types we examined, only three are unexposed at mint, namely labels under a wildcard certificate, unpublished tenant names under a platform wildcard (as for *.netlify.app and *.web.app), and labels with a private pinned certificate, which WebTunnel has supported since 2025 [31]. The exposed share ρ is thus near 1 for registrable generic-TLD domains and CT-logged names and near 0 for names that no channel exposes at mint. The rate θ at which a censor burns exposed names is unmeasured. If the censor acts promptly, only unexposed mints raise the effective mint rate (Proposition 5.5).

7 Implications for Operators

For a name-dependent deployment (Section 3), the results give one principle. Make minting, not rotation, the first-class control. Rotation must be fast enough to keep the address layer up through whole sessions (Section 5.3). Beyond that speed the burn ratio β decides whether the target is met, so operators should track β as a health metric, with an alarm as it nears β^* , or $(1 - \rho)\beta^*$ when a share ρ of mints is exposed (step 4).

For a nameless deployment, steps 1 and 2 are the whole procedure (Section 5.3).

- (1) Fix (α, T) in wall-clock terms. With rates per unit of wall-clock time, a session of length T_{wall} lasts $T = \lambda_{\text{intro}} T_{\text{wall}}$ mint intervals

and $\lambda_a T_{\text{wall}}$ mean discovery delays. Sessions must be able to move between endpoints and names (Definition 4.2).

- (2) Size the address layer for the whole window, not an instant. Measure the blocked fraction p at the rotation rate in use. At a new rate μ , recompute $p = \mathbb{E}[e^{-\mu D}]$ from the measured law of the discovery delay D , not from its mean alone (Remark B.1). Choose n and μ so that $1 - p^n(1 + n\mu T_{\text{wall}})$ is close to 1. This lower bound on the address window W_T^{addr} holds for any delay law under Poisson rotation (Corollary B.2). Making $p^n \leq 1 - \alpha$ (Lemma 4.3) is not enough, because outages inside the window grow with $\lambda_a T_{\text{wall}}$ and the name layer needs the remaining margin (Section 5.3). Discovery must also be independent across endpoints (Remark 4.4), and n counts only the endpoints to which a client’s sessions can move.
- (3) Choose the buffer k_{max} and compute $\beta^*(\alpha, T)$ for the actual $(n, \mu/\lambda_{\text{intro}}, \lambda_a/\lambda_{\text{intro}}, k_{\text{max}}, T)$ with the exact solver (Proposition 5.4, Appendix B) in about 8 ms per frontier at $k_{\text{max}} = 8$. The solver’s address window assumes exponential discovery. Otherwise use the bound of step 2. If takedowns remove b names at once, use the frontier for that b (Section 5.4).
- (4) Mint at

$$\lambda_{\text{intro}} \geq \frac{\lambda_{\text{burn}}}{(1 - \rho) \beta^*(\alpha, \lambda_{\text{intro}} T_{\text{wall}})},$$

where ρ is the fraction of mints exposed at mint time (Section 6.3). With exposed names burned at once, the rule is conservative, and at $\rho = 1$ no mint rate suffices (Proposition 5.5). Bisection in λ_{intro} solves the rule, since more mints never leave fewer live names (Lemma A.2). Count names at the censor’s granularity (Table 1), so that all hostnames under a blocked suffix or registrable domain form one name.

- (5) Keep a reserve of r of the live names, unserved and unexposed until the active pool is empty. It recovers part of the frontier lost to a strike-all censor and leaves it unchanged against a constant-rate censor (Section 5.4). If every strike empties the active pool, use the strike-all frontier for the chosen r in steps 4 and 7, not the burst frontier of step 3.
- (6) Prefer platforms whose tenant names the censor blocks one at a time (Table 1), within their quotas and terms of service. Spreading names over independent providers moves the frontier toward that of unit burns but never beyond it, with β counting takedowns (Section 5.4). It does not slow a censor whose discovery spans providers, for example through CT.
- (7) Estimate the rates online in each censored region. Burns occur at rate λ_{burn} only while $K \geq 1$, so the maximum-likelihood estimate is

$$\hat{\lambda}_{\text{burn}} = \frac{\text{burns of names in use}}{\text{time with } K \geq 1}.$$

It is consistent if each burn is observed when it occurs, and missed or late detections bias it low. Burns divided by elapsed time instead estimate $\lambda_{\text{burn}}(1 - \pi_0)$, which hides $\beta > 1$ because $\beta(1 - \pi_0)$ never exceeds 1 (Theorem 5.2). Also record b . Names burned before first use reveal exposure at mint, so their share of mints bounds ρ from below. Censors differ in what they burn (Table 1), so meet step 4 in every region, and redo steps 3 and 4 when $\hat{\lambda}_{\text{burn}}/\lambda_{\text{intro}}$ nears $(1 - \rho)\beta^*$.

8 Discussion

What the censor can do. Most countermoves change a model parameter and have a defense (Sections 5.4 and 7). Escalation from tenant blocks to a suffix rule (Section 6.1), limited by spreading names over platforms, and range blocking in a crisis are regime changes. The model thus turns “the censor will adapt” into parameters to measure and defenses to size, although the censor’s best timing against interval availability remains open.

Generality and future work. Nothing in the model is specific to proxies, so it applies to any service on rotating infrastructure whose clients present a censor-visible name, such as a news mirror or a messaging entry point. Its rates are dimensionless (Section 5.3), so only the wall-clock calibration changes from one service to another. The rotation layer analyzed here is the foundation of a complete system, which also needs a discovery layer that resists enumeration and a traffic-shaping layer that resists fingerprinting. Both build on the rotation layer, and extending the timing model to them is future work.

Limits. We do not estimate β for any deployed system, because public sources record which names were blocked, not when a deployment minted them, and test popular or listed names. Measuring β needs mint dates and repeated block observations of the same names, from data such as access-gated zone files of ICANN’s Centralized Zone Data Service [19] or unpublished GFWatch tests [18]. We use none of them, so β and ρ remain swept parameters for operators to measure (Section 7). Our data are snapshots of particular censors, periods and vantage points (Appendix D), and censors may since have changed what they block. Protocol fingerprinting (Section 3) can only lower availability at fixed address and name rates, so against a censor that also fingerprints, the model’s values are upper bounds.

9 Conclusion

We model moving-target circumvention for deployments whose clients must present a name the censor can block. Redundant endpoints rotated fast enough relative to discovery keep the address layer up, provided the censor finds them one at a time, and past that speed faster rotation barely moves the frontier. The name layer then binds, because the burn ratio β keeps time-average availability below $1/\beta$ however fast endpoints rotate. The sustainability frontier $\beta^*(\alpha, T)$ tells an operator how fast to mint, and bursty takedowns, timed blocks and burns of names exposed at mint can lower it. Public data show that censors burn different units on different platforms, and that even a peer-to-peer system depends on names at rendezvous. How fast censors burn privately minted names, and hence β in the field, remains to be measured. Still, the advice is simple. Once rotation is fast enough, mint names to keep β below β^* , and track β as a health metric.

AI Use

The model and the thesis are the authors’ work. AI-based tools (Claude Code) were used, under the authors’ direction, to implement the simulation, analysis and plotting code. We have manually verified and are responsible for the accuracy, originality, and integrity of the output of all AI-based tools.

References

- [1] Spiros Antonatos, Periklis Akritidis, Evangelos P. Markatos, and Kostas G. Anagnostakis. 2007. Defending Against Hitlist Worms Using Network Address Space Randomization. *Computer Networks* 51, 12 (2007), 3471–3490.
- [2] Apple. 2025. Apple's Certificate Transparency policy. Apple Support. Retrieved October 3, 2026 from <https://support.apple.com/en-us/103214>
- [3] Kevin Bock, Yair Fax, Kyle Reese, Jasraj Singh, and Dave Levin. 2020. Detecting and Evading Censorship-in-Depth: A Case Study of Iran's Protocol Filter. In *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet (FOCI 2020)*. USENIX Association.
- [4] Cecylia Bocovich, Arlo Breault, David Fifield, Serene, and Xiaokang Wang. 2024. Snowflake, a censorship circumvention system using temporary WebRTC proxies. In *Proceedings of the USENIX Security Symposium (USENIX Security 2024)*. USENIX Association, 2635–2652.
- [5] Jin-Hee Cho, Dilli P. Sharma, Hooman Alavizadeh, Seunghyun Yoon, Noam Ben-Asher, Terrence J. Moore, Dong Seong Kim, Hyuk Lim, and Frederica F. Nelson. 2020. Toward Proactive, Adaptive Defense: A Survey on Moving Target Defense. *IEEE Communications Surveys & Tutorials* 22, 1 (2020), 709–745.
- [6] Matthew Dunlop, Stephen Groat, William Urbanski, Randy Marchany, and Joseph Tront. 2011. MT6D: A Moving Target IPv6 Defense. In *Proceedings of the IEEE Military Communications Conference (MILCOM 2011)*. IEEE, 1321–1326.
- [7] Arun Dunna, Ciarán O'Brien, and Phillipa Gill. 2018. Analyzing China's Blocking of Unpublished Tor Bridges. In *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet (FOCI 2018)*. USENIX Association.
- [8] Roya Ensafi, David Fifield, Philipp Winter, Nick Feamster, Nicholas Weaver, and Vern Paxson. 2015. Examining How the Great Firewall Discovers Hidden Circumvention Servers. In *Proceedings of the ACM Internet Measurement Conference (IMC 2015)*. ACM, 445–458.
- [9] Hassan Fares, Omkar Fulsundar, and Nicholas Hopper. 2026. The Game Has Changed: Revisiting proxy distribution and game theory. *Free and Open Communications on the Internet* 2026, 1 (2026), 14–22.
- [10] David Fifield. 2020. Turbo Tunnel, a good way to design censorship circumvention protocols. In *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet (FOCI 2020)*. USENIX Association.
- [11] David Fifield, Chang Lan, Rod Hynes, Percy Wegmann, and Vern Paxson. 2015. Blocking-resistant communication through domain fronting. *Proceedings on Privacy Enhancing Technologies* 2015, 2 (2015), 46–64.
- [12] David Fifield and Lynn Tsai. 2016. Censors' Delay in Blocking Circumvention Proxies. In *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet (FOCI 2016)*. USENIX Association.
- [13] Arturo Filastò and Jacob Appelbaum. 2012. OONI: Open Observatory of Network Interference. In *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet (FOCI 2012)*. USENIX Association.
- [14] Google Chrome. 2026. Chrome Certificate Transparency Log List, Version 93.3. Google. Retrieved October 3, 2026 from https://www.gstatic.com/ct/log_list/v3/log_list.json
- [15] Google Chrome. 2026. Chrome Certificate Transparency Policy. Certificate Transparency in Chrome. Retrieved October 3, 2026 from https://googlechrome.github.io/CertificateTransparency/ct_policy.html
- [16] Nico Heitmann, Niklas Niere, Felix Graf Lange, and Juraj Somorovsky. 2026. On Russia's Early Introduction of QUIC SNI Censorship. *Free and Open Communications on the Internet* 2026, 2 (2026), 1–6.
- [17] Nguyen Phong Hoang, Jakub Dalek, Masashi Crete-Nishihata, Nicolas Christin, Vinod Yegneswaran, Michalis Polychronakis, and Nick Feamster. 2024. GFWeb: Measuring the Great Firewall's Web Censorship at Scale. In *Proceedings of the USENIX Security Symposium (USENIX Security 2024)*. USENIX Association, 2617–2633.
- [18] Nguyen Phong Hoang, Arian Akhavan Niaki, Jakub Dalek, Jeffrey Knockel, Pellaeon Lin, Bill Marczak, Masashi Crete-Nishihata, Phillipa Gill, and Michalis Polychronakis. 2021. How Great is the Great Firewall? Measuring China's DNS Censorship. In *Proceedings of the USENIX Security Symposium (USENIX Security 2021)*. USENIX Association, 3381–3398.
- [19] ICANN. 2026. About Zone File Access. Internet Corporation for Assigned Names and Numbers. Retrieved October 3, 2026 from <https://www.icann.org/resources/pages/zfa-2013-06-28-en>
- [20] IODA. 2026. IODA: Internet Outage Detection and Analysis. Georgia Institute of Technology. Retrieved October 3, 2026 from <https://ioda.inetintel.cc.gatech.edu/>
- [21] Jafar Haadi Jafarian, Ehab Al-Shaer, and Qi Duan. 2012. OpenFlow Random Host Mutation: Transparent Moving Target Defense Using Software Defined Networking. In *Proceedings of the Workshop on Hot Topics in Software Defined Networks (HotSDN 2012)*. ACM, 127–132.
- [22] Dayeong Kang, Jade Sheffey, Mingshi Wu, Pubali Datta, and Amir Houmansadr. 2026. CensorLess: Cost-Efficient Censorship Circumvention Through Serverless Cloud Functions. *Proceedings on Privacy Enhancing Technologies* 2026, 3 (2026), 556–574.
- [23] Leonard Kleinrock. 1975. *Queueing Systems, Volume I: Theory*. Wiley-Interscience, New York, NY, USA.
- [24] Patrick Tser Jern Kon, Sina Kamali, Jinyu Pei, Diogo Barradas, Ang Chen, Micah Sherr, and Moti Yung. 2024. SpotProxy: Rediscovering the Cloud for Censorship Circumvention. In *Proceedings of the USENIX Security Symposium (USENIX Security 2024)*. USENIX Association, 2653–2670.
- [25] Aron Laszka, Gabor Horvath, Mark Felegyhazi, and Levente Buttyán. 2014. FlipThem: Modeling Targeted Attacks with FlipIt for Multiple Resources. In *Proceedings of the International Conference on Decision and Game Theory for Security (GameSec 2014) (Lecture Notes in Computer Science, Vol. 8840)*. Springer, 175–194.
- [26] Ben Laurie, Adam Langley, and Emilia Kasper. 2013. Certificate Transparency. RFC 6962.
- [27] Let's Encrypt. 2026. Rate Limits. Let's Encrypt documentation. Retrieved October 3, 2026 from <https://letsencrypt.org/docs/rate-limits/>
- [28] Henger Li and Zizhan Zheng. 2019. Optimal Timing of Moving Target Defense: A Stackelberg Game Model. In *Proceedings of the IEEE Military Communications Conference (MILCOM 2019)*. IEEE, 1–6.
- [29] Zhen Ling, Junzhou Luo, Wei Yu, Ming Yang, and Xinwen Fu. 2015. Tor Bridge Discovery: Extensive Analysis and Large-scale Empirical Evaluation. *IEEE Transactions on Parallel and Distributed Systems* 26, 7 (2015), 1887–1899.
- [30] Mohammad Mahdian. 2010. Fighting Censorship with Algorithms. In *Proceedings of the International Conference on Fun with Algorithms (FUN 2010) (Lecture Notes in Computer Science, Vol. 6099)*. Springer, 296–306.
- [31] meskio and shelikhoo. 2025. Staying ahead of censors in 2025: What we've learned from fighting censorship in Iran and Russia. The Tor Project Blog. Retrieved October 3, 2026 from <https://blog.torproject.org/staying-ahead-of-censors-2025/>
- [32] Mozilla. 2026. Public Suffix List. Project web site. Retrieved October 3, 2026 from <https://publicsuffix.org/>
- [33] Milad Nasr, Sadegh Farhang, Amir Houmansadr, and Jens Grossklags. 2019. Enemy At the Gateways: Censorship-Resilient Proxy Distribution Using Game Theory. In *Proceedings of the Network and Distributed System Security Symposium (NDSS 2019)*. Internet Society.
- [34] Open Observatory of Network Interference (OONI). 2026. OONI Data. Project web site. Retrieved October 3, 2026 from <https://ooni.org/data/>
- [35] Porkbun LLC. 2026. Domain Name Pricing. Web page. Retrieved October 3, 2026 from <https://porkbun.com/products/domains>
- [36] Reethika Ramesh, Ram Sundara Raman, Matthew Bernhard, Victor Ongkowijaya, Leonid Evdokimov, Anne Edmundson, Steven Sprecher, Muhammad Ikram, and Roya Ensafi. 2020. Decentralized Control: A Case Study of Russia. In *Proceedings of the Network and Distributed System Security Symposium (NDSS 2020)*. Internet Society.
- [37] David Robinson, Harlan Yu, and Anne An. 2013. *Collateral Freedom: A Snapshot of Chinese Internet Users Circumventing Censorship*. Report. Open Internet Tools Project (OpenITP).
- [38] Quirin Scheitle, Oliver Gasser, Theodor Nolte, Johanna Amann, Lexi Brent, Georg Carle, Ralph Holz, Thomas C. Schmidt, and Matthias Wählisch. 2018. The Rise of Certificate Transparency and Its Implications on the Internet Ecosystem. In *Proceedings of the ACM Internet Measurement Conference (IMC 2018)*. ACM, 343–349.
- [39] Scott Seidenberger, Marc Beret, Raveen Wijewickrama, Murtuza Jadhwal, and Anindya Maiti. 2026. NinjaDoH: A Censorship-Resistant Moving Target DoH Server Using Hyperscalers and IPNS. In *Proceedings of the Workshop on Measurements, Attacks, and Defenses for the Web (MADWeb 2026)*. Internet Society.
- [40] Sailik Sengupta, Ankur Chowdhary, Abdulhakim Sabur, Adel Alshamrani, Di-jiang Huang, and Subbarao Kambhampati. 2020. A Survey of Moving Target Defenses for Network Security. *IEEE Communications Surveys & Tutorials* 22, 3 (2020), 1909–1941.
- [41] Jonas Tai, Karthik Nishanth Sengottuvelavan, Peter Whiting, and Nguyen Phong Hoang. 2025. IRBlock: A Large-Scale Measurement Study of the Great Firewall of Iran. In *Proceedings of the USENIX Security Symposium (USENIX Security 2025)*. USENIX Association, 705–722.
- [42] The Tor Project. 2025. [Russia] Blocking of webtunnel. Tor Project GitLab, censorship-analysis issue #40064. Retrieved October 3, 2026 from <https://gitlab.torproject.org/tpo/anti-censorship/censorship-analysis/-/issues/40064>
- [43] The Tor Project. 2026. CollecTor. Tor Metrics. Retrieved October 3, 2026 from <https://collector.torproject.org/>
- [44] The Tor Project. 2026. Metrics Timeline. Tor Project GitLab. Retrieved October 3, 2026 from <https://gitlab.torproject.org/tpo/network-health/metrics/timeline>
- [45] The Tor Project. 2026. rdsys-admin. Tor Project GitLab repository, circumvention settings file conf/circumvention.json. Retrieved October 3, 2026 from <https://gitlab.torproject.org/tpo/anti-censorship/rdsys-admin>
- [46] The Tor Project. 2026. Tor Metrics. Project web site. Retrieved October 3, 2026 from <https://metrics.torproject.org/>
- [47] The Tor Project. 2026. WebTunnel Bridge. Tor Project Community, Relay Operations. Retrieved October 3, 2026 from <https://community.torproject.org/relay/setup/webtunnel/>
- [48] Marten van Dijk, Ari Juels, Alina Oprea, and Ronald L. Rivest. 2013. FlipIt: The Game of "Stealthy Takeover". *Journal of Cryptology* 26, 4 (2013), 655–713.

- [49] Philipp Winter and Stefan Lindskog. 2012. How the Great Firewall of China is Blocking Tor. In *Proceedings of the USENIX Workshop on Free and Open Communications on the Internet (FOCI 2012)*. USENIX Association.
- [50] Mingshi Wu, Jackson Sippe, Danesh Sivakumar, Jack Burg, Peter Anderson, Xiaokang Wang, Kevin Bock, Amir Houmansadr, Dave Levin, and Eric Wustrow. 2023. How the Great Firewall of China Detects and Blocks Fully Encrypted Traffic. In *Proceedings of the USENIX Security Symposium (USENIX Security 2023)*. USENIX Association, 2653–2670.
- [51] Mingshi Wu, Ali Zohaib, Zakir Durumeric, Amir Houmansadr, and Eric Wustrow. 2025. A Wall Behind A Wall: Emerging Regional Censorship in China. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P 2025)*. IEEE, 1363–1380.
- [52] XTLS. 2023. REALITY. GitHub repository. Retrieved October 3, 2026 from <https://github.com/XTLS/REALITY>
- [53] Diwen Xue, Benjamin Mixon-Baca, ValdickSS, Anna Ablove, Beau Kujath, Jeddiah R. Crandall, and Roya Ensafi. 2022. TSPU: Russia's Decentralized Censorship System. In *Proceedings of the ACM Internet Measurement Conference (IMC 2022)*. ACM, 179–194.
- [54] Mahdi Zamani, Jared Saia, and Jeddiah Crandall. 2017. TorBricks: Blocking-Resistant Tor Bridge Distribution. In *Proceedings of the International Symposium on Stabilization, Safety, and Security of Distributed Systems (SSS 2017) (Lecture Notes in Computer Science, Vol. 10616)*. Springer, 426–440.
- [55] zapret-info. 2025. z-i: Register of Internet Addresses filtered in Russian Federation. GitHub repository. Retrieved October 3, 2026 from <https://github.com/zapret-info/z-i>
- [56] Ali Zohaib, Qiang Zao, Jackson Sippe, Abdulrahman Alaraj, Amir Houmansadr, Zakir Durumeric, and Eric Wustrow. 2025. Exposing and Circumventing SNI-based QUIC Censorship of the Great Firewall of China. In *Proceedings of the USENIX Security Symposium (USENIX Security 2025)*. USENIX Association, 783–802.

A Proofs

This appendix gives two auxiliary lemmas and proves Lemma 4.3, Theorems 5.1 and 5.3, and Propositions 4.5 and A.3. Appendix B proves Proposition 5.4, and Appendix C derives the extensions. All rates are positive, $k_{\max} \geq 1$, and time is measured in mean mint intervals $1/\lambda_{\text{intro}}$. Bold symbols denote vectors. Every chain below is finite and irreducible, so its unique stationary law gives the long-run fraction of time in each set of states.

PROOF OF LEMMA 4.3. An unblocked endpoint becomes blocked at rate λ_a , and a blocked one becomes unblocked at rate μ , since its next rotation moves it to an unblocked address. This two-state Markov chain is blocked with stationary probability $\lambda_a/(\lambda_a + \mu)$. The n chains are driven by independent Poisson processes, so they are independent and all n are blocked with probability p^n . $\square$

PROOF OF PROPOSITION 4.5. Mints and burns occur at the constant rates of Definition 4.1, so K is a birth–death chain [23, Section 3.6]. Detailed balance, $\pi_k \lambda_{\text{intro}} = \pi_{k+1} \lambda_{\text{burn}}$, gives $\pi_{k+1} = \pi_k / \beta$, and normalizing the geometric series gives (2). The drift is $\lambda_{\text{intro}} - \lambda_{\text{burn}} = \lambda_{\text{intro}}(1 - \beta)$. For $\beta < 1$, (2) equals $(1 - \beta)\beta^{k_{\max}} / (1 - \beta^{k_{\max}+1})$, which is at most $\beta^{k_{\max}}$. $\square$

LEMMA A.1 (EMPTY-POOL PROBABILITY). Let $G_k(\beta) = \sum_{i=0}^k \beta^{-i}$, so that $\pi_0(\beta, k) = 1/G_k(\beta)$ for a pool with buffer k (Proposition 4.5). The following hold for $\beta > 0$ and $k \geq 1$.

- (a) π_0 is continuous and strictly increasing in β , and it tends to 0 as $\beta \rightarrow 0$ and to 1 as $\beta \rightarrow \infty$.
- (b) π_0 is strictly decreasing in k .
- (c) $1 - \pi_0(\beta, k) = \beta^{-1} G_{k-1}(\beta) / G_k(\beta) < \min\{1, 1/\beta\}$.
- (d) As $k \rightarrow \infty$, $1 - \pi_0(\beta, k)$ increases to $\min\{1, 1/\beta\}$.

PROOF. For $i \geq 1$ the term β^{-i} is continuous and strictly decreasing in β , from ∞ to 0, which gives (a). Part (b) follows from $G_{k+1} = G_k + \beta^{-(k+1)}$. For (c), $1 - 1/G_k = (G_k - 1)/G_k$ and $G_k - 1 =$

$\beta^{-1} G_{k-1} < \beta^{-1} G_k$. For (d), G_k increases to $\beta/(\beta - 1)$ if $\beta > 1$ and to ∞ if $\beta \leq 1$. $\square$

LEMMA A.2 (MONOTONE COUPLING). Let K and K' be birth–death chains on $\{0, \dots, k\}$ and $\{0, \dots, k'\}$, with $k' \leq k$, birth rates η_j and η'_j (from state j), and death rates ξ_j and ξ'_j (from state $j \geq 1$). Suppose that $\eta'_j \leq \eta_j$ for $j < k'$ and $\xi'_j \geq \xi_j$ for $1 \leq j \leq k'$, and start each chain in its stationary law. Then the chains can be coupled so that $K'(t) \leq K(t)$ for all $t \geq 0$, and for every $T \geq 0$,

$$\Pr[K'(t) \geq 1 \text{ for all } t \in [0, T]] \leq \Pr[K(t) \geq 1 \text{ for all } t \in [0, T]].$$

At $T = 0$ this compares the stationary probabilities of a nonempty pool.

PROOF. For the stationary laws π of K and π' of K' , detailed balance gives $\pi_{j+1}/\pi_j = \eta_j/\xi_{j+1}$ and $\pi'_{j+1}/\pi'_j = \eta'_j/\xi'_{j+1}$. Hence π_j/π'_j is nondecreasing on $\{0, \dots, k'\}$, and $\pi'_j = 0$ for $j > k'$. This likelihood-ratio order implies the usual stochastic order, so the initial states can be drawn with $K'(0) \leq K(0)$. While $K' \neq K$, let the chains move on independent clocks. While $K' = K = j$, let one clock of rate η'_j move both up and another of rate $\eta_j - \eta'_j$ move only K up (with $\eta'_{k'} = 0$). Likewise, let one clock of rate ξ_j move both down and another of rate $\xi'_j - \xi_j$ move only K' down. Each chain then jumps at its own rates, so each is a stationary copy of its law. A single jump of size one cannot reverse a strict order, and at equality only K can rise alone and only K' can fall alone. Hence $K'(t) \leq K(t)$ for all t , and the event on the left is contained in the event on the right. $\square$

PROOF OF THEOREM 5.1. The address layer (state c) and the name layer (state K) are driven by disjoint sets of independent Poisson clocks, and no transition of one depends on the state of the other. The joint chain (c, K) therefore has the generator $Q = Q_c \otimes I + I \otimes Q_K$, the Kronecker sum of the generators of the layers. The product of their stationary laws π_c and π_K satisfies

$$(\pi_c \otimes \pi_K)Q = (\pi_c Q_c) \otimes \pi_K + \pi_c \otimes (\pi_K Q_K) = 0,$$

and the joint chain is irreducible, so the product is its stationary law. The deployment is reachable iff $c \geq 1$ and $K \geq 1$. Hence $A = \Pr[c \geq 1] \Pr[K \geq 1] = (1 - p^n)(1 - \pi_0)$ by Lemma 4.3 and Proposition 4.5. $\square$

PROOF OF THEOREM 5.3. Write $h(\beta) = 1 - \pi_0(\beta, k_{\max})$, so that $A = c_a h(\beta)$. By Lemma A.1(a), h decreases continuously and strictly from 1 to 0 as β runs from 0 to ∞ .

(i) Existence. If $c_a > \alpha$, the equation $h(\beta) = \alpha/c_a$ has a unique root β_{avg}^* , and $A \geq \alpha$ iff $\beta \leq \beta_{\text{avg}}^*$. If $c_a \leq \alpha$, then $A < c_a \leq \alpha$ for every $\beta > 0$, so no β suffices.

(ii) Monotonicity. By Lemma A.1(b), a larger $k_{\max}$ raises h at every β . A larger μ lowers p , and a larger n lowers p^n , so either raises c_a and lowers α/c_a . Each change moves the root strictly to the right.

(iii) Bound and limit. If $c_a \leq \alpha$, then $\beta_{\text{avg}}^* = 0$. Otherwise, at the root, $\alpha/c_a = h(\beta_{\text{avg}}^*) < 1/\beta_{\text{avg}}^*$ (Lemma A.1(c)), so $\beta_{\text{avg}}^* < c_a/\alpha \leq 1/\alpha$. As $k_{\max}$ grows, the roots increase and stay below c_a/α . Suppose their limit L were below c_a/α , and take β' in $(L, c_a/\alpha)$. Then $h(\beta') \rightarrow \min\{1, 1/\beta'\} > \alpha/c_a$ (Lemma A.1(d)), so for large $k_{\max}$ the root would exceed $\beta' > L$, a contradiction. Hence $\beta_{\text{avg}}^* \rightarrow c_a/\alpha$.

(iv) Sustainability frontier. By Proposition 5.4, $A_T = W_T^{\text{addr}} W_T^{\text{name}}$, the product of the address window and the name window. A larger λ_{burn} raises every death rate of the pool, so Lemma A.2 shows that W_T^{name} , and with it A_T , is nonincreasing in β . Each is also continuous in β . The set of β with $A_T \geq \alpha$ is therefore an interval $(0, \beta^*(\alpha, T)]$, possibly empty. Since $A_T \leq A$, $\beta^*(\alpha, T) \leq \beta_{\text{avg}}^*$. Raising k_{max} by one adds a state without changing the other rates, so Lemma A.2, with $k' = k_{\text{max}}$ and $k = k_{\text{max}} + 1$, shows that $\beta^*(\alpha, T)$ is nondecreasing in k_{max} . Likewise, a larger n or μ raises the birth rates $\mu(n - c)$ of the address chain, so W_T^{addr} , and with it $\beta^*(\alpha, T)$, is nondecreasing in n and μ . In all,

$$\beta^*(\alpha, T) \leq \beta_{\text{avg}}^* < c_a/\alpha \leq 1/\alpha. \quad \square$$

PROPOSITION A.3 (CONSERVATION OF NAMES). *Let mints arrive by any point process whose mean count grows at rate λ_{intro} , each admitted while $K < k_{\text{max}}$, and let burns, each removing at least one live name, arrive with stochastic intensity at least λ_{burn} whenever $K \geq 1$. Then the expected fraction of $[0, t]$ with $K \geq 1$ is at most $1/\beta + o(1)$ as $t \rightarrow \infty$, and in stationarity $A \leq \Pr[K \geq 1] \leq 1/\beta$.*

PROOF. Each burn removes a name that was admitted earlier or present at time 0, so the number $B(t)$ of burns in $[0, t]$ is at most the number $M(t)$ of admitted mints plus k_{max} . By the intensity assumption, $\lambda_{\text{burn}} \mathbb{E}[\int_0^t \mathbb{I}\{K(s) \geq 1\} ds] \leq \mathbb{E}[B(t)] \leq \mathbb{E}[M(t)] + k_{\text{max}} \leq \lambda_{\text{intro}} t + o(t)$. Dividing by $\lambda_{\text{burn}} t$ gives the first claim. In stationarity the expected fraction is $\Pr[K \geq 1]$, and a reachable deployment needs a live name. $\square$

B Exact Interval Availability

PROOF OF PROPOSITION 5.4. Let X be a finite irreducible chain with generator Q , stationary law π and up set $\mathcal{U}$, and let σ be the first time X is outside $\mathcal{U}$. The chain killed on leaving $\mathcal{U}$ has the subgenerator $Q_{\mathcal{U}\mathcal{U}}$, the restriction of Q to $\mathcal{U}$, and the transition matrix $e^{Q_{\mathcal{U}\mathcal{U}}t}$. Hence $\Pr_i[\sigma > T] = (e^{Q_{\mathcal{U}\mathcal{U}}T} \mathbf{1})_i$ for $i \in \mathcal{U}$, and averaging over the stationary initial state gives

$$W_T = \Pr_{\pi}[\sigma > T] = \pi_{\mathcal{U}} e^{Q_{\mathcal{U}\mathcal{U}}T} \mathbf{1},$$

the window formula of (4). In the model the joint chain is the Kronecker sum of the two layers (proof of Theorem 5.1), and $\mathcal{U} = \{c \geq 1\} \times \{K \geq 1\}$. The restricted generator is the Kronecker sum of the restrictions Q_1 and Q_2 of Q_c and Q_K to their up sets, with $e^{(Q_1 \otimes I + I \otimes Q_2)T} = e^{Q_1 T} \otimes e^{Q_2 T}$, and $\pi_{\mathcal{U}}$ is a product. So $A_T = W_T^{\text{addr}} W_T^{\text{name}}$, the window formula applied to each layer. $\square$

Remark B.1 (General discovery delays). Keep rotation Poisson at rate μ , but let each new address be discovered after an independent delay D with any law. In an exponential rotation interval V of mean $1/\mu$, the address is blocked for $(V - D)^+$, so an endpoint is blocked with probability $p = \mu \mathbb{E}[(V - D)^+] = \mathbb{E}[e^{-\mu D}] = \Pr[D < V]$, and $c_a = 1 - p^n$ still holds for independent endpoints. At a given mean, a deterministic delay minimizes p , because $\mathbb{E}[e^{-\mu D}] \geq e^{-\mu \mathbb{E}[D]}$. A mixture of exponentials gives at least the value $1/(1 + \mu \mathbb{E}[D])$ of a single exponential. In general p is fixed by neither the mean nor the tail of D . With $\mathbb{E}[D] = 1/\lambda_a$ and $\mu = 3\lambda_a$, the exponential gives $p = 0.25$ and a deterministic delay 0.05, but a light-tailed hyperexponential law that is usually fast (mean $0.1/\lambda_a$, weight 0.9) and occasionally slow (mean $9.1/\lambda_a$, weight 0.1) gives 0.70, which

leaves $n = 8$ endpoints with $c_a = 0.945$. The exponential model is therefore not conservative, and an operator should size n from the measured blocked fraction p , not from a mean discovery delay.

COROLLARY B.2 (A CLOSED-FORM LOWER BOUND). *Let a stationary process be up with probability γ , and let ω be the stationary rate of its transitions from up states to down states. It stays up throughout a window of length T with probability at least $\gamma - \omega T$. For the two layers of the model,*

$$W_T^{\text{addr}} \geq 1 - p^n(1 + n\mu T), \quad W_T^{\text{name}} \geq 1 - (1 + \lambda_{\text{intro}} T) \pi_0.$$

The address bound holds under Poisson rotation for every discovery-delay law of Remark B.1.

PROOF. A window fails if the process is down at its start (probability $1 - \gamma$) or enters the down set during the window, and the expected number ωT of such entries bounds the probability of at least one. In the setting of Remark B.1, an endpoint becomes blocked once in every rotation interval with $D < V$, that is, at rate μp . Since the endpoints are independent, the all-blocked state is entered at rate $n \cdot \mu p \cdot p^{n-1} = n\mu p^n$. For the pool, entries into $K = 0$ come from $K = 1$ at rate λ_{burn} , and $\lambda_{\text{burn}} \pi_1 = \lambda_{\text{intro}} \pi_0$. $\square$

The address bound needs only the measured p and the mean number μT of rotations per endpoint in a window, so it gives a closed-form check when sizing n (Section 7). At the canonical configuration it is 0.99815, against the exact $W_5^{\text{addr}} = 0.99825$. Both bounds lose accuracy as μT or $\lambda_{\text{intro}} T$ grows.

Numerical method. Both layers are birth-death chains, which are reversible. With $\Pi = \text{diag}(\pi_{\mathcal{U}})$, the matrix $S = \Pi^{1/2} Q_{\mathcal{U}\mathcal{U}} \Pi^{-1/2}$ is then symmetric and tridiagonal, and

$$W_T = \mathbf{y}^\top e^{ST} \mathbf{y} = \sum_m (\mathbf{z}_m^\top \mathbf{y})^2 e^{v_m T},$$

where $\mathbf{y} = \Pi^{1/2} \mathbf{1}$ and $(v_m, \mathbf{z}_m)$ are the orthonormal eigenpairs of S . The weights are nonnegative, so the sum involves no cancellation. Above 200 states, SciPy's `expm_multiply`, a scaled truncated Taylor method, replaces the eigendecomposition. The non-birth-death chains of the extensions and the sweep chain of Remark 4.4 use an LU solve for stationary laws and a dense matrix exponential up to 400 states, with `expm_multiply` above. Frontiers are Brent roots (tolerance 10^{-10}), and each root is the frontier, by the monotonicity of availability in β for the birth-death pools (Lemma A.2) and by a single-crossing grid check for the other chains.

Accuracy. Window probabilities are accurate to about 10^{-12} . On 40 random birth-death chains the symmetric form agrees with a dense matrix exponential to 2×10^{-14} , and on constant-rate name pools with buffers of 300 to 1,200 it agrees with `expm_multiply` to 4.4×10^{-15} . An independent implementation, which uses different algorithms for stationary laws (elimination), window probabilities (uniformization) and roots (bisection), reproduces the 120 interval frontiers for $\alpha \in \{0.9, 0.95, 0.99\}$, $T \in \{1, 2, 5, 10, 20\}$ and $k_{\text{max}} \in \{2, 4, 8, \dots, 256\}$ to 2.5×10^{-11} .

Implementation check. The simulator (Appendix E) and the exact solver implement the model independently. Across nine simulation experiments and a grid of 65 frontiers computed from the closed form (about 2,800 values in all), they agree within Monte Carlo

error where standard errors exist and to within 0.009 elsewhere. In a 148-point sweep of interval availability at $T = 5$ against β ($k_{\max} \in \{4, 8, 16, 32\}$, 24 seeds), the largest difference is 0.0052, and 99.3% of the differences lie within three standard errors. Both assume independent layers, so their agreement checks the implementations, not the product form or the model's realism.

C Extension Models

Each extension changes one ingredient of the model, keeps the canonical address layer and is a finite chain solved exactly (Appendix B). Independent Monte Carlo simulations also checked the chains for exposure, strategic timing, providers and the sweeps of Remark 4.4 (16 seeds per configuration, all differences within 2.4 standard errors).

Mint-time exposure. The chain counts exposed and unexposed live names, K_e and K_u . A mint, at rate λ_{intro} while $K_e + K_u < k_{\max}$, is exposed with probability ρ . The censor burns each exposed name preemptively at rate θ , and usage burns arrive at aggregate rate λ_{burn} while $K_e + K_u \geq 1$ and hit a uniformly chosen live name. The chain is up iff $K_e + K_u \geq 1$. At nine tested points ($k_{\max} = 8$, $T = 5$, $\rho \in \{0.25, 0.5, 0.9\}$, $\beta \in \{0.1, 0.3, 0.6\}$), the window probability differs from its limit as $\theta \rightarrow \infty$ by at most 1.5×10^{-4} at $\theta = 10^3$ and 1.5×10^{-5} at $\theta = 10^4$.

PROOF OF PROPOSITION 5.5. In the limit, unexposed mints form a thinned Poisson process of rate $(1 - \rho)\lambda_{\text{intro}}$, and usage burns hit only unexposed names. The count K_u is then the pool of Proposition 4.5 with mint rate $(1 - \rho)\lambda_{\text{intro}}$, whose stationary law depends only on β_{eff} . This gives the time-average statement. In units of $1/((1 - \rho)\lambda_{\text{intro}})$, a window of T mint intervals has length $(1 - \rho)T$, so the name window is that of an unexposed pool at burn ratio β_{eff} over $(1 - \rho)T$, which is at least its value over T . Every $\beta \leq (1 - \rho)\beta^*$, with β^* the frontier at $\rho = 0$, has $\beta_{\text{eff}} \leq \beta^*$ and so meets the target by Lemma A.2, since the address layer is unchanged. At $\rho = 1$ no mint enters the usable pool. $\square$

At $k_{\max} = 8$ the bound $(1 - \rho)\beta^*$ gives 0.496, 0.331, 0.165 and 0.066 for $\rho = 0.25, 0.5, 0.75$ and 0.9 , 3% to 15% below the exact frontiers of Section 5.4, so the mint-rate rule of Section 7 is conservative at $\theta = \infty$.

Strategic block timing. The state (K, R, J) counts K active names in use, $J \leq K$ of them discovered but not yet blocked, and $R \leq r$ reserve names, never served and so undiscovered. A mint, at rate λ_{intro} while $K + R < k_{\max}$, joins the reserve if $K \geq 1$ and $R < r$, and the active pool otherwise. When the active pool empties, the whole reserve becomes active at once. The censor discovers undiscovered active names at aggregate rate λ_{burn} while $J < K$, and the chain is up iff $K \geq 1$. The constant-rate policy blocks each name at discovery, which makes $K + R$ the constant-rate pool of Proposition 4.5, on which a reserve has no effect. The batch policy of size j blocks all discovered names when $J = j$, and strike-all when $J = K$, emptying the active pool. With $r = 0$, minimizing time-average availability is an average-reward MDP for a censor that observes (K, J) and may block any number of discovered names after each event. Relative value iteration finds strike-all optimal at all 17 points tested (β from 0.05 to 2.0 at $k_{\max} = 8$ and from 0.1 to 1.5 at $k_{\max} = 32$, plus each

buffer's time-average strike-all frontier). Intuitively, an early block only frees a slot that the defender refills with an undiscovered name, whereas blocking a fully discovered active pool empties it.

Bursts and providers. A takedown, at rate λ_{burn}/b , moves the pool from K to $\max\{0, K - b\}$ names, so the nominal burn rate stays λ_{burn} . With P providers of $k_p = k_{\max}/P$ names each and pooled minting, a mint, at total rate λ_{intro} while fewer than $k_{\max}$ names are live, goes to a uniformly chosen provider that is not full. Takedowns, at total rate λ_{burn} , each empty a uniformly chosen nonempty provider, so $\beta = \lambda_{\text{burn}}/\lambda_{\text{intro}}$ counts takedowns, each removing up to k_p names. With $P = 1$ the model is the burst model with $b = k_{\max}$ at nominal burn ratio $k_{\max}\beta$.

LEMMA C.1 (DIVERSIFICATION AND THE SINGLE POOL). *For every $T \geq 0$, the P -provider pool stays nonempty throughout a window of length T with probability at most that of the single pool with buffer $k_{\max}$ and unit burns at rate λ_{burn} . Equality holds when $k_p = 1$.*

PROOF. Run both models from empty on a common mint clock of rate λ_{intro} and a common takedown clock of rate λ_{burn} , and let K_P and K count their live names. A mint adds a name to each model with fewer than $k_{\max}$ names. A takedown removes at least one name from a nonempty P -provider pool and exactly one from a nonempty single pool. Both events preserve $K_P \leq K$, so $K_P(t) \leq K(t)$ for all t , and the joint chain stays in the closed set of ordered states. That set carries a stationary law whose marginals are those of the two models, and under it $K_P \geq 1$ throughout the window implies $K \geq 1$ throughout it. With $k_p = 1$ every takedown removes exactly one name, and the two totals coincide. $\square$

Per-name discovery. A censor that finds each live name at its own rate δ burns at rate δK , so the pool is the birth-death chain with death rate δk in state k , and its burns slow as the pool empties. At $k_{\max} = 8$ the target (0.95, 5) holds up to $\delta = 0.212$ per mint interval. There the censor burns 0.952 names per mint interval while a name is live, the rate that step 7 of Section 7 estimates, against $\beta^* = 0.661$ for a constant-rate censor. By flow balance that rate is $\Pr[K < k_{\max}]/\Pr[K \geq 1]$, which rises with δ , because a larger δ never leaves more live names (Lemma A.2). At this buffer and target, keeping that estimate below β^* is thus conservative against this censor too.

D Data Sources and Processing

Sections 5.3 and 6 rest on four analyses of public data, covering burn units, Snowflake, literature parameters and minting economics. None of them made a new measurement or used an account, key or approval, and all requests were paced. No human subjects are involved. The Tor and Snowflake statistics are published aggregates. We refer to issue threads by URL and date, to comments by position and time, and to commits by hash, never by author, and we do not redistribute them. Each analysis has a fetch script and an analysis script. The fetch script downloaded the raw files on 2026-10-03 and recorded where and when each file came from, with its hash and license. The analysis script rebuilds the derived data offline from those files, and a rerun on the same files reproduced the derived tables exactly. A separate reimplement, not part of the artifact, recomputed every number that the analyses report.

Registry data. Russia’s blocking registry, read through the zapret-info mirror [55], lists the names, URLs and addresses under blocking orders, some of them for illegal content. Each registry file was streamed through a filter that kept only the hostnames that lie under one of 19 platform suffixes, with their decision dates. No URL path was kept. The full files were hashed for provenance and then deleted. No listed name, URL or address was resolved or fetched, and the paper and the artifact report registry content only as aggregates, namely counts and dates per platform.

Licenses. The artifact ships scripts, manifests and derived data. OONI-derived data are licensed CC BY-NC-SA 4.0, as their source requires. Sources without a license are cited and not redistributed. Live sources such as the registrar’s price list can change, but the shipped derived data keep the paper’s numbers.

Burn units. Table 1 draws on GFWatch [18], which lists only the names that China blocked by DNS, from 2020-03-20 to 2024-09-06. OONI aggregates [13, 34] cover China from July to September 2026 (azurefd.net from August 2025) and Iran and Russia from October 2024 to September 2026. IRBlock [41] lists names found blocked by scans from outside Iran between November 2024 and 2025-01-15. The registry data are seven snapshots [55] from 2021-12-31 to 2025-10-01. A tenant name is a registrable domain under a platform suffix [32], or an Azure Front Door endpoint name. In OONI, a hostname with at least three non-failed measurements counts as tested, and as blocked if at least half are anomalous or confirmed. Single-label workers.dev account names do not resolve and are excluded. First dates of tenant blocks in GFWatch are upper bounds on their onset, while suffix rules are dated by daily tests of the bare suffix, so tenant-to-suffix gaps are lower bounds. A missing suffix rule shows absence of evidence, not absence of blocking.

Snowflake. Section 6.2 uses CollecTor’s Snowflake broker statistics [43], Tor Metrics’ bridge-user estimates and event timeline [44, 46], 48 versions of Tor’s Connection Assist settings [45] from 2022-02-25 to 2026-03-18, and Internet Outage Detection and Analysis (IODA) events [20] (from 2022-01-26) to exclude shutdown days. For the Turkmenistan block and the end of Fastly’s fronting, both open-ended, each change compares the median over the initial week with the median over the 14 valid days before. For Turkmenistan, user counts are the lower bound of Tor Metrics’ estimate. User counts measure usage, not availability, and polls are not users. Derived tables omit the client credentials of Connection Assist.

Parameters and minting. Section 5.3 draws on a table of 110 values from 18 papers and one field report, each with a verbatim quote found at its cited location. No source measures λ_{burn} . Minting facts come from one registrar’s price list on one day [35], three free certificate authorities, hosting platforms’ documentation, crt.sh lookups of platform wildcard certificates, ICANN’s zone-file access terms [19], a CT honeypot study [38], and the CT protocol, policies and log list [2, 14, 15, 26]. No source shows whether any censor watches CT logs or zone files.

E Simulator, Solver and Reproducibility

Our artifact is openly available at:

<https://github.com/SecretLabOU/BlockAMole>

Event-driven simulator. Our simulator draws sample paths of the model with the next-event method and tracks the number c of unblocked endpoints and the number K of live names. Since the two layers have independent clocks, we simulate them separately. The deployment is down on the union of their down intervals. On a sample path, its interval availability is the fraction of window starts t_0 for which $[t_0, t_0 + T]$ meets no down interval $[s_j, e_j]$. This fraction equals one minus the share of window starts in $\bigcup_j (s_j - T, e_j)$, so we compute it exactly, without discretization. A run lasts 1.2×10^4 to 4×10^4 mint intervals. After discarding the first 10% of every run as warm-up, we average each point over 12 to 24 independent seeds. We compare the simulator with the exact solver in Appendix B.

Reproducing the results. Both the simulator and the solver are in the artifact, along with every result file, the two figures and the data pipelines of Appendix D. In simulations/code, start with the solver exact.py, which runs its self-test. On a 20-core machine, exact_experiments.py recomputes the exact results in about five minutes. The nine simulation experiments and the closed-form grid of Appendix B come from run_experiments.py, and rerunning them takes about an hour on one core. The artifact’s README lists the library builds behind our results and the checks we ran. With those builds and the fixed simulation seeds, reruns reproduce the stored results to within 3×10^{-14} . Most of the values that no result file stores, such as the wall-clock thresholds of Section 5.3, appear in the output of extra_values.py. To redraw Figures 1 and 2 and check them against the solver, run paper/figures/make_figures.py from the artifact’s root. The independent implementation of Appendix B and the reimplementations of Appendix D are separate checks outside the artifact.