Cache Locality and Scalability of Registration-Based Encryption Systems

by

Kai Iverson

A thesis accepted and approved in partial fulfillment of the

requirements for the degree of

Master of Science

in Computer Science

Thesis Committee:

Jun Li, Chair

Yingjiu (Joe) Li, Member

University of Oregon

Spring 2026

© 2026 Kai Iverson

This work is openly licenced via **CC BY 4.0**

ABSTRACT

Kai Iverson

Master of Science in Computer Science

Title: Cache Locality and Scalability in Registration-Based Encryption systems

A single encryption context covering ten million authenticated identities fits in 1.2 megabytes. This fact, made possible by recent advances in Registration-Based Encryption, suggests that large-scale Internet authentication need not be as expensive as the certificate model has always made it seem. But compact cryptography does not automatically produce compact deployment behavior. This thesis asks what it actually takes for that compactness to become practically useful.

The answer is not cryptographic. It is structural. Registration-Based Encryption associates a compact public object, an Encryption Context, with a Key Curator rather than with an individual identity, so that a single cached object can authenticate every contact a client makes with identities registered under that curator. Whether that potential for reuse is realized depends on how identities are grouped across curators. When identities that are used together are grouped together, traffic concentration produces cache concentration. When they are scattered, it does not. Registration assignment policy is the mechanism that converts usage pattern into cache locality.

This thesis develops an analytical model of public setup storage across multi-curator deployments, introduces the Universal Authentication Footprint as the aggregate storage required for global authentication coverage, and builds a simulation framework to study how registration structure, traffic distribution, and caching behavior interact at scale. The central finding is that client working sets can be substantially smaller than the Universal Authentication Footprint under realistic traffic distributions, but only when registration structure is aligned with demand. Switching from

random to popularity-aware assignment under Zipf-distributed traffic increases cache hit rate at small cache sizes, while both policies approach saturation once the cache is large enough and the system has reached steady state. Under stronger traffic concentration, a client holding a modest fraction of all encryption context material achieves a high cache hit rate.

The practical implication is that RBE scalability is not primarily a cryptographic problem. It is a systems optimization problem, and the existing locality structure of Internet traffic may already contain much of what is needed to make large-scale deployment practical.

CONTENTS

Abstract **3**

I INTRODUCTION **9**

II BACKGROUND **13**

2.1 Public-Key Infrastructure 13

2.2 Registration-Based Encryption 15

2.2.1 Identity-Based Encryption and the Key Escrow Problem 15

2.2.2 RBE as a Solution to Key Escrow 16

2.2.3 The Encryption Context 16

2.2.4 Scaling Properties and Constructions 18

2.3 Encryption Context as a Distributed System Resource 19

2.3.1 Certificate Distribution in PKI 19

2.3.2 Encryption Context Distribution and the Caching Opportunity 20

2.3.3 Working Sets and Traffic Locality 21

2.3.4 Caching in Large Distributed Systems 22

2.4 Terminology and Definitions 23

III ANALYTICAL MODEL OF ENCRYPTION CONTEXT SCALABILITY **25**

3.1 Encryption Context Scaling 25

3.2 Universal Authentication Footprint Scalability 27

3.2.1 The Disjoint Registrations Case 28

3.2.2 The General Case and Computational Complexity 29
3.3 Discussion and Implications 31

IV SIMULATION FRAMEWORK **32**

4.1 Simulation Goals 33
4.2 System Model 33
4.2.1 Traffic Generation Model 34
4.2.2 Key Curator Registration Assignment Model 35
4.2.3 Client Cache Model 35
4.3 Metrics and Measurements 36
4.4 Experimental Methodology 37
4.5 Limitations and Assumptions 39

V EXPERIMENTAL RESULTS **41**

5.1 Baseline Policy Comparison 41
5.2 Cache Capacity Sweep: Random Assignment 44
5.3 Cache Capacity Sweep: Popularity-Aware Assignment 45
5.4 Sensitivity to Traffic Skew 46

VI LOCALITY, OPTIMIZATION, AND SYSTEMS IMPLICATIONS **48**

VIICONCLUSION AND FUTURE WORK **51**

7.1 Conclusion 51
7.2 Future Work 52

LIST OF FIGURES

V.1 Cache hit rate as a function of cache capacity (as a fraction of the UAF) under Zipf-distributed traffic with random registration assignment, for $\alpha \in \{0.8, 1.0, 1.2\}$. 44

V.2 Cache hit rate as a function of cache capacity (as a fraction of the UAF) under Zipf-distributed traffic with popularity-aware registration assignment, for $\alpha \in \{0.8, 1.0, 1.2\}$. 45

V.3 Comparison of figure V.1 and figure V.2. Blue is the random assignment curve and orange is the popularity-aware assignment curve. This chart shows the effectiveness of each policy given different cache sizes. 46

V.4 Cache hit rate as a function of the Zipf parameter α under popularity-aware registration assignment and a 10% LRU cache. The shaded region indicates the empirically motivated range $\alpha \in [0.8, 1.2]$ observed in Internet traffic measurements. 47

LIST OF TABLES

II.1 Comparison of ciphertext and encryption context sizes across representative RBE constructions. 19

IV.1 Baseline simulation parameters used throughout the evaluation unless otherwise specified. Individual experiments deviate from these values along a single axis; all other parameters are held fixed. 38

V.1 Comparison of cache performance and request concentration under different traffic distributions and registration policies. All clients use an LRU cache with a capacity of 10% of the UAF. 42

CHAPTER I

INTRODUCTION

Modern Internet authentication systems achieve universal trust through widely deployed public-key infrastructures that enable secure communication between mutually untrusted parties [1, 2]. In practice, this is realized through certificate authorities, certificate workflows, and key mechanisms that maintain trust as identities and keys evolve over time [1]. While this approach has demonstrated great success at Internet scale, it introduces large operational complexity in managing certificate lifecycles, trust relationships, and system-wide consistency [3, 4]. Large organizations may manage enormous certificate collections, often needing significant manual effort to assess security condition and avoid outages caused by expiration or misconfiguration [5]. Despite these costs, PKI remains the foundation for Internet authentication and trust establishment [1, 2]. This persistence is not because of PKI simplicity, it is because of the absence of a broadly deployed alternative architecture with comparable trust reach.

Registration-Based Encryption (RBE) is a cryptographic primitive that has matured significantly in recent years and is beginning to warrant serious consideration as the basis for an alternative Internet authentication architecture [6, 7, 8]. Rather than issuing a separate certificate per identity, RBE aggregates many authenticated public keys into a single compact object, an *Encryption Context* (EC), maintained by a *Key Curator* (KC) [6]. To encrypt to an identity, a sender need the encryption context associated with the relevant Key curator and the identity itself; logically, the EC behaves like a directory of authenticated public keys, even though the physical construction is far more compact than such a directory would suggest [7]. A single EC can cover ten million registered identities in 1.2 megabytes [7] —roughly one bit per key. RBE is not yet a

drop-in replacement for PKI; important challenges persist around ciphertext size, efficiency, identity space, and revocation [6, 7, 8]. Still, as the constructions continue to mature, the operational question of what large-scale RBE deployment would actually look like has gone largely unasked [6, 7, 8].

A key observation this thesis makes is that the total storage needed to authenticate all identities throughout a system is surprisingly small with modern RBE constructions and, in some contexts, small enough to plausibly fit within the working memory of modern clients [7]. But, interestingly, most clients do not need global coverage, and real access patterns make the gap even wider than it might seem at first. Internet traffic is largely concentrated: users contact a relatively small subset of services, and traffic to the Internet demonstrates strong locality and heavy-tailed traffic distributions [9]. This means that the working set of EC material a client actually uses may be much smaller than the total authentication footprint of the system. Users typically interact with a relatively small subset of Internet services compared to the sum of reachable identities. If those services were registered under a single key curator, one EC, fetched once and updated infrequently, would authenticate every interaction for that entire period. This is the main operational opportunity in RBE: the working set of encryption context material that a client actually needs can be a tiny fraction of the total authentication footprint.

That concentration in access patterns, however, does not translate into concentration in public setup material automatically. The alignment has to be created. It is produced by *registration assignment*: the policy that determines which identities are grouped under which key curator, and how well that grouping reflects actual access patterns determines whether clients benefit from reuse or are forced to fetch EC material they use once and discard. *Registration assignment policy is the mechanism that converts usage pattern into cache locality*. A deployment whose registrations are organized around user communities, organizations, or other repeatedly accessed structures may lead to larger cache locality. A deployment whose registrations are fragmented without care for access structure may have desirable cryptographic scaling while still behaving inefficiently in practice [10, 11].

This framing is important because existing work on RBE has focused mainly on cryptographic properties such as setup size, security guarantees, and asymptotic efficiency, while the operational behavior of RBE as a distributed system resource has been explored much less [6, 7, 8]. Particularly, the literature has yet to fully develop a systems-level account of how public setup material is distributed, cached, reused, and shaped by system structure [6, 7, 8]. To reason about this behavior, this thesis introduces two terms. An *Encryption Context* (EC) is the material required to encrypt to identities registered under a single key curator. The *Universal Authentication Footprint* (UAF) is the minimum total encryption context material required to achieve global authentication coverage across a key curator system. This thesis introduces the terms as operational abstractions that allow the encryption context layer to be analyzed as a system resource.

Motivated by these observations, this thesis investigates how registration structure, access distribution, and caching behavior influence the practical scalability of large-scale RBE deployments. The central claim is that the practical scalability of RBE systems is determined not only by cryptographic efficiency, but also by registration policy, traffic locality, and cache behavior. A compact encryption context is necessary, but it is not sufficient. System scalability emerges from the interaction between the size of the encryption context, the way identities are grouped into key curators, and the extent to which client access patterns allow that material to be reused. Put differently, compact cryptography does not lead to compact deployment behavior unconsciously. The deployment must be structured such that reuse happens.

This thesis makes four main contributions. First, it develops an analytical model for reasoning about the encryption context footprint of RBE deployments under simplified assumptions, including EC size scaling with key curator registration capacity and the aggregate storage needed across a multi-KC system. Second, it studies how traffic locality and working-set behavior lead the practical reuse of public setup material under realistic access patterns. Third, it introduces and evaluates assignment policies that pursue registration structures that align with demand structure, showing how locality can be created. Fourth, it develops a simulation

framework for comparing registration structures under different traffic and caching models, enabling the study of when EC behaves like a reusable resource and when it behaves like a fragmented collection of wasted material.

The remainder of this thesis develops these ideas in stages. Chapter II provides the needed background on public-key infrastructure, registration-based encryption, and public setup distribution, and it introduces the terms used throughout the rest of the work. Chapter III develops the analytical model of public setup scalability and defines the quantities used to reason about aggregate storage, structure, and fragmentation. Chapter IV introduces the simulation framework used to study cache locality, traffic distribution, and registration assignment. Chapter V presents the experimental results. Chapter VI extrapolates the systems implications of those results, with particular attention to optimization and deployment design. Chapter VII discusses future work, including caching effects, adaptive replacement policies, registration replication, update costs, and more realistic deployment models. Chapter VIII concludes by summarizing the main findings and discussing their implications for Internet-scale authentication.

Taken together, this thesis argues that RBE should not be understood simply as a cryptographic primitive with desirable scaling properties, but also as a distributed systems problem whose practical success depends on locality, structure, and reuse. That shift in perspective is the main intellectual contribution of the work. Modern RBE constructions make compact global authentication possible; the question explored here is how that compactness becomes practically useful.

CHAPTER II

BACKGROUND

This chapter develops the background necessary to position the contributions of this thesis. To understand this thesis, three topics must be explored. First, a careful discussion of public-key infrastructure: how it works, where its costs come from, and why its properties in operation are shaped by the per-identity structure of certificates. Second, an exploration into registration-based encryption: its origins, its constructions, its scaling properties, and its contrast with the PKI model. Third, an analysis of distributed system resources: how they must be transported, the ways they can be cached, and what factors determine their reuse.

Throughout this chapter, PKI serves as context, but also as an analogy. Because PKI is well understood, it give a reference point against which the properties of RBE can compare and contrast. The analogies must be constructed carefully: the logical similarities between a certificate and an encryption context are real, but the physical mechanisms differ in ways that change how RBE schemes behave in practice. Those differences are the source of both RBE's appeal and the open questions that this thesis addresses.

2.1 Public-Key Infrastructure

Modern Internet authentication is built on public-key infrastructure (PKI), a system of standards, protocols, and institutions that allows two parties with no past relationship to communicate securely [1]. PKI solves a fundamental problem: how can a client be sure that the server it is connecting to is who it claims to be? The answer lies in the attestation of trust. Certificate authorities (CAs), a set of trusted third parties, attest to the binding between a party's identity and its public key. A client that trusts a CA can trust any identity that the CA has certified.

The fundamental unit of PKI is the certificate. In the X.509 standard [1], a certificate is a signed data structure that binds a public key to an identity. The identity is typically a domain name, organization name, or individual, depending on the certificate's intended use. The binding is created by the signature of a certificate authority: anyone who trusts the CA's public key can verify that the CA attested to the binding between the subject identity and the subject key. Certificates are not self-sufficient, instead part of a larger trust hierarch. A small set of root CAs sit at the top of the hierarchy and are distributed in operating system and browser trust stores [12]. Those roots issue certificates for intermediary CAs, which in turn issue end-entity certificates to servers or organizations. To validate a certificate, a party must construct and verify a chain from the end-entity certificate back to the root. This structure keeps the number of trusted root manageable while still allowing many issuers to participate in the ecosystem.

Certificates are not permanent. They carry metadata: a validity window (a not-before and not-after date), permitted usage types, and the identity of the issuer. Expiration limits the time a compromised certificate can be abused, but it also introduces a maintenance obligation. When a certificate expires, it must be renewed with a freshly signed replacement. Failing to do so will cause service outages. Furthermore, if a private key is compromised before expiration, the corresponding certificate must be revoked, typically through CRLs or OCSP [1, 13, 4]. In practice, revocation is operationally awkward and often incomplete, but it is an essential mechanisms for limiting risk.

The important structural fact for this thesis is that certificates are per-identity objects. Each authenticated identity needs its own certificate, its own issuance path, and its own maintenance lifecycle. That design is simple conceptually and familiar operationally, but it also creates a linear relationship between the number of identities and the amount of public material that must be managed. An organization that has deployed ten thousand services manages ten thousand certificate lifecycles. This linear footprint is the key point of comparison with registration-based encryption.

These observations are not an argument that PKI has failed. PKI has achieved something

difficult: widely interoperable, cryptographically grounded authentication at Internet scale [1, 2]. Its persistence reflects decades of investment and the lack of a credible alternative at comparable scale. The point is narrower. The certificate model of one certificate per identity imposes a cost structure that is linear in the number of identities served, and that linear cost appears throughout the system: in storage, in lifecycle management, in revocation infrastructure, and in the engineering burden required to keep the system healthy. Any architectural alternative that preserves PKI's trust properties while changing this cost structure is worth serious investigation.

2.2 Registration-Based Encryption

Registration-based encryption is a variation of public key encryption in which an encryption context (EC) and a recipients identity are used to encrypt a message only the intended recipient can decrypt. An encryptor holding an EC can encrypt a message bound for any identity registered under the EC's Key Curator. A single EC can provide encryption coverage for millions of identities; in efficient constructions, its size grows sublinearly in the number of identities it contains [7]. Understanding why this is possible, and what constraints it imposes, requires understanding both the history from which RBE emerged and the technical structure of its construction.

2.2.1 Identity-Based Encryption and the Key Escrow Problem.

Registration-based encryption is a direct descendant of identity-based encryption (IBE). IBE was proposed by Shamir in 1984 [14] as a way to eliminate the need for certificate distribution: rather than requiring an encryptor to first obtain and verify a recipient's certificate, the encryptor would encrypt directly to the recipient's identity string (an email address, a domain name) without any prior interaction. A trusted authority, the Private Key Generator (PKG), holds a master secret from which it derives private keys for registered identities on demand. An encryptor does not need to know the recipient's public key; it encrypts to the identity string directly, using only public system parameters. The recipient proves their identity to the PKG and receives the corresponding private key.

The first practical IBE construction was given by Boneh and Franklin in 2001 [15]. The

scheme worked, but it brought the IBE model's central problem into sharp relief: the key escrow problem. Because the PKG derives all private keys from a single master secret, the PKG can decrypt any ciphertext in the system. Users must trust the PKG not to abuse this ability, not to be compromised by an adversary, and not to be coerced by a third party. For most deployment contexts, this is an unacceptable trust requirement. A compromised PKG exposes every message ever encrypted in the system, past and future. IBE therefore remained cryptographically elegant but practically marginal for general Internet authentication.

2.2.2 RBE as a Solution to Key Escrow. Registration-based encryption, introduced in 2018 [6], was proposed as a solution to the key escrow problem that preserves IBE's operational appeal. The key departure is in how private keys are generated. In RBE, users generate their own key pairs locally, as in conventional public-key cryptography. No master secret exists from which private keys could be derived. Instead, users publish their public keys by registering them with a Key Curator (KC). The KC collects registered public keys and computes from them a compact aggregation that it makes available as public material. Because the KC never generates or holds private keys, a compromised KC cannot decrypt any ciphertext in the system. The KC's trust requirement is weaker than that of a PKG: it must be trusted to aggregate correctly and to make the correct public material available, but not to keep secrets on behalf of users.

The trust model of the KC is also weaker than that of a certificate authority in a specific and important way. A CA that is compromised can issue fraudulent certificates for any identity, allowing an attacker to impersonate any entity whose certificate a client might trust. A compromised KC can corrupt public aggregation material, potentially causing encryption to fail or to encrypt for the wrong key, but it cannot retroactively compromise ciphertexts encrypted before the compromise. This distinction is relevant to the deployment security analysis, though a full treatment is outside the scope of this thesis.

2.2.3 The Encryption Context. The public material the KC produces and distributes is called the Encryption Context (EC) in this thesis. An EC is the material required to encrypt to identities registered under a single Key Curator. Across constructions, the EC consists

of two components with different lifetimes and behaviors. The first is a static component generated once when the KC is initialized and fixed thereafter; this component is called the common random string [6] or common reference string depending on the construction, and some lattice-based constructions call it the public parameter [8]. The second is a dynamic component that evolves as new identities register; this component is consistently called the public parameter across the two most widely studied constructions [7, 6], though it is called the state in the lattice construction [8]. The terminological inconsistency across constructions reflects the youth of the field. This thesis uses Encryption Context as a construction-agnostic abstraction over both components.

The EC is the operational unit of public setup material in an RBE system. An encryptor who holds the EC associated with a KC and knows a recipient identity can encrypt a message such that only the intended recipient can decrypt. Formally, the encryption, update, and decryption algorithms take the following form [7].

- **Encryption.** $\mathsf{Enc}(EC, id, m) \to ct$: Takes the Encryption Context, a recipient identity, and a plaintext message, and produces a ciphertext.
- **Update.** $\mathsf{Upd}(pp, id) \to u$: Run by the KC, takes the current public parameters and an identity, and produces update information that the identity needs to decrypt messages.
- **Decryption.** $\mathsf{Dec}(sk, u, ct) \to m$: Takes a private key, update information, and a ciphertext, and produces the plaintext or an indication that newer update information is required.

The analogy between an EC and a PKI certificate deserves careful treatment. Logically, an EC behaves like a compact, authenticated directory of public keys: it encodes the binding between many identities and their registered public keys in a single object. An encryptor who holds an EC and knows a recipient identity can encrypt for that recipient, in the same way that an encryptor who holds a certificate can encrypt for the certificate's subject. This logical picture is accurate and useful. Physically, however, the EC does not store public keys, and the encryption algorithm does not work by extracting a key from the EC and using it. Instead, the EC, the identity, and the plaintext interact through the encryption algorithm in a way that produces a ciphertext decryptable

only by the recipient. The intermediate representation, a public key indexed by identity, doesn't appear in the computation. This is why the EC can be so compact: because it does not store keys directly, it can encode the same information in far less space than a simple directory would require. The logical analogy is correct, but the physical mechanism is fundamentally better.

2.2.4 Scaling Properties and Constructions. The defining operational property of modern efficient RBE constructions is sublinear growth of EC size with the KC's registration capacity. In the construction of [7], both the static and dynamic components of the EC grow on the order of the square root of the KC's registration capacity N:

$$|EC| \propto \sqrt{N}.$$

Ciphertexts in this construction are constant size, independent of N. Together, these two properties mean that the EC remains practical even for very large registration sets: a KC with a capacity of ten million registrations produces an EC of approximately 1.2 megabytes [7].

This was not the scaling behavior of earlier constructions. The original 2018 construction [6] and several subsequent improvements achieved their security goals with ciphertexts and public parameters that grew polynomially in a way that placed total sizes in the range of gigabytes or terabytes for realistic registration counts. The result was that RBE remained of primarily theoretical interest for several years. The efficient construction of [7] changed this by achieving sublinear EC growth and constant-size ciphertexts simultaneously while also adding the property of anonymity, discussed below. This shifted RBE from a theoretical construct to a candidate practical system.

Table II.1 summarizes the scaling properties of representative constructions.

The table illustrates the trajectory of the field: early constructions were cryptographically sound but operationally impractical; efficient constructions reduced EC sizes to ranges that make Internet-scale deployment conceivable; post-quantum constructions demonstrate that this progress need not be sacrificed for quantum resistance.

Scheme	ID Space	| Ciphertext |	| EncContext |
RB-Attribute E	$\{0,1\}^*$	$O(\log N)$	$O(N^{2/3} \cdot \log N)$
Efficient RBE	$[N]$	$O(1)$	$O(N^{1/2})$
Cuckoo Commitments	$\{0,1\}^*$	$O(\log N)$	$O(N^{1/2} \cdot \log N)$
Laconic Cryptography	$\{0,1\}^*$	$O(\log N)$	$O(\log N)$
Lattice-Based RBE	$\{0,1\}^*$	$O(\log N)$	$O(\log N)$

Table II.1: Comparison of ciphertext and encryption context sizes across representative RBE constructions.

2.3 Encryption Context as a Distributed System Resource

The preceding sections described PKI and RBE as cryptographic mechanisms. This section shifts perspective and asks how the public material each system produces behaves as a resource in a distributed system. The shift is important because a cryptographic construction with excellent asymptotic scaling can still perform poorly in practice if its distribution, caching, and reuse dynamics are unfavorable. The properties of a deployed system emerge from the interaction of the cryptographic construction with the network over which its material is distributed and the access patterns of the clients that consume it.

2.3.1 Certificate Distribution in PKI. In PKI, public material is distributed as individual certificates. The dominant delivery model for web TLS is server-push: when a client initiates a TLS handshake, the server transmits its certificate chain as part of the handshake, and the client verifies it on receipt. From a caching perspective, a certificate is a small object associated with a specific server. A client that reconnects to the same server within the certificate's validity period may receive the same certificate again (unless the server has renewed), but the certificate is useful only for that specific server. No other server can be authenticated with it.

The caching unit in PKI is therefore per-identity. A client whose access patterns span many distinct servers must process many distinct certificates. Caching at the TLS session layer can reduce the number of full handshakes, but it does not reduce the number of distinct certificates a client encounters as it communicates with a diverse set of servers. The total public material a client must handle over time grows linearly with the number of distinct servers it contacts.

2.3.2 Encryption Context Distribution and the Caching Opportunity.

RBE changes the caching calculus fundamentally. An EC is associated with a KC, not with a specific server. Every identity registered under a given KC can be encrypted for using the same EC. A client that has fetched the EC for a particular KC can encrypt to any registered identity under that KC without any additional public material. If many servers a client regularly contacts are registered under the same KC, a single EC fetch covers all of them.

The caching unit in RBE is per-KC. The benefit of holding a cached EC is proportional to how many registered identities the client actually contacts that fall under the corresponding KC. An EC covering many frequently-contacted identities pays for its storage cost many times over. An EC covering identities the client never contacts is waste. This asymmetry does not exist in PKI: a certificate is either for the server being contacted or it is not, and there is no mechanism by which a single certificate covers additional servers.

The distribution mechanics of ECs have structural parallels with other forms of public material distribution in large systems. DNS responses are distributed to resolvers and clients, cached by TTL, and reused across many queries to the same domain [16, 17]. A resolver that caches a DNS response serves subsequent queries for that name without contacting the authoritative server. Content delivery networks cache content at edge nodes geographically close to users, exploiting the fact that many clients request the same content [18]. In both systems, the value of caching depends on locality of demand: if the same resource is requested repeatedly, caching is highly effective; if every client requests a unique resource, caching provides no benefit. EC distribution exhibits the same dependency. The difference from DNS and CDN caching is that the resources being cached are not content but cryptographic public material whose value is measured not in bits delivered but in identities authenticated.

Unlike certificates, which are issued fresh and expire on a fixed schedule, ECs are updated infrequently and on a schedule determined by the rate of new registrations. The dynamic component of the EC changes only when a new identity registers under the KC; in efficient constructions, the number of updates any given registered identity ever requires is bounded by

$O(\log N)$, where N is the KC's capacity [7]. This means that ECs are relatively stable resources, much more stable than web content or DNS records, which can change at any time, and that a cached EC remains valid across many interactions before it needs to be refreshed. The infrequency of updates strongly favors caching: the amortized cost of an EC fetch, spread across the many encryptions it enables, is low.

2.3.3 Working Sets and Traffic Locality. The working set of a process is the set of distinct resources it accesses within a given time window [19]. The concept was developed to characterize memory access behavior in time-sharing systems, but it applies equally to any resource for which locality of reference holds. In the context of RBE, the relevant working set for a client is the set of ECs it needs to hold to authenticate the identities it contacts within a given period.

A client whose working set consists of a small number of ECs, because its access patterns are concentrated over identities registered under few KCs, can operate effectively with a small EC cache. Once those ECs are cached, subsequent encryptions to the same identities incur no additional fetch cost. A client whose access patterns are fragmented across many KCs, each covering different sets of identities, requires either a large cache or frequent cache misses to cover its working set. The size of the EC working set is therefore the key operational quantity: it determines how much public material a client must hold and how often it must contact KC infrastructure.

The distribution of Internet traffic has been extensively studied, and the findings are consistent across many measurement contexts: traffic is heavily concentrated. A small fraction of destinations accounts for a large fraction of requests, following a pattern approximated by a Zipf-like power law [9]. Under such a distribution, the k-th most popular destination receives traffic proportional to $1/k^{\alpha}$ for some exponent $\alpha > 0$. The practical consequence is that the top handful of destinations can account for a substantial majority of all requests, while the long tail of less popular destinations collectively accounts for many more distinct identities but far less traffic. This concentration has been observed in web traffic [9], in application-layer request distributions

at large platforms [10], and across a range of other network measurement studies. The implication for RBE is that the EC working set of a typical client, in terms of traffic volume, may be substantially smaller than the full set of ECs in the system. If the identities receiving the majority of a client's traffic are covered by a small number of ECs, those ECs can be cached and reused across most of the client's interactions.

This reasoning depends on a condition that does not hold automatically: for traffic concentration to produce EC working set concentration, the the identities a client overwhelmingly use must be grouped under the same KC. A deployment in which popular identities are distributed across many KCs, each mixing popular and unpopular identities, would fail to concentrate EC demand even if the underlying traffic is highly skewed. The mechanism that creates or destroys this alignment is registration assignment: the policy that determines which identities register under which KC. This is the central operational lever in RBE deployment. Its effects are not determined by the cryptographic construction and cannot be analyzed from the EC size formula alone; they require a systems-level model of access patterns, registration structure, and cache behavior. This observation motivates the analytical and simulation work developed in the following chapters.

2.3.4 Caching in Large Distributed Systems. Caching as a systems design problem has a long and well-developed literature. The foundational observation is that practical workloads exhibit locality of reference: the same data is likely to be accessed repeatedly within a short time window, and data near recently-accessed data is likely to be accessed soon [19]. Systems that exploit this locality through hierarchical memory caches, distributed in-memory stores, or CDN edge caches can serve a large fraction of requests from fast local storage while using slower remote storage only for the infrequent cases that fall outside the working set [10].

Cache replacement policy determines which items are evicted when a cache reaches capacity. Least Recently Used (LRU) eviction is widely used as a baseline because of its simplicity, its theoretical justification, and its practical effectiveness under heavy-tailed access distributions [20]. Under LRU, popular items tend to remain in cache naturally: they are accessed frequently enough that they are never the least recently used at eviction time. Under a Zipf-like distribution, LRU

achieves near-optimal hit rates because the working set of popular items is small relative to cache capacity, and the long tail of unpopular items is accessed too rarely to displace it. This makes LRU a natural default for systems operating in the heavy-tailed traffic regime expected in Internet access patterns.

At large scale, the relationship between caching and tail latency introduces additional considerations [11]. In distributed systems handling many concurrent requests, a small number of slow or failed components can disproportionately affect overall latency if requests depend on responses from multiple subsystems. Caching mitigates tail latency by serving common requests from fast local storage, avoiding the variance introduced by remote round trips. For RBE, a client whose EC cache covers its common access patterns can encrypt for the vast majority of its intended recipients without contacting KC infrastructure, reducing both average latency and tail latency for the authentication step.

2.4 Terminology and Definitions

Two terms used throughout this thesis are introduced here to provide clear definitions before the analytical development of Chapter III.

An *Encryption Context* (EC) is the combined public material required to encrypt to identities registered under a single Key Curator. It consists of a static component, generated at KC initialization and fixed thereafter, and a dynamic component, which evolves as new identities register. The specific naming of these components varies across constructions [6, 7, 8]; the term EC abstracts over these differences and treats the full combined material as the unit of public setup that must be distributed and cached.

The *Universal Authentication Footprint* (UAF) is the minimum total Encryption Context material required to achieve global authentication coverage across a KC ecosystem. Concretely, it is the sum of the sizes of all ECs across all KCs in the system, under the assumption that each identity is registered under exactly one KC. The UAF is an operational quantity, not a cryptographic one: it measures how much public setup material must exist, in aggregate, for the system to be able to authenticate all supported identities. Its behavior under different deployment

structures is the primary subject of Chapter III.

CHAPTER III

ANALYTICAL MODEL OF ENCRYPTION CONTEXT SCALABILITY

This chapter develops the analytical model for reasoning about the scalability of large-scale registration-based encryption systems. The goal is not to model every deployment detail or reproduce the full dynamics of a production system. Instead, this chapter isolates structural properties of encryption context under simplified assumptions so that the basic scaling behavior can be understood clearly. We start by analyzing how encryption context scales with respect to system parameters. We then build off of that to grow an understanding of how a system of encryption context grows with respect to those same system parameters.

The main question this chapter asks is straightforward: when an RBE system grows in scale, how much encryption context must be created and maintained. The answer depends on the RBE construction, key curator utilization, number of key curators, and registration assignment grouping across key curators. An understanding of these properties gives us insight into the implications of different system structures. This chapter develops a model that describes things such as how much an encryption context grows when its key curators registration load doubles and how large the universal authentication footprint is.

3.1 Encryption Context Scaling

An encryption context (EC) refers to the public material needed to encrypt to identities registered under a particular key curator. In the constructions considered in this thesis, ECs contain two components: a static component acquired once that does not change, and a dynamic component needing a bounded number of updates across its lifetime. To talk about EC scalability, we need to introduce a few terms:

- **Registration Load** (L): The number of identities currently registered under a single KC.
- **Registration Capacity** (C): The maximum number of registrations a single KC is capable of servicing. This value is selected once at construction and fixed thereafter.
- **Registration Utilization** (U): The percentage of registration capacity currently in use. Defined as L/C.

Table II.1 describes the growth of encryption context across different constructions. EC scaling needs more analysis than this table, however, because because its size is dependent on the KC registration capacity, but the best way to analyze an EC is through KC registration load. To avoid repeating math across every construction, the rest of this chapter uses the Efficient Registration-Based Encryption [7] construction for all analysis. We pick this construction because it is the most practical for real world use. The worst-case asymptotic complexity of its EC is as follows:

$$|EC| = O(\sqrt{C})$$

A key property of this asymptotic complexity is that it is sublinear with registration capacity. As a result, EC size grows more slowly than registration capacity and exhibits diminishing returns with respect to capacity increases. The ratio $|EC|/C$ approaches zero as capacity grows, meaning that the amount of encryption context required per supported registration decreases at larger scales. In other words, large scale ECs approach 0 bits per registration.

Since registration utilization is defined as $(U = L/C)$, registration capacity can be rewritten as

$$C = L/U$$

Substituting this relationship into the EC complexity expression yields

$$|EC| = O\left(\sqrt{\frac{L}{U}}\right).$$

This form makes the dependence on registration load and utilization explicit. The asymptotic upper bound increases with the square root of registration load and decreases with the square root of utilization. Consequently, EC growth exhibits diminishing sensitivity to changes in either parameter.

Consider first the effect of registration load. Holding utilization constant, increasing registration load by a factor of k increases the asymptotic EC bound by a factor of $\sqrt{k}$. For example, doubling registration load increases the bound by approximately 1.41×, while quadrupling registration load increases it by 2×. Thus, substantial increases in registrations produce comparatively modest increases in EC size.

Next, consider utilization. Holding registration load constant, increasing utilization by a factor of k reduces the asymptotic EC bound by a factor of $1/\sqrt{k}$. For example, increasing utilization from 25% to 100% reduces the bound by a factor of two. Intuitively, higher utilization allows a fixed registration load to be supported with less excess capacity, reducing the amount of EC material that must be maintained.

These relationships highlight an important trade-off. For a fixed registration load, over-provisioning capacity increases EC size, while operating closer to capacity reduces it. Conversely, supporting larger registration loads requires large ECs, but the square-root dependence ensures that EC growth remains substantially slower than registration growth. As the system scales, each additional registration contributes progressively less to the overall EC size than the registrations that came before it.

3.2 Universal Authentication Footprint Scalability

A single encryption context describes the public material for one key curator. An Internet-scale deployment, however, must allow many key curators. Having multiple KCs supports decentralization efforts, distribute registration load, and accommodate the diverse communities of identities that make up the real Internet. Reasoning about a single EC in isolation is therefore

insufficient for understanding deployment-scale behavior. A complete account of EC scalability must address the aggregate storage required across all key curators at once. This aggregate quantity is the Universal Authentication Footprint.

The Universal Authentication Footprint (UAF) is defined as the minimum total EC storage required to achieve global authentication coverage across a key curator ecosystem. This section looks at two UAF cases: the disjoint registrations case in which the computation of the UAF is simple, and the general case in which the computation of the UAF is difficult.

3.2.1 The Disjoint Registrations Case. Concretely, for an ecosystem containing k key curators, the UAF is the sum of the individual EC sizes:

$$\text{UAF} = \sum_{i=1}^{k} |EC_i|$$

This definition assumes that each identity is registered under exactly one key curator, so the registration sets are disjoint and no redundancy exists across ECs. Under this non-replicated model, the individual EC sizes do not overlap in coverage, and the UAF captures precisely the minimum storage a system needs to hold in order for every registered identity to be reachable by some encryptor. Applying the EC scaling result from the previous section, where $|EC_i| = O(\sqrt{L_i/U_i})$ for registration load L_i and utilization U_i, gives the general form:

$$\text{UAF} = O\left(\sum_{i=1}^{k} \sqrt{\frac{L_i}{U_i}}\right)$$

This general form is useful but admits a cleaner expression under simplified uniform assumptions. Suppose all k key curators share the same utilization U and carry equal registration loads, so that the total number of registrations R is divided evenly across key curators, giving each a load of $L = R/k$. Then each EC contributes the same asymptotic term, and the UAF reduces to:

$$\text{UAF} = O\left(k \cdot \sqrt{\frac{R/k}{U}}\right) = O\left(\sqrt{\frac{Rk}{U}}\right)$$

This expression makes three structural dependencies visible at once, each with its own

scaling behavior.

The first is total registrations. Holding k and U fixed, increasing R by a factor of m increases the UAF by a factor of $\sqrt{m}$. Quadrupling the total number of registered identities in the system doubles the UAF. The sublinear behavior of the individual EC carries through to the aggregate: a system that grows substantially in the number of identities is supports grows only modestly in the public setup material that it must maintain.

The second is the number of key curators. Holding R and U fixed, increasing k by a factor of m also increases the UAF by a factor of $\sqrt{m}$. This so called *fragmentation penalty* is less intuitive than the registration scaling, but it follows from the same mathematical structure. Doubling the number of key curators while keeping total registrations fixed increases the aggregate EC storage by approximately $1.41\times$. Quadrupling the number of key curators doubles the UAF. The total authentication work the system must do has not grown, but the fragmentation of that work across more curators inflates the aggregate storage required.

The third is utilization. Holding R and U fixed, increasing U by a factor of m decreases the UAF by a factor of $1/\sqrt{m}$. Doubling average utilization across the ecosystem reduces the aggregate footprint by approximately $1.41\times$. As established in the previous section, higher utilization allows a given registration load to be supported with less excess capacity, reducing both individual EC sizes and, consequently, the UAF.

3.2.2 The General Case and Computational Complexity. The UAF analysis developed above assume that each identity is registered under exactly one key curator, making the minimum-coverage selection of ECs trivially the full set of all ECs. The general case relaxes this assumption: identities may be registered under multiple key curators, so some ECs redundantly cover the same identities. In that setting, a proper subset of ECs may suffice to achieve full coverage, and choosing that subset to minimize total EC storage becomes a non-trivial optimization problem.

The problem can be stated precisely. Let $\mathcal{I}$ denote the set of all identities in the system, and let each key curator i maintain an EC covering the registration set $S_i \subseteq \mathcal{I}$. A covering selection is

any subcollection $T \subseteq \{1, \ldots, k\}$ such that $\bigcup_{i \in T} S_i = \mathcal{I}$. The minimum UAF is then:

$$\text{UAF}^* = \min_{T \subseteq \{1,\ldots,k\}} \left\{ \sum_{i \in T} |EC_i| : \bigcup_{i \in T} S_i = \mathcal{I} \right\}.$$

This is exactly the Weighted Set Cover problem, with the identity space as the universe, each key curator's registration set as a set, and each EC size as the associated weight. Weighted Set Cover is NP-hard [21].

Two distinct complexity results are worth separating carefully. The first concerns the decision version of the problem: given a budget $\mathcal{W}$, does a covering selection exist with total EC size at most $\mathcal{W}$? A yes-certificate for this question is simply the selection itself. Checking that it covers $\mathcal{I}$ and that its total weight does not exceed $\mathcal{W}$ are both polynomial operations. This places the decision problem in NP, and NP-completeness follows by reduction from unweighted Set Cover [21].

The second concerns the optimization problem itself: finding the actual minimum-weight covering selection. Here, verification is substantially harder. A proposed optimal solution cannot be confirmed correct simply by verifying that it covers all identities and computing its total size; that only confirms it is a valid covering, not that no cheaper covering exists. Ruling out a cheaper solution requires confirming that every other covering selection has equal or greater total weight, which is equivalent to asking whether a cheaper covering does not exist. Verifying that a proposed solution is globally optimal would require proving that no cheaper covering exists. This is a coNP-type certification problem and is not believed to admit polynomial-time verification unless NP = coNP [22, 23].

Because the general case is weighted set cover, exact optimization is NP-hard, so practical implementations may need to rely on approximation algorithms rather than exact search. The standard greedy algorithm for weighted set cover repeatedly selects the curator that minimizes the ratio of additional EC size to newly covered identities, and it achieves an $H_n \leq \ln n + 1$ approximation guarantee, where n is the size of the identity universe [24, 25]. This guarantee is asymptotically the best possible up to lower-order factors unless widely believed

complexity-theoretic assumptions fail [26]. Consequently, greedy selection provides a computationally efficient heuristic with provable near-optimal performance when exact optimization becomes impractical.

3.3 Discussion and Implications

The analytical model demonstrates that modern RBE constructions can support extremely large authentication ecosystems with surprisingly small amounts of aggregate public setup material. Because individual encryption contexts scale sublinearly (e.g. $O(\sqrt{N})$ with a key curator's registration capacity, the aggregate Universal Authentication Footprint (UAF) requires for global coverage grows only modestly as the absolute number of registered identities expands. This mathematical compactness untethers setup storage costs from the strict linear dependencies that characterize traditional per-identity public-key infrastructure.

Understanding the operational behavior of RBE systems therefore requires simulation of dynamic access patterns and distributed caching behavior. While static mathematical limits provide bounds for a centralized or perfectly unified deployment, they cannot capture how client caching, traffic skew, and registration assignment policies interact at scale. Moving beyond static equations to evaluate client working sets under heavy-tailed workloads becomes essential to isolate the precise conditions under which RBE acts as a reusable system resource rather than an inefficiently distributed footprint. This systemic evaluation forms the structural motivation for the simulation framework detailed in the next chapter.

CHAPTER IV

SIMULATION FRAMEWORK

This chapter develops the simulation framework used to study the operational behavior of RBE systems under realistic access patterns and registration structures. The analytical model in Chapter III shows that encryption context material scales sublinearly with a key curator's registration capacity, but this result alone does not determine how a deployed system behaves in practice. In an operational setting, clients do not access the full Universal Authentication Footprint uniformly. They repeatedly contact a relatively small subset of identities and services, and those accesses are shaped by traffic locality, registration assignment, and cache behavior.

The purpose of this chapter is not to simulate the cryptographic details of registration-based encryption, but to model the system-level structure around it. The simulation abstracts the cryptographic construction into a set of key curators, servers, and clients, and studies how EC access patterns emerge under different assumptions about traffic, registration assignment, and caching. This makes it possible to isolate the factors that influence practical scalability without introducing cryptographic complexity that would obscure the system-level effects.

The central question is whether the locality needed for efficient RBE usage arises naturally from the structure of the system, or whether it must be created through registration assignment. To answer that question, the simulation measures how often clients reuse cached encryption context material, how much EC data must be fetched over time, and how traffic concentration affects the load placed on individual key curators. These measurements are used to compare random assignment against popularity-aware assignment, and to compare uniform access against skewed workloads..

4.1 Simulation Goals

The simulation is designed to answer three questions, each corresponding to a claim developed in earlier chapters.

The first question is whether client working sets are substantially smaller than the Universal Authentication Footprint under realistic traffic distributions. Chapter III established the UAF as an aggregate quantity and showed how it scales with registration structure. That analysis is static. The simulation asks what fraction of the UAF a client actually touches during normal operation — and whether that fraction is small enough to make EC caching a practical strategy at Internet scale.

The second question is whether traffic skew alone is sufficient to produce cache locality, or whether registration assignment is required. This distinction is the central empirical question of the thesis. A deployment in which popular identities are scattered across many key curators will not benefit from traffic concentration, even if that concentration is extreme. A deployment in which popular identities are grouped under the same key curator will see the same traffic concentration reflected directly in cache reuse. The simulation tests both conditions, holding traffic fixed while varying assignment policy, to isolate the contribution of each.

The third question is how cache capacity constrains the locality benefits identified above. Even if a client's working set is small relative to the UAF, it may still exceed realistic cache budgets. The simulation sweeps cache size to identify the capacity threshold below which cache hit rate degrades significantly, providing a concrete estimate of the storage requirements for efficient RBE operation.

Taken together, these three questions determine whether RBE is operationally practical at Internet. Not as a cryptographic claim, but as a systems claim about distribution, caching, and reuse.

4.2 System Model

The simulation uses a simplified model of RBE that focuses on system behavior rather than cryptographic implementation. The main entities are clients, servers, and key curators. Each server is registered under one or more key curators, and each key curator maintains an encryption

context covering its registered identities. The EC is treated as the unit of public setup material: an opaque object whose presence or absence in a client's cache determines whether a given server contact requires a fetch.

A client maintains a cache of ECs. When the client wishes to contact a server, it selects a server according to the traffic model, identifies the key curator under which that server is registered, and checks whether the relevant EC is already in its cache. A cache hit allows the client to proceed immediately. A cache miss triggers a fetch of the EC from the key curator before the contact proceeds. In the base model, each server is registered under exactly one key curator, which keeps the relationship between traffic and EC demand easy to interpret and isolates the effects of registration assignment and cache behavior from the additional complexity of replication.

The simulation also tracks per-KC request volume, which is used to assess whether assignment policy creates load concentration as a side effect of improving cache locality. The relationship between cache efficiency and load concentration is a system-level tradeoff not visible in the analytical model alone.

4.2.1 Traffic Generation Model. Traffic generation determines which servers clients attempt to contact over time. Two models are used in this simulation, each isolating a different structural feature of real access patterns.

The first is a uniform model, in which every server is selected with equal probability. This provides a control case with no built-in locality. Under uniform traffic, any cache reuse that emerges must come from the registration structure alone, not from the request distribution.

The second is a globally skewed model, in which server selection follows a Zipf distribution with a single global popularity ranking. This model captures the heavy-tailed character of Internet traffic: a small number of servers receive a disproportionately large share of requests, while the long tail of less popular servers accounts for many distinct identities but comparatively little traffic. This concentration is well documented in web traffic and large-scale platform workloads and is the primary reason to expect that EC working sets might be small relative to the UAF.

These two traffic models are ordered by interpretive complexity. The uniform model

establishes the baseline. The global skew model isolates the effect of popularity concentration.

4.2.2 Key Curator Registration Assignment Model. Registration assignment determines how servers are distributed across key curators. It is the mechanism through which access patterns are translated, or fail to be translated, into cache locality. The simulation treats it as a primary experimental variable.

The random assignment model distributes servers across key curators without regard to popularity. This is the structural null case: it represents a deployment with no deliberate alignment between registration structure and demand. Under random assignment, popular servers are scattered across many key curators, so even a client with highly concentrated traffic must hold many ECs to cover its common contacts.

The popularity-aware assignment model groups the most frequently accessed servers under the same key curator, the next most frequently accessed under another, and so on. This intentionally aligns registration structure with request structure. The purpose is not to assert that this grouping is universally optimal, but to test a specific hypothesis: that aligning KC membership with access popularity causes traffic concentration to produce EC concentration, which in turn produces cache locality.

The critical experimental comparison is between these two assignment conditions under identical traffic. Any difference in cache hit rate between random and popularity-aware assignment, holding everything else fixed, is attributable to assignment policy alone. This comparison is the primary empirical result of the simulation.

4.2.3 Client Cache Model. Client-side caching determines whether ECs must be repeatedly fetched or can be reused across multiple server contacts. Two cache models are used.

The infinite cache model retains every EC once fetched. This provides an upper bound on reuse and isolates the effect of working set structure from the additional pressure of eviction. Under an infinite cache, the only misses are compulsory: the first time a client encounters a key curator. Every subsequent contact with an identity registered under that curator is a hit. This baseline makes the maximum achievable locality visible.

The finite LRU cache model imposes a fixed capacity and evicts the least recently used EC when the cache is full. LRU is used because it is standard, well-understood analytically, and effective under heavy-tailed access distributions: popular ECs are accessed frequently enough that they are never the least recently used at eviction time. Finite cache experiments are run at several capacity levels to identify the cache size at which hit rate degrades significantly relative to the infinite-cache baseline.

4.3 Metrics and Measurements

Each metric is chosen to answer a specific question about the system. The goal is not to characterize every aspect of simulation behavior, but to provide the measurements necessary to evaluate the three claims in Section 4.1.

Cache hit rate is the primary metric. It measures the fraction of server contacts for which the required EC is already present in the client's cache. Cache hit rate is the direct operational consequence of locality: high hit rate means the client rarely contacts KC infrastructure; low hit rate means EC fetches are frequent enough to impose real distribution cost. Hit rate is reported separately for each combination of traffic model and assignment policy, and the difference in hit rate between the random and popularity-aware assignment conditions under identical traffic, the assignment-induced locality gain, is the central experimental quantity of the simulation.

Working set size as a fraction of the UAF measures the operational gap between global authentication coverage and what a client actually needs. For each traffic model, the working set is defined as the set of distinct ECs a client contacts within a fixed time window. Reporting this quantity as a fraction of the UAF converts an absolute count into a statement about the structure of the deployment: if the working set is two percent of the UAF, then a client holding two percent of all public setup material can authenticate the vast majority of its contacts. This ratio is the empirical form of the core systems insight stated in Chapter I.

Cache hit rate as a function of cache capacity maps the relationship between storage budget and reuse efficiency. This measurement is taken by fixing the traffic model and assignment policy and sweeping cache size from a few ECs to the full working set. The resulting curve identifies the

capacity threshold below which hit rate degrades sharply, and the plateau above which additional cache space yields diminishing returns. This curve provides a concrete answer to the question of whether working set concentration is sufficient to make EC caching practical under realistic memory constraints, or whether it only becomes effective at cache sizes that are unrealistic.

Key curator request concentration measures whether popularity-aware assignment concentrates load on a small number of key curators as a side effect of improving cache locality. Concentration is measured using the Gini coefficient of the per-KC request distribution, which provides a single summary statistic that can be compared across assignment conditions. This metric captures the system-level tradeoff that is not visible from the client's perspective alone: a deployment that achieves excellent cache locality by concentrating registrations may simultaneously create KC hotspots that impose operational problems at the infrastructure level.

These four metrics together provide a view of the system at two levels. Cache hit rate and working set size characterize client-side behavior: how much a client must fetch and how often it can reuse what it has. Cache hit rate as a function of capacity characterizes the sensitivity of that behavior to storage budget. KC concentration characterizes the infrastructure-level consequence of the assignment policies that produce client-side efficiency.

4.4 Experimental Methodology

The experiments are organized around a fixed number of baseline conditions, varied one factor at a time, to keep comparisons interpretable. Table IV.1 lists the parameter values used as defaults across all experiments. Any experiment that departs from these values states the deviation explicitly; all other parameters remain as given. The number of servers, key curators, and clients is held constant throughout, so that differences in measured outcomes can be attributed to the factor under study—traffic model, registration policy, or cache capacity—rather than to changes in system scale. The burn-in period of fifty timesteps is excluded from all reported metrics; figures and tables in Chapter **??** reflect steady-state behavior only.

The experimental sequence is as follows. The first experiment uses uniform traffic and random assignment. This is the structural null case: no built-in locality in the traffic, no structure

Parameter	Value
Number of clients	100
Number of servers	100,000
Number of key curators	100
Simulation timesteps	300
Burn-in timesteps	200
Traffic policy	Zipf-distributed
Default Zipf parameter (α)	1.0
Registration policy	Random
Cache policy	Infinite cache
Target KC utilization	0.5
Random seeds	{1, 2, 3, 4, 5}

Table IV.1: Baseline simulation parameters used throughout the evaluation unless otherwise specified. Individual experiments deviate from these values along a single axis; all other parameters are held fixed.

in the assignment. It provides the reference against which all other conditions are measured. The second experiment introduces globally skewed traffic while keeping assignment random. This isolates the effect of traffic concentration on cache reuse in the absence of any deliberate registration structure. The third experiment holds traffic fixed at the globally skewed model and switches to popularity-aware assignment. This isolates the effect of assignment policy by holding the demand distribution constant. The comparison between the second and third experiments is the primary empirical contribution of the simulation. A fourth set of experiments repeats the second and third conditions with finite LRU caches at several capacity levels, producing the hit-rate-versus-capacity curves described in Section 4.3.

Each experiment is run with multiple fixed random seeds to ensure that results are not artifacts of a particular sample. Results are reported separately for the cold-start phase, during which caches are empty and all misses are compulsory, and for the steady-state phase, after which cache contents stabilize and hit rate reflects the working set structure. The cold-start phase is treated as a burn-in period and is excluded from the primary hit rate measurements; it is reported separately to characterize how quickly the system converges to steady state under each condition.

All experiments hold constant the number of servers, the number of key curators, the client

population size, and the length of the simulation run. The only variables that change across conditions are the traffic model, the assignment policy, and the cache capacity. This discipline ensures that differences in measured outcomes can be attributed to the factor under study rather than to confounding changes in scale.

4.5 Limitations and Assumptions

This simulation is intentionally simplified. It does not model the cryptographic cost of RBE operations, network-level latency, or the detailed mechanics of EC retrieval. Instead, it treats ECs as discrete objects whose presence or absence in cache determines the cost of a server contact. This abstraction is appropriate because the thesis is concerned with system-level locality and scalability rather than cryptographic or network implementation cost.

The model does not include EC updates, revocation dynamics, or parameter rotation. These features are important in a full deployment, but they would substantially complicate the analysis without changing the core structural questions the simulation is designed to answer. Because ECs are updated infrequently in efficient constructions — the number of updates any registered identity ever requires is bounded by O(log N) — omitting update dynamics is a reasonable approximation for the time windows studied here.

The simulation also treats all ECs as uniform in size. In a real deployment, EC size varies with key curator capacity and utilization as described in Chapter III. Uniform EC sizes allow cache capacity to be expressed in number of ECs rather than bytes, which simplifies the presentation of hit-rate-versus-capacity curves without changing the qualitative shape of the results. Translating these results into byte-denominated storage estimates requires applying the scaling relationships from Chapter III to the working set sizes reported here.

Finally, the traffic and assignment models are stylized. Real Internet traffic is more complex than a single global Zipf distribution, and real registration structures involve communities that overlap and evolve over time. These simplifications are a deliberate methodological choice: a model that clearly isolates the structural effect of registration assignment is more analytically valuable for this thesis than a more complex model whose behavior is difficult to attribute to any

single cause.

CHAPTER V

EXPERIMENTAL RESULTS

This chapter reports the results of the simulation experiments described in Chapter IV. The experiments are organized in order of increasing structural complexity: from a uniform baseline with no deliberate registration structure, through the introduction of traffic skew and popularity-aware assignment, and finally to a systematic sweep of the Zipf parameter that reveals how the degree of traffic concentration governs the ceiling of achievable cache locality. Throughout this chapter, the focus is on what the data shows. Interpretation and systems implications are deferred to Chapter VI.

The baseline simulation parameters are given in Table IV.1. Unless otherwise noted, all experiments use the values in that table, varying only the factor under study. All results are measured after the burn-in period of fifty timesteps, during which caches are empty and all misses are compulsory; reported figures reflect steady-state behavior. Each condition is run across five random seeds, and results are averaged across seeds; variance across seeds was negligible for all reported metrics.

5.1 Baseline Policy Comparison

The first four experiments establish the principal comparison of the simulation: they hold system scale and cache capacity fixed at 10% of the UAF and vary traffic distribution and registration assignment policy, measuring the effect of each change in isolation. The results are summarized in Table V.1.

The first experiment provides the structural null case. Traffic is uniform across all servers and registration is random, so neither the access distribution nor the registration structure contains any

Traffic	Registration	Cache Size	Hit Rate	KC Gini
Uniform	Random	10%	10.2%	0.030
Uniform	Popularity-Aware	10%	10.0%	0.032
Zipf (α = 1.0)	Popularity-Aware	10%	73.6%	0.455
Zipf (α = 1.0)	Random	10%	18.6%	0.260

Table V.1: Comparison of cache performance and request concentration under different traffic distributions and registration policies. All clients use an LRU cache with a capacity of 10% of the UAF.

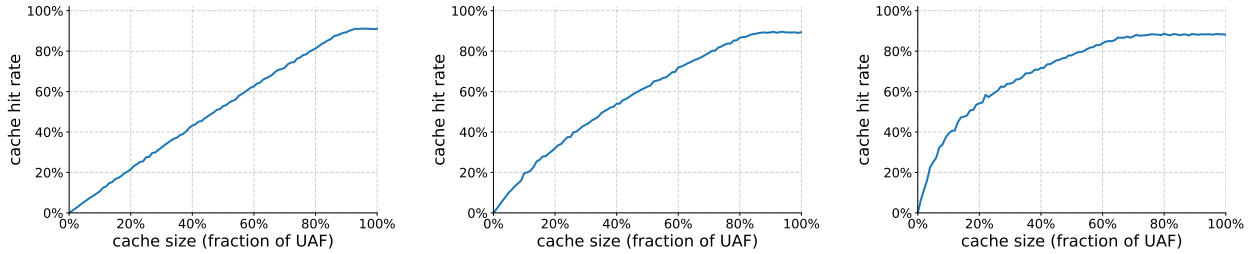
deliberate locality. Under this condition, cache hit rate reaches 10.2% in steady state. This result closely tracks the cache capacity itself: a client holding 10% of the UAF under uniform traffic will, on average, find a requested EC in cache on approximately one in ten contacts. The slight excess above 10% reflects random variation in the traffic model. The KC Gini coefficient of 0.030 confirms that request load is nearly uniformly distributed across all key curators, consistent with what is expected under uniform server selection.

The second experiment introduces globally skewed traffic following a Zipf distribution with $\alpha = 1.0$, while keeping registration assignment random. This isolates the effect of traffic concentration in the absence of any structural alignment between registration and demand. The change from uniform to Zipf traffic produces a modest improvement in cache hit rate: 18.6, compared to 10.2 in Experiment 1. The KC Gini coefficient rises to 0.260, indicating that skewed traffic begins to concentrate requests toward a smaller number of key curators. The LRU cache captures some benefit from this concentration, naturally retaining the most frequently contacted KCs and achieving a hit rate nearly double the baseline. These results demonstrate that traffic skew alone is not sufficient to produce substantial cache locality improvements when registration assignment is random. The popular servers are scattered across many different key curators, so their concentrated traffic does not concentrate EC demand proportionally. A client that repeatedly contacts the same popular server will hit the cache for that server's KC, but its other popular contacts are likely registered under different curators, requiring separate fetches. Traffic concentration is present in the demand, but the registration structure does not allow it to be fully

expressed as EC reuse.

The third experiment keeps traffic fixed at the same Zipf distribution ($\alpha = 1.0$) as Experiment 2 and changes only the registration assignment policy, from random to popularity-aware. Under popularity-aware assignment, servers are sorted by access frequency and assigned to key curators in contiguous popularity bands, so that the most frequently accessed servers share a KC, the next most frequently accessed share another, and so on. Every other aspect of the simulation is identical to Experiment 2. The effect of this change is large. Cache hit rate rises from 18.6 to 73.6, a gain of more than fifty-five percentage points attributable entirely to the change in assignment policy. The KC Gini coefficient rises substantially, from 0.260 to 0.455, reflecting the concentration of request load onto the small number of key curators whose registered servers are most popular. Notably, this hit rate is achieved with a cache holding only 10% of the UAF: the client captures nearly three quarters of all requests by caching a small fraction of the EC space. This is the central empirical result of the simulation. The demand distribution is identical between Experiments 2 and 3; only the registration structure changes. The difference in cache hit rate between the two conditions is the assignment-induced locality gain: direct evidence that registration assignment policy is the mechanism that converts traffic concentration into EC cache locality. Traffic skew creates the latent potential for locality; popularity-aware assignment realizes it.

The fourth experiment completes the two-by-two design by pairing uniform traffic with popularity-aware assignment. This condition serves as a control: it asks whether the gains observed in Experiment 3 are attributable to the assignment policy itself, or whether they depend on an interaction between the policy and the traffic distribution. The result confirms the expected null outcome. Hit rate reaches 10% and the KC Gini coefficient is 0.032, both essentially indistinguishable from Experiment 1. When all servers are accessed with equal frequency, the notion of popularity carries no information, and assigning servers to key curators in popularity order is effectively equivalent to random assignment. The assignment policy has no structure to exploit. Taken together with Experiment 3, this result isolates the interaction between skewed



(a) $\alpha = 0.8$ (b) $\alpha = 1.0$ (c) $\alpha = 1.2$

Figure V.1: Cache hit rate as a function of cache capacity (as a fraction of the UAF) under Zipf-distributed traffic with random registration assignment, for $\alpha \in \{0.8, 1.0, 1.2\}$.

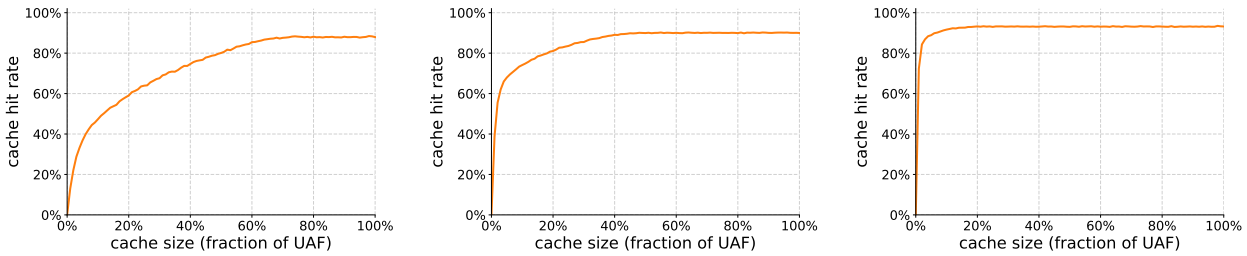
traffic and popularity-aware assignment as the necessary condition for locality: neither factor alone is sufficient.

5.2 Cache Capacity Sweep: Random Assignment

Experiments 1–3 used an LRU cache with a capacity 10% the size of the UAF. This provides a snapshot into how each policy combination behaves under these specific conditions. It demonstrates the power of projecting traffic distribution into cache hits through registration assignment. Experiment 4 takes this further by sweeping cache sizes from 0% to 100% of the UAF, holding traffic fixed at Zipf with $\alpha \in \{0.8, 1.0, 1.2\}$ and registration fixed at random assignment. The results are shown in Figure V.1.

The first thing to notice about all three curves is that that they approach a 100% cache hit rate when the cache size is 100% of the UAF. This makes intuitive sense because each simulation had a large burn-in time to fill up the caches before making any measurements. A cache capacity equal to the UAF will cache every single encryption context encountered and never drop it. We set up the experiment this way to get at something more fundamental and decoupled from the the timestep parameters. The revealing information is contained in the way the lines reach that maximum threshold.

The effect of the α parameter is visible in how quickly the curve grows with respect to cache size. When $\alpha = 0.8$, the curve forms a nearly linear relationship. This makes intuitive sense because the less aggressive traffic model and random assignment are not given usable structure to work with to create anything more efficient. Therefore, the odds of a cache hit are proportional to



(a) $\alpha = 0.8$ (b) $\alpha = 1.0$ (c) $\alpha = 1.2$

Figure V.2: Cache hit rate as a function of cache capacity (as a fraction of the UAF) under Zipf-distributed traffic with popularity-aware registration assignment, for $\alpha \in \{0.8, 1.0, 1.2\}$.

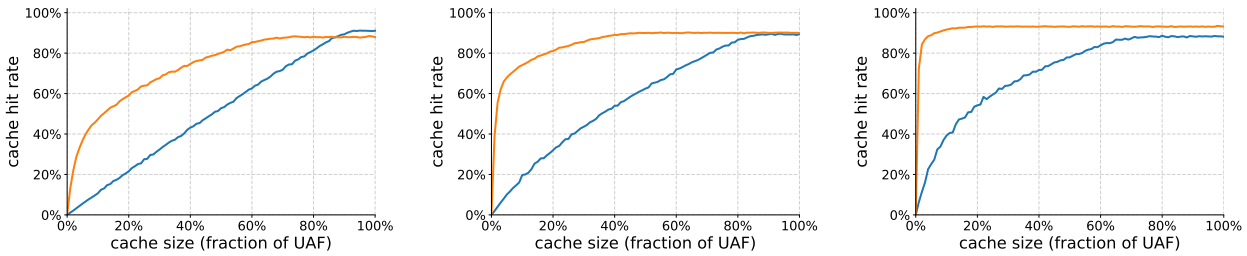
how large the cache is. When $\alpha = 1.0$, the curve starts to rise quicker than before. The more aggressive traffic patterns combine with the random assignments to make a set of key curators where some happen to be more popular than others. The model takes advantage of these randomly popular key curators to form a curve that is slightly more efficient. When $\alpha = 1.2$ the curve grows even faster and earlier. The traffic pattern is most aggressive leading to, by random chance, key curators that have even more separation in popularity that the model can take advantage of. This curve experiences jitter because it is so dependent on popular servers randomly aligning to create popular KCs. However, in all three cases, the curve does not grow significantly faster than a linear relationship. To get an 80% hit rate, each model needs at minimum 50% of the total UAF.

5.3 Cache Capacity Sweep: Popularity-Aware Assignment

Experiment 5 repeats the capacity sweep of Experiment 4 under identical traffic conditions, but switches registration assignment from random to popularity-aware. The results are shown in Figure V.2.

The contrast with Figure V.1 is immediate. Under popularity-aware assignment, the hit-rate curves rise sharply at small cache sizes and plateau significantly soon.

For $\alpha = 0.8$, the curve resembles the $\alpha = 1.2$ random-assignment case in its general shape, but reaches its plateau at a smaller cache size. For $\alpha = 1.0$, the hit rate reaches approximately 70% with a cache holding only 10% of the UAF, and 80% by 20% of the UAF, leveling off just above 90% by 40%: a plateau that required roughly 90% of the UAF to approach under random assignment. For $\alpha = 1.2$, the result is more striking: at a cache size of less than 5% of the UAF,



(a) $\alpha = 0.8$ (b) $\alpha = 1.0$ (c) $\alpha = 1.2$

Figure V.3: Comparison of figure V.1 and figure V.2. Blue is the random assignment curve and orange is the popularity-aware assignment curve. This chart shows the effectiveness of each policy given different cache sizes.

the hit rate is already near 90%, and the curve levels off just above that value.

These results make concrete the claim developed in Chapter III: that cache hit rate is higher with smaller caches under realistic traffic distributions, provided that registration structure is aligned with demand. Under random assignment, a modest cache only produces a modest hit rate. Under popularity-aware assignment, a cache covering 10–20% of the UAF is sufficient to capture the dominant working set for $\alpha = 1.0$, and less than 5% suffices for $\alpha = 1.2$.

The two figures together, Experiments 4 and 5, demonstrate that registration assignment policy is not merely a marginal optimization. The difference in cache capacity required to achieve a given hit rate between the two conditions is an order of magnitude under the stronger traffic skew values. The EC material that a client needs to cache for practical operation is largely determined not by the global UAF, but by how well the registration structure reflects the client's access patterns.

5.4 Sensitivity to Traffic Skew

Experiments 1–5 held α to specific values within a realistic range. Experiment 6 varies α continuously from 0.5 to 1.5 under popularity-aware assignment with an LRU cache at a capacity 10% of the UAF, measuring the resulting steady-state hit rate across the full range. The result is shown in Figure V.4.

Hit rate rises monotonically across the full range of α. At $\alpha = 0.5$, where traffic is only mildly concentrated, the steady-state hit rate is approximately 19%. At $\alpha = 1.0$, it reaches 75%.

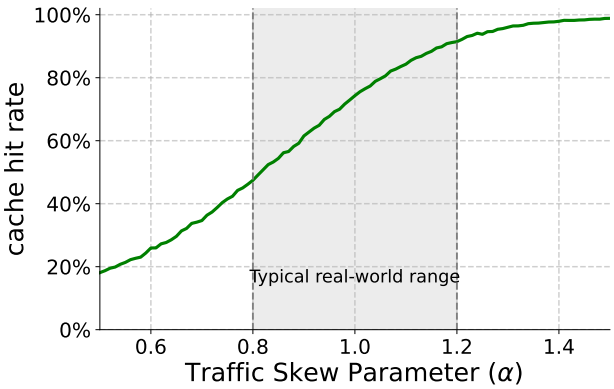


Figure V.4: Cache hit rate as a function of the Zipf parameter α under popularity-aware registration assignment and a 10% LRU cache. The shaded region indicates the empirically motivated range $\alpha \in [0.8, 1.2]$ observed in Internet traffic measurements.

At $\alpha = 1.5$, it approaches 100%. The curve is not linear: the rate of change in hit rate with respect to α is low at the extremes of the range and highest near $\alpha = 0.9$. This gives the curve a shape with a gradually rising left flank and a long right flank that accelerates before leveling off as hit rate saturates near unity at high α.

Within the empirically motivated range of $\alpha \in [0.8, 1.2]$, indicated by the shaded region in Figure V.4, hit rate spans from roughly 48% to just above 90%. This range brackets the results observed in earlier experiments: $\alpha = 0.8$ yields a hit rate consistent with the low end of plausible Internet traffic concentration, while $\alpha = 1.2$ yields a hit rate approaching what would be expected if only a small fraction of identities received the vast majority of all traffic.

The shape of this curve has a direct operational interpretation. How far can a small cache capacity take the hit rate with respect to traffic distribution. Because the derivative of hit rate with respect to α is highest near $\alpha = 0.9$, small changes in the degree of traffic concentration produce relatively large changes in cache efficiency in the regime most characteristic of real Internet workloads. Deployments operating near this inflection point are most sensitive to accurate modeling of their traffic distributions. Conversely, at both extremes, the marginal effect of α on hit rate shrinks, and the system behavior is more predictable. The upper extreme approaches a regime in which a handful of key curators receive nearly all traffic, and a client holding only those ECs achieves near-perfect hit rate regardless of other system parameters.

CHAPTER VI

LOCALITY, OPTIMIZATION, AND SYSTEMS IMPLICATIONS

The experimental results establish a clear principle: the practical efficiency of an RBE deployment is determined not by the cryptographic construction in isolation, but by how well its registration structure reflects the locality already present in client access patterns. Traffic concentration is not something a deployment engineer creates. It is a structural property of Internet workloads that exists independent of any authentication architecture. What a deployment engineer controls is whether that concentration is allowed to express itself as cache locality, or whether it is destroyed by registration structure that scatters co-accessed identities across different key curators.

The design principle that follows is straightforward: identities should be grouped under the same key curator when clients are likely to need them together. When this condition holds, a cache hit on one identity in a curator's registration set predicts future hits on others, because the access patterns that brought the client to one identity tend to bring it to the rest. An EC fetched under these conditions earns its cache footprint across many subsequent interactions. An EC fetched under fragmented registration, where a client contacts one identity from a curator and rarely returns to any other, earns nothing from the fetch beyond that single contact. The former is a reusable resource. The latter is waste.

This framing reorients the deployment problem. The question is not simply how many key curators to deploy or how large to make their registration capacities, but how to draw the boundaries between them. The UAF analysis from Chapter III introduced a fragmentation penalty: holding total registrations fixed, increasing the number of key curators increases aggregate EC storage as the square root of the curator count. This penalty is not merely an abstract accounting

cost. It is also a working set cost. A fragmented deployment requires clients to hold more distinct ECs to cover the same set of frequently contacted identities, directly increasing cache pressure. Consolidating registrations reduces the UAF, and consolidating them along lines that reflect access patterns reduces the client working set further still. Both effects point in the same direction.

The load concentration introduced by popularity-aware assignment is a real tradeoff that any deployment must confront. The Gini coefficient results in Chapter V confirm that grouping popular identities under shared curators concentrates request volume onto a small number of KCs. This creates infrastructure hotspots that must be accounted for in capacity planning. However, this concentration is also predictable and stable: the same curators that receive disproportionate traffic are the curators whose ECs are most valuable to cache. Infrastructure sizing for high-traffic curators is a tractable engineering problem, and the concentration itself is a sign that the assignment is working. A deployment whose per-KC request distribution is nearly uniform is, under skewed traffic, almost certainly one whose registration structure does not align with demand.

A further implication concerns the sensitivity of cache efficiency to the degree of traffic skew. The results of Experiment 6 show that the marginal effect of increased traffic concentration on cache hit rate is largest near the center of the empirically observed range, around $\alpha = 0.9$. This means that real Internet deployments are likely operating near the inflection point where registration structure matters most: small improvements in how identities are grouped can yield meaningful gains in cache efficiency, and small regressions can undo them. This sensitivity argues for treating registration assignment as a first-class operational concern rather than a one-time setup decision. As access patterns evolve, as new services grow popular and old ones decline, registration structure should be revisited to maintain alignment with demand.

Finally, the NP-hardness result from Chapter III bears a practical implication. Optimal registration assignment across a large, evolving identity space is computationally intractable in general. But the simulation results suggest that tractable, greedy approximations are likely sufficient in practice. The gains from popularity-aware assignment over random assignment are large even under a simple contiguous-band grouping with no sophisticated optimization. The

structure of Zipf-distributed traffic is forgiving: the most popular identities account for a disproportionate share of traffic, so grouping them correctly matters far more than perfectly optimizing the long tail. A deployment that gets the top of the popularity distribution right captures most of the available locality gain, and greedy or community-based heuristics are well-suited to that task.

CHAPTER VII

CONCLUSION AND FUTURE WORK

7.1 Conclusion

This thesis began with a single observation: a single encryption context covering ten million identities fits in 1.2 megabytes. The question it pursued was whether that compactness translates into practical deployment efficiency, and under what conditions it does so. The answer is that it does, but not automatically. Compact cryptography is a necessary condition for efficient RBE deployment, not a sufficient one. The sufficient condition is structural: identities must be grouped under key curators in a way that reflects how clients actually access them. When that alignment exists, traffic concentration produces cache concentration, client working sets shrink relative to the UAF, and the encryption context layer behaves like a reusable distributed resource. When it does not, the same cryptographic construction produces a fragmented footprint that clients must repeatedly fetch and discard.

The analytical model developed in Chapter III showed that the Universal Authentication Footprint grows as $O(Rk/U)$ under uniform assumptions, revealing a fragmentation penalty that inflates aggregate EC storage as the number of key curators increases. The simulation results of Chapters IV and V gave this penalty operational meaning: under popularity-aware assignment with Zipf-distributed traffic at $\alpha = 1.0$, a client holding 10% of the UAF achieves a 73.6% cache hit rate, compared to 18.6% under random assignment with identical traffic. Neither traffic skew nor registration structure alone is sufficient; their interaction is what produces locality. The central design implication is that registration assignment should be treated as a first-class operational

concern, not an afterthought.

7.2 Future Work

Several directions follow naturally from this thesis and are worth pursuing in depth.

Analytical cache modeling. The simulation results suggest that cache behavior under heavy-tailed traffic is tractable to model analytically, at least in simplified cases. An interesting starting point is the expected cache occupancy after a burn-in period under an infinite cache and Zipf-distributed traffic, which should admit a closed-form expression in terms of α and the number of key curators. From there, the expected hit rate as the cache continues to fill during the measurement phase can be derived as a function of the cache occupancy distribution. This would give a clean analytical counterpart to the simulation curves in Chapter V and provide tighter bounds on working set size as a function of α.

Registration replication. This thesis assumed disjoint registrations: each identity belongs to exactly one key curator. Relaxing this assumption opens the most interesting direction for future work. A realistic Internet deployment might support overlapping curators organized around natural communities. Imagine a top-500-sites KC, a university KC, a regional KC, a platform-specific KC, with identities registered under multiple curators simultaneously. It is not obvious how replication affects client working sets or cache efficiency. Hit rates may improve because clients have more options for which EC to cache. Aggregate EC storage may increase because the same identity contributes to multiple ECs. The relationship between replication structure and the NP-hard covering problem introduced in Chapter III becomes significantly more complex. The general case of overlapping registrations needs a new analytical framework and a simulation model that can represent and evaluate the additional degrees of freedom.

When is EC grouping beneficial? The thesis assumes throughout that grouping many identities under a single EC is advantageous. This deserves direct scrutiny. An EC is a sublinear aggregate public key: it encodes authenticated bindings for many identities in less space than a directory would require, but it couples those identities together operationally. A client that needs one identity from a large EC must fetch and cache the whole thing. If access patterns are

sufficiently fragmented, if clients rarely need more than one identity from any given EC, the grouping provides no reuse benefit and may impose a larger per-contact fetch cost than individual certificates would. Characterizing the conditions under which EC grouping is net beneficial, as a function of traffic structure, EC size, and individual certificate size, is a foundational question for any serious RBE deployment analysis. If grouping is not beneficial under realistic fragmentation levels, that is a significant negative result for the architecture.

Revocation and expiration. This thesis set aside the question of how registered identities are revoked or how ECs age. These are not peripheral concerns. A practical deployment must handle compromised keys and identity turnover without invalidating the entire EC or requiring expensive reissuance. One natural approach is to embed a revocation list directly in the EC, since revoked identities should represent a small fraction of total registrations, the space impact may be modest and consistent with the sublinear scaling properties of the EC itself. Expiration could be handled at the EC level rather than the per-identity level, simplifying lifecycle management at the cost of forcing coordinated renewal. Both approaches seem plausible, but neither has been analyzed carefully, and the interaction between revocation structure, EC update frequency, and cache invalidation introduces new tradeoffs that require dedicated study.

REFERENCES CITED

[1] Sharon Boeyen et al. *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*. RFC 5280. May 2008. DOI: 10.17487/RFC5280. URL: https://www.rfc-editor.org/info/rfc5280.

[2] Zakir Durumeric et al. "Analysis of the HTTPS certificate ecosystem". In: *Proceedings of the 2013 Conference on Internet Measurement Conference*. IMC '13. Barcelona, Spain: Association for Computing Machinery, 2013, pp. 291–304. ISBN: 9781450319539. DOI: 10.1145/2504730.2504755. URL: https://doi.org/10.1145/2504730.2504755.

[3] Zakir Durumeric et al. "The Matter of Heartbleed". In: *Proceedings of the 2014 Conference on Internet Measurement Conference*. IMC '14. Vancouver, BC, Canada: Association for Computing Machinery, 2014, pp. 475–488. ISBN: 9781450332132. DOI: 10.1145/2663716.2663755. URL: https://doi.org/10.1145/2663716.2663755.

[4] Yabing Liu et al. "An End-to-End Measurement of Certificate Revocation in the Web's PKI". In: *Proceedings of the 2015 Internet Measurement Conference*. IMC '15. Tokyo, Japan: Association for Computing Machinery, 2015, pp. 183–196. ISBN: 9781450338486. DOI: 10.1145/2815675.2815685. URL: https://doi.org/10.1145/2815675.2815685.

[5] Ponemon Institute. *Trends in PKI Security: A Global Study of Trends, Challenges & Business Impact*. Global study of PKI operational challenges and certificate management practices. CyberArk. 2026. URL: https://www.cyberark.com/resources/analyst-reports/ponemon-institute-trends-in-pki-security-a-global-study-of-trends-challenges-business-impact (visited on 05/26/2026).

[6] Sanjam Garg et al. "Registration-Based Encryption: Removing Private-Key Generator from IBE". In: *Theory of Cryptography: 16th International Conference, TCC 2018, Panaji, India, November 11–14, 2018, Proceedings, Part I*. Panaji, India: Springer-Verlag, 2018, pp. 689–718. ISBN: 978-3-030-03806-9. DOI: 10.1007/978-3-030-03807-6_25. URL: https://doi.org/10.1007/978-3-030-03807-6_25.

[7] Noemi Glaeser et al. *Efficient Registration-Based Encryption*. Cryptology ePrint Archive, Paper 2022/1505. 2022. DOI: 10.1145/3576915.3616596. URL: https://eprint.iacr.org/2022/1505.

[8] Michael Klooß et al. *Scalable Registration-Based Encryption from Lattices*. Cryptology ePrint Archive, Paper 2026/717. 2026. URL: https://eprint.iacr.org/2026/717.

[9] L. Breslau et al. "Web caching and Zipf-like distributions: evidence and implications". In: *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*. Vol. 1. 1999, 126–134 vol.1. DOI: 10.1109/INFCOM.1999.749260.

[10] Rajesh Nishtala et al. "Scaling Memcache at Facebook". In: *10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. Lombard, IL: USENIX Association, Apr. 2013, pp. 385–398. ISBN: 978-1-931971-00-3. URL: https://www.usenix.org/conference/nsdi13/technical-sessions/presentation/nishtala.

[11] Jeffrey Dean and Luiz André Barroso. "The Tail at Scale". In: *Communications of the ACM* 56 (2013), pp. 74–80. URL: http://cacm.acm.org/magazines/2013/2/160173-the-tail-at-scale/fulltext.

[12] Tariq Fadai et al. "Trust me, I'm a Root CA!" In: *Analyzing SSL Root CAs in Modern Browsers and Operating Systems*. Aug. 2015, pp. 174–179. DOI: 10.1109/ARES.2015.93.

[13] Stefan Santesson et al. *X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP*. RFC 6960. June 2013. DOI: 10.17487/RFC6960. URL: https://www.rfc-editor.org/info/rfc6960.

[14] Adi Shamir. "Identity-Based Cryptosystems and Signature Schemes". In: *Annual International Cryptology Conference*. 1984. URL: https://api.semanticscholar.org/CorpusID:1402295.

[15] Dan Boneh and Matthew Franklin. *Identity Based Encryption From the Weil Pairing*. Cryptology ePrint Archive, Paper 2001/090. 2001. URL: https://eprint.iacr.org/2001/090.

[16] *Domain names - concepts and facilities*. RFC 1034. Nov. 1987. DOI: 10.17487/RFC1034. URL: https://www.rfc-editor.org/info/rfc1034.

[17] *Domain names - implementation and specification*. RFC 1035. Nov. 1987. DOI: 10.17487/RFC1035. URL: https://www.rfc-editor.org/info/rfc1035.

[18] Waris Ali, Chao Fang, and Akmal Khan. "A survey on the state-of-the-art CDN architectures and future directions". In: *Journal of Network and Computer Applications* 236 (2025), p. 104106. ISSN: 1084-8045. DOI: https://doi.org/10.1016/j.jnca.2025.104106. URL: https://www.sciencedirect.com/science/article/pii/S1084804525000037.

[19] Peter J. Denning. "The working set model for program behavior". In: *Commun. ACM* 26.1 (Jan. 1983), pp. 43–48. ISSN: 0001-0782. DOI: 10.1145/357980.357997. URL: https://doi.org/10.1145/357980.357997.

[20] Claire Maıza et al. "Fast and exact analysis for LRU caches". In: *CoRR* abs/1811.01670 (2018). arXiv: 1811.01670. URL: http://arxiv.org/abs/1811.01670.

[21] Richard M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations: Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, and sponsored by the Office of Naval Research, Mathematics Program, IBM World Trade Corporation, and the IBM Research Mathematical Sciences Department*. Ed. by Raymond E. Miller, James W. Thatcher, and Jean D. Bohlinger. Boston, MA: Springer US, 1972, pp. 85–103. ISBN: 978-1-4684-2001-2. DOI: 10.1007/978-1-4684-2001-2_9. URL: https://doi.org/10.1007/978-1-4684-2001-2_9.

[22] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[23] Sanjeev Arora and Boaz Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[24] Vašek Chvátal. "A Greedy Heuristic for the Set-Covering Problem". In: *Mathematics of Operations Research* 4.3 (1979), pp. 233–235.

[25] Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2001.

[26] Uriel Feige. “A Threshold of ln n for Approximating Set Cover”. In: *Journal of the ACM* 45.4 (1998), pp. 634–652.