# Evaluating Registration-Based Encryption for Use in Internet Protocols

by

Ryan Kovatch

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

Dan Carrere, Member

University of Oregon

Spring 2026

© 2026 Ryan Kovatch
CC BY-NC-ND 4.0

# THESIS ABSTRACT

Ryan Kovatch

Master of Science in Computer Science

Title: Evaluating Registration-Based Encryption for Use in Internet Protocols

Recent developments in registration-based encryption (RBE) have produced methods that scale efficiently to millions or even billions of users, but no attempts have been made to apply them to true systems of this size on the Internet. By evaluating a set of leading constructions, this thesis argues that RBE has become feasible for providing security functions in key Internet services. We compare asymptotic behaviors of each algorithm with constraints posed in Internet protocol design to determine RBE's viability as a key encapsulation mechanism in modern hybrid cryptosystems. We also conduct a case study on ReDNS, an internally-developed encrypted DNS protocol that leverages RBE, to elucidate further practical considerations when applying the encryption scheme to Internet communications.

ACKNOWLEDGMENTS

My most sincere and heartfelt thanks are extended to my committee, Jun Li and Dan Carrere, for believing in the work and extending every deadline for me. Jun, your advice throughout the ReDNS project was invaluable, and your passion for RBE is infused throughout this work. Dan, people like you are the reason I love cybersecurity. Your mentorship from my undergrad to now has been a vital resource and I am very proud to call you a friend.

To my research partner, Kai Iverson. Getting to know you this year has been a privilege I'll never take for granted. You are whip-smart, the most thoughtful and knowledgeable software developer — nay, *engineer* — that I've ever met. I've learned so much from you in what, looking back, seems like such a short time. Thank you for all the afternoons spent whiteboarding the hell out of our protocol, for all the debates on terminology and nomenclature, and for all the patience as I debugged the thousandth serialization issue. I owe you a beer mile.

To the rest of my incredible lab, the CCSP — or more affectionately, the Jeffy B Fan Club: Eric and Ethan. There's no one I'd rather procrastinate for hours with. Kicking around that poor Kirby plush, let's-playing obscure video games based on hit animated films, or posing for photos in the funhouse mirror webcam made for some bright spots in otherwise tough times. Long live that brown recliner.

To Joe, who was not part of the lab but somehow made himself a fixture of it: thanks for stopping by, champion. I'll always pick up the phone for you. And to all the other regulars of colloquium: Tamanna, Hritvik, Zac, Jonny, and Michael. Without you all, I genuinely could not have finished grad school. I don't say that lightly; too many ciders with a bunch of goofballs every Friday on the dot turns out to be very high up on my hierarchy of needs. Thank you all for

embracing me so readily, and for busting down my door when I was hesitant to go out. Every decision to hang with y'all is a good one.

To Gaysexy, the greatest trivia team to ever exist (among other things, I guess). Jess, Ally, Marina — you are my absolute rocks, the tried to my true blue, the laughter I never want to stop coming home to. My favorite memories of the past five years have been with y'all. I hope they never find a cure for whatever is wrong with us.

Finally, to my family — Mom, Dad, Katie — for creating the conditions that allowed me to thrive, all the way from the start of my schooling. Thank you for your endless belief in me and in the paths I charted for myself. This Master's degree is the culmination of years of tireless work. No matter how far it takes me, home will always be near to my heart.

For Natasha Bedingfield.

The rest — as they say — is still unwritten.

TABLE OF CONTENTS

Chapter Page

I. INTRODUCTION ..... 10

Background ..... 12

Public-Key Infrastructure ..... 12

Registration-Based Encryption ..... 13

Problem Statement ..... 15

Contributions ..... 16

II. ANALYSIS OF LEADING RBE CONSTRUCTIONS ..... 18

RBE Construction Taxonomy ..... 18

Ciphertext Size ..... 19

Pairing-Based ..... 20

Lattice-Based ..... 21

Encryption Context ..... 22

Pairing-Based ..... 23

Lattice-Based ..... 25

Computational Efficiency ..... 28

Pairing-Based ..... 28

Lattice-Based ..... 29

Summary: Pairings vs. Lattices ..... 30

Chapter Page

III. ReDNS: A CASE STUDY....32
The Encrypted DNS Problem.... 32
Protocol Overview.... 34
Selecting an RBE Construction.... 35
Results....36
Advantages....36
Challenges....38
Supporting Resource-Constrained Devices.... 38
The Bounded Identity Problem....39
Securing the Key Curator....40
Multi-KC Systems.... 42
IV. CONCLUSION....44
REFERENCES CITED....47

LIST OF TABLES

Table Page

1. Estimated concrete ciphertext sizes and size complexities for each construction, given a large N value. 20
2. Asymptotic size complexities for static and dynamic EC components in each construction. 23
3. Concrete sizes for static and dynamic EC components in each construction, given $N = 2^{30}$. 23
4. Asymptotic complexities for encryption and decryption functions by construction. 28
5. Concrete encryption and decryption times by construction for certain large N values. 28
6. Satisfaction of key Internet encryption constraints by construction. 30

# INTRODUCTION

Encryption serves as the backbone of all private communications online. An estimated 85% of all Internet traffic is encrypted [1], with Google reporting 98% of web browsing in the United States occurring over encrypted channels [2]. This hardens the vast majority of online services against eavesdropping and manipulation by obscuring and tamper-proofing message contents while they are in transit [3]. Encryption can occur at all seven layers of the OSI model, from the physical to the application layer [4], though it is most visible at the higher layers in protocols that users encounter day-to-day, such as HTTPS.

The most widely-used cryptographic protocol on the Internet is known as Transport Layer Security (TLS). Serving as the backbone of many other protocols (like HTTPS), it uses public-key cryptography to exchange a secret key that is then used in a symmetric cipher to encrypt packets [5]. This is a traditional example of a *hybrid cryptosystem*, where asymmetric cryptography provides an authenticated key-encapsulation mechanism (KEM) during a handshake, after which point clients can switch to a much more efficient symmetric encryption scheme for data confidentiality and integrity protection. This allows systems to take advantage of the convenience of public-key encryption as well as the speed of secret-key encryption to transmit data privately [6]. TLS is just one example; most encrypted Internet protocols use hybrid cryptosystems, including IPSec/IKE [7], SSH [8], and QUIC [9].

Because public keys are the primary bootstrapping material for hybrid cryptosystems, the Internet has coalesced around a large-scale system for exchanging them known as Public-Key Infrastructure (PKI). The global PKI is responsible for distributing *certificates*, which are signed envelopes that securely bind their enclosed public keys to real-world identities. While this is a critical function for the integrity of Internet communications, the deployment of PKI creates

headaches in many systems: certificates introduce kilobytes of overhead on the wire [10], complex and error-prone management routines [11], and a fragile architecture that frequently requires entire branches of the Internet's trust system to be rebuilt [12].

A new encryption scheme called registration-based encryption (RBE), developed within the last decade, allows for the binding of public keys to identities without certificates. Instead, users register their public keys to a *key curator* (KC), which aggregates them into a compact data structure we refer to as the *encryption context* (EC). Encrypting parties use this encryption context, plus the recipient's identity (such as name, email, phone number, etc.) to encrypt a message without ever having to know the recipient's public key. Hence, RBE-based systems avoid the common pitfalls of PKI, eschewing the need for certificates entirely and forgoing complex hierarchical trust organization.

This thesis aims to evaluate leading constructions of RBE on their usability in modern hybrid cryptosystems, such as those in popular encrypted protocols like TLS, with the goal of determining whether RBE can feasibly provide security functions in Internet communications today. To this end, we use asymptotic analysis to evaluate key RBE constructions on their communication costs, cacheability characteristics, and cryptographic efficiency — critical properties for adapting RBE to the heterogeneous Internet. We also conduct a case study on the development of ReDNS, the first practical deployment of RBE at the Internet protocol level, examining breakthroughs and challenges we encountered while applying the encryption scheme to protect the domain name system. Ultimately, we show that despite its nascency, RBE has become feasible for providing security functions in key Internet services.

## Background

### Public-Key Infrastructure

In traditional PKI, certificate authorities (CAs) sign and distribute certificates to organizations that can then be used to issue additional, subsidiary certificates with specific purposes. At the very top of this system, a set of root CAs is distributed out-of-band and used to verify the trust chain of every other certificate encountered. An example of a common administrative structure might be: a university acquires one or more CAs (called "intermediate" CAs) from a root CA such as Sectigo. It then uses these CAs to issue "end-entity" certificates for particular operational needs, like serving web traffic over HTTPS or signing executable code.

There are a few difficulties with this scheme. One is that in networked applications, hybrid cryptosystems must be initialized using the server's public key, which necessitates fetching not just the server's certificate, but all certificates on the path from the end-entity to the root CA. These are referred to as certificate *chains*. Unfortunately, certificate chains are quite large, up to multiple kilobytes in size [10], which means fetching them adds overhead that can eclipse the size of the actual message needing to be sent securely. In DNS-over-HTTPS, for example, the overhead introduced by TLS' certificate fetching is an order of magnitude larger than any other layer. This creates additional latency and memory requirements that can be undesirable in certain contexts.

Another issue is that public key infrastructure *management* is difficult, and misconfigurations can be unforgiving. Large enterprises may issue hundreds of thousands of certificates over their lifetime [11], each with their own validity period that may not align cleanly with others. Certificate lifecycle management becomes a logistical nightmare as a result: over two thirds of businesses report suffering at least one certificate-related outage per month [13],

and each outage can incur costs upwards of $100,000 [11]. In extreme (though not necessarily rare) cases, this can produce large-scale outages affecting millions of users.

Finally, PKI systems rely on the weakest link in any certificate chain to uphold trust, which makes them quite fragile. A single compromised certificate (i.e., a leaked private key) immediately compromises all certificates below it in the hierarchy, sometimes requiring large swaths of the infrastructure to be rebuilt [14], which also creates significant downtime. This is especially bad the closer it occurs to the root; in fact, several root CAs have been compromised in the past [15]. A notable example of this is DigiNotar, a Dutch root CA that fraudulently issued hundreds of digital certificates to attackers for domains belonging to Google, Microsoft, and Yahoo, among others. This led to DigiNotar's CA being removed from the root of trust, triggering browser warnings for users trying to access hundreds of sites, including those of the Dutch government. As a result of the incident, DigiNotar (whose primary business was in digital trust) was forced to declare bankruptcy later that year [12].

If it were not for PKI's complete and utter domination of the online trust ecosystem, these issues would warrant serious reconsideration. A system that finds itself at the center of so much managerial headache and costly mistakes demands solutions. Fortunately, a promising alternative has recently arrived.

## Registration-Based Encryption

Registration-based encryption, first proposed by Garg et al. in 2018 [16], is an improvement on identity-based encryption (IBE) efforts that began in the 1980s. RBE specifically removes the primary limitation of IBE, which was the necessary retrieval of private decryption keys from a third party known as the private key generator (PKG). This meant that the PKG was inherently trusted not to misuse said private keys to decrypt traffic on its own, a

tradeoff that is unacceptable in many use cases. This was referred to as the *key-escrow problem,* and plagued identity-based encryption schemes for over 30 years [16]. Now, RBE has made identity-based encryption possible using self-generated keypairs.

Instead of using certificates to bind public keys to identities, RBE accomplishes this binding by aggregating public keys on a central key curator. These public keys are *accumulated* s.t. no one is extricable from the others without the identity of its originator (be it a name, email address, phone number, etc.) [17]. This makes the public key holder's identity central to the encryption process, so a client with a correct target identity can never encrypt to an incorrect public key, thus providing the same basic assurance that a traditional certificate does.

Additionally, this property of RBE enables *non-interactive* encryption — assuming we already have the aggregated public key material from the KC (the encryption context), we can encrypt straight to our recipient having never contacted them before. Compare this to PKI, where an application needs both the encryption context (its root CA store) and supplementary info retrieved interactively (the server's certificate chain) before encrypting. In RBE, the encryption context plays a much more significant role, containing all the public key material an encrypting party could ever need in just a small space.

This means that, in a theoretical world where CAs operated KCs instead of issuing certificates, one would only have to download, say, Verisign's public parameters *once* to encrypt traffic to *millions* of registered organizations. Importantly, these public parameters are *compact*, which means they grow sublinearly with the capacity of the KC [18], enabling incredibly large registration capacities while keeping communication costs low. The entire Internet's encryption could theoretically be supported by a relatively tiny set of KCs, much like it is now with the set of trusted root CAs (of which there are only around 150 [19]). This presents exciting

opportunities for real-world applications in Internet protocols, but it is yet to be seen whether the theoretical advantages of the encryption scheme hold practically.

**Problem Statement**

While RBE represents a new frontier for public-key encryption, its contributions to the field of cryptography are heretofore purely theoretical. As it stands, RBE has not yet seen any applied usage in networked systems, and thus it is not known whether RBE as a system can be adapted to modern Internet communication paradigms. In particular, if RBE is to be deployed on the Internet, it should adhere to the constraints of a modern hybrid cryptosystem. We propose three main constraints: low communication cost (objects sent over the wire are *compact* both asymptotically and concretely), cacheable encryption context (which amortizes any up-front retrieval cost), and cryptographic efficiency (for good performance even in resource-constrained environments).

At the packet level, it is crucial to save as many bytes as possible so as to leave room for additional payloads and prevent exceeding the maximum transmission unit (MTU) size, which is traditionally 1500 bytes [20]. Exceeding the MTU can result in packets being fragmented, increasing the likelihood of retransmission if a data segment is lost in transit. It can also result in packets being dropped at certain routers [21]. For these reasons, payloads that exceed the MTU are often transmitted over reliable connection-based transports instead, such as TCP. To enable transport agnosticism, we need to communicate only compact values over the wire, and the size of these values should be (in all practical circumstances) measured in bytes or single-digit kilobytes. We refer to this metric as the *communication cost*. Any candidate RBE construction that achieves secure communication with low overhead on the wire will be in competition with PKI-based protocols.

Additionally, we gain no advantage by front-loading the communication of public key material (in the form of encryption context) if that key material is too large or needs to be re-fetched often. We look for strong *cacheability* of the encryption context by examining its concrete and asymptotic sizes (to ensure it will not be too large to store in memory or on disk) as well as its frequency of invalidation (an asymptotic measure that sets an upper bound on the number of times it will ever need to be re-fetched). To achieve the best space efficiency, we desire small encryption context. We also seek out constructions whose encryption contexts are long-lived to maintain a high cache hit rate and avoid re-fetching during encryption. This allows us to preserve the benefits that RBE presents over certificate fetching in traditional PKI.

Finally, as with all other algorithms, we also care about efficiency, which determines the algorithm's usefulness in computational environments that might be pressed for processing power. The Internet of Things in particular comes to mind; IoT devices often run off of basic microprocessors and already struggle to adhere to security best practices [22]. Introducing a protocol that would significantly delay or complicate their operations would result in low adoption in that space, and thus a missed opportunity for securing the greater Internet.

Cryptographers working to develop new constructions of RBE are only peripherally considering constraints like these if at all; applied RBE is of course far outside the scope of their contributions, which are incredibly significant on their own. The field of RBE is nonetheless missing context surrounding their encryption scheme's actual usefulness in digital communications. We aim to provide some of that context in the chapters that follow.

**Contributions**

This thesis delivers two primary contributions: a comparison of leading RBE constructions to identify favorable properties for hybrid cryptosystems, and a case study on the

development of the first Internet protocol to use RBE as its key-encapsulation mechanism. The comparison (Chapter II) will weigh each construction's asymptotic behaviors against Internet-aligned constraints on object size and time complexity. The case study (Chapter III) will further serve to determine RBE's viability on the Internet by surfacing additional practical considerations when deploying it on top of existing Internet infrastructure. These parts come together to support our overarching claim that RBE is presently mature enough to provide security functions in key Internet services.

# ANALYSIS OF LEADING RBE CONSTRUCTIONS

In this chapter, we assess a handful of state-of-the-art RBE constructions for three important properties. First, communication cost will be approximated by evaluating the asymptotic growth of ciphertext objects as well as their minimum sizes. Then, EC cacheability will be determined via its concrete and asymptotic sizes as well as its frequency of invalidation. Finally, efficiency will be evaluated by observing the time complexities of functions compared to the environments they can reasonably be expected to run on. This will allow us to select a set of RBE constructions that satisfy the requirements for Internet-scale deployment.

## RBE Construction Taxonomy

There are two primary mathematical bases that RBE constructions are founded upon: bilinear pairings and lattices. Pairing-based constructions use pairing-friendly elliptic curves and rely on a form of the discrete logarithm problem for computational hardness [23]. Lattice-based constructions, on the other hand, rely on the hardness of the shortest vector problem for their security [24]. Each basis has its strengths and tradeoffs; in this thesis, we consider two constructions of each type.

The following RBE constructions have been described by their authors as "scalable," "fast," or "efficient." They account for the most recent, state-of-the-art constructions in the field. Referred to by the acronyms of their authors' names, they are:

- **Pairing-based:**
    - **GKMR:** [17], specifically the construction *without* efficient updates described in section 4.
    - **FKP:** [25], specifically the second pairing-based construction (P2) with selective compactness.

- **Lattice-based:**
    - **ZDMD:** [26], specifically the “more efficient” construction described in section 4.
    - **KLSS:** [18].

Note that we avoid constructions of RBE which expand its definition to enable attribute-based encryption (RABE), functional encryption (RFE), or other more esoteric features. While an interesting area of research, these variants of RBE improve only on functionality presently, and are markedly less efficient and practical than stricter RBE constructions [18]. The same goes for blockchain-based constructions, which require significantly more compute and are decentralized in nature, which does not represent the centralized majority of online services [27]. As a result, we do not consider them for use in Internet protocols yet.

**Ciphertext Size**

The most visible and important object in a theoretical encrypted Internet protocol is its ciphertext. This is the data that is communicated most often in a cryptographic system, so its size greatly influences the total volume of added communication overhead when encryption is enabled. On the wire, we want ciphertexts to be quite small, measured in bytes or single-digit kilobytes, due to latency requirements and constraints posed by the MTU size. A sufficiently compact ciphertext enables secure key encapsulation and exchange using only one or a small handful of packets.

| Construction | N | Ciphertext Size | Complexity |
|---|---|---|---|
| GKMR | Any | 914 B | $O(1)$ |
| FKP | Any | 2.7 KB | $O(1)$ |
| ZDMD | ~$2^{30}$ | 287 KB | $O\left(\log^2 N\right)$ |
| KLSS | $2^{128}$ | 7.1 MB | $O(\log N)$ |

Table 1: estimated concrete ciphertext sizes and size complexities for each construction, given a large $N$ value.

Pairing-Based

The only RBE construction in existence that satisfies a ciphertext size less than the standard MTU is GKMR's, which touts a vanishing 914 bytes. Unlike other constructions, the ciphertext is constant-size, so it does not grow with $N$, the maximum number of supported users, at all [17]. This allows us to securely exchange a secret in a single segment/datagram, which is critical to a lightweight protocol, especially on lower levels of the OSI model. This incredibly compact ciphertext comes at a cost, however: GKMR's is the only construction to use a bounded integer identity set, so identities such as usernames or email addresses cannot be used without hashing to a number [25]. This significantly limits the applicability of the construction.

The FKP construction builds on GKMR's by removing the limitation of the bounded identity set. It accomplishes this by implementing "cuckoo hashing," which uses multiple hash functions to map unbounded sets to bounded ones by "evicting" certain items to alternate hash values when collisions occur. This means each unbounded identity (such as "bob@email.com") maps to $k$ integer identities (with $k$ being the number of hash functions), but only occupies one at a time. To handle the "moving around" of identities, FKP add on vector commitment witness encryption (VCWE) to ensure only the present occupant of an integer identity can decrypt ciphertexts to that identity. This has the effect of increasing the ciphertext size to $|ct| =$

$k\left(914 + \left|ct_{VCWE}\right|\right)$ bytes. Because the ciphertext introduced by the VCWE scheme is constant-size, the construction maintains a constant-size ciphertext overall. The efficient 'P2' variant we select fixes $k = 2$, so the resulting ciphertext is at least $2 * 914 = 1828$ bytes in size. In reality, the authors observed a 3x increase in ciphertext size [25], indicating $|ct| \approx 2742$ bytes and thus $\left|ct_{VCWE}\right| \approx \frac{914}{2} = 457$ bytes in their implementation. While this does exceed the MTU, it's not entirely unwieldy; compare this value to the size of a single certificate, which can come out to 2 - 4 KB [10]. Because we already trade objects of this size in, say, a TLS handshake, it's not unreasonable to say this would be completely feasible in a reliable transport protocol.

Lattice-Based

The next largest ciphertext size is in the ZDMD construction, which achieved a benchmark result of 148 KB for a system of 1000 users. This is nowhere near Internet-scale, but by inspecting how the ciphertext is constructed, we can deduce how it might grow to over a billion users. The ciphertext is composed of $(k + 2m)$ elements, where $k$ is the size of the secret key (a constant not specified in the benchmark) and $m = n\left(ceil\left(\log_{64} q\right) + 1\right)$. (The log base is 64 due to the use of $d$-ary decomposition where $d = 64$.) $q$ is a prime selected such that $\frac{q}{4}$ is greater than a value which depends on $N$ (whose equation is omitted here but can be found on page 20 of the text). All elements of the ciphertext are polynomials of degree $n_R = 256$ whose coefficients are the same bit-length as $q$. If $N$ grows by a factor of 1 million (i.e. from 1,000 to 1 billion), then $q$ must grow by at least an additional ~20 bits (because $2^{20} \approx 10^6$). For the $N = 1000$ benchmark, $q$ occupied 59 bits, so we assume it would have to grow to ~80 bits. While we don't know $k$, the ciphertext size is dominated by the $2m$ term, so for the benchmark's

$n = 4$ and $q \approx 2^{80}$ [26], we have that $m \approx 4 \cdot 14 = 56$. This gives a ciphertext of approximately $2m = 112$ polynomials, each one storing 256 coefficients, with each coefficient consuming 10 bytes (80 bits). The result is $112 \cdot 256 \cdot 10 = 286,720$ bytes, or about 287 KB. While this is impressive in theory, in practice it is still many times larger than the overhead introduced by key encapsulation in PKI, requiring at least 192 segments under the usual MTU. This more or less confines it to the higher layers of the OSI model, as lower-level protocols must be highly compute- and memory-optimized.

Finally, the KLSS construction produces the largest ciphertext by an order of magnitude, but with impressive scalability properties. The authors achieve $O(\log N)$ size complexity with a remarkably low coefficient: for $N = 2^{30}$, the ciphertext is around 7 MB, and increasing $N$ to $2^{128}$ (the size of the entire IPv6 space!) only increases the ciphertext size by 100 KB [18]. This insensitivity to $N$ demonstrates Internet-scale potential out of the box. However, an initial size of 7 MB is still a deal-breaker below the application level. At that large, one would incur significant delays during key exchange on slow networks/unreliable transport, and the value will struggle to fit in memory on microcontrollers or other resource-constrained devices.

**Encryption Context**

All RBE constructions are founded on the idea that an encrypting party should never query anyone for the recipient's identity [17]. Instead, encrypting parties fetch a set of public material from the KC, which they can then use to privately encrypt such that the target identity is never revealed. We term this the encryption context (EC). The EC is usually comprised of a static component, sometimes referred to as the *common reference string* (CRS), and a dynamic component, sometimes referred to as the *public parameters*. (The EC can also contain other material besides the CRS and public parameters, hence the generic distinction of static vs.

dynamic.) The static component is fetched once-and-for-all. The dynamic component needs to be updated (re-fetched) some number of times over the lifetime of the KC; the rarer these updates occur, the better. In this section, we look at the size of each construction's EC and the relative update frequency of their dynamic components to determine the added overhead of fetching (and re-fetching) this material.

| **Construction** | **Static Component** | **Dynamic Component** | **Number of Updates** |
|---|---|---|---|
| GKMR | $O\left(\sqrt{N}\right)$ | $O\left(\sqrt{N}\right)$ | $O\left(\sqrt{N}\right)$ |
| FKP | $O\left(\sqrt{N}\right)$ | $O\left(\sqrt{N}\log N\right)$ | $O(\log N)$ |
| ZDMD | $O\left(\log^2 N\right)$ | $O\left(\log^2 N\right)$ | $O(\log N)$ |
| KLSS | $O(1)$ | $O(\log N)$ | $O(\log N)$ |

Table 2: asymptotic size complexities for static and dynamic EC components in each construction.

| **Construction** | **Static Comp.** | **Dynamic Comp.** |
|---|---|---|
| GKMR | 12.7 MB | 3.2 MB |
| ZDMD | 29.8 MB | 7.1 MB |
| KLSS | 64 B (wire)<br>849.4 KB (expanded) | 322.6 KB |

Table 3: concrete sizes for static and dynamic EC components in each construction, given $N = 2^{30}$. FKP is omitted for lack of concrete size information in their paper.

Pairing-Based

GKMR's construction has a very traditional EC in that its static component consists only of the CRS, which has a minimum size of $|crs| = \left(2ceil\left(\sqrt{N}\right) - 1\right) * (49 + 97)$ $= 292\left(ceil\left(\sqrt{N}\right) - \frac{1}{2}\right)$ bytes, which for $N = 2^{30}$ gives $|crs| = 292 \cdot 2^{15} - 146$ $= 9,568,110$ bytes or 9.5 MB. Its dynamic component consists of a set of public parameters

and a set of "updates," which are both arrays of $ceil\left(\sqrt{N}\right)$ 49-byte commitments, so an $N = 2^{30}$ system will reach a size of $49 \cdot 2^{15+1} = 3,211,264$ bytes or 3.2 MB. This brings the total footprint of the EC to ~12.7 MB for a system that supports over 1 billion users. The public parameters need to be re-fetched $O\left(\sqrt{N}\right)$ times, which is only necessary if the encrypting party tries to encrypt to an identity that registered after their last fetch. The updates object is fetched upon registration and re-fetched $O\left(\sqrt{N}\right)$ times by decrypting parties over the lifetime of the KC [17]. This is quite good for Internet protocol use; if the CRS is distributed with an operating system or library, and public parameters + decryption updates are fetched in the background every so often, encryption and decryption can be more or less transparent.

FKP inherit most of the properties of GKMR's construction in their own. Just as before, the CRS is the static component and the public parameters are the dynamic component. With the added functionality, the CRS gains additional material in the form of cuckoo hashing public parameters (CHPP) and vector commitments CRS (VCCRS). The CHPP are constant-size and the VCCRS is said to be $O\left(\sqrt{N}\right)$ [25], so the overall CRS is expected to grow but not faster than before. As for the public parameters, they now include a new vector commitment $D$ which binds unbounded identities to hashed ones, and a "stash" $S$ which stores identities that have overflowed after colliding in both hash domains. Their size is given as $|pp| = \left(\sqrt{N} + |D| + \log N\right) \cdot \log N$. Since $D$ is a constant-size group element [25], we can describe the public parameters' growth as $O\left(\sqrt{N}\log N\right)$. Unfortunately, the authors do not give any concrete sizes for the EC components in their paper (nor any source code for benchmarks), so it's difficult to gauge where they start from. As for updates, clients only need to re-fetch dynamic components $O(\log N)$ times — an improvement on GKMR's $O\left(\sqrt{N}\right)$ [25]. Overall, FKP's construction increases the size

complexity of the EC in exchange for unbounding the identity space, a tradeoff which harms viability in lightweight or unreliable-transport protocols, but may or may not outweigh the scalability benefits.

Lattice-Based

For ZDMD's construction, the CRS remains the only static component of the EC, but is now composed of several matrices $A^{n\times m}$, $W^{ln\times m}$, $T_1^{lm\times lm}$, $T_2^{m\times lm}$, $W_0^{n\times m}$, and $B^{n\times lm}$ (superscripts denote dimensions). As a reminder, $n = 4$, $m \approx 56$, and $l$ (the identity length) is 50 bits in ZDMD's benchmarks and our calculations in the previous section. Encryption only requires $A$, $W$, and $W_0$, which are the smallest matrices in the CRS, so these are the only ones a client needs to store; the others do not leave the KC and thus are functionally auxiliary data (with the exception of $B$, which is used for keypair generation). $A$, $W$, and $W_0$ are constant-size and their elements are polynomials of degree $n_R = 256$ whose coefficients are the same bit-length as $q$ (a large prime that is ~80 bits in size at a billion users, as deduced in the previous section) [26]. Using this we can derive an estimated size of $n \cdot m \cdot n_R \cdot \log_2 q = 4 \cdot 56 \cdot 256 \cdot 80$ $= 4,587,520$ bits $= 573,440$ bytes for $A$ and $W_0$, and $573440l = 573440 \cdot 50$ $= 28,672,000$ bytes for $W$, bringing the total size of the static component to $52 \cdot 573440 = 29,818,880$ bytes or 29.8 MB. As for the dynamic component, there are two: the *master public key* (MPK), an $n \times k$ matrix of polynomials (of the same size as above) that has one global value, and the *helper decryption key* (HSK), an $lm \times k$ matrix of polynomials that is unique for each registered identity (i.e. each user requests *their* HSK from the KC before decrypting). The MPK, like public parameters in previous constructions, is updated with each registration and thus must be re-fetched $O(N)$ times, once each time a user wishes to encrypt to

another who registered after their last fetch. It is required only for encryption. The HSK, conversely, is required for decryption and must be updated routinely by parties who wish to receive messages from any other user. Theorem 3 of the ZDMD paper asserts that it is possible to bound the number of HSK updates at $O(\log N)$ [26]. Concretely, the MPK will be $n \cdot k \cdot n_R \cdot \log_2 q = 4 \cdot k \cdot 256 \cdot 80 = 81920k$ bits $= 10240k$ bytes in size, and the HSK will be $l \cdot m \cdot k \cdot n_R \cdot \log_2 q = 50 \cdot 56 \cdot k \cdot 256 \cdot 80 = 57344000k$ bits $= 7168000k$ bytes. Because $k$, the dimension of the secret key, is not given in the paper, we take the minimum theoretical size of $k = 1$. This gives us an MPK of 10.2 KB and an HSK of 7.1 MB. The total size of the EC is thus at least $29,818,880 + 10,240 + 7,168,000 = 36,997,120$ bytes, or ~37 MB — quite large, especially considering that this value grows linearly with $k$. For even a moderate secret key length, the EC surpasses hundreds of megabytes. This EC size becomes prohibitively large for some single-board computers or microcontrollers which may have less than a GB of storage and memory to work with, especially as multiple KCs come into play, and of course completely outgrows the total amount of PKI material required in standard encryption protocols.

Lastly, the KLSS construction calls their static EC component the public parameters, and their dynamic component the "system state" (ST). In a significant departure from other constructions, the static public parameters can take two sizes: a constant 64-byte seed pair, or a tuple of two matrices $A^{n \times km_A}$ and $B^{n \times m_B}$ and a vector $term$ where $|term| = |st|$. The latter form is generated deterministically from the former [18], which introduces a tradeoff between communication cost and computational overhead. Since our primary evaluation metric is cacheability, we take the size of the *derived* public parameters — we assume it is fetched as the compact seed pair and expanded once per fetch, with the result stored for later use. This

minimizes overhead during actual encryption. For the sizes of $A$ and $B$, we assume the benchmark parameters of $n = m_B = 7$, $k = 3$, $l_A = 2$, and $m_A = nl_A = 7 \cdot 2 = 14$. This means that $A$ is a $7 \times 3 \cdot 14 = 7 \times 42$ in dimension and $B$ is $7 \times 7$. Each element is a polynomial whose $\varphi = 256$ coefficients each take $\log_2 q = 48$ bits $= 6$ bytes to represent (given that $q \approx 2^{48}$) [18]. Thus, the concrete size of $A$ is $7 \cdot 42 \cdot 256 \cdot 6 = 451,584$ bytes or ~451.6 KB. The concrete size of $B$ is $7^2 \cdot 256 \cdot 6 = 75,264$ bytes or ~75.3 KB. As for the "terminator" vector $term$, its size is equivalent to the size of the ST, as both are constructed the same way during KC setup. We compute both their sizes at once: each is a $b$-length vector whose elements each consist of $n$ polynomials [18]. Given $b = 30$ (which we get directly from $N = 2^{30}$), this yields $|term| = |st| = 30 \cdot 7 \cdot 256 \cdot 6 = 322,560$ bytes or ~322.6 KB. In total, the static component comes to $|A| + |B| + |term| = 849,408$ bytes or ~849.4 KB, and the dynamic component is just $|term|$ computed above. Both $term$ and $st$ grow with the number of "buckets" $b$, and since $b = \log N$ we can see plainly that both the static and dynamic components of the KLSS construction are $O(\log N)$ in size. The dynamic ST needs only to be updated $O(\log N)$ times over the KC's lifetime [18]. An order-of-magnitude improvement on previous constructions, this construction can support much smaller cache sizes at rest than its counterpart in this section and all pairing-based constructions (at the cost of much larger ciphertexts discussed earlier). Even with larger ciphertexts, however, the sum total size of all its cryptographic objects is much less than in other constructions, indicating that the field is starting to converge on size optima. This is a promising signal for the future practicality of lattice-based constructions overall.

**Computational Efficiency**

| Construction | Encryption Complexity | Decryption Complexity |
|---|---|---|
| GKMR | $O(1)$ | $O(1)$ |
| FKP | $O(1)$ | $O(1)$ |
| ZDMD | $O(polylog(\log N))$ | $O(polylog(\log N))$ |
| KLSS | $O(\log N)$ | $O(\log N)$ |

Table 4: asymptotic complexities for encryption and decryption functions by construction.

| Construction | N | Encryption Time (ms) | Decryption Time (ms) |
|---|---|---|---|
| GKMR | Any | 2 | 3 |
| ZDMD | $2^{50}$ | 1.5 | 0.5 |
| KLSS | $2^{30}$ | 15.5 | 10.5 |

Table 5: concrete encryption and decryption times by construction for certain large $N$ values. FKP is omitted for lack of benchmark data and source code.

Pairing-Based

GKMR achieves constant-time encryption and decryption thanks to its constant-size ciphertext, which keeps concrete times down to 2 - 3 milliseconds — virtually unnoticeable [17]. FKP unfortunately do not provide concrete time measurements for their cryptographic functions, but we can infer their time complexities. Since FKP use GKMR's construction as a basis and add witness encryption for vector commitments (VCWE) on top, we only need to determine the asymptotic behavior of the VCWE functions to derive an exact bound. Traditionally, witness encryption schemes are mostly theoretical, but FKP note that specifically with respect to vector commitment schemes, they can be made black-box and thus very concretely efficient: encryption and decryption in VCWE take only a constant number of pairing and group operations [25]. Thus, constant-time GKMR cryptography combined with constant-time VCWE cryptography produces a new set of functions that are also constant-time. Due to their use of cuckoo hashing,

encryption in the FKP construction may need to be done multiple times — once for each possible identity hash, if the recipient has not been stashed — however this, too, is a constant-size parameter ($k$). Thus, FKP encryption and decryption are still O(1), in line with the authors' claims on achieving "the same efficiency properties" as GKMR [25]. Constant-time cryptography is incredibly important for supporting resource-constrained devices, as it allows them to interact with incredibly large RBE systems without their sizes causing explosions in runtime. This gives pairing-based constructions a leg up over lattice-based ones when deployed.

Lattice-Based

Lattice-based constructions no longer benefit from constant-size ciphertexts, so their encryption and decryption times now grow with $N$. ZDMD state that for their construction, "the runtime of encryption and decryption algorithm[s] only have polylog (instead of linearly) dependency on the length of the identity" [26]. The length of the identity bitstring is named $l$, and is bounded s.t. $l \leq \log_2 N$. Hence, both encryption and decryption run in $O(polylog(l)) = O(polylog(\log N))$. This excellent efficiency enables them to achieve the impressive times of 1.5 ms for encryption and 0.5 ms for decryption, beating even pairing-based constructions. KLSS' cryptographic functions also depend on a parameter $l$, but this time it reflects the *variable* length of a given identity string, taken at encryption time. $l$ has a maximum value of $\hat{l} \leq \log_k N$ where $k$ is the arity of an underlying Merkle tree (fixed at $k = 3$ in the text). On encryption and decryption, the number of operations is dominated by a loop over $l$ [18]. Thus, both operations run in $O(\log N)$ time. Concretely, they report runtimes that are sometimes multiple orders of magnitude higher than ZDMD, but still usable: 15.5 ms for encryption and 10.5 ms for decryption. Despite not being constant, the cryptographic runtimes of both lattice-based

constructions are still highly efficient, especially in ZDMD's case, whose nested polylog efficiency yields great insensitivity to the growth of $N$.

**Summary: Pairings vs. Lattices**

| Construction | Compact CTXT | Cacheable EC | Efficient Crypto |
|---|---|---|---|
| GKMR | ✔ | ✔ | ✔ |
| FKP | ✔ | ~ | ✔ |
| ZDMD | ✘ | ✔ | ✔ |
| KLSS | ✘ | ✔ | ✔ |

Table 6: satisfaction of key Internet encryption constraints by construction.

In line with observations in previous works [25], [26], [28], our analysis of leading pairing- and lattice-based RBE constructions reveals certain strengths and weaknesses inherent to each approach. Pairing-based approaches generally exhibit preferable asymptotic behaviors under Internet protocol constraints — they have much smaller, constant-size ciphertexts, which is beneficial during protocol design as it enables hard-coding of ciphertext size and even allows us to remain within the MTU in some cases, which opens up applications in unreliable transport protocols. Additionally, pairing-based constructions introduce the least overhead when establishing encrypted channels, as their constant-size ciphertexts straightforwardly translate to constant-time encryption and decryption, with concrete times staying within single-digit milliseconds.

Lattice-based constructions, however, exhibit stronger cacheability of encryption context, as they have same-order-of-magnitude or smaller EC sizes to pairing-based constructions while exhibiting much less frequent updates bounded by $\log N$. Additionally, the robust insensitivity of the KLSS construction to $N$ makes the maximum number of registrations effectively unlimited, which is a significant leap forward for the field. Unfortunately, this comes with the tradeoff of

significantly larger concrete ciphertext sizes, which nullify their benefits in compact wire-format scenarios where Internet protocols operate. It is also, however, important to consider a third quality of lattice-based constructions not previously mentioned, which is their post-quantum security. Because pairing-based constructions rely on a version of the discrete logarithm problem, they can be broken by Shor's algorithm [28] and are thus vulnerable to harvest now, decrypt later attacks [29]. Lattice-based constructions, on the other hand, inherit post-quantum security from the Learning with Errors assumption [18], [26]. This is hugely preferable when developing future-proof Internet protocols, which must maintain their security even as quantum compute becomes generally useful and available.

Overall, pairing-based constructions still represent the state-of-the-art in practicality when applied to real-world networks. Their relative compactness and efficiency is difficult for present lattice-based solutions to achieve. Still, lattice-based RBE constructions are likely the future of the field due to their incredible scalability and post-quantum security properties. We believe they are a promising area for future research and will eventually overcome their drawbacks in communication cost.

## ReDNS: A CASE STUDY

ReDNS, a registration-based encrypted DNS protocol, is the first application of RBE at the session layer. Here we document the process of utilizing RBE at such a low level of the Internet protocol stack. *This chapter contains unpublished material. The protocol described here was developed by Jun Li* [30]*, Kai Iverson, and myself. All written material is by me.*

### The Encrypted DNS Problem

A backbone of the Internet since the ARPANET days, the Domain Name System (DNS) has been mapping human-readable names to IP addresses for over 40 years [31]. Google's public DNS resolver, representing only ~10% of all DNS traffic [32], has grown to process over a trillion queries per day originating from all continents, including Antarctica [33]. But where other protocols in the trillions club have majority-adopted security extensions (e.g. TCP→TLS, and by proxy HTTPS), the lion's share of DNS traffic today is still sent unencrypted and unauthenticated [34].

The plaintext nature of DNS makes queries highly vulnerable to eavesdropping and manipulation. This grants threat actors from small-time phishing campaigns to nation-state surveillance networks the ability to monitor and hijack the name resolution process on virtually any Internet-connected device. In the wild, DNS manipulation enables cache poisoning attacks [35] and mass censorship strategies [36] that aim to deceive or restrict the freedoms of Internet users.

Several solutions attempting to safeguard DNS communications exist already, but none have achieved a strong percentage of adoption. DNSSEC, the first authentication and integrity mechanism for DNS records, is nearly 30 years old and only estimated to protect around 1% of DNS query traffic [37]. This paltry takeup is attributed to complex and sensitive key

management schemes (which commonly cause large-scale resolution blackouts, like .de experienced recently [38]) as well as increased message sizes that can exceed the MTU, forcing fallback to TCP [37]. Another suite of protocols attempts to move DNS onto traditional reliable transport: they are DNS-over-TLS, -HTTPS, and -QUIC (DoT/H/Q). These protocols attempt to steer DNS away from its connectionless roots, introducing overhead that greatly increases communication and memory costs, sometimes many times the size of the original DNS message [39]. This adds up for large resolvers who have to maintain thousands of connections at once, wasting many resources on establishing reliable channels for a protocol ultimately designed for unreliable transport. Additionally, these protocols suffer from similar PKI configuration complexity to DNSSEC — around 25% of DoT resolvers in the wild were found to use invalid certificates, creating security risks [35]. The result is that DoT and DoH account for only ~12% of DNS traffic to Cloudflare's public resolver [34], and DoQ is so uncommon on the Internet that its presence is almost unmeasurable [40]. DNSCrypt, a third challenger in DNS encryption, delivers a rotating set of certificates over plain DNS whose public keys are then used to encrypt subsequent DNS messages. Instead of relying on existing trust stores, DNSCrypt requires clients to explicitly trust a DNS provider's public signing key that they have acquired out-of-band. The provider uses this signing key to issue a set of certificates that the client uses to encrypt. These certificates are expected to be rotated every 24 hours, putting additional burden on resolver operators and requiring clients to re-handshake often [41]. The configuration complexity of such a system is prohibitive; it is the least-supported encrypted DNS protocol across major OSs, browsers, and DNS resolvers [35].

The common denominator between these failures is that PKI introduces far too much complexity to the DNS system. The constant exchange of public key material adds an incredible

amount of overhead and requires configuring intricate, sensitive trust chains that are intolerant to mistakes. A natural instinct, then, is to try and avoid traditional PKI entirely. With registration-based encryption, we can theoretically perform the authenticated symmetric key exchange required by modern hybrid cryptosystems, but without transmitting any public keys at all. This is the driving force behind ReDNS, a new registration-based encrypted DNS protocol that relies on a trusted key curator setup instead of traditional PKI.

**Protocol Overview**

The ReDNS protocol maps the structure of RBE onto the parties involved in DNS resolution. DNS resolvers that want to support ReDNS are expected to register their public keys to a set of trusted KCs. ReDNS clients (stub resolvers) do not need to register — they only need to fetch each trusted KC's public parameters. This allows the client to encrypt a generated secret to the resolver from which a symmetric key can be derived. A full technical specification is outside the scope of this paper, however we do provide a message-level description of the protocol as follows.

The first message, M1, indicates the client's desire to upgrade to a ReDNS-secured session. It does this by sending a plain DNS message that includes a request for the list of KCs to whom the resolver has registered. If the resolver does not support ReDNS, it will ignore this request. If it does, it will return the list of KCs in another plain DNS message we call M2 (alongside the answers to other requests in M1).

Then, if the client has already cached the public parameters for at least one of the KCs in the resolver's list, it will use those parameters to encrypt a generated pre-shared key (PSK) to the resolver's IP address. (If it has not cached the parameters yet, it must first reach out to the KC and request them.) The encrypted pre-shared key (EPSK) is then sent in a third message, M3,

which is the first message in ReDNS wire format. M3 also encapsulates the client's first symmetrically-encrypted DNS message. Upon receiving M3, the server decrypts the EPSK using RBE, derives a symmetric key from the resulting PSK, and uses that symmetric key to decrypt the payload. It resolves the names enclosed and encrypts the response using the same symmetric key, sending it back in M4.

Now, the ReDNS client and resolver have established a session. M5 and M6 are encrypted DNS requests and responses that repeat for the lifetime of the session. The ReDNS client does not need to re-bootstrap until either party has forgotten the session (which may occur after a certain time-to-live or in cases of higher-than-normal load on the resolver).

**Selecting an RBE Construction**

As established in the previous chapter, pairing-based RBE constructions are preferable in Internet protocol contexts due to their incredibly small constant-size ciphertexts. Additionally, GKMR's construction fits a ciphertext within the standard MTU leaving room to spare. These qualities enabled convenient design decisions. First, because we can fit a ciphertext in one packet, we need not worry about establishing reliable transport like other encrypted DNS protocols do. Second, because the ciphertext is constant-size, we can allocate it a fixed number of bytes in the wire format and know for certain there will be enough room for our first DNS message after it in the datagram. For these reasons, we decided to move forward with deploying the GKMR construction.

We also gained other advantages with this decision that conformed well to the existing domain name system. Particularly, the most common operations (encryption, decryption, and update retrieval), occur in constant-time, and rare operations (key curator setup, key-pair generation, and registration) take at most $O\left(\sqrt{N}\right)$ time. This means our encryption scheme can

feasibly support billions of resolvers without compromising on communication cost or efficiency, although the real number of resolvers in existence is much less — around 10 million [42]. In fact, at only 10 million registerable identities, we obtain multiple order-of-magnitude decreases in size across all non-constant cryptographic objects when compared to our estimates in the previous chapter — the total encryption context, for example, would shrink to less than 1.5 MB [17]. That is a negligible amount of storage to theoretically support encrypting to every DNS resolver on the Internet. This demonstrated favorable scaling properties early on and made us optimistic about applying RBE to this long-standing problem.

**Results**

Using the GKMR construction, we were able to achieve our goal of certificateless, authenticated, non-interactive encryption between DNS resolvers. We produced three components: the ReDNS client, the ReDNS proxy, and the key curator server. The client establishes ReDNS sessions with the proxy, which runs on top of a traditional DNS server. The proxy decrypts DNS messages sent from the client and forwards them to the underlying DNS layer, encrypting and returning the response. During these exchanges, the client and proxy contact the key curator as needed to fetch updated encryption context. Using this scheme, we were able to fulfill DNS requests in as fast as 19 ms when handshaking and 9 ms in steady state.

<u>Advantages</u>

Our landmark accomplishment was achieving key encapsulation in a hybrid cryptosystem over the Internet without the need for certificates. Particularly, we were able to reproduce the incredibly compact encryption context sizes described in the previous chapter and the GKMR paper, which proved the encryption context's viability as a replacement for certificate material. For a system supporting the entire IPv4 space ($N = 2^{32}$), the encryption context came out to

around 25.6 MB in total, with the dynamic component representing around 6.5 MB of that size. This was a remarkable result — using just 25 MB of storage, we could theoretically encrypt to over 4 billion identities, which is a density of *less than one bit per identity*. Compare that to the same number of end-entity certificates, which would take up anywhere from 8.5 to 17.2 terabytes (assuming an individual size of 2 - 4 KB [10]). This illustrates the primary advantage of RBE in Internet protocols: by frontloading just a small amount of material, we can prevent future communication overhead that is many orders of magnitude larger.

Additionally, we determined that despite the fact that the dynamic component of the EC needs to be re-fetched occasionally, the occasion itself is actually quite rare. The primary trigger for a re-fetch of the EC is when encryption fails to a recipient; there are two ways that this can happen. In the GKMR construction, the dynamic component is formed of $\sqrt{N}$ blocks of public key material. When a new identity is registered, its public key is accumulated in one of these blocks. If you want to encrypt to someone who registered after your last fetch, you will have to re-fetch the block they registered in. This is an *optional* update if you do not have to communicate with anyone you don't already have updated blocks for. Thus, the encryption context you have can be very long-lived if you only encrypt to a small subset of the registered identities (which is true in DNS — a stub resolver may only contact a few other resolvers in its lifetime). However, the encryption context does eventually get invalidated when a new user registers in a block that you use. This results in a *mandatory* update. Fortunately, mandatory updates only increase in frequency proportionate to your utilization of the EC. Because only massive, public recursive resolvers will contact large portions of the identity space, they are the only parties who will have to update their ECs actively. The average resolver can lazily fetch updates without significant performance penalties day-to-day. This reflects a convenient

alignment with the existing DNS hierarchy; ReDNS won't make smaller resolvers do the same work that larger ones do.

Finally, because of the established cacheability of the encryption context, we were able to achieve significantly faster handshakes for the vast majority of sessions. This is relatively straightforward: because the EC is so long-lived, most handshakes will never involve contacting a key curator. Even though fetching the EC at encryption time can introduce significant latency, it is deeply amortized by the many handshakes afterward that will occur before we have to fetch the EC again. The relative infrequency of mandatory EC updates for the average client means that we can sustain cache hit rates in the high 90 percents. This is crucial for the performance of RBE when compared with PKI and shows another hurdle the encryption scheme has cleared in practice.

Not all hurdles are equally tall, however. Despite these significant advantages, RBE (and GKMR's construction specifically) also posed limitations that produced challenges for us while developing ReDNS. Our solutions had knock-on effects and security implications that required remediation. These solutions are detailed in the sections that follow. We predict that, because the field is so young, future constructions of RBE are likely to shed these limitations while also bringing better functionality to the table.

## Challenges

### *Supporting Resource-Constrained Devices*

During our testing, we decided to run benchmarks on a setup of three Raspberry Pi 2s, which were devices available to us that approximated the resource-constrained environment of many IoT and edge devices. In our setup, the Pis were all connected to the same LAN and each had a different function: one was a key curator, one was a ReDNS client (stub resolver), and one

was a ReDNS server (recursive resolver). During our testing, we observed initial handshake times of around 485 ms — this included messages 1 through 4 described in a previous section as well as the retrieval of encryption context from the KC. At almost half a second, this is very noticeable latency, especially in DNS.

Fortunately, this latency only applied to the first DNS request included in M3 — future requests used symmetric encryption, which yielded response times of around 14 ms. We consider the former latency to be amortized by the latter, as many more DNS requests will take place in the protocol's steady state than in its handshake. Additionally, because these handshakes involved an out-of-band HTTP request to the KC, future handshakes will be reduced by hits to the EC cache. Still, we would like to see this time reduced — as it stands, software implementing ReDNS would have to hide that latency somehow, perhaps by sending a dummy DNS message during boot/initialization and thus before the first "actual" request.

*The Bounded Identity Problem*

Another significant struggle we encountered during development of ReDNS was deciding on what information to use to identify resolvers, and how. In ReDNS, the $N$ value is analogous to the maximum number of DNS resolvers that a KC can support. The set of identities supported by the GKMR construction is $\{1, \ldots, N\}$, meaning that identities are not arbitrary strings (such as email addresses) but rather integers (such as phone numbers) less than or equal to $N$. Because encrypting parties in RBE must encrypt to an identity itself, this means the encryptor (the ReDNS client) must know, or be able to derive, an integer identity for the recipient (the ReDNS-capable resolver).

On its face, the integer identity for a DNS resolver is obvious: its IP address, which is already necessary for the client to know before sending plain DNS messages. It's relatively

straightforward to implement this with IPv4; the identity set becomes 1, ..., $2^{32}$, and the encryption context is only 25.6 MB in size, as established earlier. IPv6, however, drastically increases the theoretical identity set to 1, ..., $2^{128}$, which increases the non-constant parameters to 7,194,230,188,746,725,130,094 bytes or 7 zettabytes, which is approximately 1.8 trillion DVDs' worth of data [43]. This unfortunately necessitates a mapping function from the IPv6 space onto a smaller RBE identity space. The function must be shared between resolvers and the KC so that each party can derive the identity of another, preserving the structure of RBE. Because such a function is necessarily non-injective, we now have to worry about identity collisions.

Earlier we noted that the FKP construction solved this issue using cuckoo hashing and vector commitment witness encryption [25]. While incredibly useful for our case, increasing the ciphertext size to the next order of magnitude made it difficult for us to use it in our protocol. A ciphertext of a couple thousand bytes necessitates a more reliable transport, either by ad-hoc packet numbering or switching wholesale to TCP, both of which increase the amount of state stored on a given resolver and overall complicate the handshake process. Instead, we attempted to adapt the GKMR construction for our use case.

Our solution involved using a hash function as part of a negotiated cipher suite (agreed upon in M2) to map an encryption target's IPv6 address to a number in the IPv4 space. In the event of collisions at registration time, the KC rejects the registration request. This strategy, if rudimentary, did work to enable IPv6 support. We regret introducing collisions, however, and are confident that future constructions will achieve similar usability to GKMR's without this issue.

*Securing the Key Curator*

Of all the architectural hurdles, though, the secure deployment of the key curator is the most challenging. The KC is intended to be a completely transparent and auditable entity such

that, if the complete contents of its storage and memory were public, no secrets would be compromised [25]. This, of course, is a rather idealistic view — in practice, there are many threats that can compromise secure communication if the key curator is breached or any of its messages are modified in transit.

Our current implementation uses HTTP to communicate with the KC. If we wanted to protect against man-in-the-middle attacks, we would enable HTTPS. This involves a traditional PKI certificate authority issuing a certificate for our use, which we use to facilitate key encapsulation in TLS handshakes. In theory, this isn't a horrible solution — since the KC is contacted infrequently, the overhead of an occasional TLS connection is not too high a cost, plus DNS infrastructure operators are likely to have their own PKI systems in place already. The problem is that this method requires that we store the certificate's private key on the KC, which violates the auditability assumption posed by RBE.

There are strategies in the field of zero-trust that can help us verify communications without storing any secrets; instead, we verify the KC's *computations* were unmodified during setup, registration, or update of the encryption context. One of these strategies is called zk-SNARKs, which are "Zero Knowledge Succinct Non-interactive Arguments of Knowledge." In essence, a SNARK is a generated proof that a specified algorithm executed honestly. To produce a SNARK, a program must be compiled into circuit form, from which an arithmetic representation of the satisfiability problem for that circuit is derived. Then, a proving system allows a prover (the key curator) to generate a compact proof of correct execution for the verifier (a ReDNS client) using the arithmetic representation of the program [44]. In theory, this would allow us to verify that modification of the encryption context occurred without tampering; if an incorrect public key was injected on the route to the KC, or incorrect encryption context were

received by a client, either party could verify the computational proof that the other provided to assure themselves of no foul play. We leave research of this avenue to future work.

*Multi-KC Systems*

Somewhat related to the above problem is the orchestration of secure multi-KC systems. In an ideal world, ReDNS is deployed to the entire Internet, and all major resolvers adopt it. Naturally, we might expect KC operators to fall roughly along the same lines that certificate authorities do — a handful of very popular ones serving the busiest resolvers, and hundreds of other smaller ones that serve the rest. A single KC is able to securely bind an identity to its public key, but when you introduce multiple trusted KCs in a broader ecosystem, it's possible for the same identity to correspond to multiple public keys.

Let's say Bob trusts two KCs on the Internet. The ReDNS resolver he uses is registered at KC-A. On KC-B, the slot that Bob's resolver occupies on KC-A is empty. Eventually, an attacker registers their own public key to the open slot on KC-B. They intercept the traffic between Bob and his resolver and rewrite the handshake to indicate that his resolver is now using KC-B. Bob is tricked into using the encryption context for KC-B and the identity slot that the attacker occupies!

This is a difficult problem that is not entirely solved in our implementation. As a partial remediation, we compute asymmetric digests of the plaintext handshake messages (M1 and M2) and verify them on each side, similar to how TLS protects the integrity of its handshake, however this verification doesn't complete until after M4, when the first DNS request has already been serviced.

The problem becomes even more difficult when you consider the hash collisions introduced earlier. Let's say two honest parties register at two different trusted KCs, and their

identities hash to the same slot. Those two parties are now theoretically able to decrypt each others' traffic if a client is redirected to the wrong KC! The security of multi-KC systems requires extensive threat modeling and could produce an entire thesis of its own. For this reason, we also leave it to future work.

# CONCLUSION

This thesis has demonstrated that RBE has matured sufficiently enough to provide security functions for key Internet services in a scalable, efficient, and most importantly *certificateless* manner. By addressing fundamental inefficiencies and systemic fragilities inherent to traditional public-key infrastructure — particularly its reliance on unwieldy certificate chains and complex, error-prone management routines — RBE presents a more versatile, scalable, and resilient future for encrypted Internet protocols.

Our comparative analysis of leading RBE constructions (GKMR, FKP, ZDMD, and KLSS) identified important trade-offs to consider when adapting RBE to hybrid cryptosystems typically deployed on the Internet. We found that pairing-based schemes like GKMR and FKP offer compact, constant-size ciphertexts that are highly favorable in lightweight protocols that use unreliable transport or are otherwise MTU-conscious, but they struggle to scale beyond billions of users and have reduced functionality. Conversely, lattice-based constructions suffer from prohibitively large ciphertexts but provide improved encryption context cacheability and insensitivity to $N$, as well as post-quantum security, establishing them as an area to watch for future Internet applications.

Furthermore, to prove the practicality of RBE on the actual Internet, we created ReDNS — an encrypted DNS protocol that is the first to apply RBE at the session layer. By successfully deploying RBE to the existing domain name system, we proved that important theoretical advantages of RBE, such as non-interactive key encapsulation and effective caching of encryption context, are actually achievable under real-world network conditions. Ultimately, we confirm that despite its nascency relative to other public-key encryption schemes, RBE is ready to serve critical security roles in modern, high-impact Internet protocols.

Still, RBE faces mounting pressures to continue improving; over the course of our research, we encountered certain challenges when adapting the encryption scheme to common cryptosystem architectures on the Internet. Namely, we struggled to conform to limitations imposed on the identity set, secure communications with the key curator, and responsibly orchestrate multi-KC trust ecosystems. While these challenges required complex workarounds involving hashing and collision management, and some problems remain unsolved, the overall performance results validate the present feasibility of RBE and promote further research into practical, networked RBE systems.

In the future, we hope to see RBE reach feature-parity with PKI, as it is presently missing many of the features that make certificates desirable — alternate identities, usage constraints, revocation mechanisms, and even validity periods. Promising work is being done to augment RBE with attribute-based encryption, which enables many of these features [45], [46], [47]. Additionally, we would like to see the field explore zero-trust methodologies for deploying RBE securely in the real world, perhaps through zk-SNARKs or other mechanisms that can prevent MITM attacks. Finally, we believe there's more work to be done with regards to RBE-enabled Internet protocols; a reliable encrypted transport protocol taking advantage of the FKP construction would lift many of the limitations we faced in this work and further prove RBE's feasibility at Internet scale.

Overall, this research establishes that the traditional paradigm of certificate-based encryption is no longer the only path forward for Internet privacy. By replacing the ailing architecture of the global public-key infrastructure with a scalable, certificateless solution utilizing registration-based encryption, we can prevent large-scale Internet service outages, interception, and manipulation while also producing a system that is fundamentally more

efficient. While the road to widespread adoption will require fine-tuning cryptographic object sizes, navigating complex multi-KC trust models, and implementing zero-trust strategies, the conceptual and practical results presented here strongly suggest that RBE is no longer just cryptographic theory. It has become a robust, viable architecture ready to secure the next Internet age.

REFERENCES CITED


[1] N. Shah, “The Challenges of Inspecting Encrypted Network Traffic,” Fortinet Blog. Accessed: May 14, 2026. [Online]. Available: https://www.fortinet.com/blog/industry-trends/keeping-up-with-performance-demands-of-encrypted-web-traffic

[2] Google, Inc., “HTTPS encryption on the web.” Google Transparency Report. Accessed: May 14, 2026. [Online]. Available: https://transparencyreport.google.com/https/overview

[3] University of California, Berkeley, “Data Encryption in Transit Guideline,” Information Security Office. Accessed: May 15, 2026. [Online]. Available: https://security.berkeley.edu/data-encryption-transit-guideline

[4] Asterfusion, “7 Layers Of OSI Model Encryption: From MACsec To HTTPS.” Accessed: May 15, 2026. [Online]. Available: https://cloudswit.ch/blogs/osi-model-encryption-macsec-ipsec-tls-https/

[5] Oracle, “Transport Layer Security (TLS) Protocol Overview,” in *Security Developer’s Guide*, 26th ed., 2026. Accessed: May 15, 2026. [Online]. Available: https://docs.oracle.com/en/java/javase/26/security/transport-layer-security-tls-protocol-overview.html#GUID-69ECD56C-3B20-47F4-AEF0-A06EFA13A61D

[6] R. Cramer and V. Shoup, “Design and Analysis of Practical Public-Key Encryption Schemes Secure against Adaptive Chosen Ciphertext Attack,” *SIAM J. Comput.*, vol. 33, no. 1, pp. 167–226, Jan. 2003, doi: 10.1137/S0097539702403773.

[7] C. Kaufman, P. E. Hoffman, Y. Nir, P. Eronen, and T. Kivinen, “Internet Key Exchange Protocol Version 2 (IKEv2),” Internet Engineering Task Force, Request for Comments RFC 7296, Oct. 2014. doi: 10.17487/RFC7296.

[8] C. M. Lonvick and T. Ylonen, “The Secure Shell (SSH) Transport Layer Protocol,” Internet Engineering Task Force, Request for Comments RFC 4253, Jan. 2006. doi: 10.17487/RFC4253.

[9] J. Iyengar and M. Thomson, “QUIC: A UDP-Based Multiplexed and Secure Transport,” Internet Engineering Task Force, Request for Comments RFC 9000, May 2021. doi: 10.17487/RFC9000.

[10] Sven, “Answer to ‘What is the common size of SSL certificate? (give or take),’” Server Fault. Accessed: May 27, 2026. [Online]. Available: https://serverfault.com/a/793564

[11] C. Crane, “PKI Mistakes That Were So Bad They Made Headlines (12 Examples),” Hashed Out by The SSL Store. Accessed: May 18, 2026. [Online]. Available: https://www.thesslstore.com/blog/pki-mistakes-that-were-so-bad-they-made-headlines/

[12] N. Van Der Meulen, “DigiNotar: Dissecting the First Dutch Digital Disaster,” *J. Strateg. Secur.*, vol. 6, no. 2, pp. 46–58, Jun. 2013, doi: 10.5038/1944-0472.6.2.4.

[13] CyberArk, “2025 State of Machine Identity Security Report,” 2025. Accessed: Jun. 03, 2026. [Online]. Available: https://www.cyberark.com/resources/i/1533259-state-of-machine-identity-security-report-2025/

[14] B. Uhri, “Why Attackers Love Mismanaged PKIs,” Keyfactor. Accessed: May 18, 2026. [Online]. Available: https://www.keyfactor.com/blog/why-attackers-love-mismanaged-pkis/

[15] Access Now, “The Weakest Link in the Chain: Vulnerabilities in the SSL certificate authority system and what should be done about them,” Nov. 2011. [Online]. Available:

https://www.accessnow.org/wp-content/uploads/archive/docs/Weakest_Link_in_the_Chain.pdf
[16] S. Garg, M. Hajiabadi, M. Mahmoody, and A. Rahimi, “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*, Berlin, Heidelberg: Springer-Verlag, 2018, pp. 689–718. doi: 10.1007/978-3-030-03807-6_25.
[17] N. Glaeser, D. Kolonelos, G. Malavolta, and A. Rahimi, “Efficient Registration-Based Encryption,” in *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*, in CCS ’23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 1065–1079. doi: 10.1145/3576915.3616596.
[18] M. Klooß, R. W. F. Lai, J. N. Siemer, and M. Swarnakar, “Scalable Registration-Based Encryption from Lattices,” presented at the 2026 IEEE Symposium on Security and Privacy (SP), IEEE Computer Society, May 2026, pp. 3128–3146. doi: 10.1109/SP63933.2026.00199.
[19] Apple, Inc., “List of available root certificates in iOS 26, iPadOS 26, macOS 26, tvOS 26, visionOS 26 and watchOS 26,” Apple Support. Accessed: May 27, 2026. [Online]. Available: https://support.apple.com/en-us/126047
[20] C. Hornig, “A Standard for the Transmission of IP Datagrams over Ethernet Networks,” Internet Engineering Task Force, Request for Comments RFC 894, Apr. 1984. doi: 10.17487/RFC0894.
[21] Cloudflare, Inc., “What is MTU (maximum transmission unit)?,” Cloudflare Learning Center. Accessed: May 27, 2026. [Online]. Available: https://www.cloudflare.com/learning/network-layer/what-is-mtu/
[22] A. Gerodimos, L. Maglaras, M. A. Ferrag, N. Ayres, and I. Kantzavelou, “IoT: Communication protocols and security threats,” *Internet Things Cyber-Phys. Syst.*, vol. 3, pp. 1–13, 2023, doi: 10.1016/j.iotcps.2022.12.003.
[23] Y. Sakemi, T. Kobayashi, T. Saito, and R. S. Wahby, “Pairing-Friendly Curves,” Internet Engineering Task Force, Internet Draft draft-irtf-cfrg-pairing-friendly-curves-10, Jul. 2021. Accessed: Jun. 04, 2026. [Online]. Available: https://datatracker.ietf.org/doc/draft-irtf-cfrg-pairing-friendly-curves-10
[24] S. Khot, “Hardness of approximating the shortest vector problem in lattices,” *J ACM*, vol. 52, no. 5, pp. 789–808, Sep. 2005, doi: 10.1145/1089023.1089027.
[25] D. Fiore, D. Kolonelos, and P. de Perthuis, “Cuckoo Commitments: Registration-Based Encryption and Key-Value Map Commitments for Large Spaces,” 2023, 2023/1389. Accessed: May 21, 2026. [Online]. Available: https://eprint.iacr.org/2023/1389
[26] T. Zhang, X. Ding, G. Malavolta, and N. Döttling, “Fast and Compact Lattice-Based Registration-Based Encryption,” 2026, 2026/628. Accessed: May 20, 2026. [Online]. Available: https://eprint.iacr.org/2026/628
[27] J. Zarrin, H. Wen Phang, L. Babu Saheer, and B. Zarrin, “Blockchain for decentralization of internet: prospects, trends, and challenges,” *Clust. Comput.*, vol. 24, no. 4, pp. 2841–2866, Dec. 2021, doi: 10.1007/s10586-021-03301-8.
[28] M. Roetteler, M. Naehrig, K. M. Svore, and K. Lauter, “Quantum Resource Estimates for Computing Elliptic Curve Discrete Logarithms,” in *Advances in Cryptology – ASIACRYPT 2017*, T. Takagi and T. Peyrin, Eds., Cham: Springer International Publishing, 2017, pp. 241–270. doi: 10.1007/978-3-319-70697-9_9.

[29] D. Riley, “Half of organizations worry about quantum ‘harvest now, decrypt later’ attacks,” SiliconANGLE. Accessed: Jun. 07, 2026. [Online]. Available: https://siliconangle.com/2022/09/20/half-organizations-concerned-quantum-harvest-now-decrypt-later-attacks/
[30] J. Li, “ReDNS—Scalable DNS Security Through Registration-Based Encryption,” Sep. 2025.
[31] P. Mockapetris and K. J. Dunlap, “Development of the domain name system,” in *Symposium Proceedings on Communications Architectures and Protocols*, in SIGCOMM ’88. New York, NY, USA: Association for Computing Machinery, 1988, pp. 123–133. doi: 10.1145/52324.52338.
[32] APNIC, “DNS Recursive Resolver Use Metrics.” APNIC Labs Measurements and Data, https://labs.apnic.net/measurements/. Accessed: Apr. 29, 2026. [Online]. Available: https://stats.labs.apnic.net/rvrs/XA?hc=XA&hl=1&hs=0&ht=10&w=1&t=10&s=0&x11=1
[33] A. Dupuy, “Google Public DNS turns 8.8.8.8 years old,” Google Online Security Blog. Accessed: Apr. 29, 2026. [Online]. Available: https://security.googleblog.com/2018/08/google-public-dns-turns-8888-years-old.html
[34] Cloudflare, Inc., “DNS transport protocol worldwide,” Cloudflare Radar. Accessed: Apr. 29, 2026. [Online]. Available: https://radar.cloudflare.com/dns#dns-transport-protocol
[35] C. Lu *et al.*, “An End-to-End, Large-Scale Measurement of DNS-over-Encryption: How Far Have We Come?,” in *Proceedings of the Internet Measurement Conference*, in IMC ’19. New York, NY, USA: Association for Computing Machinery, Oct. 2019, pp. 22–35. doi: 10.1145/3355369.3355580.
[36] P. Pearce *et al.*, “Global measurement of DNS manipulation,” in *Proceedings of the 26th USENIX Conference on Security Symposium*, in SEC’17. USA: USENIX Association, 2017, pp. 307–323. doi: 10.5555/3241189.3241215.
[37] G. Huston, “Measuring the use of DNSSEC,” APNIC Blog. Accessed: May 03, 2026. [Online]. Available: https://blog.apnic.net/2023/09/18/measuring-the-use-of-dnssec/
[38] S. Neuteboom, C. Elmerot, and M. Worsley, “When DNSSEC goes wrong: how we responded to the .de TLD outage,” The Cloudflare Blog. Accessed: May 07, 2026. [Online]. Available: https://blog.cloudflare.com/de-tld-outage-dnssec/
[39] T. Böttger *et al.*, “An Empirical Study of the Cost of DNS-over-HTTPS,” in *Proceedings of the Internet Measurement Conference*, in IMC ’19. New York, NY, USA: Association for Computing Machinery, 2019, pp. 15–21. doi: 10.1145/3355369.3355575.
[40] K. Hynek, “The prevalence of DNS over HTTPS,” APNIC Blog. Accessed: May 26, 2026. [Online]. Available: https://blog.apnic.net/2021/09/13/the-prevalence-of-dns-over-https/
[41] F. Denis, “The DNSCrypt Protocol,” Internet Engineering Task Force, Internet Draft draft-denis-dprive-dnscrypt-08, Mar. 2026. Accessed: May 28, 2026. [Online]. Available: https://datatracker.ietf.org/doc/draft-denis-dprive-dnscrypt-08
[42] J. Abley, “Ten Million DNS Resolvers on the Internet,” ICANN Blogs. Accessed: May 28, 2026. [Online]. Available: https://www.icann.org/en/blogs/details/ten-million-dns-resolvers-on-the-internet-22-3-2012-en
[43] C. Arthur, “What’s a zettabyte? By 2015, the internet will know, says Cisco,” *The Guardian*, Jun. 29, 2011. Accessed: Jun. 07, 2026. [Online]. Available: https://www.theguardian.com/technology/blog/2011/jun/29/zettabyte-data-internet-cisco

[44] J. Liang, D. Hu, P. Wu, Y. Yang, Q. Shen, and Z. Wu, “SoK: Understanding zk-SNARKs: The Gap Between Research and Practice,” in *Proceedings of the 34th USENIX Conference on Security Symposium*, in SEC ’25. USA: USENIX Association, 2025. doi: 10.5555/3766078.3766186.

[45] S. Hohenberger, G. Lu, B. Waters, and D. J. Wu, “Registered Attribute-Based Encryption,” in *Advances in Cryptology*, C. Hazay and M. Stam, Eds., Cham: Springer Nature Switzerland, 2023, pp. 511–542. doi: 10.1007/978-3-031-30620-4_17.

[46] S. Murshid, R. Sarkar, and M. Mandal, “Registered Attribute-Based Encryption with Publicly Verifiable Certified Deletion, Everlasting Security, and More,” 2025, 2025/2314. Accessed: Jun. 08, 2026. [Online]. Available: https://eprint.iacr.org/2025/2314

[47] R. Yang *et al.*, “PR-RABE: Privacy-preserving and Revocable Registered Attribute-Based Encryption with lightweight encryption,” *Comput. Stand. Interfaces*, vol. 98, p. 104158, 2026, doi: https://doi.org/10.1016/j.csi.2026.104158.