Correlation-Resistant Storage via Keyword-Searchable Encryption

Lucas Ballard
Johns Hopkins University
lucas@cs.jhu.edu

Matthew Green
Johns Hopkins University
mgreen@cs.jhu.edu

Breno de Medeiros
Florida State University
breno@cs.fsu.edu

Fabian Monrose
Johns Hopkins University
fabian@cs.jhu.edu

Abstract

We consider the problem of using untrusted components to build *correlation-resistant* survivable storage systems that protect file replica locations, while allowing nodes to continuously re-distribute files throughout the network. The principal contribution is a chosen-ciphertext secure, searchable public key encryption scheme which allows for dynamic re-encryption of ciphertexts, and provides for *node-targeted* searches based on keywords or other identifiers. The scheme is provably secure under the SXDH assumption which holds in certain subgroups of elliptic curves, and a closely related assumption that we introduce.

Keywords: targeted search, key privacy, bilinear maps, survivable storage

1 Introduction

The past decade has seen growing interest in techniques for protecting critical data even in the face of catastrophic storage failure. Recently, a number of loosely-related approaches have surfaced which guarantee data availability by massively replicating records across decentralized, potentially untrusted, "survivable" storage networks. To ensure the continued availability of content after storage nodes fail or leave the network, survivable storage networks continuously re-distribute replicas from machine to machine. Ideally, content redistribution is provided as a service of the network, and should not require the active participation of content publishers.

Recently, Srivatsa *et. al.* [29] showed that a number of survivable storage systems (e.g, [20, 1]) are vulnerable to targeted denial of service attacks, as these systems make no attempt to hide the location of content replicas within the network. An adversary can locate selected file replicas via the network's search mechanism, or by manually examining stored collections for identical instances of a replica. Once located, the adversary can limit access to the selected files (and defeat survivability) by disabling the small subset of storage nodes which host the target content.

In this work, we propose techniques for *correlation-resistant storage*, which protect content replicas from targeted attacks while allowing for continuous re-distribution by the storage network. The approach we describe allows untrusted nodes to dynamically re-encrypt (i.e., randomize) file replicas such that an adversary cannot link the new replicas to others within the system. Simultaneously, we provide a flexible search mechanism which allows authorized receivers to locate any matching replica by querying storage nodes on information such as a keyword or other identifier. We note that maintaining correlation-resistance while achieving this remote search facility is challenging when storage nodes are untrusted, as one must prevent malicious nodes from re-using search queries to locate matching replicas at other locations in the network. In that regard, the primary contribution of this paper is a new form of searchable public key encryption scheme which allows for *node-targeted* keyword search, i.e., queries sent by users to a specific node cannot be re-played by that node to locate files stored elsewhere. Our keyword search scheme is related to the schemes of [12, 34], but enables randomization of indexes and is provably secure in the standard model.

2 Related work

The importance of correlation-resistance in the setting of survivable networks was recently highlighted in [29], where it was shown how targeted denial of service attacks can be deployed against protocols based on Distributed Hash

Tables (DHTs). In these attacks, the adversary is able to thwart survivability by compromising nodes and selectively removing files, which is less noticeable than, for example, removing every file on a given node. To avert such attacks, the solution proposed in [29] requires publishers to generate unique replicas of a file using a randomized encryption scheme, and to identify each with a unique record key for later retrieval. While this approach prevents an adversary from correlating distinct replica instances, it does not allow the network to generate new replicas on demand—a critical component of a survivable storage network. In fact, unless publishers periodically inject new replica instances into the network, the availability of content may decrease over time as nodes fail or leave.

Another area of research where correlation-resistance is particularly compelling (and that has received significant attention over the last few years) is censorship-resistant publishing [3, 32, 31, 18]. There, the goal is to provide schemes that allow for the anonymous replication of data across the network. In that setting, a number of censorship-resistant networks have been deployed, some of such which, like Freenet [15], Publius [32] and GnuNet [9], encrypt stored data in order to protect content and provide plausible deniability to storage node operators. While encryption achieves this goal to some degree, these distributed networks are still vulnerable to correlation attacks when files are dynamically transmitted from one node to another. If the adversary can identify such correlations, she may potentially distinguish nodes hosting particularly controversial content, or reveal information about queries made by network users.

The targeted keyword search scheme we define extends previous work in the field of keyword search on encrypted data (see for example, [28, 12, 17, 34, 8]). Such schemes provide mechanisms that allow users to remotely search for keywords contained in encrypted documents residing on a potentially untrusted server. However, existing schemes have two shortcomings which prevent their use in our application: first, dynamic redistribution (i.e., universal re-encryption) of searchable ciphertexts is not permitted by some constructions (e.g., [12, 34]). Secondly, these systems do not allow users to *target* search queries to a particular server, which is necessary to prevent replays in our setting. Finally, our keyword search scheme is secure in the standard model, and does not require random oracles. We believe that the ideas we propose here may have applications to other searchable encryption settings as well.

Lastly, the approach that we explore to dynamically re-encrypt files is similar to the concept of Universal Re-encryption, formalized by Golle *et. al.* [19]. Informally, Universal Re-encryption allows an untrusted third party to re-encrypt (randomize) ciphertexts without knowledge of the corresponding public key. In the schemes of Golle *et. al.*, identification of ciphertexts required that the recipient decrypt each candidate using her secret key. Our work can be viewed as extending the original setting of [19] to allow for targeted remote searches of ciphertext collections. Moreover, while the schemes of of [19] are only semantically secure, our schemes offer improved security by addressing chosen ciphertext security under the IND-RCCA definition of Canetti *et al.* [14].

3 Correlation-resistant storage and node-targeted keyword search

Survivable storage networks consist of a collection of unreliable, untrusted nodes, which collaborate to ensure the availability of content. To protect content in the event of node failure, each record is typically replicated across multiple nodes. Content is inserted into the network by a *publisher*, and later recovered by a *retriever*. In practice, these entities may be one and the same, or they may be distinct individuals. A publisher inserts a record into in the storage network by transmitting it to some subset of storage nodes. These nodes may continue to offer the content for later retrieval, may fail, or may re-distribute their contents to other nodes. Therefore, even when a retriever knows the identity of the nodes initially hosting the content, she has no guarantee that the desired content will remain at those nodes. As such, to locate content, retrievers require a search mechanism.

The principal goal of a correlation-resistant network is to prevent unauthorized parties from discovering the location of a given piece of content, even when given knowledge of the plaintext or some set of ciphertext replicas. This requirement must hold against adversaries who interact with nodes remotely (via the correct protocol), but should also hold against an adversary who can compromise a storage node and directly view its contents. More importantly, since many survivable networks have a low barrier to participation, a correlation-resistant network must resist even malicious storage nodes which use "insider knowledge" (e.g., legitimate search queries) to locate content on other nodes.

To summarize, we list the specific security goals of our application and the strategies we employ to achieve them.

- CONFIDENTIALITY: The contents of a stored file should not be intelligible to the nodes storing them. This

implies that the files need to be encrypted under a strong encryption scheme. In particular, we require that the encryption scheme provide indistinguishability against certain chosen-ciphertext attacks—specifically, IND-RCCA secure encryption as defined later in this section.

- KEY-OBLIVIOUS REPUBLICATION WITH UNLINKABILITY: Continuous redistribution of records for increased availability requires that we allow any unstrusted storage node to dynamically generate new, unlinkable instances of a ciphertext without knowledge of the ciphertext's public key. This requires that our encryption scheme be universally re-encryptable (a notion introduced in [19]), and key-private. Note that such a scheme certainly provides plausible deniability: the confidentiality of the content and and publishers' identities nodes ignore not only the contents they carry, but also their publishers' identities.

- NODE-TARGETED KEYWORD SEARCH CAPABILITY: The encryption scheme should support remote searches which are initiated by users and executed by the storage nodes. To achieve this, we implement a form of *keyword searchable encryption* [12, 34] which allows users to search a particular storage node for files that are encrypted under their public key and contain desired keywords. We require that search queries for a triple $(\texttt{user}, \texttt{keyword}, \texttt{node})$ do not reveal any information about an index unless the index is stored at node `node`, and the index also embeds the pair $(\texttt{user}, \texttt{keyword})$.

  The next sections formalize these goals.

3.1 Re-encryption and CCA security

Ciphertext indistinguishability under chosen ciphertext attacks (IND-CCA2) has become the de facto gold standard to assess the security of encryption schemes. However, it has been noted in a series of works [14, 2, 27, 21] that IND-CCA2 may be overly restrictive for some applications. This is particularly important for re-encryption (randomization) schemes, which are partially malleable by design, and thus *cannot* satisfy the requirements of the IND-CCA2 security definition. As a consequence, earlier re-encryption constructions, such as those of Golle *et. al.* [19] provide only chosen-plaintext security (i.e., semantic security).

Recently in [14], Canetti, Krawczyk and Nielsen advanced a relaxed security notion called ciphertext indistinguishability under *re-playable* chosen ciphertext attack (IND-RCCA). The schemes that we present in this work are designed to satisfy this definition, which [14] shows is sufficient to prevent even subtle attacks on message security. Intuitively, the IND-RCCA definition relaxes the non-malleability requirements of IND-CCA2 to allow for operations which do not alter the underlying *plaintext* of a given ciphertext. This simple relaxation is sufficient to permit general re-encryption schemes.

IND-RCCA can be defined as follows: consider a public (or hybrid) encryption scheme $S = (gen, enc, dec)$, specified by its key generation, encryption, and decryption algorithms, respectively. A prover $\mathcal{S}$ generates a key pair (sk, pk), releases pk to the adversary $\mathcal{A}$, and uses sk to implement a decryption oracle $\mathcal{O}_S^{dec(SK,\cdot)}$. The game proceeds with the adversary submitting a number of ciphertext decryption queries, and then choosing a pair of messages (m_0, m_1). The security prover selects a random bit b, encrypts m_b under the public key, and sends the resulting ciphertext to the adversary. The adversary may continue to query the decryption oracle $\mathcal{O}_S^{dec(SK,\cdot)}$, but now the oracle returns decrypted messages only when they differ from m_0 or m_1, otherwise returning a fixed value which is not a valid message. The game ends with the adversary pronouncing his guess bit b' to indicate which message was encrypted by the prover. It succeeds if $b' = b$.

Definition 3.1 (IND-RCCA [14]) *The encryption scheme $\mathcal{S}$ is IND-RCCA secure if there does not exist a p.p.t. algorithm $\mathcal{A}$ that wins the above game with success probability at least $1/2 + \varepsilon$, where ε is a non-negligible value.*

3.2 Node-targeted keyword searchable (indexing) schemes

To prevent the replay of queries by malicious nodes, our *node-targeted* keyword search scheme incorporates an additional feature, which allows query authors to embed the public key of a particular search party into the query itself. This embedding process can only be performed by the original author of a query, which prevents search parties from (successfully) evaluating replayed queries. The security of such a scheme depends on the hardness of "re-targeting"

fully-formed queries so that they embed a different party's public key than the one embedded by the legitimate retriever. The protection offered by this feature does not prevent *colluding* parties from sharing queries. However, in the context of a correlation-resistant storage system, it is sufficient to prevent malicious storage nodes from re-playing queries to non-colluding nodes. We define a node-targeted keyword search scheme as follows:

Definition 3.2 (Node-targeted keyword search) *A node-targeted keyword search scheme is a tuple of (possibly probabilistic) polynomial time algorithms* (Setup, KeyGen_1, KeyGen_2, CreateIndex, GenerateQuery, MatchQuery)*, where:*

- Setup takes a security parameter τ, and outputs system-wide parameters.
- KeyGen_1 outputs a key pair $\langle pk_{\mathsf{user}}, sk_{\mathsf{user}} \rangle$ used to implement access control.
- KeyGen_2 outputs a key pair $\langle pk_{\mathsf{node}}, sk_{\mathsf{node}} \rangle$ associated with a storage node. This key pair is used for *targeting* queries.
- CreateIndex takes a public key pk_{user} and keyword w, outputing the index value I.
- GenerateQuery accepts a secret key sk_{user}, a keyword w, and the public key of a storage node pk_{node}, and outputs the node-targeted query T.
- MatchQuery accepts an index I, a node-targeted query T, and a storage node's private key sk_{node}, and outputs TRUE iff the query matches the index, and is evaluated by the correct node.

In the context of correlation-resistant storage, where search indexes are distributed along with encrypted records, a search scheme must also provide the means for storage nodes to universally re-encrypt (randomize) indexes, such that the resulting instances are unlinkable with the originals. Therefore we add the following algorithm:

Definition 3.3 (Randomizable node-targeted indexing scheme) *A node-targeted keyword search scheme is said to be randomizable if it also supports the algorithm:*

- RandomizeIndex *universally re-encrypts an index I, outputting a new index I' which contains the same key and keyword as the original index.*

Informally, a record index embeds a keyword and a user's public key. Regardless of how many times an index is randomized, correctness requires that a query match provided that it was generated from the same keyword and the user's secret key, along with the public key pk_{node} of the storage node that executes MatchQuery. We define the correctness of the scheme:

Correctness: If an index I is created by the call $\mathsf{CreateIndex}(pk_{\mathsf{user}}, w)$, and later randomized n times to value I' by calls to $\mathsf{RandomizeIndex}(\cdot)$, and T is the query obtained by computing $\mathsf{GenerateQuery}(sk_{\mathsf{user}}, w, pk_{\mathsf{node}})$, then $\mathsf{MatchQuery}(I, T, sk_{\mathsf{node}}) = \texttt{true}$.

Soundness follows from the security definitions of our scheme (see below).

Indistinguishability of Index Values. Clearly a correlation-resistant storage system requires that ciphertexts be indistinguishable. Because index values are included with ciphertexts, this indistinguishability must extend to indexes as well, i.e., they must be indistinguishable to any party who does not possess the key sk, or a properly-formed trapdoor query T. This indistinguishability must hold even when the adversary has access to valid queries based on chosen key/keyword pairs. Boneh *et. al.* [12] formalize this property (as a variation of Goh's *Indistinguishability under Adaptively Chosen Keyword Attack* [17]) in the public-key setting. We adapt this definition in §6.

Query Re-Targeting. While the above definition is sufficient to address the indistinguishability of stored index values, it does not address the unique features of *node-targeted* search. In discussing this property, we first note that node-targeted search does not prevent the intended recipient of a query from *colluding* with another party in order to share queries. For example, a dishonest storage node might simply publish its value of sk_{node}, which would allow any party to evaluate queries targeted to the key pk_{node}.

The goal of the targeting mechanism is instead to prevent *honest* nodes from inadvertantly replying to re-played queries. We refer to attacks on this mechanism as *query re-targeting*. In such an attack, the recipient of a node-targeted query T attempts to produce a useful search query targeted to a different (non-colluding) party. Intuitively, a

node-targeted keyword search scheme is secure if an adversary, given T embedding $(\texttt{user}, \texttt{keyword}, \texttt{node})$, is unable to create a new query T' which will reveal information about (non-adversarially-generated) indexes stored on honest nodes.

Our construction makes use of three-part records that contain a keyword-searchable index, a randomizable public encryption key, and a payload. The payload is a ciphertext (key encapsulation and data encapsulation values) produced using a key-private, IND-RCCA hybrid scheme. Re-encryption (i.e., randomization of) ciphertexts treats the entire payload as a plaintext, and encrypts it using the hybrid scheme (see §5); since the hybrid scheme is semantically secure (a consequence of being IND-RCCA), it does not compromise the unlinkability of two re-encryptions of the same text. By using standard hybrid arguments (omitted here for the sake of brevity), it is only needed that we independently consider (1) the security of the indexing scheme and (2) the security consequences of modifying the basic IND-RCCA scheme by layering encryption steps to support universal re-encryption. In §6, we prove that each is secure under appropriate assumptions.

4 Cryptographic setting and assumptions

Our scheme is constructible within paired groups: Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be groups of prime order p such that there exists an efficiently computable *bilinear map* $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$, where $\mathbb{G}_T$ is also a p-order group, and the following properties hold:

1. Non-degenerate: If P is a generator of $\mathbb{G}_1$ and Q is a generator of $\mathbb{G}_2$, then $e(P, Q)$ generates $\mathbb{G}_T$.
2. Bilinear: $e(aP, Q) = e(P, aQ) = e(P, Q)^a$, where we write the group operation as addition in $\mathbb{G}_1$ and $\mathbb{G}_2$, but as multiplication in $\mathbb{G}_T$.

In order for the scheme to be secure, we require that certain cryptographic assumptions hold, namely:

Assumption 1 *(Symmetric External Diffie-Hellman assumption, or SXDH): Both $\mathbb{G}_1$ and $\mathbb{G}_2$ are DDH-hard groups, i.e., given (P_0, P_1, P_2, P_3) in $\mathbb{G}_1^4$ it is infeasible to decide if there is a value x such that $P_1 = xP_0$ and $P_3 = xP_2$ simultaneously. The same requirement must hold for $\mathbb{G}_2$.*

In appendix A, we review the favorable evidence for the existence of paired groups which satisfy the SXDH assumption. Variants of the XDH assumption [13, 4, 6] including SXDH [5] have been used elsewhere.

Our proofs also make use of a new assumption, named Implicit External Diffie-Hellman assumption, below. In order to promote confidence in the assumption, we include in appendix B a proof of validity in the generic group model [24, 26].

Assumption 2 *(Implicit External Diffie-Hellman, or IXDH) Given groups $\mathbb{G}_1$ and $\mathbb{G}_2$ where the SXDH assumption holds, generated respectively, by P and Q, and values $\tilde{P}$ in $\mathbb{G}_1$ and $(\tilde{Q}_1, \tilde{R}_2, \tilde{Q}_3, \tilde{R}_3)$ in $\mathbb{G}_2^4$. Let $\tilde{Q}_2$ be implicitly defined by the equation $e(P, \tilde{Q}_2) = e(\tilde{P}, Q)$. Let r be implicitly defined by the relation $r\tilde{Q}_3 = \tilde{R}_3$. Then the input is further required to satisfy $r\tilde{Q}_2 = \tilde{R}_2$. If it is computationally infeasible to produce a pair $(Q', a^{-1}bcQ)$, where $\tilde{Q}_3 = aQ$, $\tilde{Q}_2 = bQ$, and $\tilde{Q}_1 = cQ$, then it is said that the Implicit External Diffie-Hellman assumption holds.*

We re-emphasize that, in the above assumption, $\tilde{Q}_2$ is not part of the inputs, but it is given implicitly by either the pair $\tilde{Q}_3$, $r\tilde{Q}_3$, as well as by the point $\tilde{P}$.

5 CRES: A Correlation-Resistant Encryption Scheme

In this section we describe CRES, a scheme for correlation-resistant storage and targeted retrieval from a storage network. CRES allows for efficient encryption and re-encryption of stored records in order to maintain correlation-resistance in a survivable environment. Specifically, the scheme adapts the Universal Re-encryption setting of Golle *et. al.* [19] to incorporate the targeted keyword search scheme described in §5.1. This search scheme enables recovery

of files by allowing retrievers to author search queries which distinguish specific records. Storage nodes can use these queries to identify matching records and return them to users, but cannot re-target them to other honest nodes in order to locate replicas. Moreover, the flexible query construction we present can be based on a combination of user keys and keyword identifiers.

5.1 A Targeted Search Scheme from XDH

A CRES record index is a construct that embeds a set of keywords which can be matched to a query. Indexes are generated by encryptors using the recipient's public key, and are transported along with stored records. To achieve correlation-resistance in a storage network, indexes may be efficiently randomized by any party, *without* access to keying material. Although it would be convenient to use an existing public-key keyword search scheme, earlier schemes (e.g., [12, 34]) are limited (by their setting) to embedding only two elements into a search query (user key, keyword). By employing the novel SXDH setting, our scheme permits searchers to embed an additional public value into the query– the node's public key– without compromising the security of the scheme. Additionally, our setting allows us to build a scheme which is secure in the standard model, without the use of random oracles. Our keyword search scheme is loosely based on the IBE scheme of Waters [33], which we adapt to the SXDH setting in order to guarantee the indistinguishability of index values. [1]

The CRES keyword search scheme assumes a pair of SXDH subgroups $\langle \mathbb{G}_1, \mathbb{G}_2 \rangle$, with generators (P, Q) respectively. Keywords must be selected from a dictionary of size 2^t, where each keyword is uniquely identified by a value $\{0,1\}^t$. The global parameters of our scheme include two vectors $\vec{P} = (P_1, \ldots, P_t) \in \mathbb{G}_1^t$ and $\vec{Q} = (Q_1, \ldots, Q_t) \in \mathbb{G}_2^t$ with the requirement that $e(P_i, Q) = e(P, Q_i)$ for $0 < i \leq t$. A trusted party generates these vectors prior to deployment (see below). We also define an injective, public function and two functions $F_1 : \{0,1\}^t \rightarrow \mathbb{G}_1$ and $F_2 : \{0,1\}^t \rightarrow \mathbb{G}_2$ as:

- $F_1(\cdot)$: let b_i be the i-th bit of the bitstring b. Then $F_1(b) = \sum b_i P_i$.
- $F_2(\cdot)$: let b_i be the i-th bit of the bitstring b. Then $F_2(b) = \sum b_i Q_i$.

We define our scheme as:

- $\mathsf{Setup}(\tau)$. Given a security parameter τ, output a description of SXDH groups $\langle \mathbb{G}_1, \mathbb{G}_2 \rangle$, both of order p, with an efficiently computable pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, and values P and Q such that $\langle P \rangle = \mathbb{G}_1, \langle Q \rangle = \mathbb{G}_2$. Output $\vec{P} = (P_1, \ldots, P_t)$ and $\vec{Q} = (Q_1, \ldots, Q_t)$ by computing $P_i = z_i P$ and $Q_i = z_i Q$ for $z_i \in_R \mathbb{Z}_p$, and discard z_i.
- $\mathsf{KeyGen}_{\mathsf{user}}$. Output a user's keypair $sk_{\mathsf{user}} = s \in_R \mathbb{Z}_p$, $pk_{\mathsf{user}} = sP$.
- $\mathsf{KeyGen}_{\mathsf{node}}$. Output a storage node's keypair $sk_{\mathsf{node}} = x \in_R \mathbb{Z}_p$, $pk_{\mathsf{node}} = xQ$.
- $\mathsf{CreateIndex}(pk_{\mathsf{user}}, w)$. Given a user's public key $pk_{\mathsf{user}} = sP$, and a keyword $w \in \{0,1\}^t$, compute the value $Z = F_1(w)$, select a value $r \in_R \mathbb{Z}_p$, and output the index $I = \langle rsP, rZ \rangle$.
- $\mathsf{RandomizeIndex}(I)$. Given an index $I = \langle \alpha, \beta \rangle$, select a value $r \in_R \mathbb{Z}_p$, and output the new index $I' = \langle r\alpha, r\beta \rangle$.
- $\mathsf{GenerateQuery}(sk_{\mathsf{user}}, w, pk_{\mathsf{node}})$. Given a user's private key $sk_{\mathsf{user}} = s \in \mathbb{Z}_p$, a keyword $w \in \{0,1\}^t$, and a node's public key $pk_{\mathsf{node}} = xQ$, first compute $Y = F_2(w)$. Next, select $k \in_R \mathbb{Z}_p$ and output a query $T = \langle ks(xQ), kY \rangle$.
- $\mathsf{MatchQuery}(I, T, sk_{\mathsf{node}})$. Given an index $I = \langle \alpha, \beta \rangle$, a targeted query $T = \langle \gamma, \delta \rangle$, and a node's secret key $sk_{\mathsf{node}} = x \in \mathbb{Z}_p$, we say that the query matches the index if $(\alpha, \beta, \gamma, x\delta)$ is a co-DDH tuple. If $e(\alpha, x\delta) = e(\beta, \gamma)$, output TRUE, otherwise output FALSE.

Correctness: The above indexing scheme allows for retrieval, as keyword indexes will match those queries which were created using the same keyword, the correct secret key, provided that $\mathsf{MatchQuery}$ is executed by the intended node. This remains true even when the indexes are randomized, due to the bilinearity property of the pairings. The key to generating useful keyword indexes is the ability to compute the functions F_1 and F_2 on an input w such that

[1] We note that the techniques used in this construction might also be adapted to construct a key-private IBE scheme secure in the standard model.

$\log_P F_1(w) = \log_Q F_2(w)$. If we refer to this (unknown) discrete log value as z, we can express the randomized index pair I as $\langle rsP, rzP\rangle$, and the randomized query T as $\langle xksQ, kzQ\rangle$ (for arbitrary values of k, r and $sk_{\mathsf{node}} = x$). Following the process described in MatchQuery, it should be evident that the combined values $(rsP, rzP, xksQ, xkzQ)$ form a co-DDH tuple.

Security: The above scheme allows users to ensure both indistinguishability for search indexes, as well as protection against query re-targeting *without* relying on the use of random oracles. In §6 we formally prove the security of this scheme.

5.2 Design of CRES Records and Algorithms

In what follows we disregard issues related to ciphertext length, although we note that in practice correlation-resistance requires that publishers disguise record lengths through the use of padding, and by partitioning long plaintexts across multiple records. CRES records consist of a searchable index appended to an RCCA-secure ciphertext containing the record data. In addition to the ciphertext and search index, each CRES record includes a *randomized* encryption key which is used to facilitate re-encryption. The ciphertext indistinguishability afforded by each of these elements ensures that an adversary who does not possess the record-specific key will be incapable of linking two equal-length encrypted records.

For reasons of efficiency, CRES records are encrypted using a hybrid encryption scheme, defined by *key encapsulation* and *data encapsulation* mechanisms with the following notation.

- ENC-KEM(pk, r): Given a random seed r, generate a session key k and encrypt k under an asymmetric (possibly randomized) encryption key pk to encapsulate the session key. Output the pair $\langle k, KE\rangle$ where KE is the encapsulated key.
- DEC-KEM(sk, KE): Given an encapsulation KE along with an asymmetric decryption key sk, output the session key k.
- ENC-DEM(k, M): Given a key k (derived from ENC-KEM), encrypts a plaintext M and outputs a ciphertext $\mathcal{C}$.
- DEC-DEM$(k, \mathcal{C})$: Given a key k (derived from DEC-KEM), decrypts a ciphertext $\mathcal{C}$ and outputs the plaintext M.

We require that the KEM/DEM combination used to encrypt CRES records provide IND-RCCA or IND-CCA2 security (the latter definition implies the former). An example of such a scheme is PSEC-KEM [27].

5.3 The CRES Scheme

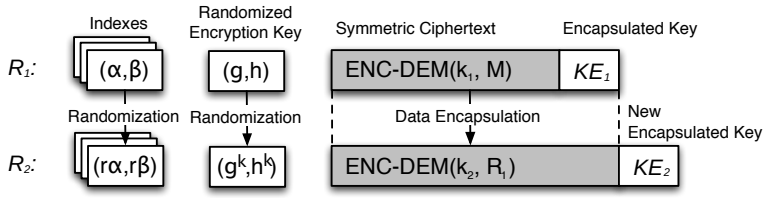

Figure 1: Re-encryption in the CRES protocol. The record index and re-encryption key are first randomized. The KEM uses the randomized key to output a new key encapsulation and a session key which is used to re-encrypt the entirety of the original record. Finally, the new key encapsulation is appended to the resulting ciphertext.

INITIAL SETUP Encryption in CRES requires a set of system-wide parameters defining a pair of XDH groups $\langle \mathbb{G}_1, \mathbb{G}_2\rangle$ with generator points P and Q, respectively, and the vectors $\vec{P}$ and $\vec{Q}$. Group parameters can be shared across any number of CRES deployments. As each node joins the network, it generates a unique public/private key $\langle sk_{\mathsf{node}} = x, pk_{\mathsf{node}} = xQ\rangle$, and publishes pk_{node} using a public-key certification mechanism. Each user in the system

generates a keypair for use with the indexing scheme $\langle sk_{\mathsf{user}} = s, pk_{\mathsf{user}} = sP\rangle$. The user also generates an Elgamal keypair $\langle sk^{\mathsf{dec}}_{\mathsf{user}}, pk^{\mathsf{enc}}_{\mathsf{user}}\rangle = \langle x, (g, g^x)\rangle$ where $\langle g\rangle$ is a DDH-hard group of order q that is distinct from $\mathbb{G}_1$ and $\mathbb{G}_2$, and $x \in_R \mathbb{Z}_q$. The user then publishes his public keys $(pk_{\mathsf{user}}, pk^{\mathsf{enc}}_{\mathsf{user}})$, and delivers the secret keys to authorized receivers. Note that users are not required to certify their public keys. The scheme only requires that nodes have PKI-certified keys.

INITIAL RECORD ENCRYPTION Given a user's public keys $(pk_{\mathsf{user}}, pk^{\mathsf{enc}}_{\mathsf{user}})$, a plaintext M, and a set of n keywords $w_1, w_2, \ldots w_n$, the encryptor first generates a new record index I based on pk_{user} by invoking $\mathsf{CreateIndex}(pk_{\mathsf{user}}, \mathtt{null})$. This index allows retrievers to search for all records belonging to a specific user, without specifying any keyword. Additional indexes are added to allow keyword search for the words $w_1, w_2, \ldots w_n$. ENC-KEM is then invoked using $pk^{\mathsf{enc}}_{\mathsf{user}}$, in order to generate a new session key k and key encapsulation KE. To allow decryptors to detect the initial layer, a redundancy check is provided via appending a hash function to the plaintext. k is then used to encrypt the plaintext using the Data Encapsulation Mechanism (an authenticated symmetric encryption scheme). Next, the encryption algorithm selects a random $r \in \mathbb{Z}_q$ and generates a *randomized* encryption key $pk_R = (g^r, g^{rx})$ from $pk^{\mathsf{enc}}_{\mathsf{user}}$. The key encapsulation KE and randomized key pk_R are appended to the output, producing a re-encryptable, key-encapsulated ciphertext.

RE-ENCRYPTION AND RE-PUBLICATION Given a record and its associated search indexes, the re-encrypting party first randomizes the encryption key pk_R by exponentiating both terms by a new random value $r \in \mathbb{Z}_q$. The re-encryptor next randomizes each record index I by calling $\mathsf{RandomizeIndex}(I)$. ENC-KEM is then invoked using pk_R, which results in a new session key k' and Key Encapsulation KE'. As in the initial encryption stage, k' is used to encrypt the entirety of the input record $\mathcal{R}$ (including the previous key encapsulation). KE' is appended to the resulting ciphertext, producing a new key-encapsulated record. The re-encryption process is illustrated in Figure 1.

DECRYPTION If a record has been re-encrypted multiple times, decrypting and verifying its authenticity may require multiple iterations. To remove each layer of encryption, the decryptor first parses the record to recover the record data and key encapsulation value KE, which is then decapsulated to reveal a session key. Next, the record data is decrypted using the recovered session key. This step may fail if the ciphertext is invalid—the DEM mechanism is assumed to be IND-RCCA (or IND-CCA2) by itself. Otherwise, the decrypted value is parsed to detect the redundancy (hash) added during the initial encryption stage. If this redundancy is not discovered, then the decryptor assumes that it has uncovered an intermediate layer, and repeats the previous steps to uncover the next layer. Should it become impossible to decrypt further layers, the decryptor rejects the ciphertext.

6 Security analysis

In this section, we show that our constructions achieve the security properties set forth in §3. We initially concentrate our efforts on analyzing the security of the indexing scheme only. Later, we show how our hybrid constructions for file encryption/re-encryption achieve the appropriate security notion (IND-RCCA). We achieve this by starting with a hybrid scheme that is IND-RCCA and showing that the universal re-encryption construction preserves that property. In particular, this extend the security analysis of the hybrid construction proposed in the scheme of Golle *et. al.* [19].

6.1 Index indistinguishability

Simulation model/Non-adaptive setting: We subsume all dishonest parties as a single adversary $\mathcal{A}$ (which includes both users and storage nodes). We adopt a non-adaptive setting, i.e., there exist two sets of public keys, the *uncompromised keys* corresponding to honest nodes and/or users, and the *compromised keys* belonging to the adversary, and the character of a party as honest or adversarial not does not change during the simulation. We formulate the index indistinguishability property as an indistinguishability game. The game captures all of the security requirements of the indexing scheme except for the resistance against *query retargeting*, which we consider separately.

Setup: We relate certain forms of adversarial success with simulator success in solving the DDH problem. Suppose the simulator has been challenged with a triple $\langle \tilde{P}_1, \tilde{P}_2, \tilde{P}_3 \rangle \in \mathbb{G}_1^3$, and seeks to determine whether these values form a DH triple. At the beginning of the simulation, the simulator $\mathcal{S}$ generates the public vectors $\vec{P} \in \mathbb{G}_1^n$ and $\vec{Q} \in \mathbb{G}_2^n$ such that for $k = 0, \ldots, n$ $P_k = a_k P$ and $Q_k = a_k Q$, with a_k known to $\mathcal{S}$. The simulator also generates several private/public key pairs s_i, $pk_{\mathsf{user}}(i) = pk_{\mathsf{user}}(\mathcal{U}_i) = s_i P$, except in the case of one the users, for which it sets the public key equal to $\tilde{P}_1$. It publishes the vectors $\vec{P}, \vec{Q}$, and the honest users' public keys $pk_{\mathsf{user}}(i)$. Note that this global setup and public key generation method is indistinguishable from the correct setup of the protocol, as the values are randomly distributed.

Special simulator commit step: Let $\hat{i}$ represent the index of the user specially chosen by the simulator to have public key $\tilde{P}_1$. Let $w_{\hat{j}}$ be one of the keywords in the keyword list. The set $(\hat{i}, \hat{j})$ is the simulator commitment.

Adversarial queries: After the public keys have been generated and published, the adversary $\mathcal{A}$ knows the list of honest users' public keys $\{pk_{\mathsf{node}}(i)\}_{i=1,\ldots,n}$. It also knows the list of keywords L, of length m. During the game, $\mathcal{A}$ may choose a new user/keyword pair $(\mathcal{U}_i, w_j), i = 1, \ldots n, j = 1, \ldots, m$, and submit it to the simulator, with an extra instruction, with possible values `index` or `query`. In each case, the simulator will cause the honest node $\mathcal{U}_j$ to create an index for its own identity and keyword w_j and store it at another node. If the instruction `index` is given, the simulator instructs the honest parties to let some (potentially re-published) randomization of the index eventually reach an adversarial node. In the case of a `query` request, an adversary-targeted query will reach the adversary. Note that the adversary may place both `index` and `query` requests for the same user/keyword pair, and may interleave requests in any order.

Index generation: If the adversary requests an index for the pair $(\mathcal{U}_i, w_j)$, and $(i, j) \neq (\hat{i}, \hat{j})$, the simulator computes $F_1(w_j) = \sum_{i=1}^{t} b_{j,k} P_k$, where $(b_{j,k})$ is the bit-encoding of w_j, and $\vec{P} = (P_i)_{i=1,\ldots,t}$. It then generates the value $(rs_i P, rF_1(w_j))$, for some random value r in $\mathbb{Z}_p^*$. It republishes that value a number of times and then delivers it to an adversarial node. If, on the other hand, the adversary requests an index for the pair $(\mathcal{U}_{\hat{i}}, w_{\hat{j}})$, the simulator first computes α such that $F_1(w_{\hat{j}}) = \alpha P$—which can be done because it knows the discrete logarithm of every P_i to the basis P. It then returns $(\tilde{P}_3, \alpha \tilde{P}_2)$ as the index. This is a correct index only when the simulator's challenge triple is of Diffie-Hellman type.

Query generation: If the adversary requests an index for the pair $(\mathcal{U}_i, w_j)$, and $(i, j) \neq (\hat{i}, \hat{j})$, the adversary first computes α such that $F_2(w_{\hat{j}}) = \alpha Q$—again by using knowledge of the discrete logarithms of Q_i to Q, and then generates the value $(r\alpha \tilde{Q}, rs_i P)$, where $\tilde{Q}$ is an adversarial node's public key. If the adversary requests an index for the pair $(\mathcal{U}_{\hat{i}}, w_{\hat{j}})$, then the simulator *aborts the simulation*. It then throws an unbiased coin and returns a guess for whether the challenge triple is DH or random.

Final guess: Without loss of generality, we assume that $\mathcal{A}$ submits an `index` requests for *all* user/keyword pairs, and also submits `query` requests for *all but one* pair. At the end of a non-aborted simulation, the adversary should have at least one value which could be (1) a random value, or (2) an index corresponding to the only user/keyword pair for which it does not have the corresponding query. At this point, the adversary may either guess whether the stored value is a legitimate index or not.

Theorem 6.1 *Let $\mathcal{A}$ be an adversary that has advantage ε_{idx_ind} against the index indistinguishability game, in a system with m keywords and n users, performing $q_{\mathcal{A}}$ computational steps. Then, it is possible to construct an adversary $\mathcal{B}$ that solves DDH instances in $16mnq_{\mathcal{A}}$ steps, with advantage ε_{DDH}, where:*

$$\varepsilon_{DDH} \geq \min\{\frac{1}{4}, 8\varepsilon_{idx\_ind}\}.$$

This result is a *tight reduction* in the standard model, provided that $8\varepsilon_{idx\_ind} \leq \frac{1}{4}$. To see this, note that while the statement appears to indicate a somewhat looser reduction with deterioration factor $1/2mn$, in fact the proper measure is probability of breaking *per user/keyword* pair (since there are now many choices to an adversary), which is just $\frac{\varepsilon_{idx\_ind}}{2}$, a deterioration factor of a single bit in the effective security.

In order to prove the above claimed result, first note that whenever the simulation does not abort, the adversary ends with a perfect instance of the index indistinguishability game. Moreover, when the challenge tuple is a DH tuple,

and the adversary is challenged with a correctly formed index, it cannot distinguish the special user/keyword pair from other pairs. Therefore, in this case, the simulation must go through to the end with probability equal to $1/nm$, where n is the number of users and m, of keywords. Let $\varepsilon^{+}_{idx\_ind}$ be the advantage (over random guessing) that $\mathcal{A}$ has in correctly identifying "DH tuple" indexes conditioned on it being true. Then, in 1/2 of the cases where the challenge tuple is a Diffie-Hellman tuple the adversary has advantage $\frac{\varepsilon^{+}_{idx\_ind}}{nm}$.

When the challenge is not a DH tuple, however, it is possible that the adversary might detect the difference between the indexes and force the simulation to fail with probability higher than $1-1/nm$, say probability $1-p$. The simulator can use the difference of probability $1/nm$ and $1-p$ to break DDH, as follows. Say, for instance, that $p = \frac{1}{2mn}$. Then, repeating the simulation $16mn$ times, the expected number of successful simulations is 16 in the first case, but only 8 in the second. The difference is 8, which equals twice the standard deviation, which here equals 4.[2] By the Chebyshev inequality,[3] the the simulator can conclude that this is not a DH tuple with probability at least $3/4$, and advantage $1/4$. We can conclude that $p > \frac{1}{2mn}$. (Assuming willingness to perform $16mn$ simulations.)

Similarly, if the challenge tuple is NOT a DH tuple, and the simulation succeeds, then the simulation is a perfect reproduction of the index indistinguishability game when the adversary is challenged with a random index. In this case, the adversarial guess can also be used to provide a solution for the DDH tuple. Let $\varepsilon^{-}_{idx\_ind}$ be the adversarial advantage in correctly guessing "non-random tuple" conditioned on it being true.

Assembling it all together, we see that the simulator advantage in breaking DDH using the index indistinguishability adversary is at least $\min\{\frac{1}{4}, \frac{1}{mn}(\frac{\varepsilon^{+}_{idx\_ind}}{2} + \frac{\varepsilon^{-}_{idx\_ind}}{4})\}$, which is bound below by $\min\{\frac{1}{4}, \frac{1}{2mn}(\varepsilon_{idx\_ind})\}$.

6.2 Infeasibility of query re-targeting

This game is very similar to the index indistinguishability one, except that the simulator does not need to embed Diffie-Hellman tuples in the public keys of any honest users. Additionally, instead of requesting randomized versions of all indexes and all but one query, the adversary requests all queries. Its task is to construct a query for a particular index, targeted at an honest node.

First, we note that, since in our setting there are no efficiently computable homomorphisms from $\mathbb{G}_1$ to $\mathbb{G}_2$, neither indexes nor the values of public keys of users will help with the forgery, as they are values in $\mathbb{G}_1$. So we only need to concentrate on the received queries, which take the form: $(r_{i,j}F_2(w_j), r_{i,j}xs_iQ))$, where x is the adversary's private key, and s_i are the private keys of honest users. The $r_{i,j}$ are the randomizing values, unknown to $\mathcal{A}$.

The adversary can easily use its knowledge of x^{-1} to "untarget" the received queries, and get: $(r_{i,j}F_2(w_j), r_{i,j}s_iQ)$. The problem statement is then, given such values, as well as the public keys of honest nodes, $pk_{\mathsf{node}}(\mathcal{N}_u)$, compute $(rF_2(w_{\hat{j}}), rs_{\hat{i}}pk_{\mathsf{node}}(\mathcal{N}_u))$, for some choice $(\hat{j}, \hat{i}, \hat{u})$. Labeling $\alpha_{i,j} \leftarrow r_{i,j}\delta_j$, where $\delta_j Q = F_2(w_j)$, $\beta_{i,j} \leftarrow r_{i,j}s_i$, and $\gamma_u \leftarrow x_u$, where $pk_{\mathsf{node}}(\mathcal{N}_u) = x_uQ$, we get: Given $(\alpha_{i,j}Q, \beta_{i,j}Q, \gamma_uQ, \delta_jQ)$, find $T = (Q', \alpha^{-1}_{\hat{i},\hat{j}}\beta_{\hat{i},\hat{j}}\gamma_{\hat{u}}Q')$, for some choice $(\hat{i}, \hat{j}, \hat{u})$ of indices. We are now ready to show the reduction. Note that we modify the simulation game so that now the adversary receives queries already "untargeted," which gives the adversary an equivalent view of the problem as far as the query re-targeting task is concerned.

Theorem 6.2 *Let $\mathcal{A}$ be an adversary of the query re-targeting, with advantage $\varepsilon_{retarget}$ in $q_{\mathcal{A}}$ steps, on a system with m keywords, n honest users, and v honest nodes. Then one may use $\mathcal{A}$ to defeat IXDH in $mnvq_{\mathcal{A}}$ steps, with advantage:*

$$\varepsilon_{IXDH} \geq \varepsilon_{retarget}.$$

Again, the above result is a *tight reduction* in the standard model to the IXDH assumption. Note that while the reduction seems "loose" by a factor $1/mnv$, the appropriate comparison here is with the security per user/keyword/-node triple, in which case the reduction involves no security loss.

Now, to show this result, assume the simulator starts with a set of values $\tilde{P}$ in $\mathbb{G}_1$ and $(\tilde{Q}_1, r\tilde{Q}_2, \tilde{Q}_3, r\tilde{Q}_3)$ in $\mathbb{G}_2$, where $e(P, \tilde{Q}_2) = e(\tilde{P}, Q)$. It wishes to obtain $(Q', a^{-1}bcQ')$, where Q' is any element of $\mathbb{G}_2$, and $\tilde{Q}_3 = aQ$, $\tilde{Q}_2 = bQ$, $\tilde{Q}_1 = cQ$. $\mathcal{S}$ chooses random values Q_k during the setup phase such that it knows the discrete logarithm of

[2] The variance of a sequence of independent events given by the same distribution equals the square root of the expected value.

[3] The Chebyshev inequality [23] says that the probability that the number of observed occurrences of an outcome in a series of repeated experiments differs from the expected value by $u\sigma$, where σ is the variance, is no larger than $1/u^2$.

each Q_k with respect to $\tilde{Q}_3$. It sets $\tilde{P}$ to be a user's public key, and $\tilde{Q}_1$ to be an honest nodes' public key, and generates other user's public keys so that it knows the respective private keys. Now, the simulator can answer any adversarial queries for pairs $\mathcal{U}_i$, w_j, where $\mathcal{U}_i$ does not have public key $\tilde{P}$. In the case for a query with $\tilde{P}$ as the user's identity, the simulator can use the knowledge of the discrete logarithm δ_j of $F_2(w_j)$ with respect to $\tilde{Q}_3$ to answer the query with the value $(r'r\tilde{Q}_2, r'r\delta_j\tilde{Q}_3)$. It is straightforward to verify that the simulation is perfect, and that the chance that the adversary will choose the same values prepared by the simulator equal $\frac{1}{mnv}$, where m, n, and v are respectively the number of keywords, honest users, and honest nodes, respectively. The result follows from the IXDH assumption stated in §3.

6.3 IND-RCCA security of hybrid universal re-encryption schemes

The principal result described in this section is the notion that RCCA-secure universal re-encryption schemes may be constructed from an RCCA or CCA2-secure hybrid encryption scheme which allows for the randomization of public keys.

It is clear that no re-encryptable construction can meet IND-CCA2 security requirements, because IND-CCA2 security implies a strong non-malleability of ciphertexts—namely, that it be infeasible to produce new ciphertexts from an existing ciphertext, *even* if the new ciphertexts encrypt the same plaintext. However, as observed in [14], a re-encryptable ("publicly randomizable") scheme may indeed be IND-RCCA secure. To illustrate this point, [14] presents a simple composition in which a message is first encrypted using a CCA2 or RCCA-secure scheme, and the resulting ciphertext is encrypted under a re-encryptable CPA-secure scheme. The remainder of this section simply extends this result to the case of *universal* re-encryption (where the public key is not known), and demonstrates that RCCA-secure universal re-encryption schemes may be constructed from existing IND-RCCA (or IND-CCA2) KEM/DEM constructions that accept Elgamal-type keys.

Theorem 6.3 *If a hybrid scheme $\mathcal{HE}$ is IND-RCCA, then so is the universal re-encryption scheme $\mathcal{UHE}$ obtained from $\mathcal{HE}$ via the layered construction. More explicitly, suppose that an adversary to $\mathcal{UHE}$ succeeds with probability at most $\varepsilon_{\mathcal{UHE}}$ after q_D decryption queries to the decryption oracle, in which it may forward ciphertexts of total length L_D. Then the adversary to $\mathcal{HE}$ success with same probability after $\max\{q_D, \frac{L_D}{\ell}\}$ queries to the decryption algorithm, where ℓ is the minimum amount of ciphertext expansion by application of the $\mathcal{HE}$-encryption scheme.*[4]

Proof: Let $\mathcal{A}$ be an IND-RCCA adversary of $\mathcal{UHE}$, and we show how to implement an IND-RCCA adversary $\mathcal{B}$ for $\mathcal{HE}$.
Key generation: The challenger gives public key e to $\mathcal{B}$, which is forwarded to $\mathcal{A}$.
First stage: Each time $\mathcal{A}$ generates a query of the form $(\texttt{ciphertext}, \tilde{c})$, $\mathcal{B}$ simulates the universal decryption algorithm, making a call to the $\mathcal{HE}$-decryption oracle for every layer of decryption. It forwards the final result to $\mathcal{A}$.
Choice and Challenge: $\mathcal{A}$ generates a choice $(\texttt{choice\_messages}, m_0, m_1)$, which $\mathcal{B}$ forwards to the challenger. When $\mathcal{B}$ receives the challenge c, it sends c to $\mathcal{A}$.
Second stage: $\mathcal{B}$ interprets $\mathcal{A}$'s and the challenger's actions exactly as in the first stage.
Guessing stage: When $\mathcal{A}$ puts forth a guess b_0, $\mathcal{B}$ forwards this guess to the challenger.

It should be clear that the simulation succeeds, with equal probability in both cases. We only need to account for the fact that one query of $\mathcal{A}$ may result in several $\mathcal{B}$-queries to a $\mathcal{HE}$-oracle, while successive encryption layers are removed. Each query of $\mathcal{B}$ to $\mathcal{HE}$ results in a length decrease of at least ℓ bits. So $\mathcal{B}$ cannot make more than L_D/ℓ queries, where L_D is the total length of the ciphertext, except if $\mathcal{A}$ puts forward queries smaller less than ℓ length (guaranteed to fail), in which case $\mathcal{B}$ makes at least one query for each of $\mathcal{A}$'s queries. Hence the bound $\max\{q_D, L_D/\ell\}$.

[4] ℓ is a non-zero number, as it includes at least the KEM length and the DEM minimum ciphertext expansion, which is non-zero since it implements an authenticated mode of the symmetric block cipher.

7 Conclusion

We present the requirements for a new survivable storage model with the property of correlation-resistance. To achieve this we offer a specific construction that meets our privacy goals without imposing unnecessary rigidity. In doing so, we provide a provably-secure public-key encryption scheme which allows for dynamic re-encryption as well as targeted keyword search.

References

[1] A. Adya, W. Bolosky, M. Castro, R. Chaiken, G. Cermak, J. Douceur, J. Howell, J. Lorch, M. Theimer, and R. Wattenhofer. Farsite: federated, available, and reliable storage for an incompletely trusted environment. *SIGOPS Oper. Syst. Rev.*, 36(SI):1–14, 2002.

[2] Jee Hea An, Yevgeniy Dodis, and Tal Rabin. On the security of joint signature and encryption. In *EUROCRYPT '02: Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pages 83–107, London, UK, 2002. Springer-Verlag.

[3] Ross Anderson. The eternity service. In *Proceedings of Pragocrypt '96*, 1996.

[4] Giuseppe Ateniese, Jan Camenisch, and Breno de Medeiros. Untraceable RFID tags via insubvertible encryption. In *Conference on Computer and Communications Security – CCS'05*, Alexandria, Virginia, USA, November 2005. ACM, ACM Press.

[5] Giuseppe Ateniese, Jan Camenisch, Breno de Medeiros, and Susan Hohenberger. Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385, 2005. `http://eprint.iacr.org/`.

[6] Lucas Ballard, Seny Kamara, and Fabian Monrose. Achieving efficient conjunctive keyword searches over encrypted data. In *Proceedings of the Seventh International Conference on Information and Communications Security (ICICS '05)*, to appear December 2005.

[7] Paulo S.L.M. Barreto, Hae Yong Kim, Ben Lynn, and Michael Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology – CRYPTO '02*, volume 2442 of LNCS, pages 354–368, 2002.

[8] Mayank Bawa, Roberto J. Bayardo Jr., and Rakesh Agrawal. Privacy-preserving indexing of documents on the network. In *VLDB 2003*, pages 922–933, 2003.

[9] Krista Bennett, Christian Grothoff, Tzvetan Horozov, and J. T. Lindgren. An encoding for censorship-resistant sharing, 2003. `http://www.ovmj.org/GNUnet/download/ecrs.ps`.

[10] D. Boneh and X. Boyen. Short signatures without random oracles. In *Proceedings of Advances in Cryptology – Eurocrypt'04*, volume 3027 of *Lecture Notes in Computer Science*, pages 56–73. Springer-Verlag, 2004.

[11] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *Proceedings of Advances in Cryptology – Crypto'04*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55. Springer-Verlag, 2004.

[12] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT 2004*, pages 506–522, 2004.

[13] Jan Camenisch, Susan Hohenberger, and Anna Lysyanskaya. Compact e-cash. Cryptology ePrint Archive, Report 2005/060, 2005. `http://eprint.iacr.org/`.

[14] Ran Canetti, Hugo Krawczyk, and Jesper Nielsen. Relaxing chosen-ciphertext security. Cryptology ePrint Archive, Report 2003/174, 2003. `http://eprint.iacr.org/`.

[15] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: A distributed anonymous information storage and retrieval system. *Lecture Notes in Computer Science*, 2009:46+, 2001.

[16] S. D. Galbraith and V. Rotger. Easy decision Diffie-Hellman groups. *Journal of Computation and Mathematics*, 7:201–218, 2004.

[17] Eu-Jin Goh. Secure indexes. Cryptology ePrint Archive, Report 2003/216, 2003. `http://eprint.iacr.org/2003/216/`.

[18] Ian Goldberg and David Wagner. TAZ servers and the rewebber network: Enabling anonymous publishing on the world wide web. *First Monday*, 3(4), August 1998.

[19] Philippe Golle, Markus Jakobsson, Ari Juels, and Paul Syverson. Universal re-encryption for mixnets. In *Proceedings of the 2004 RSA Conference, Cryptographer's track*, San Francisco, USA, February 2004.

[20] J.Kubiatowicz, D.Bindel, Y. Chen, S. Czerwinski, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, C. Wells, and B. Zhao. Oceanstore: an architecture for global-scale persistent storage. In *Proceedings of the ninth international conference on Architectural support for programming languages and operating systems*, pages 190–201. ACM Press, 2000.

[21] Hugo Krawczyk. The order of encryption and authentication for protecting communications (or: How secure is ssl?). In *CRYPTO '01: Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pages 310–331, London, UK, 2001. Springer-Verlag.

[22] A. Miyaji, M. Nakabayashi, and S. Takano. New explicit conditions of elliptic curves for fr-reduction. In *IEICE Transactions on Fundamentals*, volume E84-A, no.5, pages 1234–1243, 2001.

[23] Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1st edition, 1995.

[24] V. I. Nechaev. Complexity of a determinate algorithm for the discrete logarithm. *Mathematical Notes*, 55:165–172, 1994.

[25] Mike Scott. Authenticated ID-based key exchange and remote log-in with simple token and PIN number. Technical Report 2002/164, International Association for Cryptological Research, 2002. Available at http://eprint.iacr.org/2002/164.

[26] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *Proc. of Advances in Cryptology (EUROCRYPT'97)*, LNCS, pages 256–266, 1997.

[27] Victor Shoup. A proposal for an iso standard for public key encryption (version 2.1). Manuscript, 2001. `http://www.shoup.net/papers/iso-2_1.pdf`.

[28] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *IEEE Symposium on Security and Privacy*, pages 44–55, 2000.

[29] Mudhakar Srivatsa and Ling Liu. Countering Targeted File Attacks using LocationGuard. In *Proceedings of the 14th USENIX Security Symposium*, to appear August 2005.

[30] Eric R. Verheul. Evidence that XTR is more secure than supersingular elliptic curve cryptosystems. *J. Cryptology*, 17:277–296, 2004.

[31] Marc Waldman and David Mazières. Tangler: a censorship-resistant publishing system based on document entanglements. In *Proceedings of the* 8^{th} *ACM Conference on Computer and Communications Security (CCS 2001)*, pages 126–135, November 2001.

[32] Marc Waldman, Aviel D. Rubin, and Lorrie Faith Cranor. Publius: A robust, tamper-evident, censorship-resistant, web publishing system. In *Proc. 9th USENIX Security Symposium*, pages 59–72, August 2000.

[33] Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT 2005*, pages 114–127, 2005.

[34] Brent R. Waters, Dirk Balfanz, Glenn Durfee, and D. K. Smetters. Building an encrypted and searchable audit log. In *Proceedings of Network and Distributed System Security Symposium 2004 (NDSS'04)*, San Diego, CA, February 2004.

A DDH-hard Pairing Groups

In most applications of pairings to date, *two mathematical properties* are exploited: The existence of the bilinear pairing (defined between distinct, prime order subgroups of the elliptic curve) and a *distortion map*. The latter is required for optimization reasons: One of the paired groups is defined over the base field, typically $\mathbb{F}_q$, where q is a prime in the 160–300 bit range; while the second group is defined over a larger extension field $\mathbb{F}_{q^\ell}$ (for $\ell = 6$, elements of this field require in the range of 960–1800 bits to represent), wherein group operations are less efficient.

To avoid the performance penalty, super-singular curves are adopted. In such, there is a computable homomorphism $\psi : \mathbb{G}_1 \to \mathbb{G}_2$ from the base field-defined group to the group defined over the extension field. One may then define a distorted pairing entirely within $\mathbb{G}_1$ as $\tilde{e}(g, h) := e(g, \psi(h))$. Because the pairing "internalizes" to the group admitting short representations, one obtains better performance for many pairings-based internalized protocols. Another consequence of the existence of an "internal" pairing is that $\mathbb{G}_1$ admits an efficient Decision Diffie-Hellman solver, and is therefore DDH-easy.

Until recently, it was thought that the existence of a distortion map was key to reasonable performance with pairing operations. However, implementations by Barreto et al. [7] demonstrate that the non-supersingular MNT curves (after Miyaji, Nakabayashi, and Takano [22]), result in *more* efficient implementations of pairings-based protocols than supersingular curve implementations, when one also considers that non-supersingular curves achieve comparable security to supersingular ones at smaller key sizes.

One consequence of the adoption of these curves is that, while distortion maps do exist for *most* of the subgroups of these curves, they have been proven NOT to exist on the most natural subgroup choice, namely $\mathbb{G}_1$ above which is defined over the base field. In fact, it is a result of Verheul [30] that for both eigenspaces of the Frobenius map for the field extension $\mathbb{F}_{q^\ell}/\mathbb{F}_q$ there are no distortion maps. The group $\mathbb{G}_1$ defined over the base field is the eigenspace where the Frobenius action is trivial (eigenvalue equal to 1), and the other eigenspace corresponds to a group $\hat{\mathbb{G}}_1$ where the trace of Frobenius equals 0. We re-state Verheul's results here, for the sake of making this appendix self-contained.

Theorem A.1 *(Verheul's theorem [30]) Let E be an MNT curve defined over a base field $\mathbb{F}_q$, and such that $E(\mathbb{F}_q)$ contains a large subgroup $\mathbb{G}_1$ of prime order p. Let $\ell > 1$ be the (small) embedding degree of $\mathbb{G}_1$, and let $\mathbb{G}_2$ be* any *p-subgroup of the elliptic curve different from $\mathbb{G}_1$ and from $\hat{\mathbb{G}}_1$. Then, there does* not *exist an efficiently computable distortion map ψ from $\mathbb{G}_1$ to $\mathbb{G}_2$, while there* exists *a distortion map from $\mathbb{G}_2$ to $\mathbb{G}_1$.*

In the above, $\hat{\mathbb{G}}_1$ again refers to the p-order trace-0 Frobenius eigenspace within $E(\mathbb{F}_{q^\ell})$.

Notice that the statement of Verheul's theorem remains true if the roles of $\mathbb{G}_1$ and $\hat{\mathbb{G}}_1$ are reversed. On the other hand if $\mathbb{G}_2$ is chosen to equal $\hat{\mathbb{G}}_1$, we get instead a situation in which neither of the two paired groups admits a distortion map. In the first case, we pose the assumption that $\mathbb{G}_1$ is a DDH-hard, while in the second, that both $\mathbb{G}_1$ and $\mathbb{G}_2$ are DDH-hard. This lead to the asymmetric and symmetric versions of the XDH assumption:

Assumption 3 *(Asymmetric XDH assumption) Let E be an MNT curve defined over $\mathbb{F}_q$, such that $E(\mathbb{F}_q)$ has a large subgroup $\mathbb{G}_1$ of prime order p with small embedding degree ℓ. Let $\mathbb{G}_2$ be the any subgroup of $E(\mathbb{F}_{q^\ell})$ different from* both *$\mathbb{G}_1$, and from the trace-0 subgroup of $E(\mathbb{F}_{q^\ell})$ under the Frobenius map of $\mathbb{F}_{q^\ell}/\mathbb{F}_q$, and $e : \mathbb{G}_1 \times \mathbb{G}_2$ be the Tate pairing. If $\mathbb{G}_1$ is a DDH-hard group, we say that the* Asymmetric external Diffie-Hellman assumption *holds (XDH) for the pair $(\mathbb{G}_1, \mathbb{G}_2)$.*

We are not the first authors to use the above assumption for realizing cryptographic constructions [25, 11].

For the sake of completeness, we also describe the symmetric XDH assumption. This assumption is not used in this paper, but can be used to provide additional privacy guarantees in our setting (these extensions are presented in the full version of this paper.) As far as we know, the assumption below has not been used for cryptographic protocols:

(Symmetric XDH assumption, or SXDH) Let E be an MNT curve defined over $\mathbb{F}_q$, such that $E(\mathbb{F}_q)$ has a large subgroup $\mathbb{G}_1$ of prime order p with small embedding degree ℓ. Let $\mathbb{G}_2$ stand for $\hat{\mathbb{G}}_1$, i.e., the trace-0 subgroup of $E(\mathbb{F}_{q^\ell})$ under the Frobenius map of $\mathbb{F}_{q^\ell}/\mathbb{F}_q$, and $e : \mathbb{G}_1 \times \mathbb{G}_2$ be the Tate pairing. If BOTH $\mathbb{G}_1$ and $\mathbb{G}_2$ are DDH-hard groups, we say that the Symmetric external Diffie-Hellman (SXDH) assumption holds for the pair $(\mathbb{G}_1, \mathbb{G}_2)$.

Note that, while Verheul's theorem does not prove that the Frobenius eigenspaces in MNT curves are DDH-hard groups, it does rule out the only methods known to date that can be used to solve DDH in elliptic curve subgroups, namely the use of pairings in combination with distortion maps. Therefore, it would seem to indicate that the XDH assumptions (symmetric and asymmetric) are reasonable extrapolations of our current state-of-knowledge about the hardness of DDH problems.

The work of Galbraith and Rotger [16] is also directly relevant to our constructions; this latter paper expands on Verheul's investigations on distortion maps, providing practical constructions of distortion maps between subgroups (in the cases where they exist).

B Proof of IXDH in the generic group model

We now proceed to prove that IXDH holds in the generic group model. The model, introduced by Nechaev [24], and further developed by Shoup [26], uses the artifice of considering the binary encodings of group elements as black box algorithms (oracles). Therefore, the only algorithms allowed in "generic groups" are those that do not exploit the characteristics of a particular encoding.

Definition B.1 *The encodings $\chi_i, i = 1, 2,$ or T, are arbitrary mappings of elements of $\mathbb{Z}_p$ to elements of $\mathbb{G}_1$, $\mathbb{G}_2$, or $\mathbb{G}_T$, respectively. This encodings are moreover assumed to represent homomorphisms of the group $(\mathbb{Z}_p, +)$ to each $\mathbb{G}_i$ (under its defined group law).*

Generic Group Model: Assume that there exist efficient oracles to compute each encoding above, but otherwise these mappings are opaque, i.e., black boxes which are invulnerable to cryptanalysis and cannot be distinguished from random oracles. The resulting computational model is the so-called Generic Group Model, or Generic Model (GM). Under such computational model, the cost of computing discrete logarithms can be shown to be exponential, with the most efficient algorithm being of the baby-step/giant-step type, such as Pollard's rho.

Extensions of the GM: In the generic group model, the discrete logarithm problem (DLP), (computational) Diffie-Hellman problem (CDH) and decisional Diffie-Hellman problem (DDH) can all be shown to require exponential effort. The model has been extended [10] for (pairs of) groups with pairings, by adding other mappings. The pairing oracle $e(\cdot, \cdot)$ accepts as inputs encoded pairs of elements in $\mathbb{G}_1 \times \mathbb{G}_2$, and returns an encoded element of $\mathbb{G}_T$. It implements a bilinear, non-degenerate mapping, which is otherwise opaque to analysis. The net result of adding this oracle to the model is that the co-DDH problem becomes efficiently computable, while the co-CDH, and the DDH problem in $\mathbb{G}_i, i = 1, 2$, or T, remain exponentially hard. These results are in accordance to the current empirical knowledge of low embedding degree MNT-type curves, within the range of key sizes for which the MOV attack does not enable faster algorithms for the DLP (see above paragraph).

We now prove that the IXDH conjecture holds in the generic group model:

Theorem B.2 *Let $\chi_1(1)$, $\chi_2(1)$, and $\chi_T(1)$ be given encodings of generators of $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$, respectively. Suppose that one is also given encodings $\chi_1(Y)$ of an element in $\mathbb{G}_1$, and of elements $\chi_2(X)$, $\chi_2(RY)$, $\chi_2(RZ)$, $\chi_2(Z)$. Then, the number of steps that an algorithm will need to perform on average before it may output a pair of encodings $\chi_2(SZ)$, $\chi_2(SXY)$ with probability ε is at least $\frac{\sqrt{\varepsilon p}}{2}$, where p is the order of the group.*

A generic algorithm maintains a list of polynomials F_i, $i = 1, 2$, or T, for each group. Let $\tau_{i,t}$ stand for the length of the i-th queue at step t of the algorithm. The j-th element of list F_i is denoted by $F_{i,j}$. Note that the lists are initialized as:

$$F_1 = \{1, Y\}, \tau_{1,0} = 2;\ F_2 = \{1, X, RY, Z, RZ\}, \tau_{2,0} = 5;\ F_T = \{1\}; \tau_{T,0} = 1.$$

At any particular step, the algorithm may use the group operation oracle within one of the groups to update $F_{i,t} \leftarrow F_{i,t'} \pm F_{i,t''}$, where t' and t'' are smaller than t. The new polynomial is then added to the list, and therefore $\tau_{i,t} \leftarrow \tau_{i,t-1} + 1$, while the other lists do not grow. The algorithm may also instead choose to consult the pairing oracle to obtain $F_{T,t} \leftarrow F_{1,t'}F_{2,t''}$ for some t', and t'' smaller than t (also $\tau_{T,t} \leftarrow \tau_{T,t-1} + 1$). At any particular point in time, the polynomials $F_{i,j}$ have the following general form:

$$\begin{aligned} F_{1,t} = &\quad \alpha_t + \beta_t Y \\ F_{2,t} = &\quad \gamma_t + \delta_t X + \varepsilon_t RY + \zeta_t Z + \theta RZ \\ F_{T,t} = &\quad \eta_t + \iota_t X + \kappa_t RY + \lambda_t Z + \mu RZ + \nu_t Y + \pi_t XY + \rho_t RY^2 + \sigma_t ZY + \upsilon RZY \end{aligned}$$

Now, each of these polynomials correspond to one element of one of the groups, that is computed by the generic algorithm. The encoding oracle keeps a list of the above polynomials and performs equality testing. If a polynomial is new, it responds with a new value in $\mathbb{G}_i$ at random for the computed by the algorithm. Otherwise, it looks up the previously returned value, in order to provide for consistent, deterministic answers.

In order to produce the answer we need to have two of these steps produce equal coefficients for the term XY and the term Z, in the list F_2. In other words, we need that $F_{2,t'}(X, Y, R, Z) - F_{2,t''}(X, Y, R, Z \leftarrow XY) \equiv 0$, with $F_{2,t'}(X, Y, R, Z) - F_{2,t''}(X, Y, R, Z) \not\equiv 0$. (I.e., the polynomials became identical after—not before—the substitution $Z = XY$.)

However, substituting $Z = XY$ in the equations for $F_{2,t}$ above, and subtracting, we get:

$$(\gamma_{t'} - \gamma_{t''}) + (\delta_{t'} - \delta_{t''})X + (\varepsilon_{t'} - \varepsilon_{t''})RY + \zeta_{t'}Z - \zeta_{t''}XY + \theta_{t'}RZ - \theta_{t''}RXY.$$

If the above polynomial is to be identically 0, then $\gamma_{t'} = \gamma_{t''}$, $\delta_{t'} = \delta_{t''}$, and $\varepsilon_{t'} = \varepsilon_{t''}$, and all other coefficients must equal 0. But that implies that the two polynomials were already equal before the substitution $XY \leftarrow Z$.

Since the polynomials are not identically 0, they cannot provide solutions in general. They may be able to produce solutions for some instantiations of the variables R, X, Y, and Z. There are $\binom{\tau_{2,max}}{2}$ different pairs of elements in the list F_2, where $\tau_{2,max}$ is the final number of elements in the list F_2, in particular a smaller number than the number of steps performed by the algorithm. Moreover, the probability of any such pair evaluating to equal values is at most $3/p$ (considering that the F_2 are quadratic polynomials). It follows that the above solution is only achieved with probability at most $3\tau_{2,max}^2/2p$. We now need to consider the possibility of the simulation to fail. This could happen whenever two polynomials in F_2 evaluate to the same value for an instantiation of the variables X, Y, R, Z, without being identical polynomials, and similarly for the polynomials with the substitution $Z \leftarrow XY$. The first case happens with probability $\binom{\tau_{2,max}}{2}(2/p) \simeq \tau_{2,max}^2/p$, and the second with probability $\binom{\tau_{2,max}}{2}(3/p) \simeq 3\tau_{2,max}^2/2p$. The total probability of adversarial success is therefore at most $4\tau_{2,max}^2/p$. The result follows.

Note that, in the absence of morhphisms from $\mathbb{G}_1$ to $\mathbb{G}_2$ (as in the setting for the XDH conjecture) it is useless to perform operations outside of $\mathbb{G}_2$ at all, and this is reflected in that above only the polynomials in the list F_2 were useful.