Black Forest Database™

Craxel, Inc.

ABSTRACT

We now live in the era of irreversible reliance on technology with consistently eroding digital trust. Enterprises have entrusted their sensitive information to increasingly expensive software and systems that simply can't protect the data, user privacy yet scale to future data demands. Customers still expect their information to be completely protected and used responsibly in their best interests. Currently encryption is the only provable way to secure our information. So why don't we encrypt all sensitive information throughout the entire technological ecosystem? There are 3 key problems. The first is that encryption is only as secure as the encryption keys and to properly compartmentalize information, many keys are required. Key management with lots of encryption keys is very hard to manage. There is also a fear of losing keys and the information those keys unlock. The second problem is that nobody wants to slow down operations or suffer any performance loss caused by encryption. Although modern ciphers are very fast and today's CPU chips provide acceleration, this hasn't always been the case. Third, nobody wants to lose capability because their information is encrypted and they can't easily find it or access it on-demand.

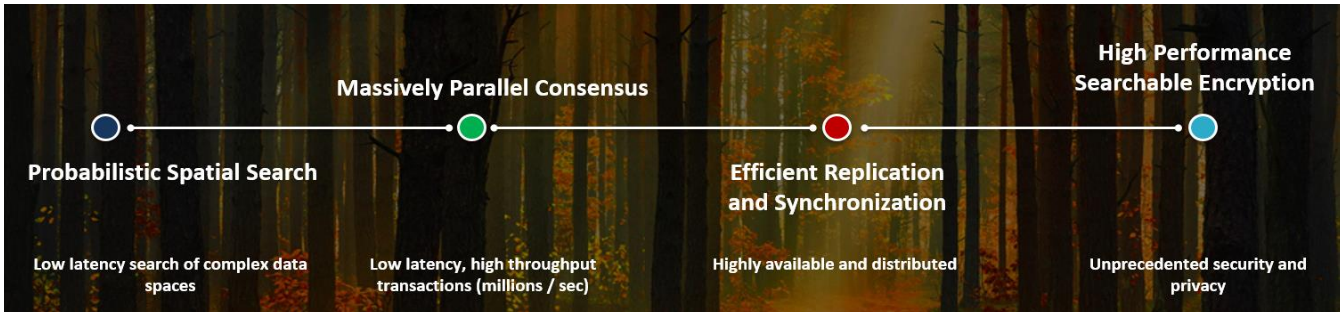

Massively Parallel Consensus
High Performance Searchable Encryption
Probabilistic Spatial Search
Efficient Replication and Synchronization
Low latency search of complex data spaces
Low latency, high throughput transactions (millions / sec)
Highly available and distributed
Unprecedented security and privacy

Figure 1 Craxel's Key Innovations

Craxel's high-performance secure data management solution delivers a compelling solution to each of these problems. Craxel has developed a platform to enable data to be encrypted, then stored and managed with incredible performance and scale, while never seeing the contents of the data. By combining high performance searchable encryption with a novel database architecture to store and organize strongly encrypted data, the entire data layer of the IT architecture can be zero knowledge and not susceptible to data breaches. Zero knowledge means the infrastructure can still manage and protect data without ever knowing its contents to ***eliminate the greatest attack surface within an enterprise: the central repository of data***. Enabling application layer encryption means that information can be highly compartmentalized, limiting the potential damage of successful attacks on any remaining attack surface. Further, shifting the attack surface to small portions of the IT architecture only where data must be decrypted can reduce cybersecurity expenditures while increasing risk mitigation.

Craxel's high performance secure data management enables a new information architecture that improves information security across the enterprise and enables two key capabilities:

1. The ability to store and organize strongly encrypted data without making it vulnerable in the data layer (Black Forest Database "BFDB")

2. The ability to exchange value and information securely without having trusted intermediaries look at the data (Black Forest Distributed Ledger "BFDL")

Organize information with unprecedented security
Black Forest Database™
Securely exchange value or information with high level of assurance
Black Forest Distributed Ledger™

Figure 2 Key Functions of Digital Trust Platform

Together, BFDB and BFDB are key components of a "Digital Trust Platform" that can index and search strongly encrypted data without decryption, without the encryption keys present, and with remarkable performance and at massive scale.

Black Forest DB™ (BFDB), the subject of this white paper, is a high performance, full-featured database with searchable encryption that provides real-world opportunities to change how business protect and share information. Key benefits to the enterprise architecture ecosystem include:

1. Cost effective elimination of IT attack surface and corresponding reduction in cyber risk
2. Highly accessible, highly available, efficiently organized, and easy to find secure information
3. Streamlined compliance with data privacy and confidentiality regulations
4. Low latency and scalable operations to securely manage exponentially growing volume and velocity of data

Enterprises spend billions of dollars on cybersecurity, yet massive data breaches continue. Businesses have accepted that their efforts at protecting the perimeter are not effective. Databases can't protect our information because they can only index and search data while it is in a vulnerable state. Neither encryption at rest nor transport layer encryption can solve this problem. The Black Forest DB™, with built-in high-performance searchable encryption, offers both performance and functionality to address these growing concerns, and provides a ground breaking – and incredibly effective -- approach to application architecture and security.

1. INTRODUCTION

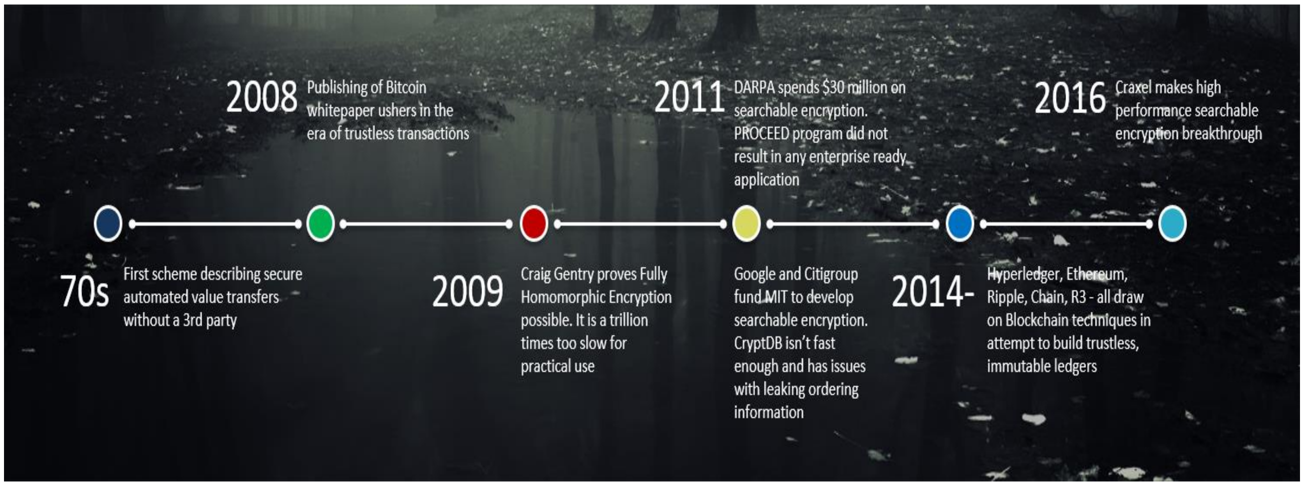

2008 Publishing of Bitcoin whitepaper ushers in the era of trustless transactions
2011 DARPA spends $30 million on searchable encryption. PROCEED program did not result in any enterprise ready application
2016 Craxel makes high performance searchable encryption breakthrough
70s First scheme describing secure automated value transfers without a 3rd party
2009 Craig Gentry proves Fully Homomorphic Encryption possible. It is a trillion times too slow for practical use
Google and Citigroup fund MIT to develop searchable encryption. CryptDB isn't fast enough and has issues with leaking ordering information
2014- Hyperledger, Ethereum, Ripple, Chain, R3 - all draw on Blockchain techniques in attempt to build trustless, immutable ledgers

Figure 3 The Path to High performance secure data management

Researchers have been studying searchable encryption for years, looking for the breakthrough that would make it practical enough to be widely adopted. The ability to search encrypted data without decrypting it and without the keys being present is widely accepted as revolutionary for data security. Unfortunately, previous methods simply were not fast enough to be practical and/or leaked too much information about the encrypted data, allowing attackers to learn sensitive details and infer information about the data.

Perhaps the most well-known of these efforts is Craig Gentry's 2009 paper proving that Fully Homomorphic Encryption was possible. Gentry's technique allowed arbitrary operations to be performed on encrypted data without decrypting it. Unfortunately, this technique was orders of magnitude too slow to be practical. While there are certainly uses for Fully Homomorphic Encryption in secure computation, it is not currently (and may never be) a practical answer for searchable encryption; search performance will always be dependent on efficient indexing. In 2011, Google, MIT, and DARPA began searching for practical searchable encryption. Ultimately, their solutions were not fast enough or had unacceptable information leakage.

In addition to Fully Homomorphic Encryption, cryptography experts often refer to a rather simple method of searchable encryption called Deterministic Encryption. A Deterministic Encryption scheme simply takes a key/value pair and encrypts the key field using something like AES-256 in electronic code book (ECB) mode, then stores this in a map like a NoSQL database. To query, the user simply encrypts the key the same way again and looks up the associated value. Although this method is much more secure than leaving NoSQL data in plaintext (AES-256 is for all practical purposes unbreakable), researchers have shown that with a-priori knowledge of the data distribution they can infer significant information about the encrypted data. Thus, the mantra in cryptography is that deterministic encryption is not secure. Additionally, using deterministic encryption for indexing and searching sorted/ordered data is even less secure than the key/value case because it leaks ordering information. Ordering information is information that can be gleaned about the underlying data without knowing the data itself (i.e. I don't know your account balance, but I can infer it is greater than $1M). This makes it easy to infer the range of a given value.

In 2016, Craxel's founders applied an innovate technique in probabilistic spatial indexing to achieve a breakthrough in high performance secure data management. This breakthrough makes it practical to organize and search strongly encrypted records without decrypting them or having the keys present on the servers. Unlike simple deterministic encryption, Craxel's proprietary technology even supports range and spatial query of encrypted records and does so with various mechanisms to prevent/mitigate order leakage attacks. This means data can be queried, stored and organized with unprecedented speed and privacy.

2. BLACK FOREST DATABASE™ SEARCHABLE ENCRYPTION

Black Forest DB™ (BFDB) is a high performance, full-featured database with searchable encryption.

"Our breakthrough high performance searchable encryption allows BFDB servers to index and efficiently query strongly encrypted data without decrypting it or having access to the encryption keys. "

How can a database server respond to a client's query without decrypting the records that need to be searched? The answer is probabilistic spatial search. Applications begin by using the BFDB Driver to insert their strongly encrypted records in BFDB servers. Every strongly encrypted record has special metadata attached to it that secretly identifies its location in a set of secret data spaces. To query the system, the client identifies the areas within the secret data spaces it is interested in and the servers simply return the records that have metadata that match the areas of the data space being searched. Only a client with the right secret key can understand the secret data spaces and how to perform database operations in those spaces.

Cryptography is highly complex; how can anyone trust this method is secure? Simple. BFDB servers don't (and can't) even look at the strongly encrypted record contents. The application encrypts the records completely independent of BFDB. This greatly simplifies validation of the security of BFDB, <u>as it is easy to validate that the BFDB servers only have strongly encrypted records and never possess the decryption keys for the records.</u> The security of the strongly encrypted records held in the BFDB servers is therefore based on the application's encryption method. Even if someone retrieved an encrypted record or hacked the server, they can't decrypt it without the right encryption key. We recommend AES-256 but applications can choose what encryption system to use. We call BFDB servers "zero-knowledge": they know nothing about the contents of the records they hold.

So, given that there are encryption systems that applications can use such as AES-256 that can't currently be brute-force attacked, the only security concern related to BFDB servers is if the metadata leaks information about the records? Yes, absolutely. Previous academic attempts at searchable encryption have used metadata that leaked ordering and other information or used deterministic encryption, which researchers have proven is not secure under certain conditions. For instance, hashing or encrypting a record the same way every time is called deterministic encryption. This allows statistical analysis to be performed to try to figure out something about the record. Some other very crude attempts have used very insecure cryptographic techniques such as substitution ciphers to create metadata for indexing and search.

What is special about Craxel's breakthrough related to metadata? Craxel's method is based on our patented probabilistic spatial search technology. We previously invented a method of efficiently indexing and searching huge numbers of shapes in a data space. This led us to our breakthrough discovery; the creation of secret data spaces that prevent/mitigate order leakage attacks but can still support range and spatial queries efficiently. In addition, patent-pending techniques that use widely adopted, proven cryptographic primitives are used in creating and generating data space permutations to ensure that an attacker can't figure out the ordering of the space without the secret key. Further, because this method is based on probabilistic spatial indexing and search, random uncertainty can be added to thwart highly unlikely, but theoretical statistical inference attacks. Since our searchable encryption method is probabilistic, the servers can return false positives to the clients that aren't really matches to the query. This also thwarts those highly unlikely, theoretical statistical inference attacks. The client-side driver works with the application to filter out the false positives, so they don't impact the application. It is important to note that although false positives are possible, our method never fails to send back ***ALL*** the actual matches.

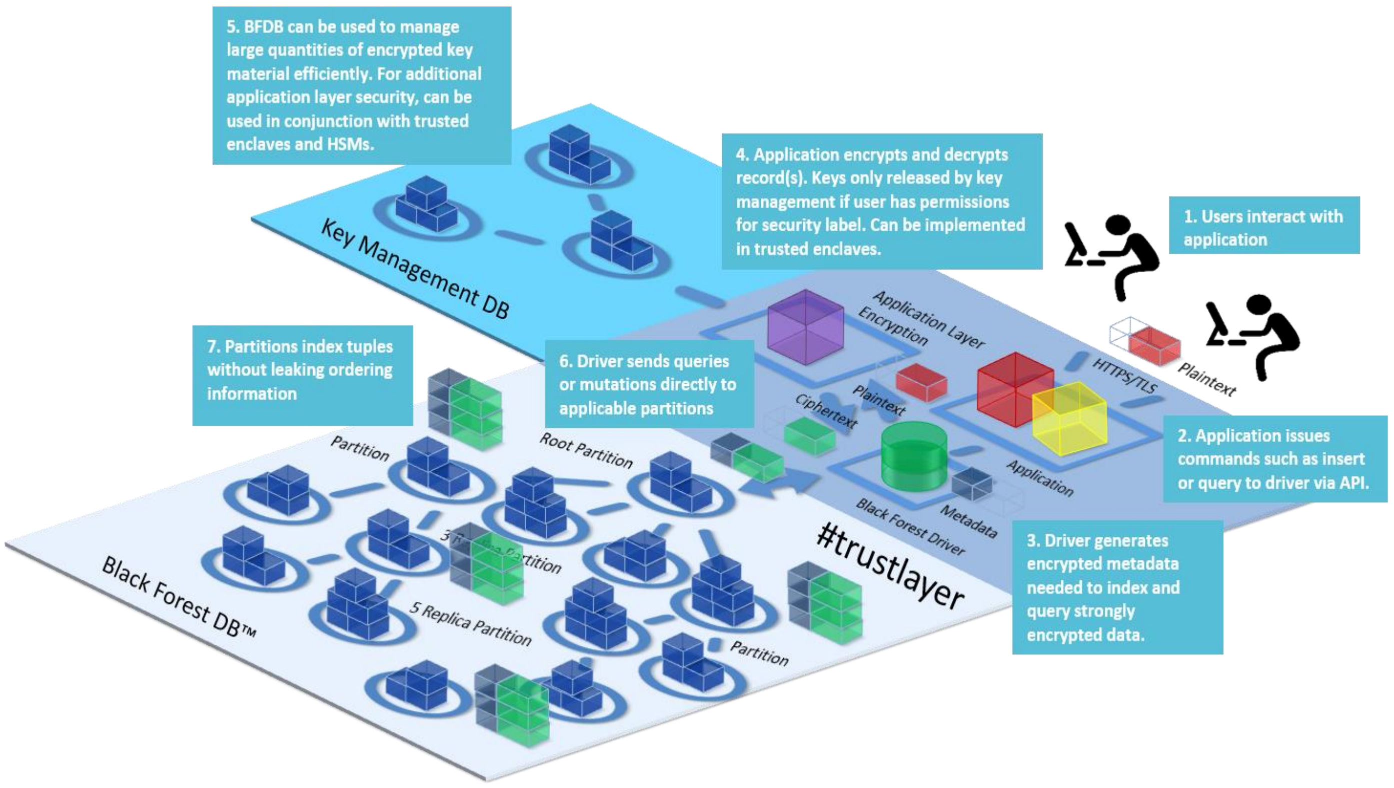
How can searchable encryption be fast enough to use in place of existing databases? Our probabilistic spatial search techniques are incredibly fast at finding the candidate results for a query and are incredibly selective in terms of keeping false positives to a minimum. For many use cases, this method is faster server-side than searching Btree indexes in relational databases. Server-side query processing consists of tree traversals and simple bit comparisons. More work must be performed on the client-side to generate metadata, encrypt and decrypt records, but it is not that demanding. Our architecture scales better than typical databases as it is possible to have many clients doing that work instead of a smaller set of servers.

If this is secure, fast, scalable, and functional, then the 99.999% of time information is in the data layer, it can be safe, yet instantly and efficiently accessible? That's right. Information can now be securely organized and efficiently accessed, without it ever being vulnerable in the data layer. The only way to decrypt the encrypted records is to have the right encryption keys. We believe that this breakthrough will fundamentally change how the world organizes information.

3. APPLICATION ARCHITECTURE

An instance of Black Forest DB™ consists of a cluster of one or more servers that are accessed by an application through the BFDB Driver. Figure 4**Error! Reference source not found.** shows the architecture of an application using Black Forest Database™.

The BFDB Driver is a software library accessed directly by an application. Mutations such as inserts, updates, and deletes are presented to the BFDB Driver by the application. The BFDB Driver sends the tuples and associated security labels through an application-provided callback for transformation or to get the keys to use to encrypt or decrypt the tuples prior to sending a mutation or transaction to the BFDB servers. This is called application layer encryption and makes it trivial to validate most of the security aspects of BFDB. The BFDB Driver also creates the encrypted metadata that is used to index and search the tuples on the BFDB servers.


5. BFDB can be used to manage large quantities of encrypted key material efficiently. For additional application layer security, can be used in conjunction with trusted enclaves and HSMs.

4. Application encrypts and decrypts record(s). Keys only released by key management if user has permissions for security label. Can be implemented in trusted enclaves.

1. Users interact with application

7. Partitions index tuples without leaking ordering information

6. Driver sends queries or mutations directly to applicable partitions

2. Application issues commands such as insert or query to driver via API.

3. Driver generates encrypted metadata needed to index and query strongly encrypted data.

Figure 4 BFDB™ Example Application Architecture

For queries, the application passes the query to the BFDB Driver. The BFDB Driver sends the query parameters to the application-provided callback for transformation and/or to get the keys needed to encrypt the query. The BFDB Driver then creates encrypted query metadata for search. The encrypted query is sent to BFDB servers and a candidate result set is returned. The query result set and their associated security labels are sent to the application callback to be transformed or to get the keys required to decrypt the results. Search is probabilistic to the extent that the query result can contain false positives. BFDB servers do not and cannot know for certain if a tuple matches a query. Query results are validated by the BFDB driver against the query parameters after decryption so that applications never receive these false positives. There are other searchable encryption schemes that return false positives. A key to our performance breakthrough is our ability to minimize the number of false positives returned with each query.

Applications can manage their own encryption keys. Since the application layer performs the encryption and decryption and has possession of the encryption keys, it is trivial to prove that the security of the tuples themselves stored in a BFDB server is equivalent to the encryption method used on the client. Security of BFDB is reduced to the tuple encryption methods chosen and the security characteristics of the metadata encryption scheme.

4. PERVASIVE COMPARTMENTALIZATION

Mandatory Access Control

Every tuple has a security label. BFDB only allows users with permission to access a given security compartment to retrieve the encrypted tuples.

Graph Data Model

Each tuple has a subject, predicate, and object. Applications can create rich directed graphs of information about an object. Since each tuple has its own security label, the graph can be compartmentalized so that only authorized users can access a given node in the graph.

Application Encryption

Since the application performs the encryption and decryption of the tuples themselves, the application can choose which encryption keys to use for each tuple. If the application doesn't have access to the key for a security label, it can't decrypt and access the information.

Key Release for Policy Enforcement

Application layer encryption enables a very powerful, compartmentalized information architecture that enables fine-grain access control. "Least privilege" and "need to know" can be controlled at the database server AND by controlling access to encryption keys. Access to information can be granted by allowing access to encryption keys for a compartment.

BFDB enforces access control using digitally signed user assertions and security labels. All data on BFDB remains encrypted, and only valid parties with the right encryption keys can access the encrypted records. Secure granularity and pervasive compartmentalization at scale are key requirements for a range of use cases including identity management, GDPR compliance, and supply chain track and trace. Application layer encryption enables a very powerful, compartmentalized information architecture that enables fine-grain access control. "Least privilege" and "need to know" can be managed by controlling access to encryption keys. Access to information can be granted on a granular level by allowing access to encryption keys for a specific transaction.

5. BLACK FOREST DATABASE™ KEY FEATURES

Black Forest Database with high performance searchable encryption makes data both safe and useful. As an enterprise offering, BFDB include a rich set of features. Figure 5 provides a summary of BFDB's features.

High Performance Searchable Encryption

Black Forest DB™ servers index and search strongly encrypted records without decrypting them or ever having the encryption keys present on the server.

Massively Parallel Consensus

Nodes can be distributed over any number of servers. Data operations and queries are performed in parallel. This includes spatial, temporal, and semantic search operations. BFDB's novel data partitioning method supports incredibly fast parallel queries and transactions.

Multiple Query Types

Craxel's breakthrough in searchable encryption allows BFDB to supports a variety of query types including range, spatial, and temporal queries without leaking ordering information.

Schema-less

Black Forest DB™ is schema-less, providing flexibility and extensibility to developers. This extremely flexible and powerful data model supports very simple key/value data as well as very sophisticated semantic relationships.

Partitioning

Black Forest DB™ utilizes a novel approach to partitioning data that doesn't sacrifice speed or data integrity. Data partitions can be created, removed, or replicated amongst multiple servers seamlessly and on the fly.

ACID Properties

Black Forest DB™ transactions are atomic, consistent, isolated, and durable.

Fine-grain Access Control

Black Forest DB™ enforces access control using digitally signed user assertions and security labels. Although BFDB has zero knowledge of the contents of any tuple, only valid parties have access to even encrypted data.

Security Labels

Data tuples stored within Black Forest DB™ can have application assigned security labels, which add an additional layer of protection and control. Applications can manage their encryption keys based on the security level, compartment and group they assign to their data tuples.

Figure 5 Black Forest DB™ Feature Summary

BFDB scales horizontally across many commodity servers, providing high availability through consensus-based replication. BFDB is not a NoSQL or relational database. BFDB's data model is schema-less and supports inserting, updating, removing, and querying records in the form of N-tuples. This powerful model can be thought of as a directed acyclic graph (DAG), much like those used by graph databases. BFDB provides strongly consistent, ACID transactions. Due to BFDB's unique parallel architecture, these transactions can be performed at very high rates.

In addition to searchable encryption, BFDB provides a rich, robust security model based on every tuple having a security label. BFDB enforces mandatory access control using security labels and digitally signed user assertions. The combination of BFDB security labels and application-layer encryption allow enterprises to easily implement cryptographic compartmentalization or segmentation of their valuable information.

Security and Privacy

BFDB has no knowledge of the encrypted data within it. Since the database server never has the keys, the data can't be stolen and decrypted without stealing the keys from somewhere else.

Pervasive Compartmentalization

BFDB enables pervasive compartmentalization protected by security labels, trusted user assertions, and key release-based policy enforcement.

Performance

BFDB provides unprecedented performance through massively parallel consensus. BFDB can support hundreds of thousands to millions of operations per second. Mutations are measured in milliseconds and queries are performed in parallel.

Scale

BFDB can support thousands of partitions, each with their own replica group. Each partition supports many thousands of operations per second. Replication and synchronization after network faults are highly efficient.

6. PERFORMANCE

Black Forest DB™ is an extremely high-performance database due to its novel and unique indexing technology. BFDB has been tested on a cluster of 33 Amazon AWS instances and achieved insert performance of 8 million inserts per second and hundreds of thousands of queries per second. Query performance is dependent on the type of query and distribution of data. A single BFDB instance on a single server can handle thousands of queries per second for both tuple, spatial, and temporal queries. Performance can be scaled by adding more database instances to the cluster.

7. MASSIVE HORIZONTAL SCALE

BFDB is a distributed, multi-master transactional database that can be partitioned to achieve massive scale. An instance of BFDB consists of one or more servers that are operating within a cluster. The partitions within BFDB are arranged as a tree structure.

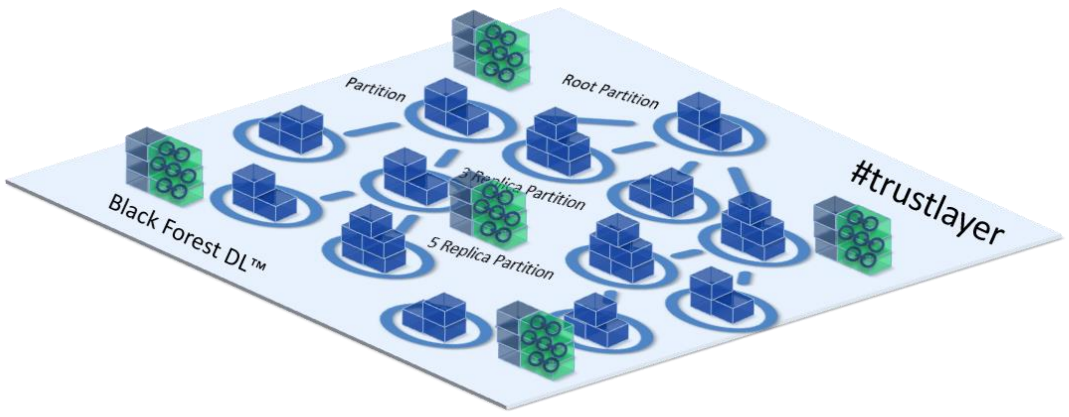


Figure 6 BFDB Partitioning

Figure 6 shows BFDB's partitioning scheme. Partitions are essentially branches of a tree structure. Each partition has a set of replicas. Those replicas use a consensus algorithm to provide serialized ordering of transactions within each tree node. BFDB provides multi-master replication of each partition. BFDB treats each partition as N disjoint sub-partitions which allows for unprecedented scale and replication performance. BFDB uses a highly parallelized Paxos implementation for achieving consensus between the replicas of each sub-partition.

Tables are overlaid, and their contents indexed using this single, sparse, fixed tree structure. Each BFDB cluster includes one or more root servers. The root servers maintain the partitioning, table, and user metadata. Root servers also operate as full members of the cluster, servicing their branch of the tree. Unlike distributed hash tables used by NoSQL databases, BFDB's tree structure elegantly provides an ordered index that efficiently supports range and spatial queries. Queries that span multiple branches are performed in parallel. Although BFDB's tree structure is ordered, it employs various mechanisms to prevent/mitigate order leakage attacks when indexing encrypted metadata.

8. CONSENSUS AND REPLICATION

Multiple servers can be assigned to any partition. For any mutation to be performed, a majority of servers must agree. BFDB uses the Paxos consensus algorithm and can run an extremely large number of Paxos sessions in parallel. Servers can dynamically join or leave a partition based on administrator action. Data is synchronized to ensure that a majority of servers always have the latest data for the partition. Servers that have been disconnected resync upon reconnection.

Synchronization upon reconnection after network failure is a significant issue for replicated systems. In most databases, log shipping is used to replay any missing transactions. BFDB utilizes log shipping for short outages but also implements an extremely efficient synchronization algorithm that can quickly determine differences between any two replicas.

9. STORAGE

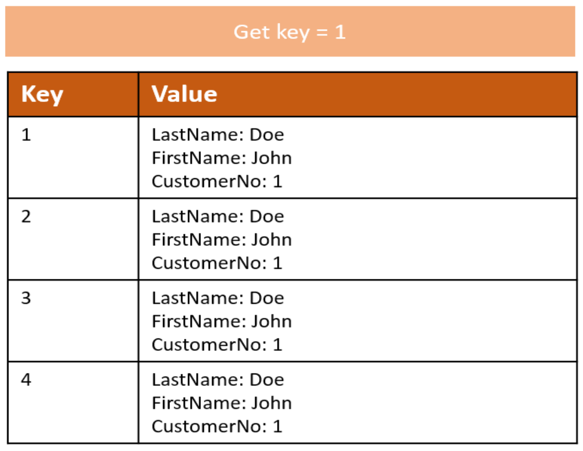
Black Forest DB storage consists of a write-ahead log storage subsystem and a page storage subsystem. Each subsystem is append-only for writes. Each subsystem consists of several segmented files. Files that no longer contain enough active data are garbage collected. BFDB has an efficient log garbage collection scheme that can move any remaining active log or page data from older file segments so that they can be collected. The active data is appended to the current file segment. Once a mutation has been persisted into page storage, the Paxos acceptances and commits for the mutation kept in the write-ahead log are no longer needed. Unlike other databases, BFDB can dynamically minimize the size of the write-ahead log, saving storage space and ensuring rapid recovery if a server must restart.

Data pages are variable length, reducing the wasted page overhead seen in other databases. BFDB can do this because of its append-only storage. BFDB can partition its data within a server onto multiple log and data disks, elegantly providing substantial storage IO bandwidth without expensive storage solutions. SSD storage is a good match for BFDB due to its append-only operation.

10. DATA MODEL

Black Forest DB™ has a completely different architecture than relational databases. Relational databases keep information normalized in multiple tables, each with multiple columns. Specified columns are indexed for search. Tables and columns are specified upfront in a schema. While powerful data relations can be modeled in relational databases, the inability to easily change the schema on the fly without serious side effects is a major problem.

NoSQL databases can be categorized as key-value databases, document databases, wide-column databases, and graph databases.



Get key = 1

Key	Value
1	LastName: Doe FirstName: John CustomerNo: 1
2	LastName: Doe FirstName: John CustomerNo: 1
3	LastName: Doe FirstName: John CustomerNo: 1
4	LastName: Doe FirstName: John CustomerNo: 1

Figure 7 Key/Value Data Model

Key-value databases have very simple data models. Data is accessed and stored with a get and a put command. Some key-value databases have added additional features beyond get/put since most applications require more sophisticated operations. Since the data model of key-value databases is so simple, no schema is required. A downside of key-value data models is that they are very ad-hoc and typically one-off implementations. They also require additional indexing features to be performant for range and other complex queries.

Document databases are key-value databases that represent values as semi-structured data instead of opaque values. These databases typically index the semi-structured data for search, such as MongoDB does with JSON data. Wide-column databases have been derived from Google's BigTable paper. Wide-column databases support the addition of columns for a given row instead of all the columns being predefined. A column is a key-value pair. One way to think of wide-column databases is that a row is a key-value pair whose value is a nested set of key-value pairs (columns).

Graph databases store their data in the form of a directed labeled graph. Resource Description Framework (RDF) stores are the most common. Given the sample tuples and the given query, the result would be all the tuples where the Subject = 'JohnDoe', providing John Doe's SSN, Birth Date, and Spouse in a single query. This is the data model used by the semantic web.

```
SELECT* FROM Customers WHERE Subject='JohnDoe' and Predicate = '*'
```

Metadata	Subject	Predicate	Object	Security Label
	JohnDoe	SSN	000-00-0001	TS/1
	JaneDoe	SSN	000-00-0002	TS/2
	JackDoe	SSN	000-00-0003	TS/3
	JohnDoe	BirthDate	1/1/1980	TS/1
	JohnDoe	Spouse	JaneDoe	TS/1

Figure 8 Graph Data Model Example

Black Forest DB has a very powerful, schema-less data model. BFDB's data model allows the creation of graphs of information within the database. It is very similar to RDF and semantic triples. Records in BFDB are called tuples. They are in the form:

(subject, predicate, object, isLiteral, security label)

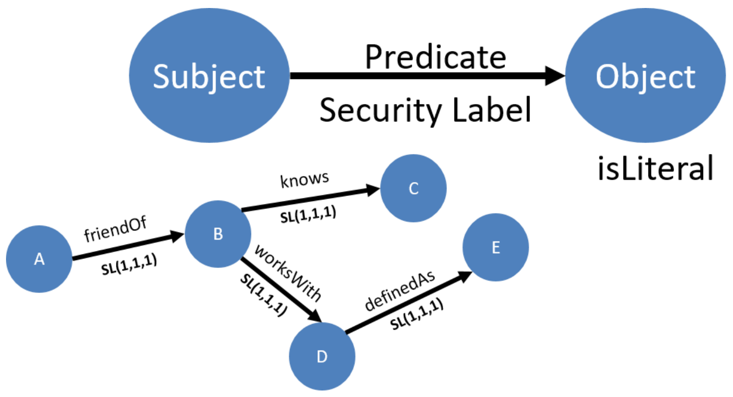


Figure 9 Graph of Tuples

Figure 9 shows how a graph of tuples can be created using a tuple. Subjects, predicate, and objects are arbitrary byte arrays. Applications can query by subject, subject and predicate, or by subject, predicate, and object.

Data tuples stored in Black Forest DB can be thought of as a directed labeled graph wherein nodes are subjects, edges are labeled with predicates, and objects are either other subjects or literal values. An object can be any arbitrary sequence of bytes. For instance, an object could store a JSON object, an AVRO serialized object, or the bytes representing an image. Black Forest DB data tuples also have a security label, which restricts access to each data tuple. Users cannot traverse an edge in the graph if they do not have permission to access that security label. Black Forest DB provides a simple and uniform data model. While NoSQL databases have ad-hoc data models, Black Forest DB has a well-specified data model but remains schema-less.

Black Forest DB supports different data types for tuple subjects. Tuple subjects can be simple binary strings, semantic subjects, spatial, temporal, alphanumeric, lexigraphic, numeric, or combined data types. For instance, a subject can be a spatial location or shape such as a polygon. A subject can be a specific time or time period. A subject can even be a value with uncertainty, such as a probability range. A subject can also be a combination type such as an identifier and a time period. Combined data types can be effectively used for significant performance gains for time series queries about an object. The reason Black Forest DB can so easily support such a wide variety of data types is that they can all can be mapped into a data space and then treated as spatial data. Probabilistic spatial search allows this to be very fast, efficient, and scalable.

11. COMPARISON WITH OTHER DATABASE TECHNOLOGIES

Figure 10 shows a comparison of properties between BFDB, relational, and NoSQL databases. BFDB has substantial advantages for most use cases while at the same time using high performance searchable encryption to deliver unprecedented security without sacrificing performance and functionality.

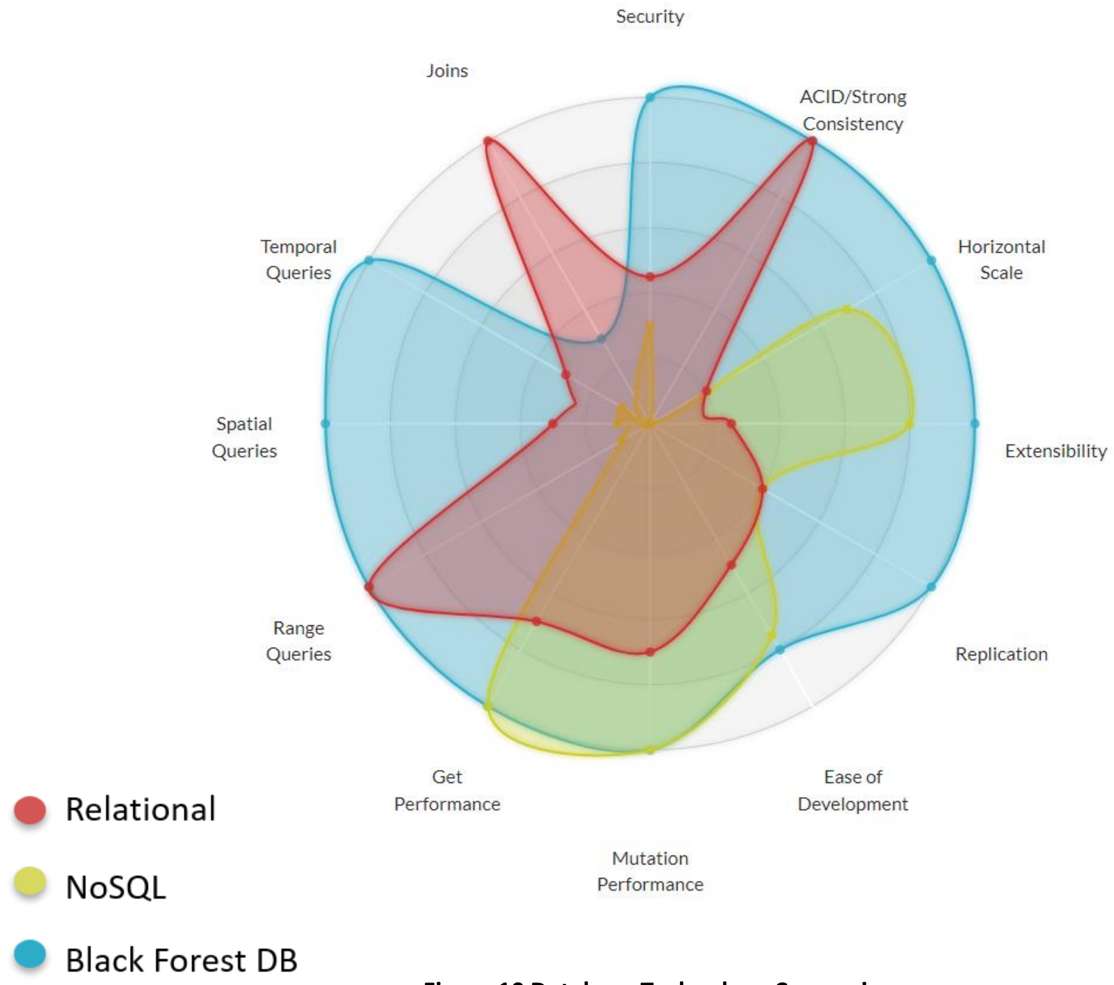


Figure 10 Database Technology Comparison

12. CONCLUSION

Databases are generally not optimized for protecting information. This is proven again and again in the headlines every day. Once a hacker or insider threat gets inside a database server they can steal everything, even if the database employs encryption features. Databases typically must index on plaintext and a hacker or insider can easily access this data. Even when databases do provide encryption at rest or column encryption features, they typically require many complex configurations, are hard to manage, and are prohibitively expensive. When a database does encrypt columns, a hacker or insider simply has to find the encryption keys inside that database server, exfiltrate the keys, and then decrypt the data and steal it. Instead of stealing the data, they could also surreptitiously change the data, leading to data integrity issues.

This problem is compounded through the increasing use of cloud infrastructures to house critical databases. While cloud providers are working hard to provide better security, the complexity of the attack surface means that customers will never be able to fully trust the security of their information in the cloud. There are varying estimates about the cost of data theft, but the reality is that the financial, legal, and reputational consequences of not protecting data are extremely high.

Although many firms have expensive security processes and controls in place, they have little confidence that their data is safe. Using today's databases to protect sensitive information is unacceptably risky. Information spends 99.9% of its life sitting in a database waiting to be accessed. Can you imagine the impact of eliminating any threat to information during 99.9% of its life? We believe Craxel can bring the performance and functionality needed to replace nearly everything today's databases can do, with the privacy and security of practical searchable encryption.

Black Forest Digital Trust Platform™ delivers the enterprise trust layer that drives digital transformation. Our breakthroughs in high performance secure data management safeguard your information while making it usable, valuable, and quickly accessible; relying solely on cryptography for trust.