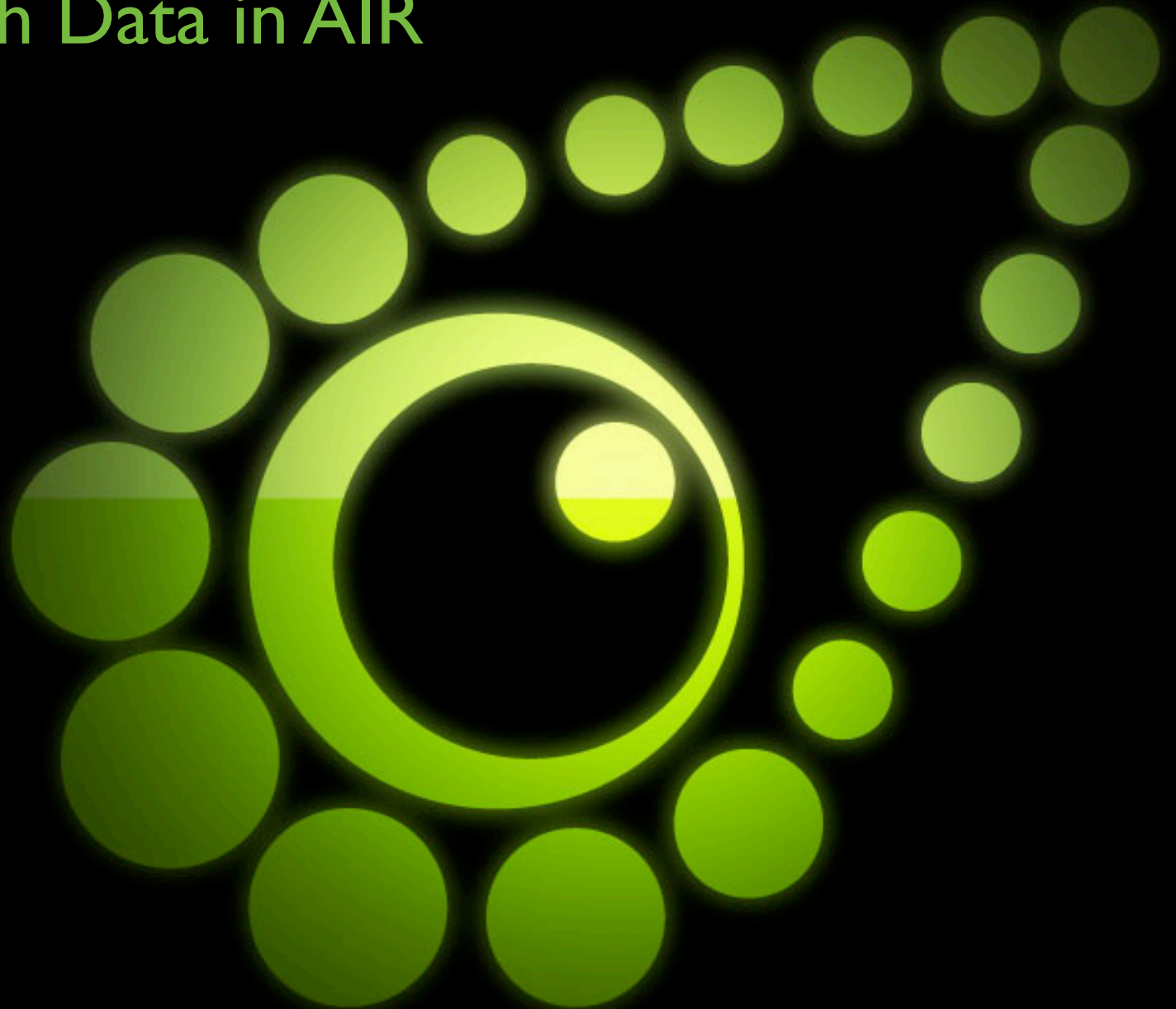


# ADOBE AIR

## Working with Data in AIR



David Tucker

# Who am I

- Software Engineer II, Universal Mind
- Adobe Community Expert
- Lead Author, *Adobe AIR 1.5 Cookbook*
- Podcaster, *Weekly RIA RoundUp* at InsideRIA
- Author, *AIR for Flash Developers* at Lynda.com



COMMUNITY  
EXPERT

# How I Will Be Approaching Topics

- Explaining the background for a **beginner**
- Showing **intermediate** level examples
- Providing **advanced** tips for production applications

# Agenda: AIR Data

# Agenda: AIR Data

- Encrypted Local Store

# Agenda: AIR Data

- Encrypted Local Store
- Local File System

# Agenda: AIR Data

- Encrypted Local Store
- Local File System
- Embedded SQLite Database

# Agenda: AIR Data

- Encrypted Local Store
- Local File System
- Embedded SQLite Database
- Online/Offline Sync with LCDS (time permitting)

# Encrypted Local Store

The **encrypted local store** provides secure storage for binary data which can be retrieved via a string key. It utilizes the native operating system encryption store (such as Keychain on the Mac or DPAPI on Windows) to secure data with 128-bit encryption. Data in the Encrypted Local Store is accessible only from your AIR application.

# API Dictionary

flash.data.EncryptedLocalStore - The class that handles all interaction with the Encrypted Local Store has four static methods for working with its data

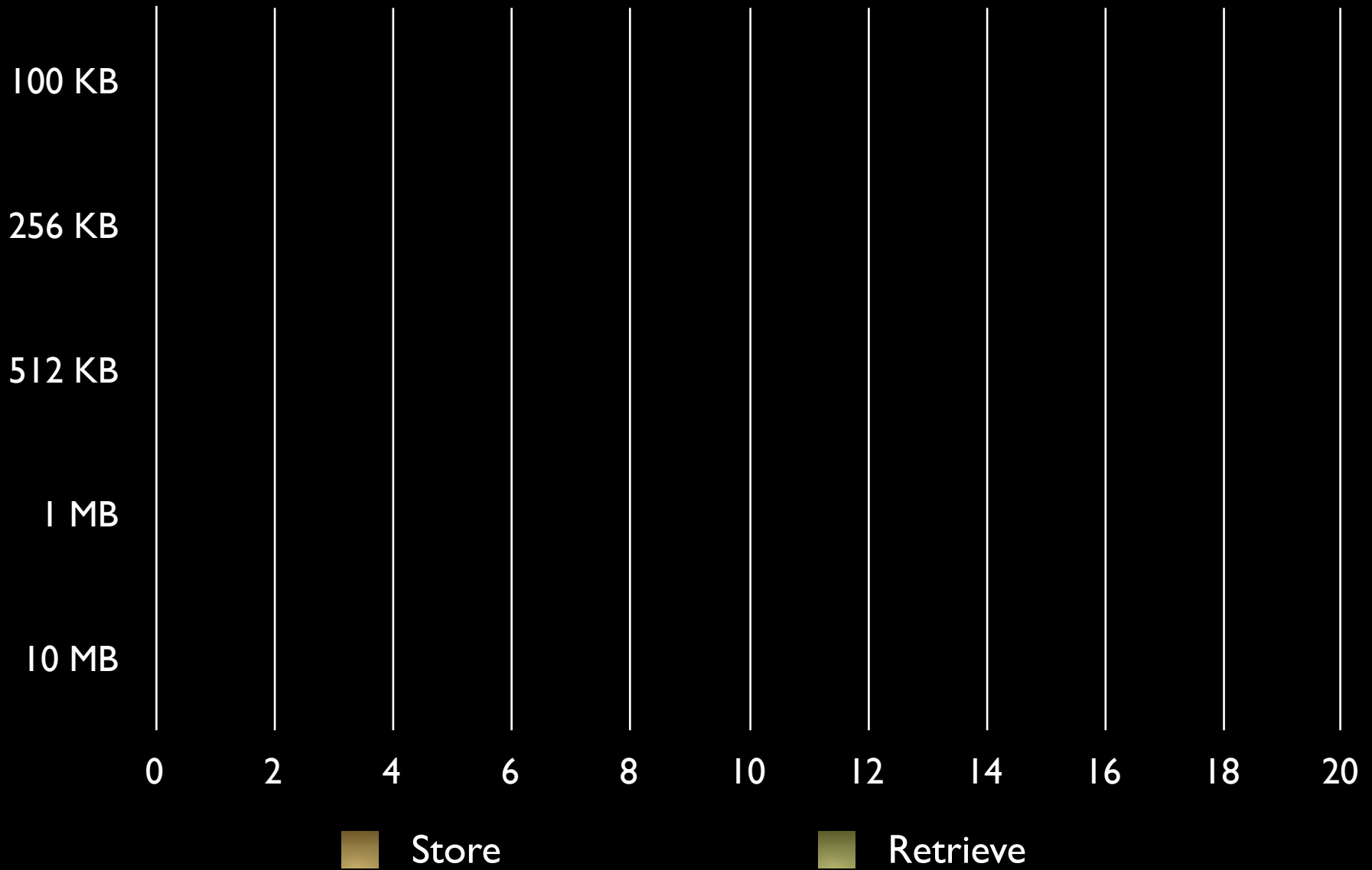
- **getItem**(name:String):ByteArray
- **removeItem**(name:String):void
- **reset**():void
- **setItem**(name:String, data:ByteArray, stronglyBound:Boolean = false):void

# Code Sample: Encrypted Local Store

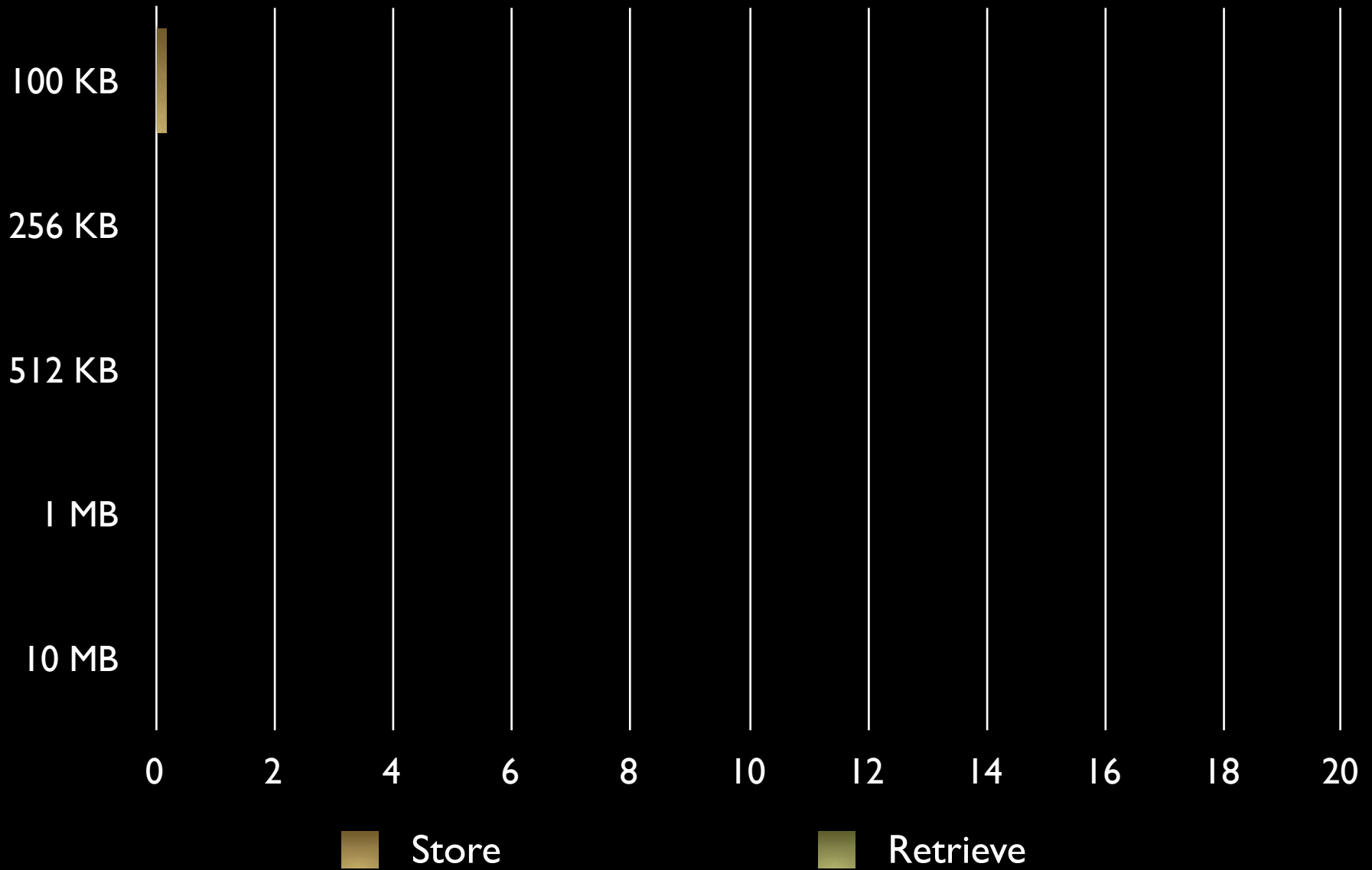
■ Store

■ Retrieve

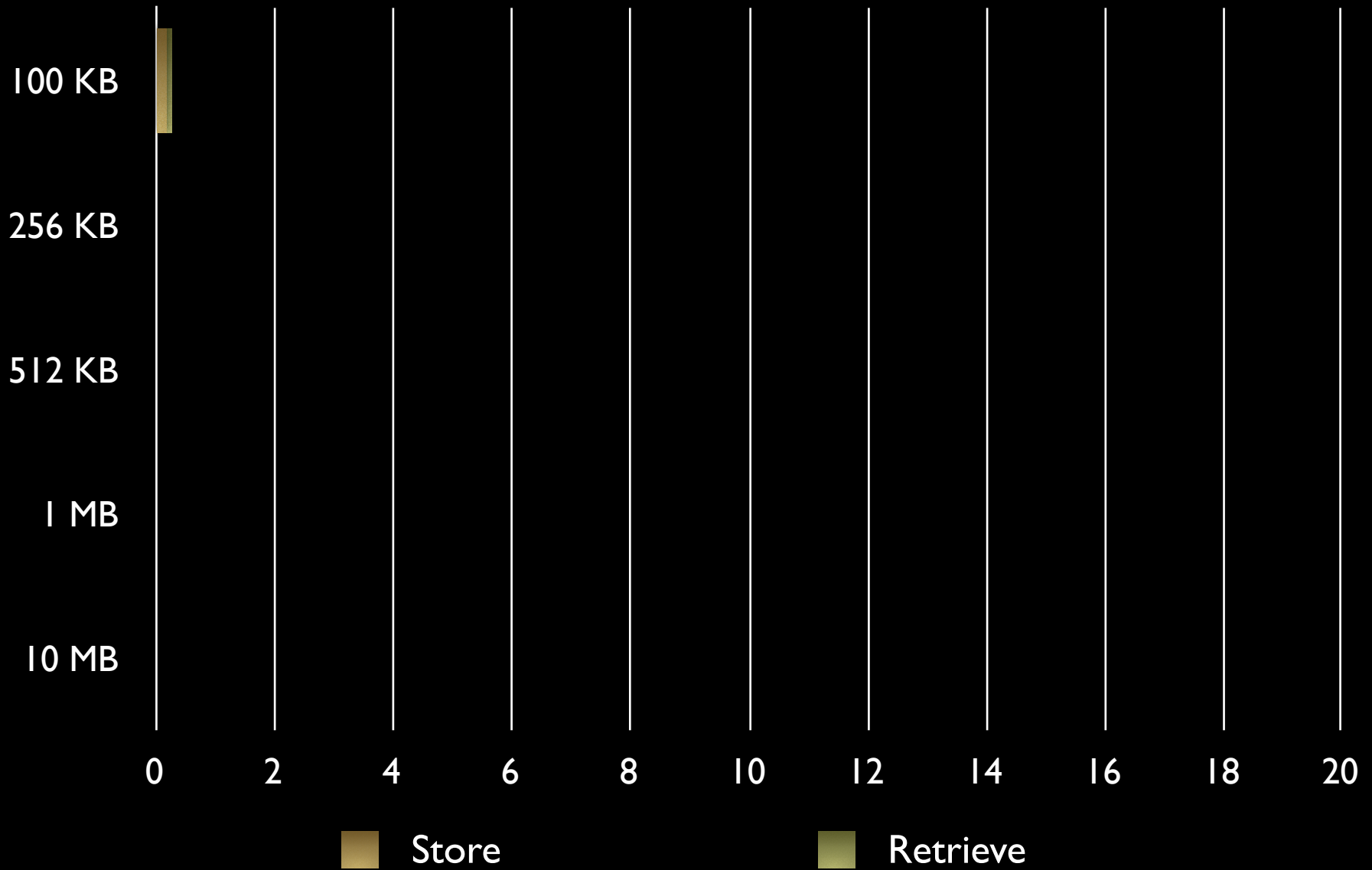
# Encrypted Local Store Performance Analysis



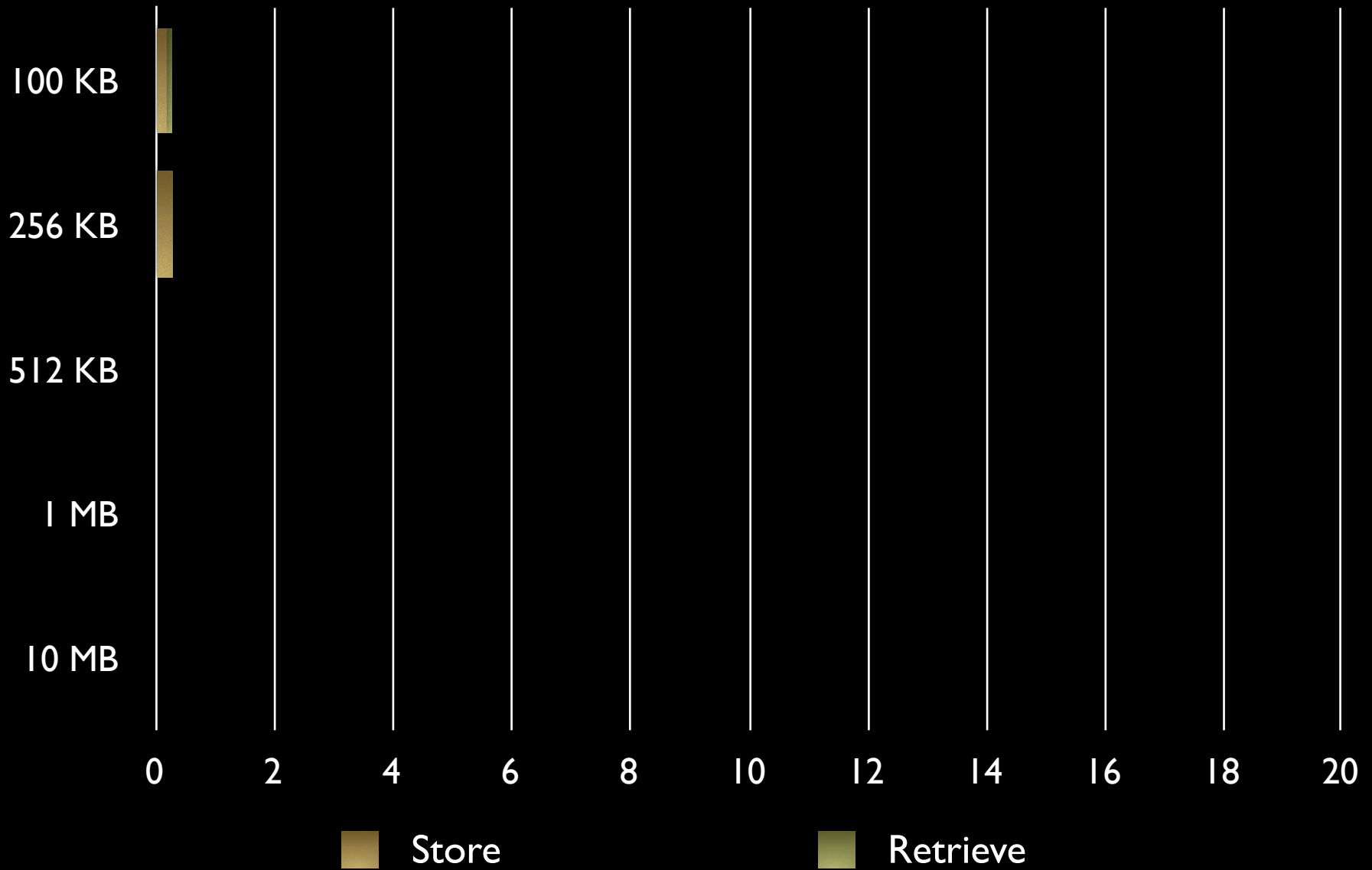
# Encrypted Local Store Performance Analysis



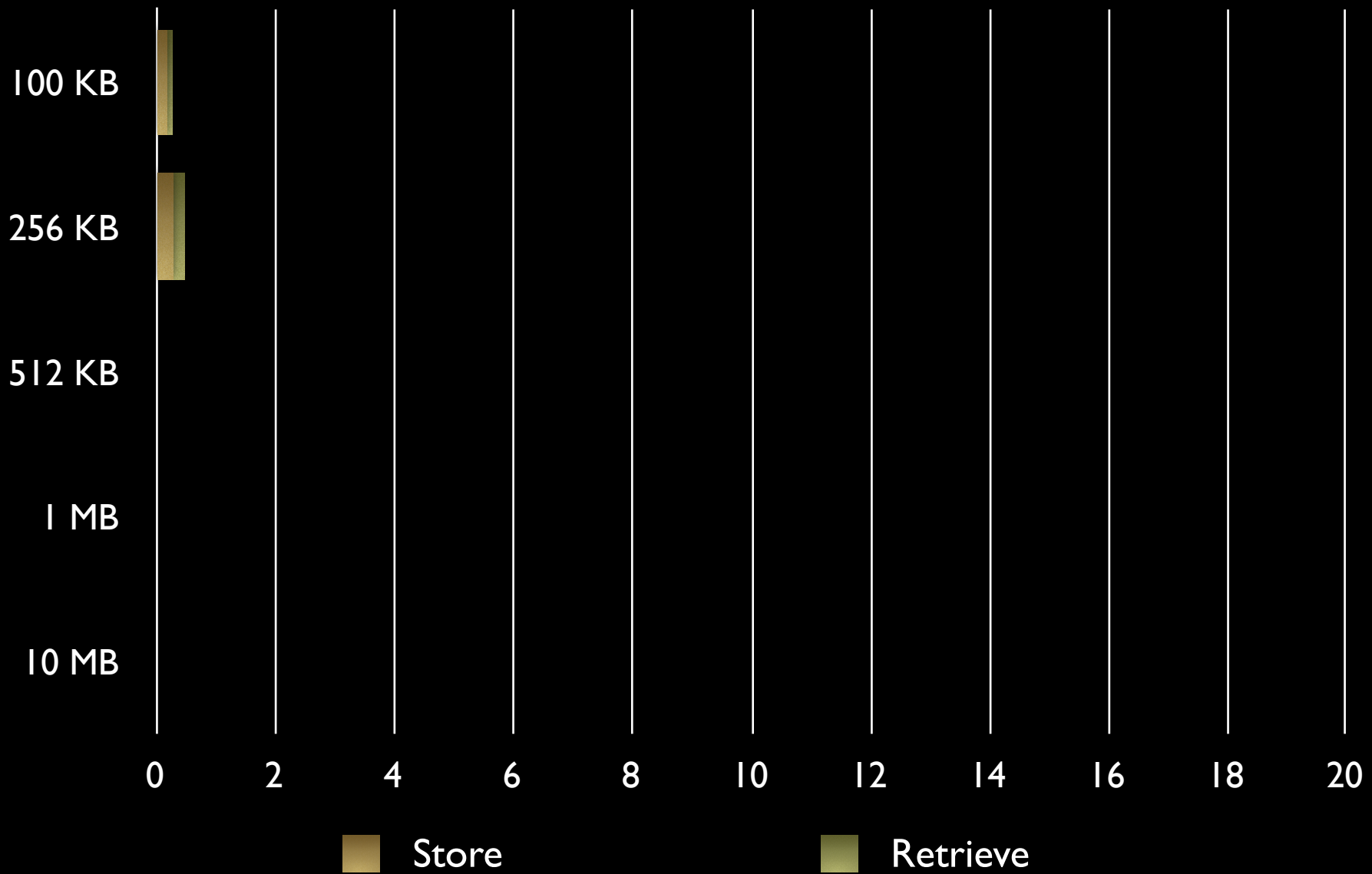
# Encrypted Local Store Performance Analysis



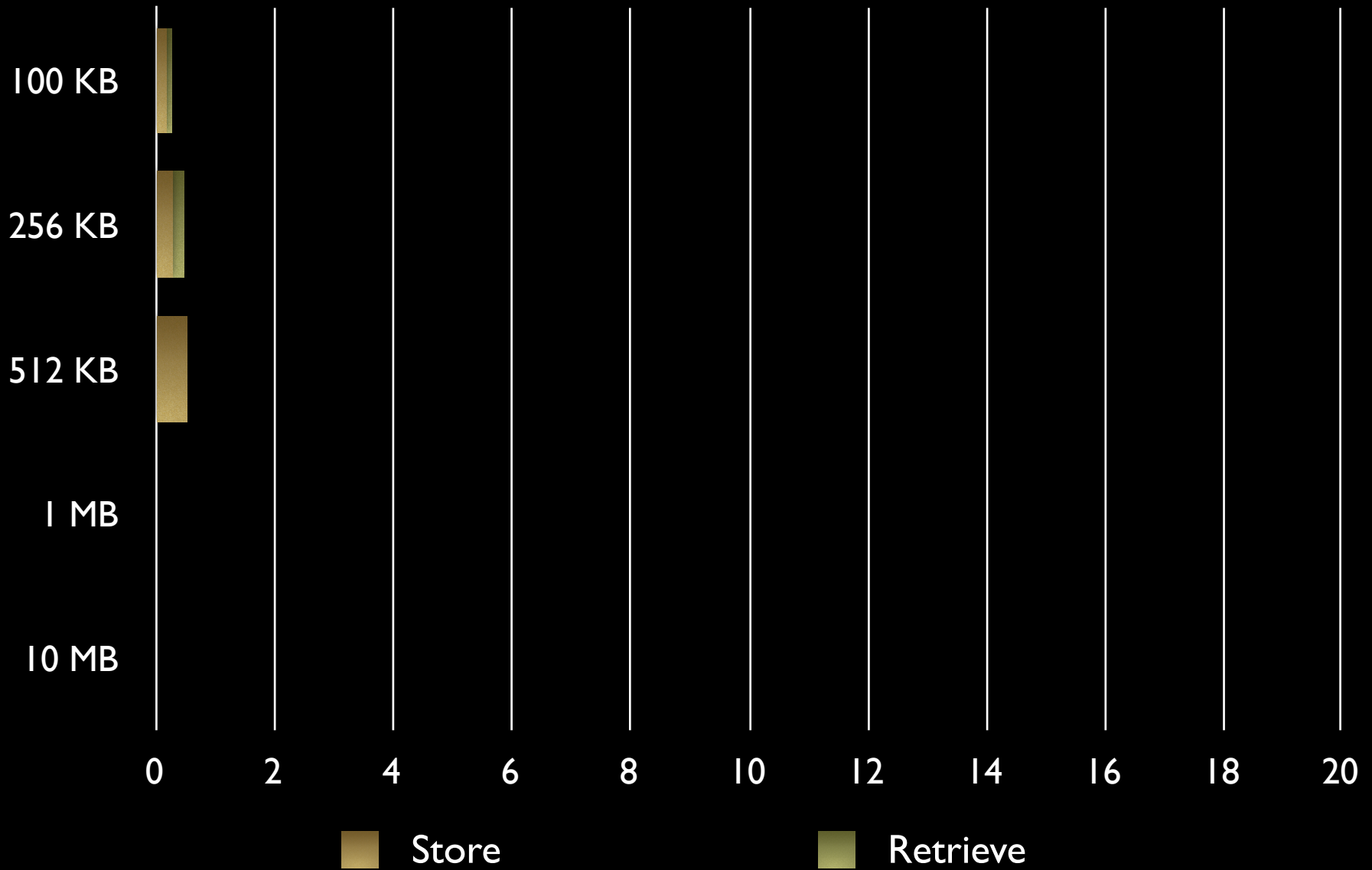
# Encrypted Local Store Performance Analysis



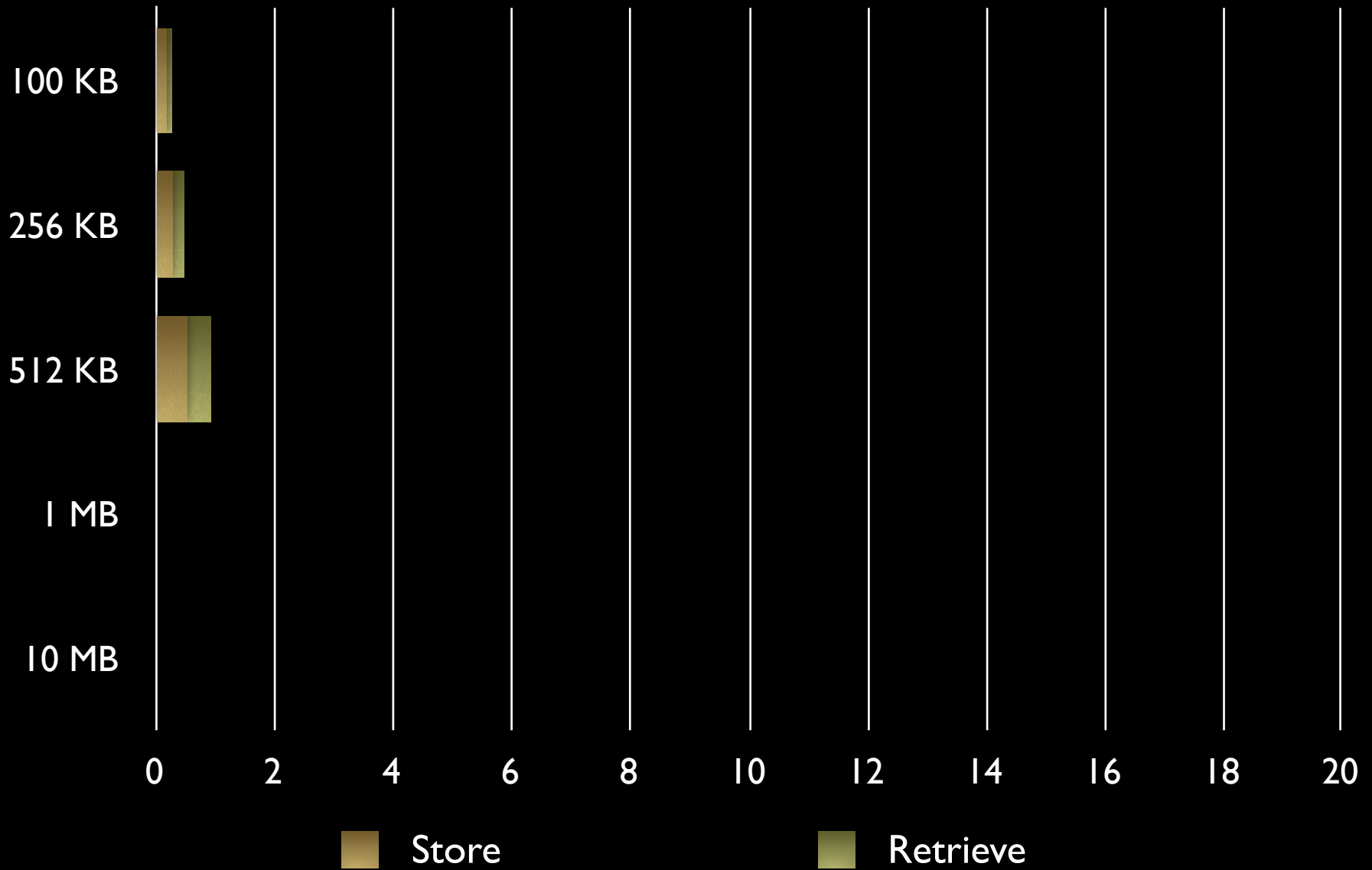
# Encrypted Local Store Performance Analysis



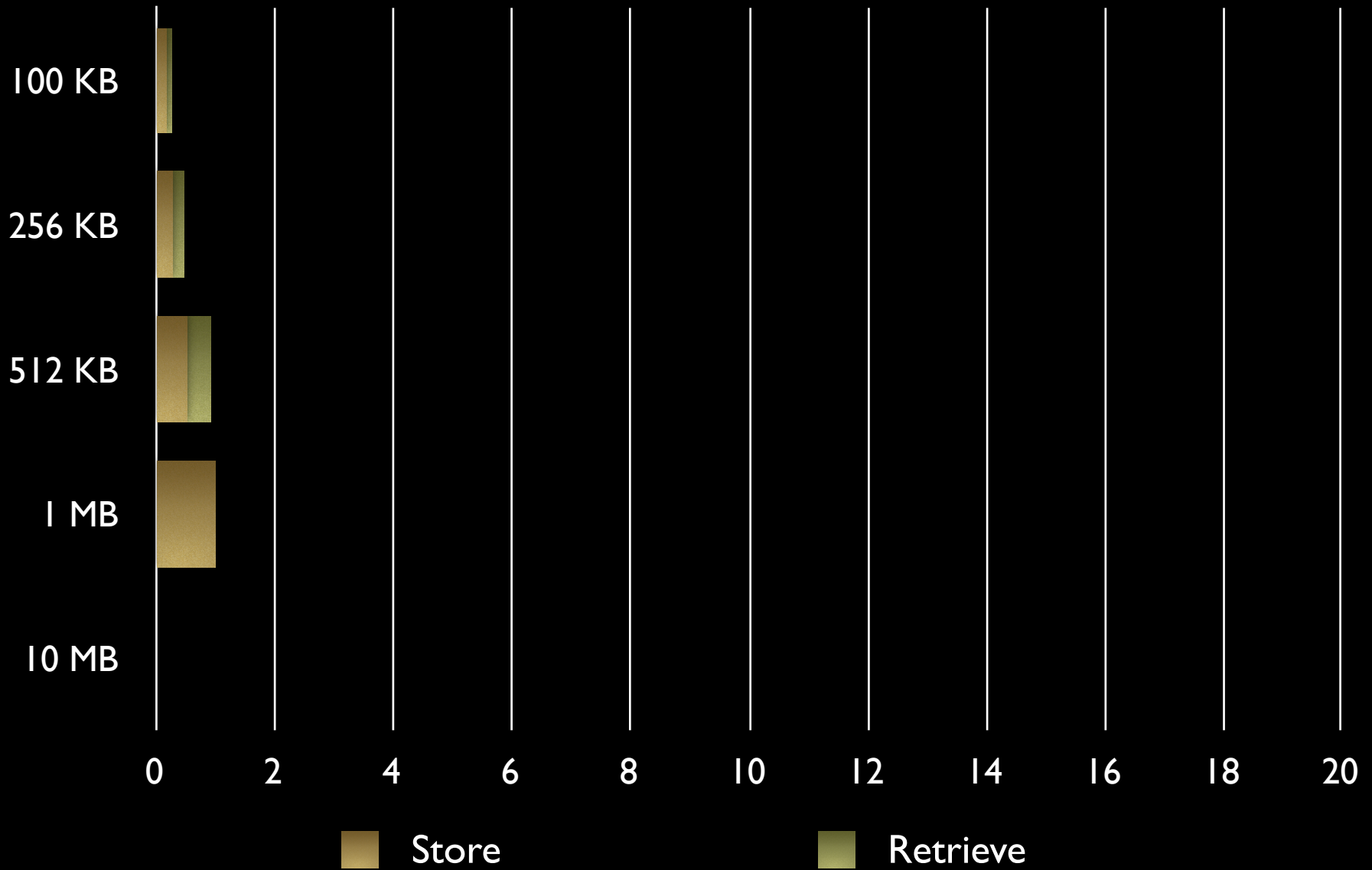
# Encrypted Local Store Performance Analysis



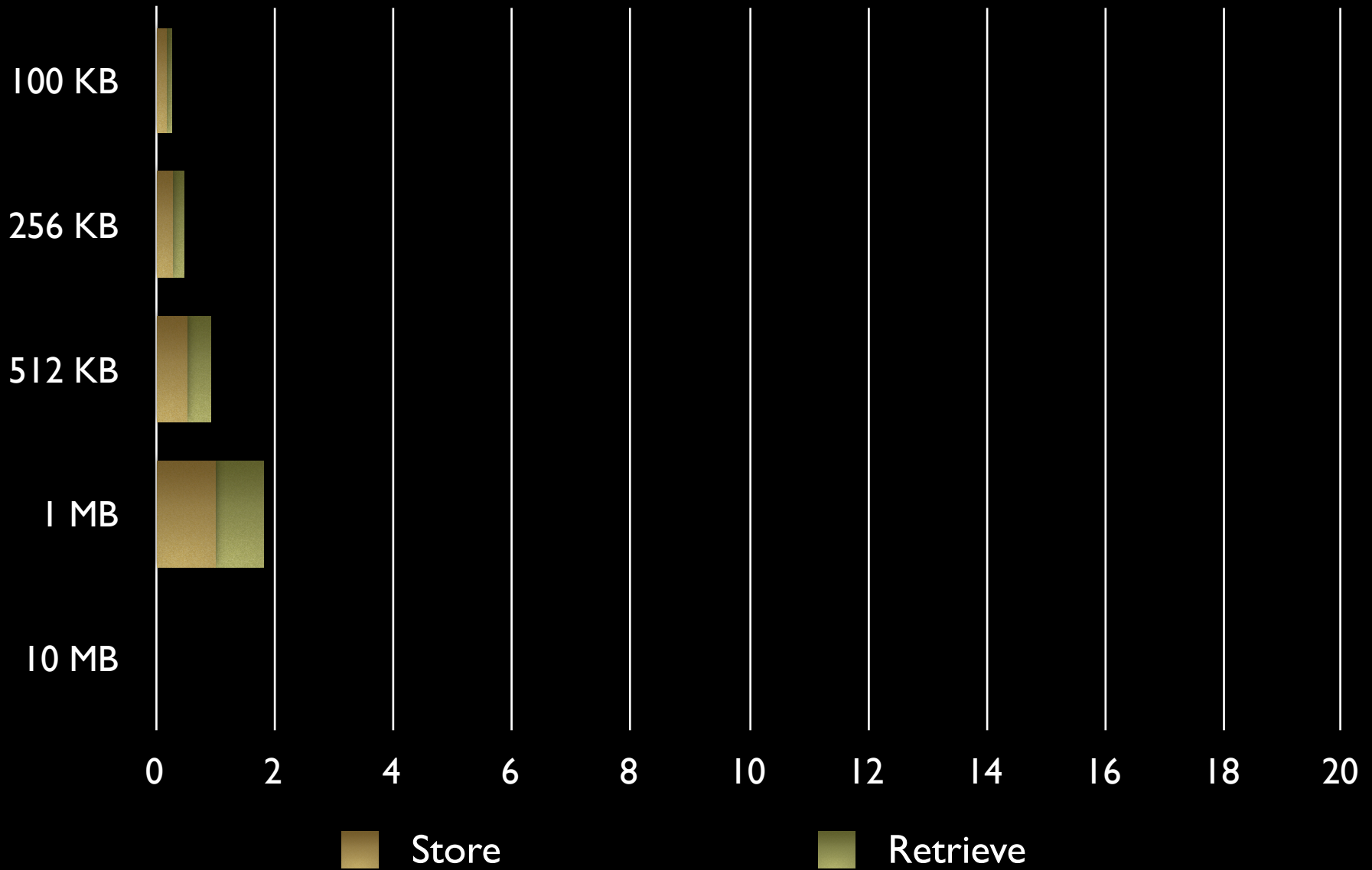
# Encrypted Local Store Performance Analysis



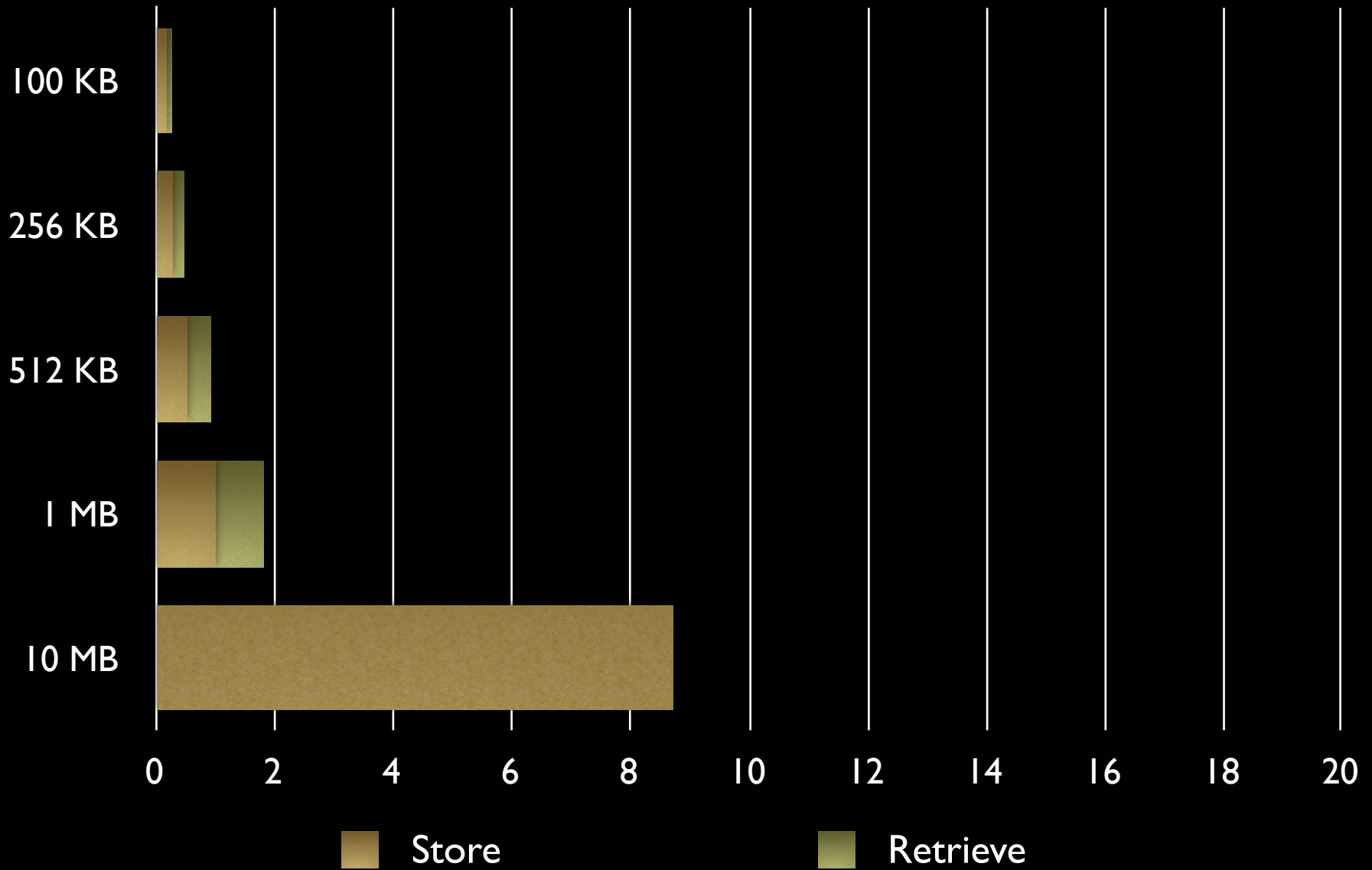
# Encrypted Local Store Performance Analysis



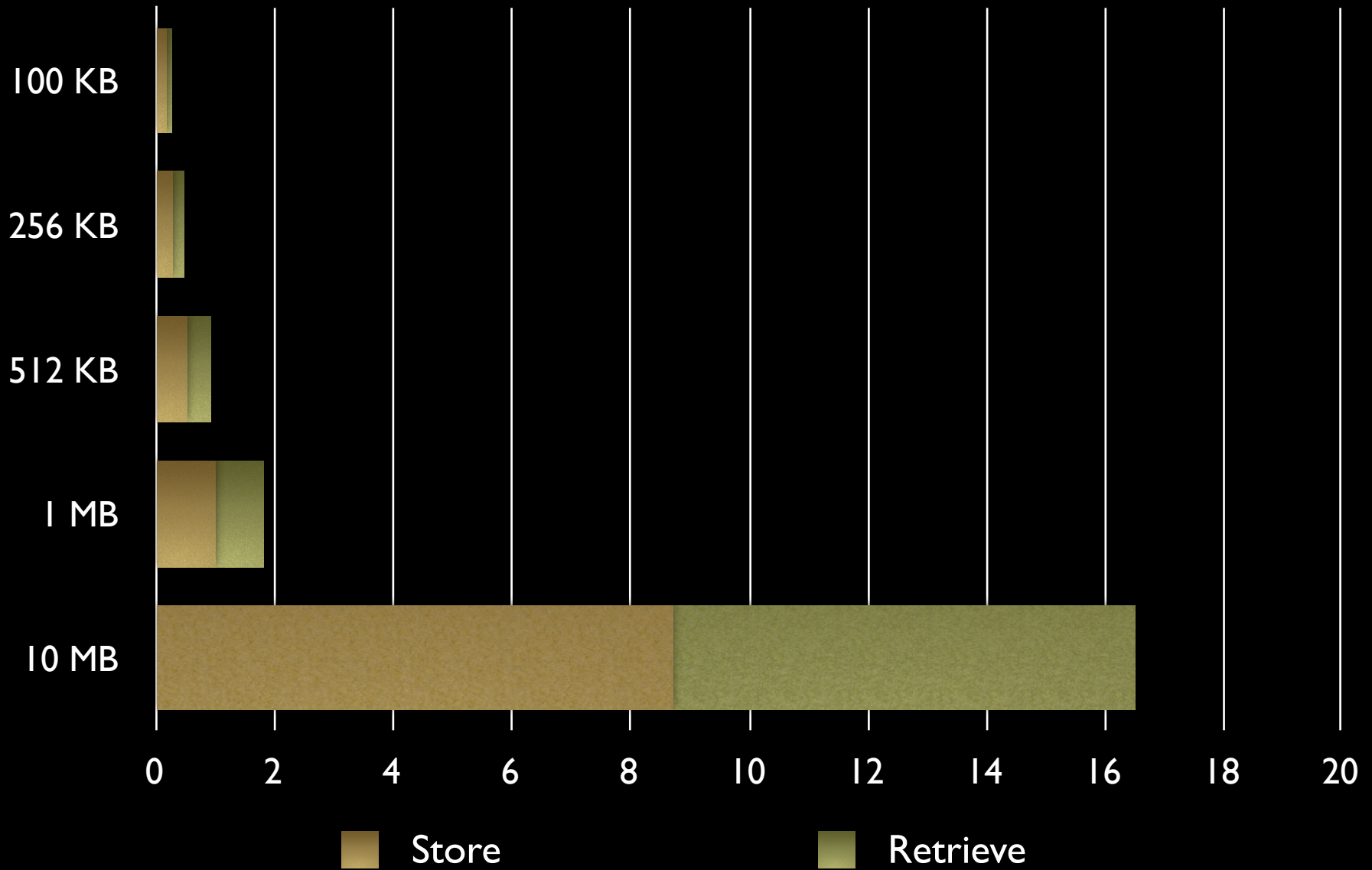
# Encrypted Local Store Performance Analysis



# Encrypted Local Store Performance Analysis



# Encrypted Local Store Performance Analysis



# Asynchronous vs. Synchronous AIR API's

AIR does allow you to work with multiple threads using the Asynchronous API's for certain pieces of functionality:

- Encrypted Local Store - No Async API
- Local File System - Sync and Async API
- SQLite Database - Sync and Async API

You cannot utilize multiple threads outside of these Asynchronous API's.

# Local File System

# API Dictionary: File System

- flash.filesystem.File - The class that contains a reference to a file system object (either file or directory).
- flash.filesystem.FileStream - This class can read binary data into and out of a file on the filesystem.
- flash.filesystem.FileMode - A class of constants that defines the mode in which a file is opened.

# Two Ways to Open a File

- `open( file:File, FileMode:String ):void`
- `openAsync( file:File, FileMode:String ):void`

# Code Example: Image Cache

# Serialization and IExternalizable

With the local file system, you are not limited to just storing binary data from files, you can also store complex object graphs by having your model implement IExternalizable.

[Serializing Objects](#) - DavidTucker.net

# Local File System Tips

- Vista will not allow you to copy, update, or delete any files that exist in the application directory. Because of this, always utilize the application storage directory for storing application data.
- Utilize the constants defined in [flash.filesystem.File](#) to help preserve cross-platform compatibility. Utilize the **url** property when possible.

# Embedded SQLite Database

# SQLite Tools

It can be very helpful to have an SQLite administration tool:

- [SQLite Admin AIR Application](#)
- [SQLite Manager Firefox Extension](#)
- [SQLite Database Browser](#)

# API Dictionary: SQLite

- flash.data.SQLiteConnection - The actual connection to a database stored either as a file or in memory
- flash.data.SQLiteStatement - The class that contains an SQL statement that will be executed on the database
- flash.data.SQLiteResult - The class that holds the result set for a given query.

# Code Sample: SQLite Database

# Encrypted Databases

- AIR 1.5 added support for encrypted databases
- Database must be encrypted when created
- The encryption key can be changed using the **reencrypt** method

# Generating Encryption Keys

You will need to use a library that will allow you to generate the required binary key for the encrypted database. Two common options are **as3corelib** and **as3crypto**.

- [as3corelib](#)
- [as3crypto](#)

# Code Sample: Encrypted Database

# Advanced Database Tips

- Each statement is in a transaction by default, so batch commands should be wrapped in a transaction for performance.
- When multiple SELECT queries need to be run simultaneously, utilize connection pooling.

# ORM Solutions

There are two promising solutions for simplifying interaction with the SQLite database in AIR.

- Flex ORM - Provides a metadata-based framework for defining the database model from the application model.
- DAO-Ext - Automates the implementation of the DAO pattern in your application.

# AIR Sync with LCDS

**LiveCycle Data Services** provides an enterprise solution for synchronizing data between the server and an occasionally connected AIR application.

- Data Synchronization in AIR with LCDS -  
John C. Bland II at InsideRIA

# Questions?

# Keep Up with RIA's

The **Weekly RIA RoundUp Podcast** at **InsideRIA** covers both Flex and AIR news as well as the rest of the RIA landscape.

