Arvados - Feature #2706

[SDKs] Python SDK Keep client can enable "HEAD before PUT" mode when writing. Expose this as an arv-put option.

04/29/2014 11:17 AM - Tom Clegg

Status:	New	**Start date:**	
Priority:	Normal	**Due date:**	
Assigned To:		**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	Deferred		

Description

Related issues:

Related to Arvados - Feature #4122: [Keep] Keepstore respects HTTP HEAD metho...	**Resolved**

History

#1 - 07/04/2014 04:35 PM - Tom Clegg

- Target version set to Deferred

#2 - 07/04/2014 04:38 PM - Tom Clegg

- Subject changed from Python SDK Keep client can enable "HEAD before PUT" mode when writing. Expose this as an arv-put option. to [SDKs] Python SDK Keep client can enable "HEAD before PUT" mode when writing. Expose this as an arv-put option.

#3 - 11/01/2015 01:32 AM - Tom Clegg

With permissions enabled, HEAD (like GET) returns an error if the client does not supply a valid permission signature. In order to achieve "skip blocks that have already been written" a client needs either:

- some way to look up a hash in a rainbow table to get the permission signature (and in order for this to work for *old* blocks, the rainbow table has to be on the API server, otherwise permission signatures will expire), **or**
- some other way to prove to keepstore that it has the data (CRAM?)

Some relevant use cases:

- arv-copy should resume an interrupted copy (a rainbow table in ~/.cache could address this use case *if* the transfer resumes and finishes before the permission signatures expire).
- arv-copy should not receive or transmit the content of data blocks that already exist on the destination side (a more general case that does not require arv-copy itself, or anything else on the client host, to have ever seen the data/hash in question).
- arv-put should not transmit the data when writing a block that already exists at the target keepstore node.

The easiest and most efficient way to enable this is to add a keepstore feature that accepts a *proof* from a client that the client has a data block in front of it, and responds with a signed locator. (In this case keepstore should also be able to *provide* such a proof, such that a client who has a signed locator for a block stored at site A can obtain a signed locator from site B without even receiving the data from site A -- assuming, of course, that the data already exists at both sites.) This creates an information leak which will be unacceptable for some sites: a client who has a chunk of data could use this feature to determine whether *anyone else* has stored that same chunk of data. There must be a server configuration option to disable the feature and prevent the information leak.

An alternate way to reduce excess copies, without the above information leak, is to add a locator-signing API that accepts a manifest (or a set of Keep locators in some other format), determines whether any of the given blocks are already readable by the current user, and returns signatures for those locators. This will require an index or table that can efficiently map locators to the current user's collection permissions (e.g., a table of {block locator, portable data hash}).