Key Management in the Cloud

Large Scale Genomics / Encrypted File Formats

Date: Tuesday, March 2, 2021
Time: 21:00 - 22:30 UTC
Meeting Chair(s): Alexander Senf

	Agenda Item	Speaker	Time
1.0	Introduction and brief Overview of Crypt4GH	Alexander Senf	10
2.0	Current ideas about key management	Alexander Senf	10
3.0	Discussion	All	60
4.0	Recap Actions and Key Takeaways	Alexander Senf	10

Abstract:

The Crypt4GH encrypted file format is targeted at individual researchers working with secure data. An example workflow would be to receive a data set from the European Genome-phenome Archive (EGA) and analyze them using one's own code.

Crypt4GH is random-accessible and compatible with existing index files. It allows these files to be stored on disk in an encrypted format, and the tools (e.g. samtools) using the files automatically decrypt data on the fly into main memory for computation; results can also be written directly to encrypted files. The advantage is that data on permanent storage is always encrypted, limiting the attack surface of exposed unencrypted data to the main memory of the computer running the tool.

Crypt4GH files are encrypted for individual users, using Curve 25519 based asymmetric keys. The user's private key is required to access files. This works great on local machines, but poses problems if the user wants to send Crypt4GH-encrypted files into the cloud for processing, because the cloud machines would need access to a private key for the files in some form—which would expose the key to XYZ .

This session is to exchange ideas and to discuss ways to enable Crypt4GH-encrypted files to be processed in the cloud natively; ideally within the standards developed by the Cloud Work Stream (DRS, WES, CWL, etc.). The session is also for the Crypt4GH task team to learn about options available to us in the cloud.

Here is a presentation showing more detail, and some of the ideas we have had so far: https://docs.google.com/presentation/d/1Beo_hHrX4nGCy_nD4SakPKbwy3ea5xVHkFSiuAymyWc/edit?usp=sharing.

[The scope of Crypt4GH is personal researchers. As such it is separate from the efforts currently pursued by the Data Security Work Stream (DSWS) & Cloud Work Stream (CWS) with regards to enabling homomorphic encryption and secure multiparty computation systems. The goal is to enable individual research to happen as it does today, with the difference that the files used are all encrypted instead of plain, in a way that is transparent to the user.]

Minutes

AWS, Azure contacts needed

Integration with Passports - has anyone there looked at this yet? Looked but not discussed…

Shrems 2 changes how we approach this stuff, fully-controlled keys, with clouds, enclave computing,

How does it speak to our AAI protocols? Would enhance that security, bc clouds can't even see what's going on

Adding keys to passport was rejected previously, in this case not carrying keys around in passport,s just want auth to be strong enough

Protocol that establishes feedback between cloud and data API

David Jackson - solution ideas, changing the header of encrypted file either append or replace, that might be awkward, don't want to change first few bytes on end of file

Normal bucket tech, can't write in, have to rewrite whole file

Can you have a temp key in side channel, auxiliary file with temporary key

ASenf: possible, look at way EGA stores crypt4gh in archive, header and data separated, header generated on the fly

KRodarmer: SRA toolkit was going down that road for a while; now NCBI is relying on clouds for greater security, unclear where that will lead

AS: Use case involving DRS, data downloaded from the origin to destination ,only one header generated at the source; or work locally and send to cloud, has my header attached to it

Can see benefit of splitting, would have to adjust implementations to work with that

Currently have standard itself - file format, what it looks like - no prescriptions on how to implement; key handling is an implementation detail

Thought shouldn't be part of standard, but if you create diff ways of handling the key then file wouldn't be interchangeable after all, need to ensure tool matches the file

DB: has done in all three environments, all similar conceptually but different implementation wise, require different way to think about it; concept works in all three, external keys, enclave computing, replication process is different in each, but all have a replication process

Google has ability to do External Key Management (EKM) - on prem or wherever, key exchanges that way; works okay, integrates with customer "off-site" keys. SEB ...only in alpha, not GA yet, but does work

Azure HSM, AWS - less experience but work similarly

Looking at full year for any three platforms to fully support all your stuff

Crypt4GH and Cloud APIs are probably on same timeline, so that's not terribly concerning

BO'Connor: will there be changes recommended to DRS or extension to support this? How do confidential compute APIs from CPs interact with DRS, transparent to end user? Add'l back and forth required to DRS server?

ASenf: Part of the reason for this discussion :)

Curious whether DRS is interested in an addition to enable something like this, or look at other ways of addressing

OH: How does SAMtools, htslib recognize this? Is it needee?

RDavies: inspects start of file

You start agent, load in your private key, samtools needs to decrypt, gets header, ...uses that to decrypt rest of file

OH: would something on DRS side be helpful for this, server side rather than client side? Or nice to know what's coming back?

AS: if cloud requests data from an origin, negotiate key pair between cloud destination and origin; don't have to submit private key into cloud or anywhere, key pair generated for this particular session; what cloud receives - that would work as SAMtools does now

Just diff when download to local machine

BO: dynamically created key pairs, something running on my behalf in cloud, I as worker node, user passed me DRS...crypt4gh file, so before I request it, send some info (dynamically created public key), encrypts header and sends file back

User doesn't have to pass in anything there, data comes back safely encrypted

How does that tie into secure compute infrastructure within the clouds? Dynamically created on worker host? Some other infrastructure?

Extension that allows you to request something, header encrypted with particular public key you provide

RD: relatively simple to generated key pair, DRS server has to validate the key pairer, allowed to see that data; may have to do that anyway

BO: DRS servers typically not 3rd party managing access that they themselves can't read

Server serving up bytes for particular project, group running that DRS server is the authoritative source for that data

CVoisin: Billing could become an issue? Where does encryption/decryption happen, if requestor paid scenario, ensure end user billed, not host? Or not an issue

RD: Ephemeral key pair, DRS would gen new header for file, rest of file is unchanged, if you can send as two diff bits, it's fairly simple, shouldn't have a problem; only sending about 100 bytes and then a giant file

KR: but then are you claiming no signed url to the data file? That will cause the signer to be charged for the network costs

OH: cost management will hit TES, WES, long before headering charge becomes an issue, I think

KR: Cost of issuing new header is negligible, but that either means you give open access to your bucket and have user negotiate payment with CP, or closed access, then mechanism is to use signed URL, transfers all costs/permissions to the signer

OH: How is this different from what DRS does?

BO: spec doesn't cover this use case, when I'm requesting signed URL - original use case, if Im authorized I'll get signed URL without question; reality is most projects that will provide signed URLs if doing that outside cloud, not insignificant cost of pulling data out of the region, need

something where signed URL request is made, end user provides minimal amount of account info necessary so CP bills egress to person requesting signed url rather than owner of the data

Part of the question here, we can be careful about making sure using temporary dynamically created key pair, safe for worker host, but passing in passport and account info so signed url process can happen appropriately - other information going into this worker node on this cloud that I don't necessarily trust

Make data secure through crypt4gh but how to ensure secure with account info and passports, so if either is intercepted, just request a file encrypted from the system and use it

How do you encrypt the entire system, beyond just the transfer of bytes?

OH: cost management isn't necessarily a crypt4gh question

RD: server with original file has to know key used to encrypt that file, has to be stored somewhere, take that key and public key you've been given and use that to generate another encrypted header with file key in it - give that back

Thing you're encrypting is very small file key

OH: spin up local ega, eg., ideally like crypt4gh encoded content, saves me from re-encrypting file for every user, don't need multiple copies around, storage costs goes down

KR: might consider not having a header file or not calling it a header, pass around in a token and keep it ephemeral

RD: one reason for doing with htsget, rest of file give url to,

AS: currently it's a crypt4gh file, looks at beginning,would have to change that slightly in htsget

RD: could be enhancement to add extra record in header, this is where you get the data

AK (via chat): Couldn't a temporary key be minted (e.g., by DRS after checking the user has access)and the data header be modified to have the matching section for that key, sorta like a (one-time or time-expired) bearer token? If I understand correctly, Crypt4GH assumes (or needs to assume) that tools working on the data need to be trusted. If that is the case, I suppose they might as well be required to have functionality to remove the header section that matches that temporary key?

BO: need to do some DRS-extended-for-crypt4gh proposal, response has some information

Take a DRS ID and ask the server what is this, tells you various ways you can get the data

Could include “it supports cryp4gh” and that’s one option for getting data, subsequent step for that access mechanism, as parameter requires encrypted public key

Two step process would work well with what needs to happen here, not mentioned in spec now; would need to be extension

How do we do secure compute in cloud environment, part of the answer is crypt4gh…

How do we send passports to an environment we don’t particularly trust, encrypted file flow works fine for DRS, but if it can steal my passport and use for other stuff without my knowledge, no matter how well encrypted the file is, won’t work without ability to work with my passport

What is the correct place to discuss full life cycle of computing in cloud environment we don’t quite trust

RD: if someone files up a random host asks to access yoru file, how do you knwo it’s the one that should be allowed to?

KR: There are ways to approach it but the way we’ve been trying to use based on oAuth conventions, not set up to handle this, bearer tokens, other problem that Brian described, anyone can use those tokens; fragility of the system.

DSteinberg (via chat): Have any PKI services been considering for public key sharing? Would PGP work and why not?

RD: PGP is a signing thing, demonstrating the person who sent the message is the person you think sent the message. If alice sends bob a message, confirm alice sent it

DS: what mechanisms are in place for sharing public keys, or just user’s responsibility to figure out how to share their key in a single request

RD: no opinion on that, but if you can make a secure channel between yourself and channel, easy to send key down it, eg., HGVS which work fine for th purpose, tricky part is “I’m talking to the right person”

DS: third party to verify data request, PI granting access to a researcher to do data request, centralized key sharing mechanism might allow for that

RD: EBI eg., could store public keys for the people allowed ot use that service

KR: not sure centralized approach is viable across the world

RD: tricky and arguably PGP never really solved it

KR: did what they could within decentralized system, wouldn't work for all conditions, wouldn't work for NIH

DJ: Concentrated on idea of ind'l researcher with their own private key being able toa access data bc header being written with public key to allow them to read it; but going back to the idea of combining this with a DRS server, valuable even without that, base encrypted files, DRS server is where all auth, pushed back to DRS layer, provides URL could be to an integrated htsget server, provides a rewritten header on the fly, second part of response feeds back to static file which you don't have to continuously change

Files encrypted at rest, rely on DRS / htsget provide signed header for that ind'l and that particular request...perhaps?

APatterson: why issue a header with pub/private key immediately decrypted, why not have DRS server issue a...if the encryption key for whole static file is just a key you could give dynamically, once proved authorized by server, why go through pub/priv key - that's for identity stuff, once solved, just give key to file

KR: keep in mind every process is running as a user agent, under power of attorney of original user, handing out bearer tokens, several hands in the mix all have ability to exercise bearer token, naked key sent back can be for any of them, to the user only, under user's control to get at that key; provides a level of confidentiality.

DJ: to address that, submit public key to combined htsget/drs server and use that to provide header on the fly

AS: run samtools in the cloud, eg., file that will be received will then be processed by samtools to do something, should run same way on my machine as in cloud, same version of the program, wrapped in docker container, works everywhere, should be able to handle all of these situations, important to make this a ubiquitous file format

RD: designed so you can save your encrypted file on disc and come back to it later, assuming you haven't forgotten your

DJ: use in cloud is more ephemeral

AP: in cloud you're not wanting that, in cloud file can be thrown away, doesn't fit that use case

RD: same infrastructure works for both

DJ: keys used might differ, create temp keypair to submit to DRS/htsget server, on laptop use more permanent long term key

RD: yes, but essentially the same thing, could send a file key over, without all the extra wrapping around it, assuming secure method to do it, need slightly different code to handle that

AP: Because of what DRS, passports - all attempting to solve that auth problem, by the end maybe you can just give them encryption key to the file without header...probably a bad idea ;)

AS: one way around authentication issue, if you do require user to upload private key, that's something that happens outside passports or any API, source produce data that requires private key to be read, even if token is intercepted, couldn't read the data

Might be argument for requiring users to upload key first

RD: then go back to original problem, you have a terribly secret thing

KR: with all this protection around source data, what happens when someone dumps it as FASTQ

RD: nothing to stop you from dumping it out and leaving it somewhere, argument was to make it sufficiently simple

KR: we created a fuse-based environment, made it so encrypted sra files appeared as plain text, but that was never necessary bc our toolkit decrypts in memory, all of the plain text output would be recorded on same encrypted volume, without any effort, as long as they mount that volume as their workspace, able to work in plain text

OH: is that really part of the spec, or...implementation? Meant for me to pass data onto users, what you do after that is up to you, not meant to be enclave for files to be encrypted at all times

KR: yes, what we've been pursuing we've found to be an awful lot of work, in the end defeated by the average use case; for all of our encryption, didn't buy us much, bc first thing someone did was dump as FASTQ and throw away encrypted version

OH: from perspective of data steward, we need to keep data secure in our environment, AWS, secure layer on top as I pass it on to someone else; after that it's their responsibility to, I also need to ensure their server is secure.

From data provider perspective don't necessarily need that, would be nice, but getting into DRM space

KR: discussing a lot of security, secure enclaves only to be thrown away

KR via chat: Our conclusion was that we should give users the tools they need to protect privacy. We did not prevent them from violating it - only make it possible. And I think this is the spirit of what I am hearing here.

Frederic Haziza: for EGA we'd do the reading, but there's another version that uses crypt4gh for read and write

AS: that's the other option, send files into cloud, access them from the api, access happens from Fuse layer

Still has same issues of key exchange, would make easier for using all tools, legacy tools, but also means one more step for user on laptop, would always have to start up fuse layer and remember to use that, rather than dumping into a director they're used to

Tried to add as native format to as many tools as possible, remove as many steps as possible so user is never tempted

RD: could be tricky to persuade everyone; idea with SAMtools, make you recognize it as encrypted file…

AS: that would be a great goal, data you download, use and results stored in encrypted format, ideal use case, but that requires a lot of tools to support it

RD: consider things like accidental spillage, eg., sort, might do temporary copies all over the place - need to make sure they're encrypted as well

Next steps:

- Reach out to clouds, but what is most useful for spec maintainers to do next?
    - write up the use case in some detail, and a draft pitch or 2-3
    - Discuss with TASC - so coming as GA4GH, not just a bunch of academics asking for help

KR: interested to gather information to contribute to the use cases from NIH, volunteer to do some of that and feed back into the conversation

RD: don't think block chain would be useful here, BC good for storing a large number of things you want to remember, but in this case don't want to remember the private keys, that's the problem we have here

AS: idea to get path the data took...but expensive way to do it

DS: nucypher uses partial key encryption, allows cloud neutral way of doing encryption in this spirit

RD: secure multiparty computation?

DS: key exchange mechanism, system for doing symmetric and asymmetric keys together, so you have file key like one you're talking about, split up across multiple people, other agents

have to agree before someone can reconstitute the file key, block chain can see every time access a given file

RD: Secret sharing type thing

AS: should we think about doing with DRS to make this useable as a proof of concept,

BO: straightforward to mock up what this would look like, what a request would look like, and how DRS would tell you this is a crypt4gh access method...etc

HSofia: Is what's missing here a threat model? A clear concept of the actual hazard or threat, easy to get too much security/too little, but better to be up front with what you think the hazards really are

AP: Agree, say this part of the solution is aimed at this part of the threat model vs. another

HS: that would be a reaction the DSWS might have, also good to say up front, "we don't think this will protect against everything" already doing that when saying you trust the researcher to some degree; may be sloppy, but reasonable to assume not malevolent, if you're wrong, still probably better than those things called privacy/security, inadvertently becomes theater.

feedback/response loop, decide how well is this working, decide too much or too little

RD: did some threat modeling when we put the spec together, mentioned but didn't fully cover this key management part, put it out of spec

AS: for the use cases we envisioned back then, users downloading to systems they have control over, mentioned there were issues to keys in the cloud, but not part of what the spec did back that

RD: cloud providers give you secure ways of storing keys make them safer than legacy HPC file

AS: have been talking about this, but without cloud expertise, hard to know what really is the threat, has to be good enough that data owners agree to send into the cloud, different for different data owners, pretty sure there is a standard model of what's acceptable / not of data currently residing in the cloud

What is considered secure? When should we worry and when is it way too much security?

DB: depends on the culture, given Schrems 2, changes how whole continent views security of cloud, some things may be overkill, external keys in diff environments will become normal to accommodate that

If Schrems 2 really becomes the guidance, everyone agrees, that will have to be the new normal

AS: good example of eviro we need to consider

DB: customer managed keys with enclave computing, customer managed keys, cloud supports this seamlessly, click button once set up but if you lose your keys all data is dead

Huge complications but from an implementation perspective, not terrible

DB: enclave computing in the clouds, triples pricing of most of their offerings

AS: what is current security consideration in cloud WS?

DB: this is super new, so not much yet; cloud WS - JP's security group, about more esoteric/advanced things, homomorphic encryption, etc., those do solve all this - with that security/privacy apparatus, but way more complex and not built into clouds yet; customer managed keys is built in

BO: for me this is the beginning of the conversation, relatively new; interested to see what changes… Cloud APIs are on purpose simple APIs, WES/TES/DRS/TRS - complications go toward the implementers, interested to see what bubbles up to required supplemental interface changes; work is really those that implement DRS, WES will have to implement in more complex way under hood, but what does that translate to in changing the simple interface for calling end user?

What would a prototype look like, to pull of back/forth with DRS server providing crypt4gh, few changes from DRS server, implementer would have the work behind the scenes

Don't know of a DRS server that advertises htsget as an access option, but pretty sure its on the list of available access methods; if not in there, relatively small change; DRS just reporting that you exist and you can get data at this address

DS: byte range queries on bams or have to go for whole thing?

AS: want to support access in same way you'd access BAM

DS: features for using index to go to specific location works; plans to do similar things with VCF?

RD: agnostic to whatever file is inside it; VCF, SAM, CRAM, etc and all the indexes as well

DS: curious if anyone looking at homomorphic encryption, don't see BAM being handled that way bc amount of computation needed, for VCF may be fairly tractable though; would the standard support new encryption methods?

RD: that is out of spec, would need to talk to JP about this!

AS: next steps:

- Use cases that explain what it is that would be done in cloud/crypt4gh in more detailed way, in collaboration with cloud WS
- Extend threat model into cloud, now that have better idea of what exists (see supplementary data in recent bioinformatics paper)
- Reach out to Cloud, esp DRS, to get basic interaction working
- Reach out to Cloud providers to see how implementing key store, secure enclave would work, how to integrate with access to crypt4gh files