
Dropbox Gearup

Checkpoint

At this point, you should have a design
and be ready to implement it!

Make sure you have done your design meeting!
If you have not, contact us ASAP

What the client looks like

```
# Make a user
client.create_user("usr", "pswd")

#           . . .

# Log in
u = client.authenticate_user("usr", "pswd") # Returns a User object
```

What the client looks like

```
# Make a user
client.create_user("usr", "pswd")

#
    . . .

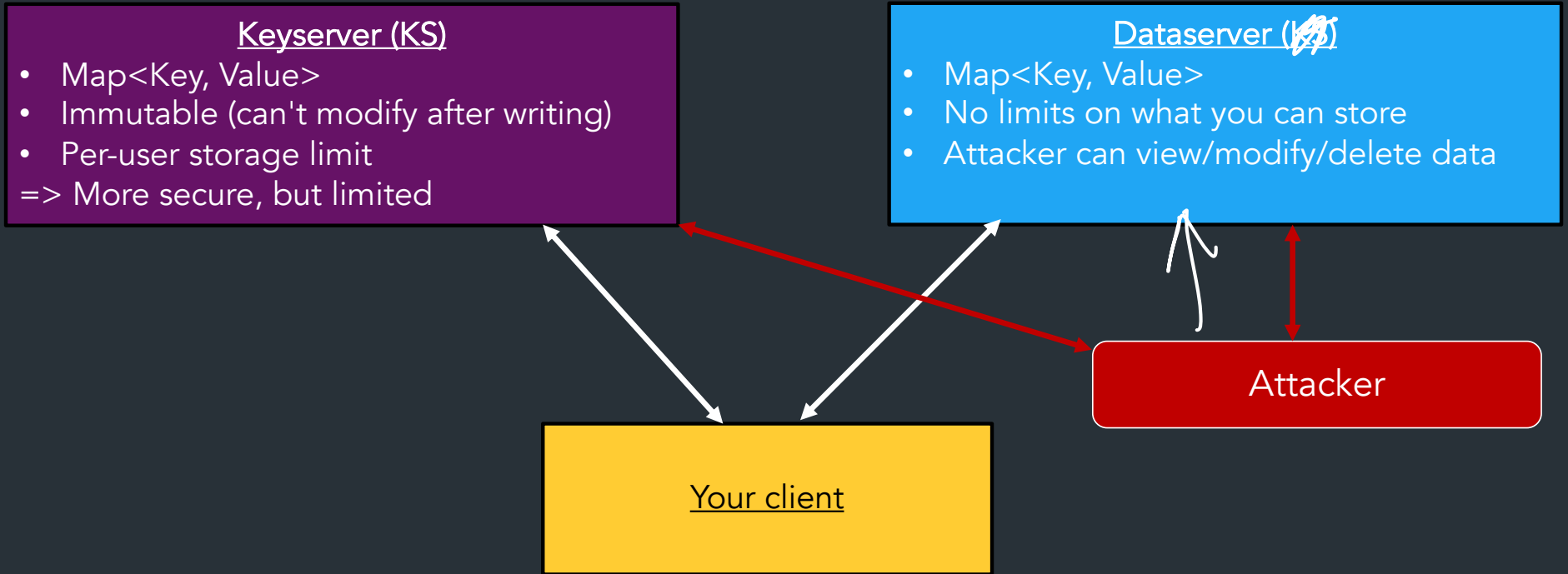
# Log in
u = client.authenticate_user("usr", "pswd") # Returns a User object

# Make some data to upload
data_to_upload = b'testing data'

# Upload it
u.upload_file("file1", data_to_be_uploaded)

# Download it again
downloaded_data = u.download_file("file1")
assert downloaded_data == data_to_be_uploaded
```

System Overview



Setup and Stencil

Container setup & Environment

For this project, we'll use the "Development container" (same as project 1)

- Some slight updates—see setup guide for instructions
- Stencil uses a Python virtual environment
 - See setup guide for instructions
 - Like VSCode? You can use it with the container!

Python stencil/testing overview

- Demo!
- (See posted examples)

Serializing stuff on the dataserver

Remember: to store or encrypt/MAC/sign anything, it must be of type bytes

Example: strings

```
s = "Hello world!"
s_bytes = s.encode("utf-8") # Or write s = b"Hello world"

m = memloc.Make()
dataserver.Set(m, s_bytes)

sb = dataserver.Get(m)
s_check = sb.decode("utf-8")

assert(s == s_check)
```

For lots more info: see the [serialization examples](#), linked in the setup guide/Ed FAQ

Example: more complex types

=> Use our serialization API to help you!

```
def test_serialize_dict(self):
    info = {
        "a": 1,
        "b": crypto.SecureRandom(20),
        "c": [1, 2, 3, 4],
    }

    addr = memloc.Make()
    info_bytes = util.ObjectToBytes(info)
    dataserver.Set(addr, info_bytes)

    info_check_bytes = dataserver.Get(addr)
    info_check = util.BytesToObject(info_check_bytes)
    assert(info == info_check)
```

Can serialize any nested
int, bool, string, bytes, dict, list

But can we do better? What if you like types?

What about more complex types?

Yes! Can serialize arbitrary classes, dataclasses, etc.
You just need to write some helpers.

See serialization examples for more info!

Example: Asymmetric keys

Asymmetric keys use a special type, need to serialize specially...

Note: this generates a key for encryption
SignatureKeyGen() generates a key for signing

```
k_pub, k_priv = crypto.AsymmetricKeyGen()

# How do I know which types these are?
m = s_addr("k_pub")
k_pub_bytes = bytes(k_pub) # Convert to bytes
dataserver.Set(m, k_pub_bytes)

kpub_check_bytes = dataserver.Get(m)

# Convert back to Asymmetric key object
kpub_check = crypto.AsymmetricEncryptKey.from_bytes(kpub_check_bytes)

assert(k_pub == kpub_check)
```

AsymmetricEncryptKey: public key
AsymmetricDecryptKey: private key

Working with memlocs

Demo/example!

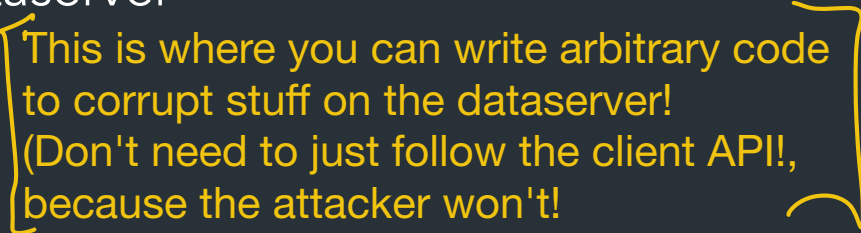
See [setup guide](#) and [serialization examples](#) for more info!

(And the [recording...](#))

Security analysis/Attack tests

For your final writeup, should try to write tests for some attacks

What could this include?

- Setup phase: what data should be on the dataserver
- Attack phase: what does the attacker do? — 

This is where you can write arbitrary code to corrupt stuff on the dataserver!
(Don't need to just follow the client API!, because the attacker won't!
- Test phase: what does the client do that detects the attack?
=> Next operation after attack should raise a DropboxError

See examples for more info!

Tests that we provide

- test_client.py: Some basic tests, can add your own tests here
 - => Your security analysis tests (ie, tests on attacks) should go here
- test_functionality.py: Almost all of the the autograder tests (more comprehensive test suite)

=> See setup guide for examples and tips on how to run the tests! Make your life easier!

What you'll submit

- Your code: autograder will be up soon
- Final writeup: your design document
 - Updated design based on what you submitted
 - Security analysis: speculate on some attacks, write about how your design prevents them

For full details on what you should include, see the handout

Some final design advice

1. Make sure you look at the serialization examples + resources in the setup guide

2. Plan things out on paper: if you need to make changes to your design, try to work them out on paper/a whiteboard first to discuss => you don't want to backtrack on your implementation if you find a problem

3. Don't try to implement everything all at once! We've given you a lot of tests: try to implement the major bits (upload/download, append, sharing) one at a time and test them

4. ✨ ✨ ✨ Don't repeat yourself--use helper functions!!! ✨ ✨ ✨
=> You're going to do a lot of small things repeatedly (serialization, deserialization, encrypt+MAC, integrity checking) => make helper functions!!
=> Spending some time to think about and build these WILL absolutely save you time later when you need to debug! We really mean it!

About autograding

Questions?
