Crypton Overview

Hack In The Box, Malaysia, 2014

David Dahl, SpiderOak

Cam Pedersen, SpiderOak

David Dahl

Crypton Director, SpiderOak

Mozilla Privacy Engineer

Mozilla DevTools Engineer

ILM Engineer

Cam Pedersen

Crypton Principal Engineer, SpiderOak

Crypton

Crypton Setup

2 Methods

- Docker Install Full Infrastructure
- Web (Firefox Scratchpad)

Docker setup

crypton.io/blog

Web (Firefox Scratchpad)

- In Firefox go to https://nulltxt.se:1025
- Open Firefox Scratchpad: Devtools->Scratchpad

What is Crypton?

- Client framework
- Storage server
- Realtime server

Build applications where the server
doesn't know what data the user is saving

Client

First client (POC) is built in JavaScript

We're working on native clients soon

Code delivery

Cordova / node-webkit

Top Level Object: crypton

crypton.generateAccount

crypton.authorize

crypton.<...>

Predictable callback pattern

crypton.<APICALL>(argument[s], function callback(***error***, ***successObject***){})

argument[s]

callback function

error

successObject

Client

APIs

- Keyring / Account generation
- Authorization
- Peer Discovery & Trust
- Inbox
- Container lifecycle

Crypton Account Generation

```
crypton.generateAccount(username, password, function callback(err, account) {

  if (err) {
    console.error(err);
    return;
  }

  console.log(account);

  myapp.account = account;
})
```

Account


```
{
 "username": "ddahl",
 "keypairSalt": "[-1737095913,1934775537,-319606855,-924854634,-33963282
 "keypairMacSalt": "[1161055642,43367311,-1050404689,1691693954,17032129
 "signKeyPrivateMacSalt": "[1674657353,-602795583,-769601665,-598924700,
 "srpVerifier": "71c40c4b0176ab34e23ed42...",
 "srpSalt": "ijhEwD8qpQSD1NQXupBmPe",
 "pubKey": "{Object}",
 "signKeyPub": "{Object}",
 "signKeyPrivate": {Object} ,
 "hmacKeyCiphertext" {Object},
 "containerNameHmacKeyCiphertext": {Object},
 "keypairCiphertext": {Object},
 "keypairMac": 99598a3abb5c633ad2882d4f0b24ee4151412d2931690a431e47f8ab0
 "signKeyPrivateCiphertext": {Object},
 "signKeyPrivateMac": "70eb04c4d9b95e679ffc5778dfecaa14c759735879491bf21
}
```

Crypton Session

```
crypton.authorize(username, password, function callback(err, session) {

  if (err) {
    console.error(err);
    return;
  }

  myapp.session = session;

});
```

```
{
 id: "2BoHGj0iMO6PrM7qyS1FK8zd",
 peers: Array[0],
 events: Array[0],
 containers: Array[1],
 inbox: Object,
 socket: Object,
 account: Object
}
```

Peer Discovery

```
myapp.session.getPeer('alice', function callback(err, alicePeer) {
  if (err) {
    console.error(err);
    return;
  }
  // The alice peer object will be stored in 'session.peers' array
  console.log('Success! we have: ' + alicePeer.username);
});
```

```
{
  accountId: 59,
  session: Object,
  username: "alice",
  pubKey: Object,
  signKeyPub: Object,
  fingerprint: "cd87ef5fa2f0767f120f17743e926a4edefd6fa8687cf502b87041a3fae59755",
  trusted: true
}
```

Peer Trust

Out-of-band fingerprint comparison

SMS

Email

Printed

Peer.trust(callback)

Alice must "trust" Bob before messages can be exchanged

This process is largely automate-able when using ID Card QR-code

What does trust() do?

trust() adds a peer object to an internal container

called 'trustStateContainer'

```
david: {
  trustedAt: 1413258904170,
  fingerprint: "2ce9d8f17f9f8f780f7f0dfa2e6b9202a4442ee202b04a0ee878a23c5f2e5ae8"
},
...
```

Inbox

Crypton's internal app messaging API

Meant for short messages, notifications

Can be used for ephemeral messaging

```
myapp.session.inbox.getAllMetadata(function callback(err, metadata) {
  if (err) {
    console.error(err);
    return;
  }
  console.log('Success! we have a message metadata array!);
  // The metadata array can be added to a queue to pull down and decrypt the message content
});
```

Inbox.get(msgId)

Once we have the message ID, we can get() it

```
myapp.session.inbox.get(876, function callback (err, message) {
  // get message #876, once the message is handed to this callback function,
  // it is already decrypted
  if (err) {
    console.error(err);
    return;
  }
  myapp.displayMessage(message);
});
```

Containers

Containers allow versioned data storage

Containers can be larger data sets

Containers can be ***shared***

Containers API

- session.create()
- session.load()
- session.deleteContainer()

Note: container names are not stored in plain text on the server, they are HMAC'd

session.create()

```
myapp.session.create('my-notes', function callback (err, container) {
  if (err) {
    console.err(err);
    return;
  }
  // Now we are handed a container object,
  // and the container is cached in session.containers
  console.log(container.name);
});
```

session.load()

```
myapp.session.load('my-notes', function callback (err, container) {
  if (err) {
    console.err(err);
    return;
  }
  // Now we are handed the container object,
  // which may have been cached in session.containers
  console.log(container.name);
});
```

session.deleteContainer()

```
myapp.session.deleteContainer('my-notes', function callback (err) {
  if (err) {
    console.err(err);
    return;
  }

  console.log('Container was deleted');
});
```

Add data to a container

Containers have keys & values

```
container.add('myList', function (err) {
  if (err) {
    console.log(err);
    return;
  }
  console.log('container property "myList" was created');

  // Now we can add data to the container:
  container.keys['myList']['one'] = 'First item in my list';
});
```

Share a container

Containers can be shared with one or more peers

```
container.share(peer, function (err) {
  if (err) {
    console.error(err);
    return;
  }
  // Success
  console.log('Container was shared');
  // Let's get notified of edits to the container:
  container.watch(function listener() {
    console.log(container.name + ' was edited');
    // We can then do session.load() to see the latest updates
  });
});
```

Live Code

Crypton Messaging Application: The skeleton

Live code functions

- Create User
- Authenticate User
- Lookup Peer
- Trust Peer
- Message Peer
- Get Message