Identity, Security, etc. API Issues

Eric Rescorla

ekr@rtfm.com

Overview of Topics

- DTLS
  - Controlling my own DTLS key
  - Examining remote DTLS parameters
- Identity
  - Examining my own identity
  - Channel between chrome and content
  - `noaccess` and `peerIdentity` constraints

DTLS Key Control Requirements

- Keys are scoped to origin
- Be able to use the same key repeatedly
  - Avoid repeatedly generating keys
  - Enable key continuity/auditing
- Be able to use multiple distinct keys
- Be able to generate a temporary key

- Application needs to be able to control this

DtlsKeyName Constraint

```
{
  mandatory : [
   {
     DtlsKeyName : "ekr@example.com"
   }
  ]
}
```

- DTLS Keys are stored under `DtlsKeyName` value D
- If no key exists with name D it is made and stored
- If key exists with name D that key is reused
- "falsy" (`false`, `null`, ...) `DtlsKeyName` values never match anything
  - ... this means make a fresh key pair for this call

DTLS Key Control using WebCrypto

- JS creates a key using WebCrypto
  - `pc.setDtlsKey()` API call to impose the key
- JS is responsible for figuring out what keys to use
  - Keys can be stored using usual WebCrypto mechanisms (`wrap()`, `unwrap()`, etc.)

WebCrypto Example

```
function new_key(label){
  // Algorithm Object
  var algorithmKeyGen = {
    name: "RSASSA-PKCS1-v1_5",
    // RsaKeyGenParams
    modulusLength: 2048,
    publicExponent: new Uint8Array([0x01, 0x00, 0x01]),
  };

  window.crypto.subtle.generateKey(
     algorithmKeyGen,
     false, ["peerconnection"]).then(
    function(key) {
      index.put(key, label);
      pc.setDtlsKey(key);
    }
  );
};
```

```
function set_key(label) {
  var req = index.get(label);
  req.onsuccess(
    function() {
      if (req.result === undefined) {
        new_key(label);
      }
      else {
        pc.setDtlsKey(req.result);
      }
    }
  );
}

set_key("ekr-key");
```

What about the other side's public key

- Would be nice to know the other side's public key
  - For key continuity
- We Justin, Martin, EKR went back and forth on this
  - And decided that less is more
- Proposal: a binary version of the other side's keys

New API

- `pc.remoteCertificates` contains a list of other side's certificate chain
  - As ArrayBuffer
- The raw certificate can just be used as a lookup key
  - ... or parsed with WebCrypto (when available)
- No claims about the browser's opinion of the certificates

Recap: remote identity

- Remote identity is directly observable

```
dictionary RTCIdentityAssertion {
    DOMString idp;
    DOMString name;
    // Extensible
};
```

- Stored as `pc.peerIdentity`

What about my own identity?

- Would be nice to be able to observe this
- We have `pc.onidentityresult` to notify when assertion obtained
  - It doesn't have a defined argument ("TODO")

Proposal

- `onidentityresult` takes a `RTCIdentity` argument corresponding to the obtained identity
- Rename `peerIdentity` to `remoteIdentity` to match `remoteDescription`
- `localIdentity` contains my own identity (can be `null`)

Message Channel between chrome and content

"The context must have a MessageChannel named window.TBD which is "entangled" to the RTCPeerConnection and is unique to that subcontext. This channel is used for messaging between the RTCPeerConnection and the IdP. All messages sent via this channel are strings, specifically the JSONified versions of JavaScript structs."

- This works fine in current Firefox implementation (landing soon)
- What should "TBD" be?
  - Proposal: `identityMessageChannel` (but I don't care)

`noaccess` **and** `peerIdentity`

- Current status
  - Can't attach MediaStream to inappropriate sinks
  - ... generates errors
- New proposal from Martin
  - Can attach anything anywhere
  - But unauthorized sinks just get silence/black
    - Rules for "authorized" remain the same
  - Need API flag for "authorized"; propose read-only `.accessible`

Modified Permissions Model

- Allow JS to get a `noaccess` stream w/o any permissons
  - Can map into video/audio tag
  - Usable for "hair check"
- Permissions check when constraints change
  - This means we need a `.onaccessiblechange` event
- This is generally more flexible
  - But arguably more creepy

What happens when you run out of resources?

Eric Rescorla

ekr@rtfm.com

Possible cases (mostly hardware limitations)

- Fixed number of HW decoders
- Fixed total number of HW resources (e.g., macroblocks)
- Total CPU limitations

When do you know?

- `AddStream()` (for encoders, but not sure)
- `CreateOffer()`, `CreateAnswer()` (can I create a valid offer with the known resources)
- `SetRemote()` (did the other side ask to send me more than I can process)
- When media starts to come in (if I over-allocate)???

Proposed processing model

- Iterate over tracks *in order*
- Add any track which can be added
  - Assume maximum possible resource consumption for that track
- Skip any track which cannot

Example 1

- I can receive one HD stream and one SD stream
- Other side offers HD1, HD2, SD
  - I accept HD1, SD
- Other side offers HD1, SD, HD2
  - I accept HD1, SD

Example 2

- I can receive one HD stream *or* two SD streams
- Other side offers HD, SD1, SD2
  - I accept HD1
- Other side offers SD1, HD, SD2
  - I accept SD1, SD2

How do I find out what happened?

- For `CreateOffer()`,
  - Just negotiate the maximum possible
  - Maybe can introspect with Stefan's doohickey
- For `SetRemoteDescription()` there is no notification
  - Just negotiate the maximum possible
  - Not possible to introspect into what didn't get accepted