

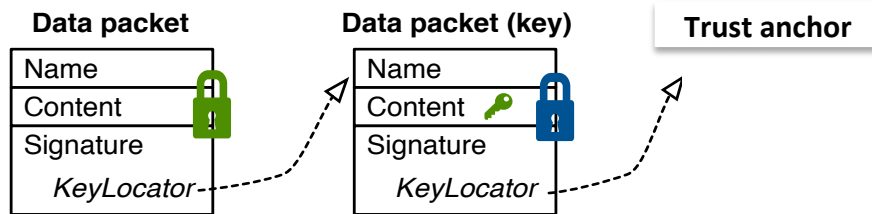
# **Trust Schema (Part 2)**

NDN Tutorial – ACM ICN 2015  
September 30, 2015

Jeff Thompson

# Goals

- See how to apply a trust schema in an application
- See how to use the trust API of the NDN client library



# Overview

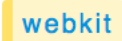

- All NDN apps need to sign/verify
- Review the client library sign/verify API
- Update FireChat to sign chat data from me and verify from others
- Deep dive: follow code to verify data

# CCL trust API

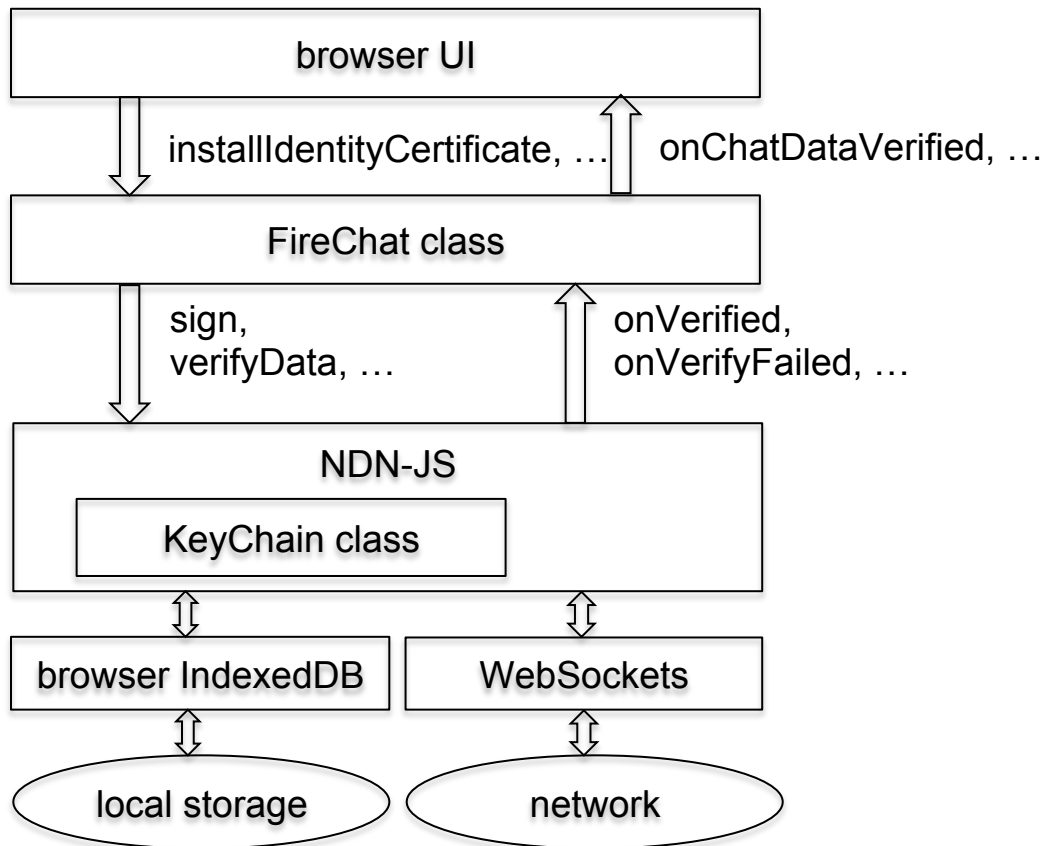
- Main class: KeyChain <http://named-data.net/doc/ndn-ccl-api/key-chain.html>
- ```
var keyChain = new KeyChain(new IdentityManager  
    (new IndexedDbIdentityStorage(),  
    new IndexedDbPrivateKeyStorage()),  
    new ConfigPolicyManager());
```
- keyChain.createIdentityAndCertificate, sign, verifyData
- Use IndexedDB key storage (next)
- ConfigPolicyManager for hierarchical verification (later)
- Key names: <http://redmine.named-data.net/projects/ndn-cxx/wiki/SecurityLibrary>

# In-browser key storage

- NDN apps need key storage. How to do this in the browser?
- IndexedDB: A new in-browser persistent storage API  
[https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB\\_API](https://developer.mozilla.org/en-US/docs/Web/API/IndexedDB_API)
- Use the Dexie wrapper <https://github.com/dfahlander/Dexie.js>
  - Implemented with promises
- NDN-JS: IndexedDbPrivateKeyStorage, IndexedDbIdentityStorage

| Chrome                                                                                    | Firefox (Gecko)                                                                                  | Internet Explorer | Opera | Safari (WebKit) |
|-------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------|-------------------|-------|-----------------|
| 23.0<br> | 10.0 (10.0)<br> | 10, partial       | 15    | 7.1             |
| 24                                                                                        | 16.0 (16.0)                                                                                      |                   |       |                 |

# FireChat application design



# Sign

- Previously in FireChat.onInterest:  
`this.keyChain.sign(data, this.certificateName, function() { ... });`
- Use certificateName when signing a data packet
- Get the key name from certificateName, get the private key from IndexedDB
- Add the SignatureSha256WithRsa with the signature bits to data
- Puts the certificate name as the data packet's signer  
`/ndn/org/icn/USER/alice%40ucla.edu/KEY/ksk-1443029460/ID-CERT`
- ChronoSync2013 already calls `sign` for sync messages

# Hierarchical validation

- NDN-JS: `policyManager = new ConfigPolicyManager();`  
`policyManager.load(policy, "chat-policy");`
- Hard-wired UCLA authority trust anchor self-signed certificate
- All chat users have a certificate under the trust anchor  
`/ndn/org/icn/USER/KEY/ksk-1442374173898/ID-CERT/%FD%00%00%01O%D45%8D%FC`
- Config details: <http://redmine.named-data.net/projects/ndn-cxx/wiki/CommandValidatorConf>



# Hierarchical validation policy

```
validator {  
  rule {  
    id "Chat Rule"  
    for data  
    filter {  
      type name  
      name /ndn/org/icmp  
      relation is-prefix-of  
    }  
    checker {  
      type hierarchical  
      sig-type rsa-sha256  
    }  
  }  
}
```

```
rule {  
  id "Sync Rule"  
  for data  
  filter {  
    type name  
    name /ndn/multicast/CHAT/CHANNEL  
    relation is-prefix-of  
  }  
  checker {  
    type customized  
    sig-type rsa-sha256  
    key-locator {  
      type name  
      regex ^<ndn><org><icmp><USER><><KEY><><ID-CERT>\$  
    }  
  }  
}
```

```
trust-anchor {  
  type base64  
  base64-string "Bv0DEQdBCAN  
uZG4IA29yZwgDaWNuCARVU0VSCANLRvkiEWtzay0xNDQ  
yMzc0MTczODk4CADJRC1DRVJUCAn9AAABT9Q1jfwUCRg  
BAhkEADbugBX9AXwwggF4MCIYDzIwMTUwOTE2MDMzMj  
jAzWhgPMjAzNTA5MTEwMzMyMDNANaMwwKgYDVQQpE  
yMvbmRuL29yZy9pY24vVFNuFi9rc2stMTQ0MjM3NDE3M  
zg5ODCCASlwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCgg  
EBALNdLoGnKHT+6YVW24MqHT3zMICgrID+YmcASLrMrF  
J0oMOR4glXPefaVAjvaQmxwNUriOxCaD/PmrudPgCPykrRl  
SL0hFYTeVcJRfMD+hDJMs1RkEo37q6i252f7v4dpYUlz96fS  
EC712YxsJ9Vh0mbaYtKGGQou0+IVewR0KQbQJS88Lyi/Vj6  
xWGxEaHAyHSPGKkip0EMehkqxeppi+Br9UGPDzNMB3OX  
eNuERcrcMS7z+qI+hgWoJAEvF7o4pEMyHkDRC6Y7JX75  
1WCTyWiEKoulC4xrQEv6Xq70A+6xmaPaxx4QX66ZZ6T+bb  
ulAZj+8bc0EApRuRmRrOqwCAwEAARY9GwEBHDgHNggD  
bmRuCANvcmclA2ljbggEVVNFUggDS0VZCBFrc2stMTQ0Mj  
M3NDE3Mzg5OAgHSUQtQ0VSVBf9AQChD7qJasfM2pLWR  
NY4Uz/GfsZyJEOQy5h9QaTNAAW3vxBg5PM3UO7joNyx  
dV1bUho5iQgutg3dLPr3NgG7sPuAjMGVoXxAKOGCEulluc0  
MV2zwNdwj/7ywp479TbDb/ysSfFi2oOV95Y/h8hZlVTRoud  
8mwc6LyeLsdkWbeY0e6BplB9Bga4Uvn+PglBoEwaWwO  
vBfvmPDwccOr22o9JVqbiWRi/ICULJ7uZUZYe82LoCTgaoQ  
qlna6FUTWfKrVhZfokinwHeDDtEw8QrzCW5kAvcPb7Ce  
FZzhFB5PH7b/f0n2ig6iLFhycl+hnrkFUD+KbHJLhWNqRA  
7TBJr"  
  }  
}
```

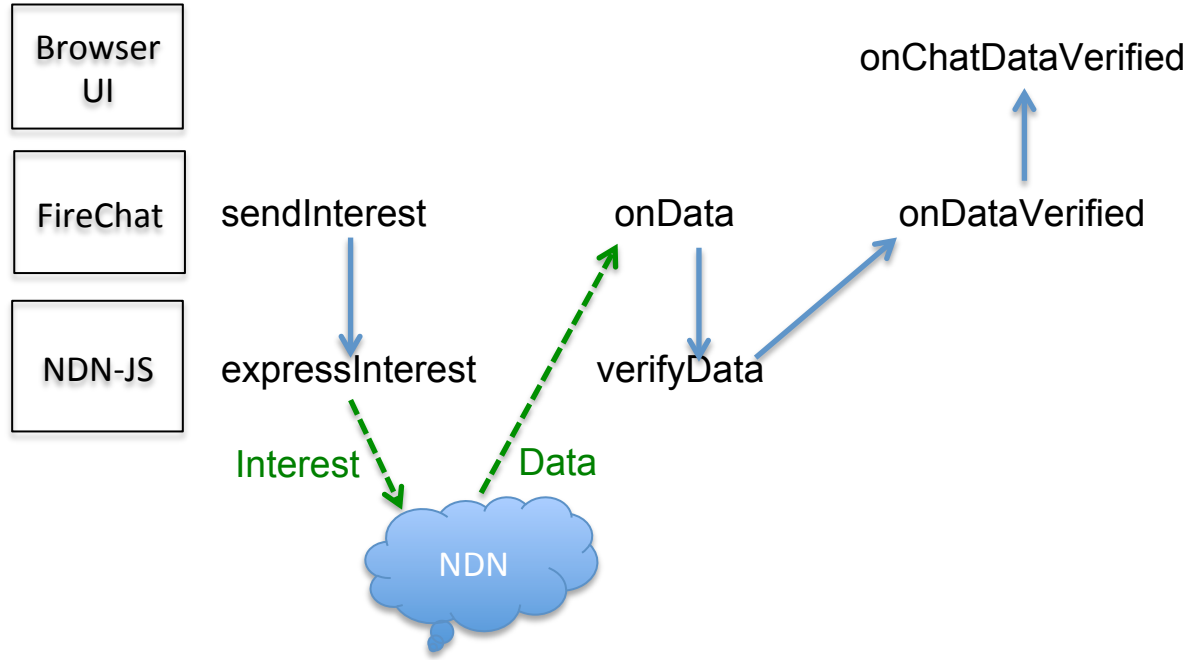
# Verify

- Name of the signer of Bob's data packet, e.g.:  
[/ndn/org/icn/USER/bob%40ucla.edu/KEY/ksk-1441413822/ID-CERT](#)

```
keyChain.verifyData(data,  
  function(data) { /* verified */ },  
  function(data) { /* failed */ });
```

- Callbacks for verified and failed
- Invokes the ConfigPolicyManager with the hierarchical policy
- Use the signer's name to fetch Bob's certificate from the test bed authority
- Use the hard-wired trust anchor to validate the chain

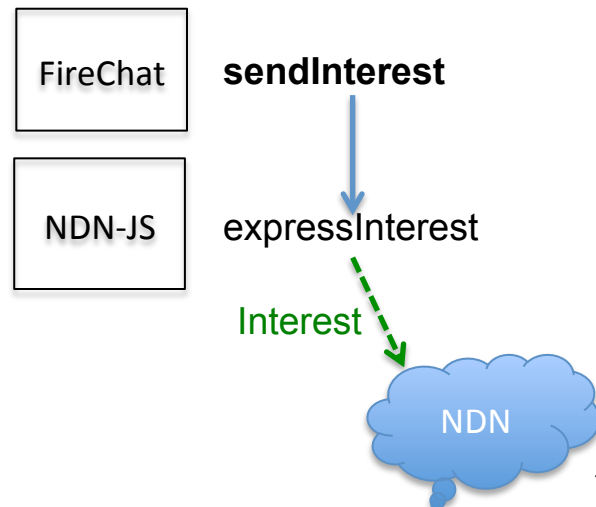
# Deep dive: Follow code to verify data



# Deep dive: FireChat.sendInterest

```
FireChat.prototype.sendInterest = function(syncStates, ...) {  
  // Scan syncStates, get the other user's new seq number.  
  var prefix =  
    "/ndn/org/icn/USER/bob%40ucla.edu/CHAT/CHANNEL/tutorial/SESSION/1442864410";  
  var seq = 5;  
  var interest = new Interest(new Name(dataPrefix).append(seq.toString()));  
  interest.setInterestLifetimeMilliseconds(this.chatInterestLifetime);  
  this.face.expressInterest(interest, this.onData.bind(this), ...);  
};
```

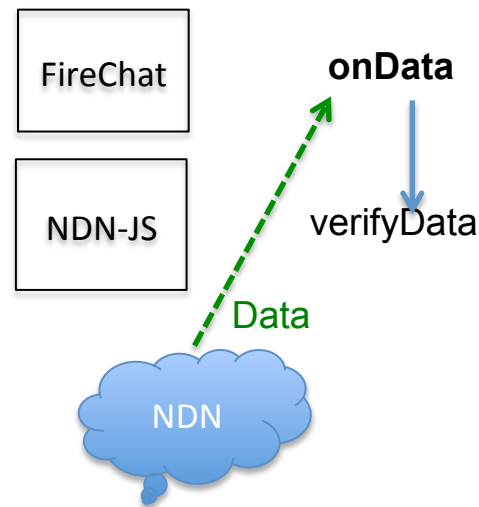
- sendInterest callback from ChronoSync2013 on receiving a new seq number
- Express an interest for Bob's chat message 5



# Deep dive: FireChat.onData

```
FireChat.prototype.onData = function(interest, data, ...) {  
  // Get the chat message from the data content.  
  this.onChatData  
    (screenName, onDataTimestamp, message, false, username, session, seqNo);  
  ...  
  var self = this;  
  this.keyChain.verifyData(data, function () {  
    self.onDataVerified  
      (data, updatePersistentStorage, content,  
        username, session, seqNo);  
    }, ...);  
  }  
};
```

- Initially display the message in gray
- Call verifyData to start verification with the ConfigPolicyManager
- onDataVerified is called when verified



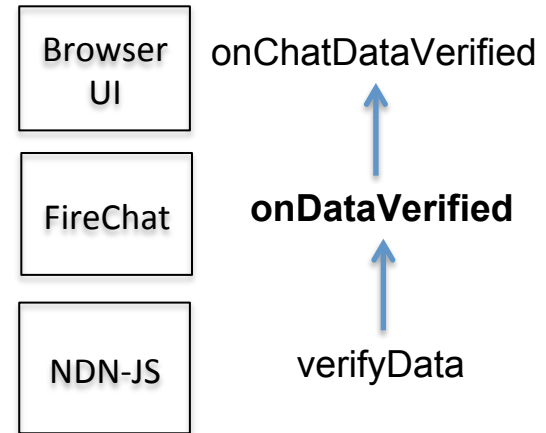
# Deep dive: KeyChain.verifyData

- Get the name of Bob's certificate from the data KeyLocator [/ndn/org/icn/USER/bob%40ucla.edu/KEY/ksk-1343029825/ID-CERT](#)
- Check if Bob's certificate is already in the ConfigPolicyManager cache
- If not, send interest, receive Bob's certificate from the certification authority, cache it
- Use Bob's public key to verify the data packet from Bob
- Get the signer name from the Bob's certificate's KeyLocator [/ndn/org/icn/USER/KEY/ksk-1442374173898/ID-CERT](#)
- Check if this is the trust anchor from the config policy (yes)
- Use the trust anchor certificate public key to verify Bob's certificate
- If verified call `onDataVerified` else `onVerifiedFailed`

# DeepDive: FireChat.onDataVerified

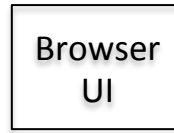
```
FireChat.prototype.onDataVerified = function  
  (data, updatePersistentStorage, content, name, session, seqNo) {  
  // Store verified chat data into persistent storage.  
  if (this.onChatDataVerified !== undefined)  
    this.onChatDataVerified(name, session, seqNo);  
};
```

- Called when data is verified



# Deep Dive: page.js onChatDataVerified

```
function onChatDataVerified(name, session, seqNo) {  
  var elementIdStr = name + session + seqNo.toString();  
  var para = document.getElementById(elementIdStr);  
  if (para) {  
    $(para).removeClass("unverified");  
    $(para).addClass("verified");  
  }  
}
```



**onChatDataVerified**

- Find the chat window line for the chat message, change to black



# Generating user keys

- Previously: `new FireChat(screenName, username, chatRoom, ...)`
  - “If needed, generate user keys”

```
var identityName = "/ndn/org/icn/USER/alice%40ucla.edu";  
keyChain.createIdentityAndCertificate  
(identityName, function(certificateName) { ... });
```

- (If the identity and keys already exist, return the self-signed certificate name)  
`/ndn/org/icn/USER/alice%40ucla.edu/KEY/ksk-1443029460/ID-CERT/%FD%00%00_OzB49`
- Generate RSA public/private keys, put in IndexedDB storage
- Set the key as the default for the identity name
- Create a self-signed certificate for the key, set as default
- `certificateName` is used in `keyChain.sign` (later)

# Get certified

- Alice authenticates `alice@ucla.edu` at authority [memoria.ndn.ucla.edu:5000](http://memoria.ndn.ucla.edu:5000)
- FireChat **Show certificate** button:
  - Use `certificateName` to get the self-signed certificate from IndexedDB
  - Display: `certificate.wireEncode().buf().toString('base64')`
- Alice pastes the base64 self-signed cert into the authority page
- The authority signs and stores Alice's certificate (and answers interests)
- Alice copies the certificate base64 from the authority's email
- FireChat **Install signed certificate** button:  

```
FireChat.prototype.installIdentityCertificate = function(signedCertString, ...) {  
  var certificate = new IdentityCertificate();  
  certificate.wireDecode(new Buffer(signedCertString, "base64"));  
  this.keyChain.installIdentityCertificate(certificate, ...);  
}
```
- `keyChain.installIdentityCertificate` stores the certificate in IndexedDB

# Goals recap

- See how to apply a trust schema in an application
- See how to use the trust API of the NDN client library