Proposal to expand the key discovery specification

The current WebCrypto key discovery specification describes a JavaScript API for discovering named, *origin-specific pre-provisioned* cryptographic keys. We propose to expand the specification by adding an API for discovering cryptographic keys not just their names but via any of their attributes; these keys will be known as "*pre-provisioned* cryptographic keys". This gives web applications a mean to find pre-provisioned keys whose names are unknown to them. Furthermore, we propose to define an AttributedKey that has additional attributes.

Use Case

Alice has a government issued signing certificate and the corresponding pre-provisioned private key for her to sign documents associated with the government.

Alice prepares a tax return using a web application of a tax preparation company (e.g. TurboTax, H&R Block, in US). Before submitting the tax return, Alice needs to sign it using her government issued signing key. The tax web application discovers the pre-provisioned key by using certain attributes of the key, for example, the issuer, the key type, the algorithm, and the key usage. With Alice's permission, the web application uses this pre-provisioned key to sign the tax return.

To support this use case, we propose the following two extensions to the key discovery specification.

Key discovery API

```
Dictionary FilterParam {
  DOMString attribute;
  DOMString value;
}

[NoInterfaceObject]
interface CryptoKeys {
    KeyOperation getKeysByName (DOMString name);
    KeyOperation getKeys( FilterParam[] filterparams );
};
```

Methods

getKeysByName

See current key discovery spec.

getKeys

When invoked, this method must perform the following steps:

Let KeyOp be a newly created object implementing the KeyOperation interface
Queue an operation to asynchronously run the following steps:

Search for a key or keys matching the FilterParam[] provided in filterparams. The FilterParam[] matches if all of the corresponding filterparams.value's match those of the key parameters' values. If filterparams is null, all keys are found.

If one or more keys are found

1. Let the result attribute of KeyOp be an object of type Key[] containing the keys
2. queue a task to fire a simple event called onsuccess at KeyOp

If no matching keys are found,

1. Let the result attribute of KeyOp be an object of empty array.
2. queue a task to fire a simple event called onsuccess at KeyOp

Otherwise

queue a task to fire a simple event called onerror at KeyOp

Return KeyOp to the task that invoked getKeys

Parameter	Type	Nullable	Optional	Description
filterparams	FilterParam[]	✘	✘	

Return type: KeyOperation

Attributed Key

```
Interface AttributedKey : NamedKey {
  readonly attribute DOMString? issuer;
  readonly attribute bool?      isPrivate;  //cannot access without authentication
}
```

AttributedKey interface members:

issuer

The Issuer of the underlying key.

isPrivate

If the key requires a prior authentication before it can be used

Example

```
var filterParams = new FilterParam(
                          [{attribute: “type”    , value: “Private”},
                           {attribute: “issuer”   , value: “eGov”   },
                           {attribute: “keyUsage” , value: “Sign”   },
                           {attribute: “isPrivate”, value: “True”   }] );

var op = window.CryptoKeys.getKeys( filterParams );

op.oncomplete = function(e) {
    var keys[] = e.target.result;
    if (keys.length > 0)
    {
        console.log( "First Key returned: " + keys[0].toString() );
    }
}
```