O'REILLY®

Fluent

THE WEB PLATFORM

fluentconf.com

#fluentconf

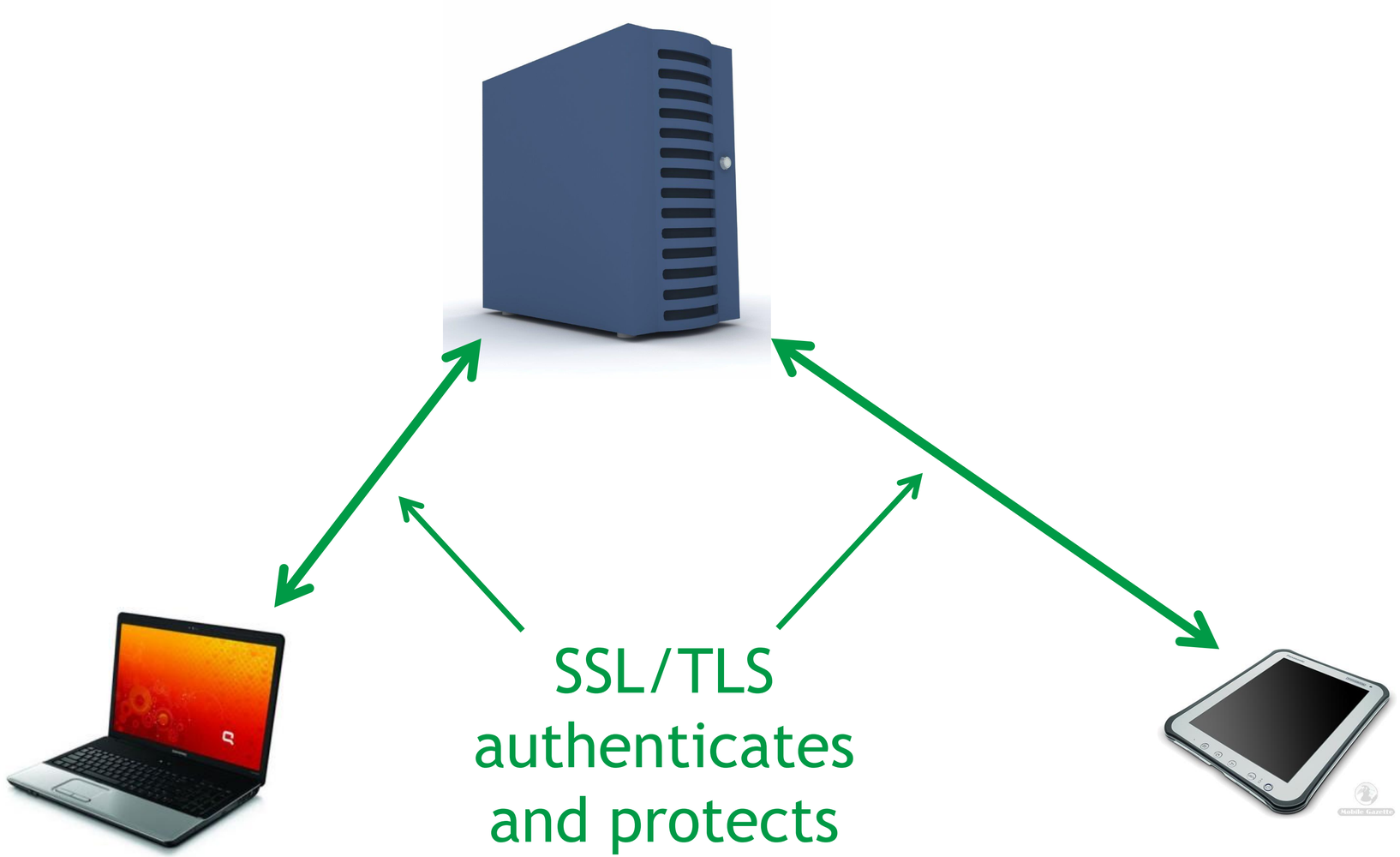
Cryptography in the Browser

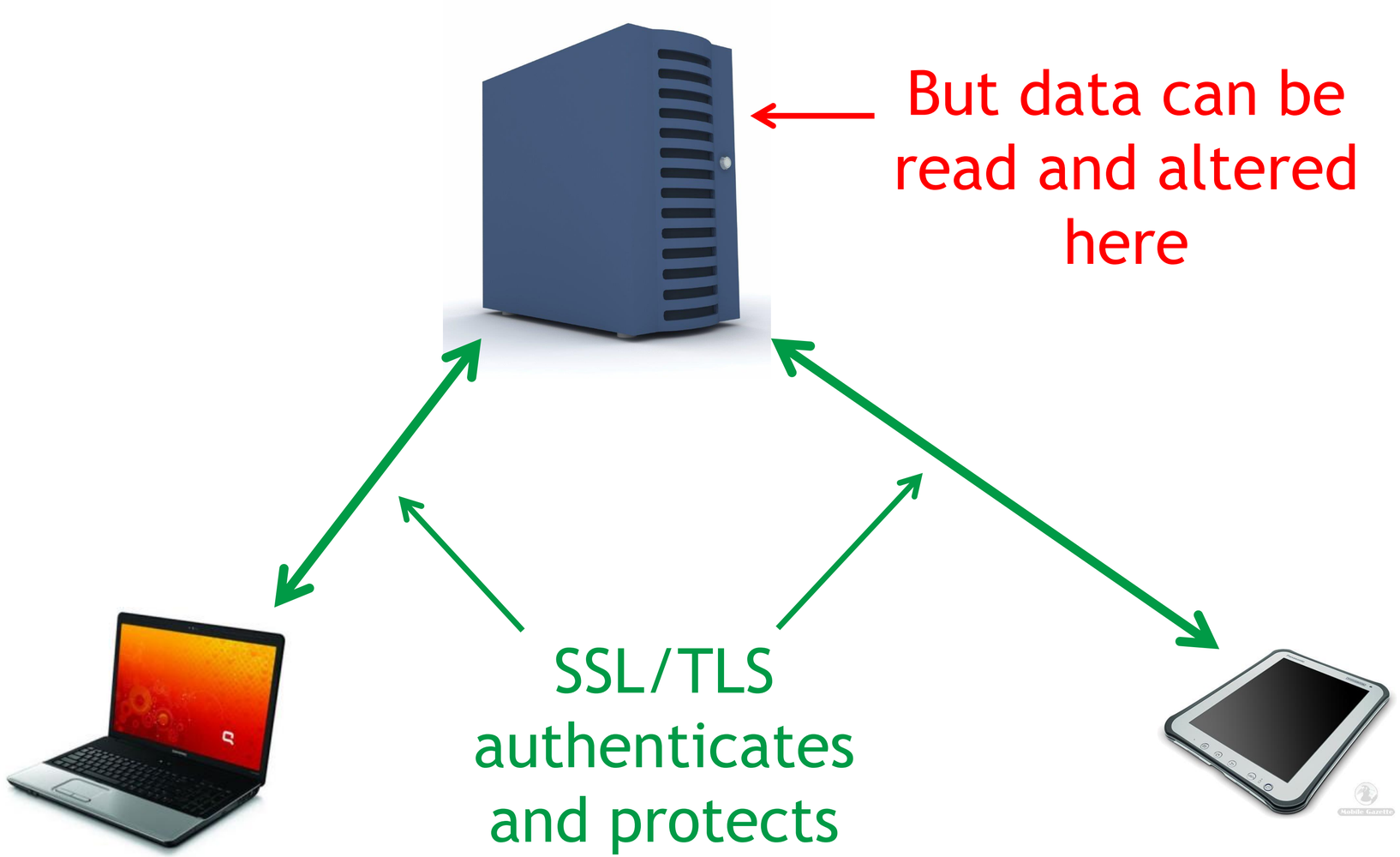
Charles Engelke
Info Tech, Inc.

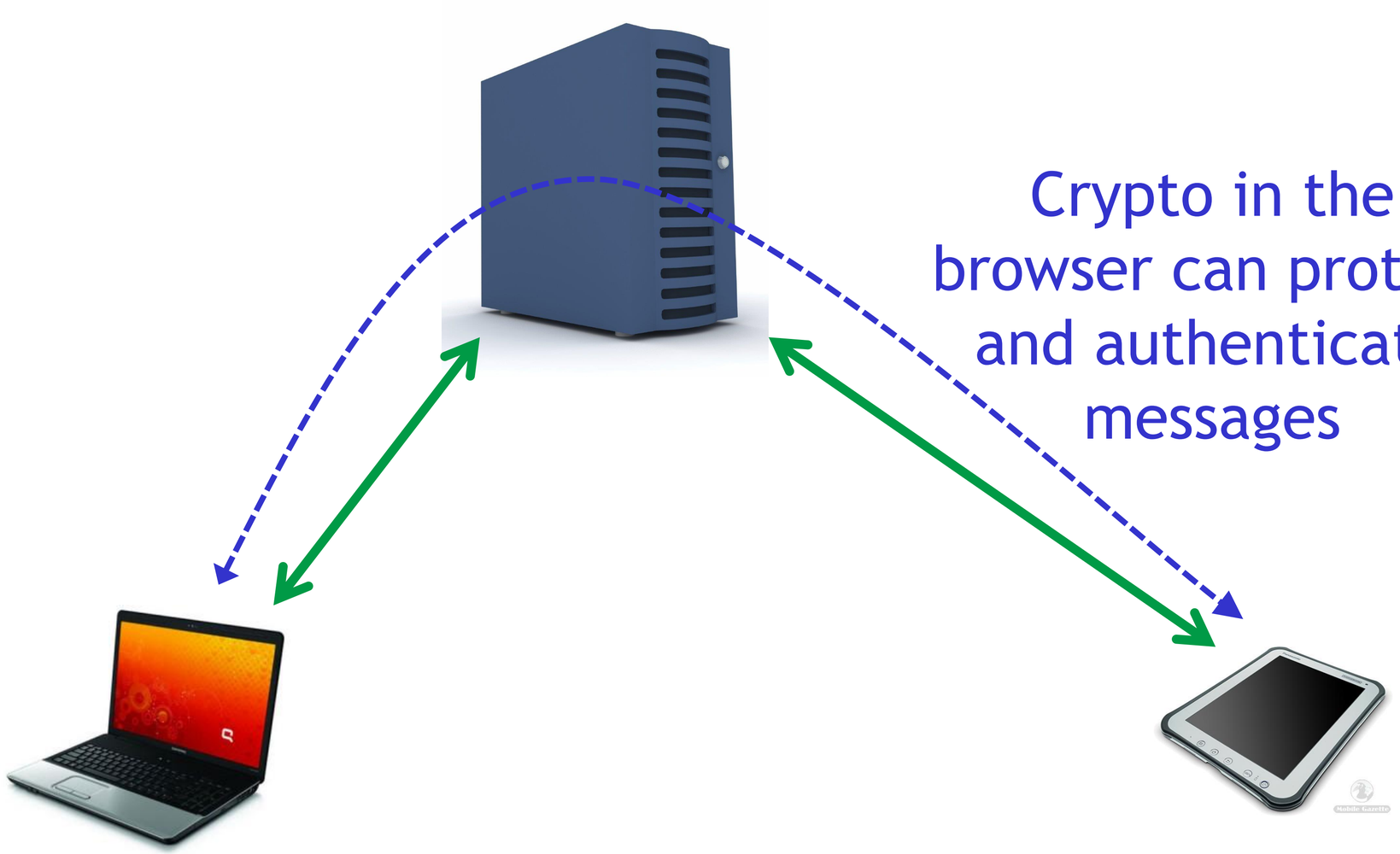
@charlesengelke
engelke.com/fluent

My Goal

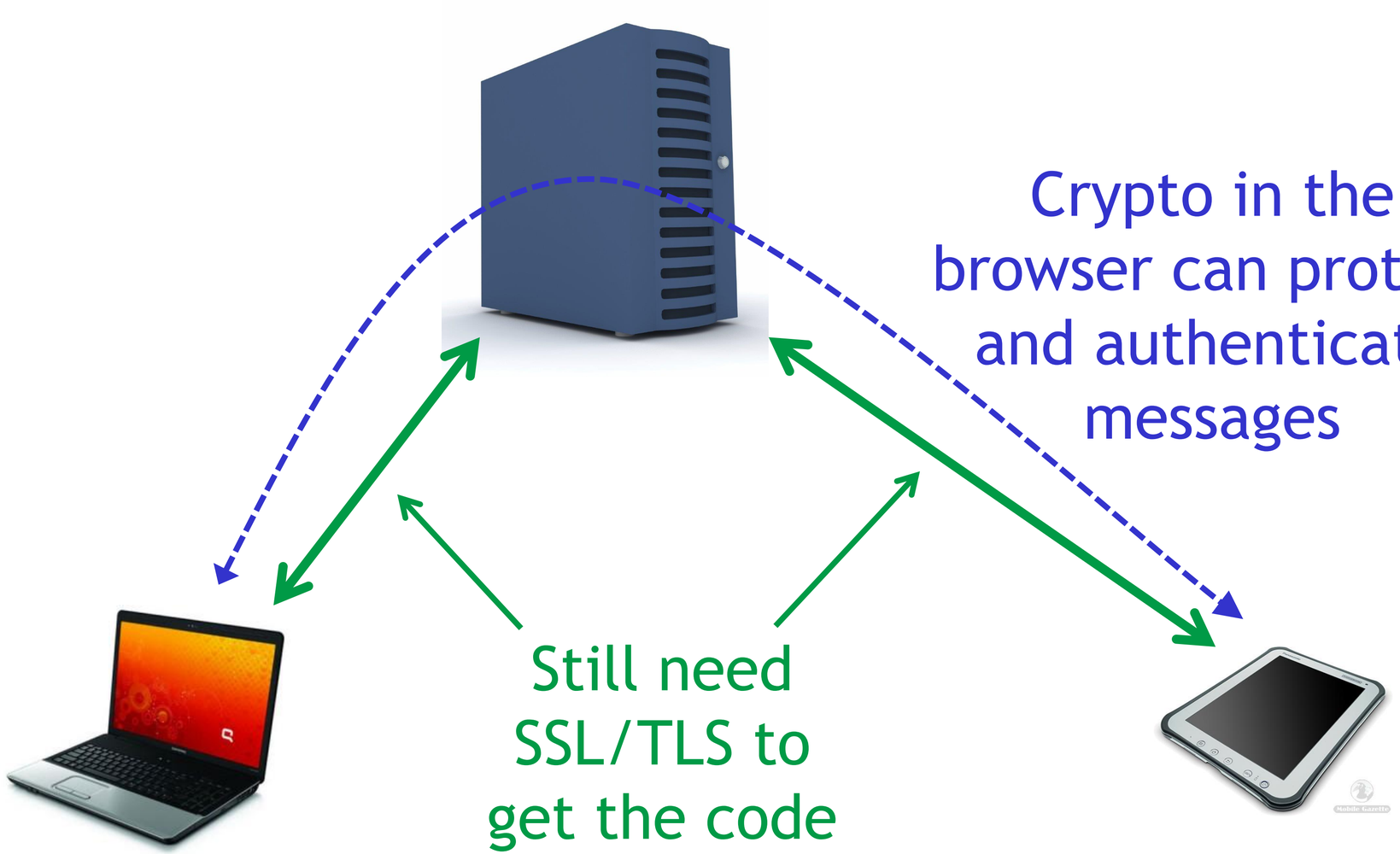
Protect messages between two browser users from disclosure or tampering


SSL/TLS
authenticates
and protects
connections


But data can be read and altered here
SSL/TLS authenticates and protects **connections**



Crypto in the browser can protect and authenticate messages

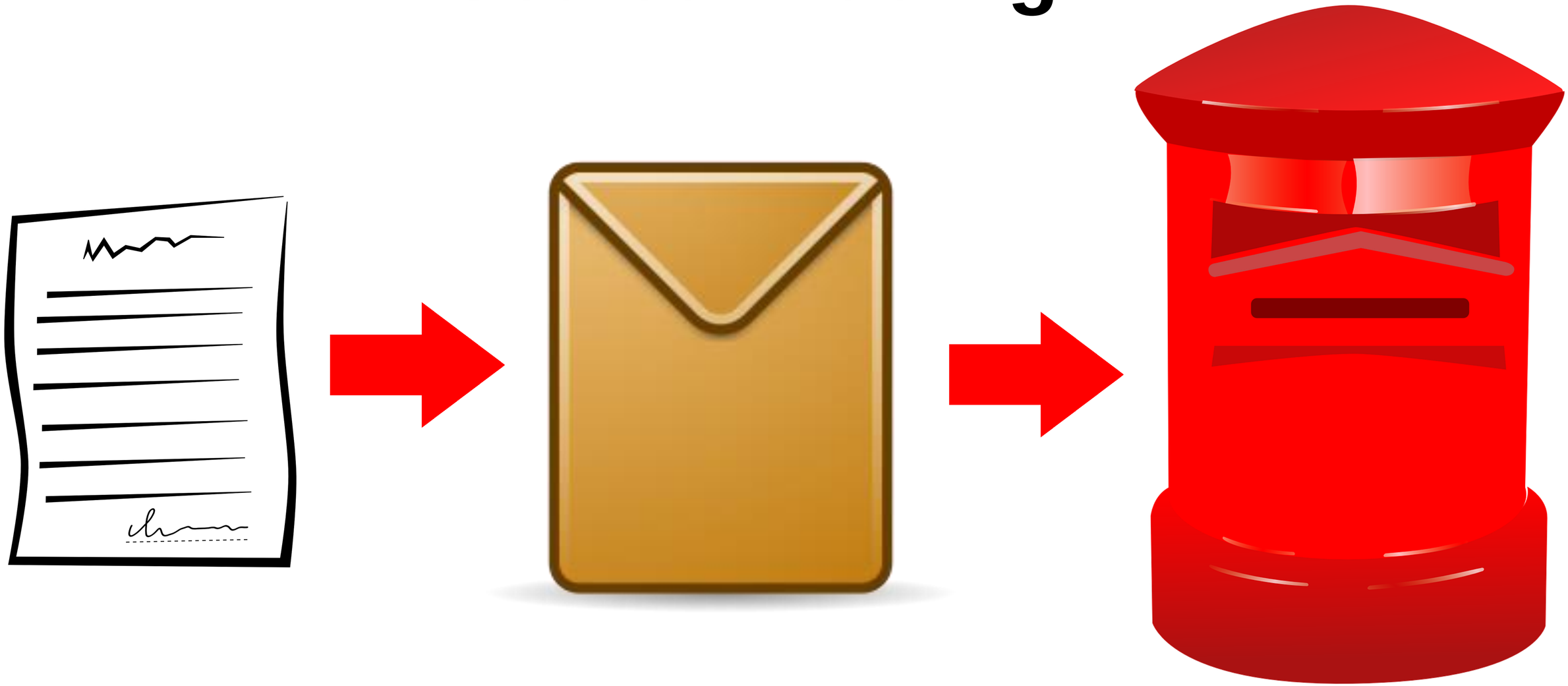

Crypto in the
browser can protect
and authenticate
messages
Still need
SSL/TLS to
get the code
to the
browser

InfoTech™
INNOVATION AT WORK

Software for Government Transportation Agencies

http://www.fhwa.dot.gov/publications/publicroads/04mar/01.cfm

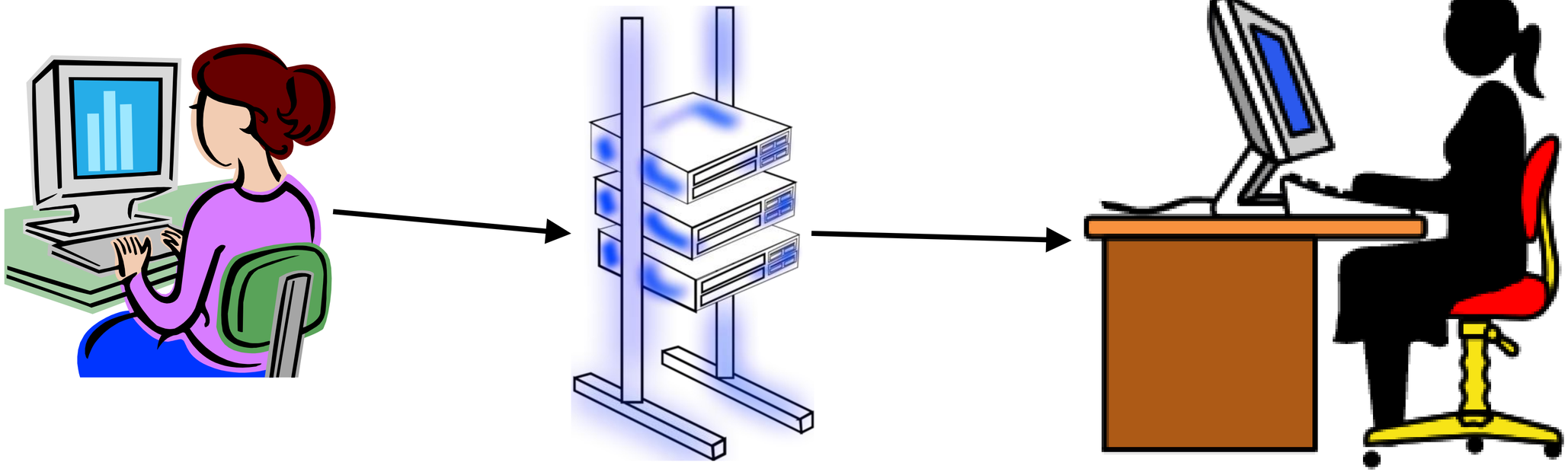
Sealed Bidding

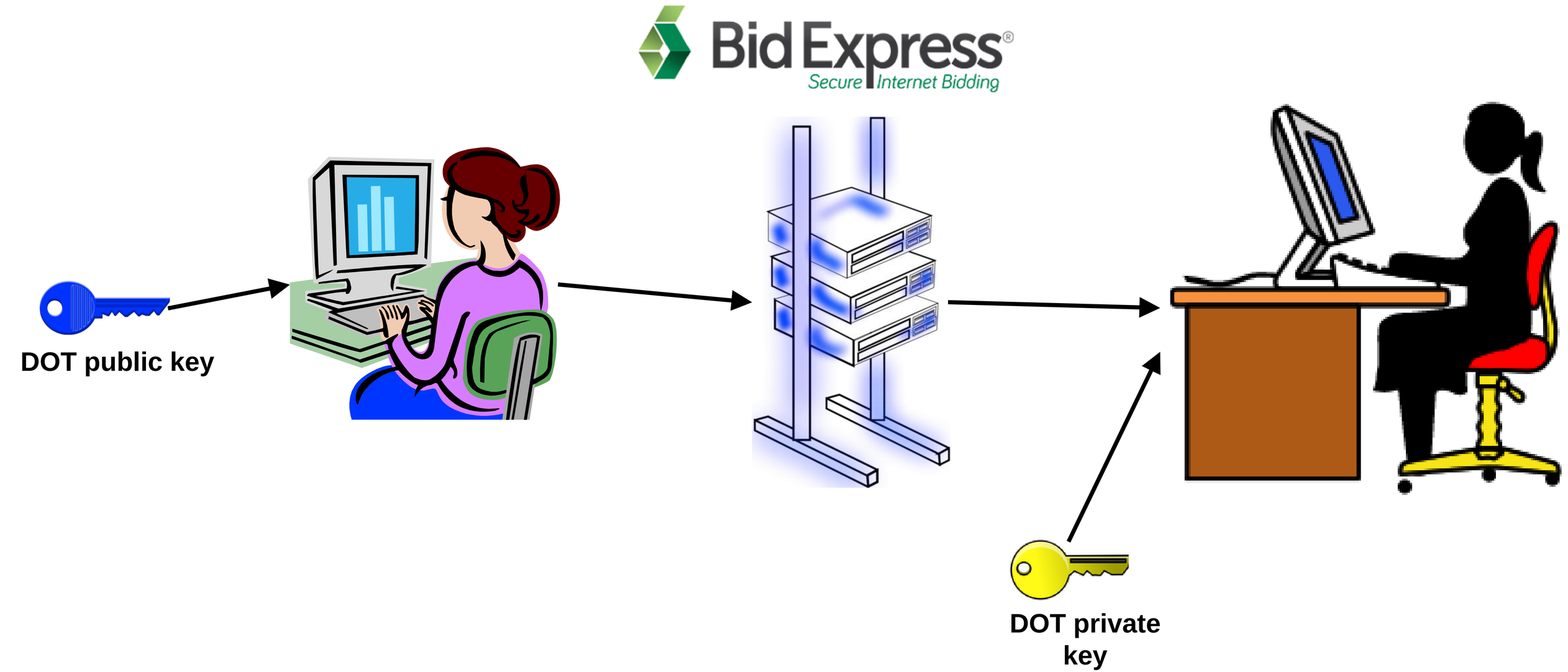


Bid Opening Day

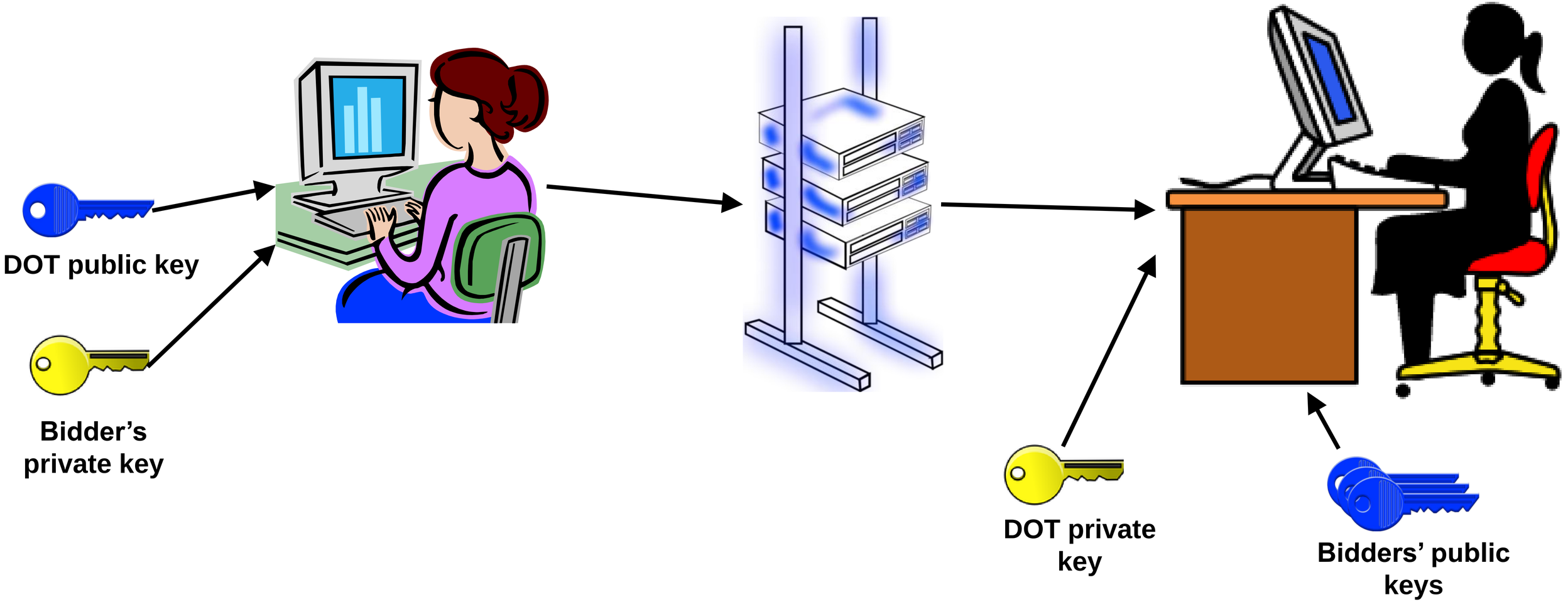
https://www.flickr.com/photos/agecombahia/5726490081

Bid Express®
Secure Internet Bidding




Bid Express®
Secure Internet Bidding
DOT public key
DOT private key

Bid Express®
Secure Internet Bidding



37 US state DOTs

1 Canadian MOT

Many other public transportation agencies

Branching out to other sealed bid users

> $1,000,000,000,000 in bids

Could we use a browser instead of a Windows program?

JavaScript and the browser VM are not well suited to cryptography.

Obstacles can be overcome (and have been)… but:

We'd rather use existing, time-tested libraries

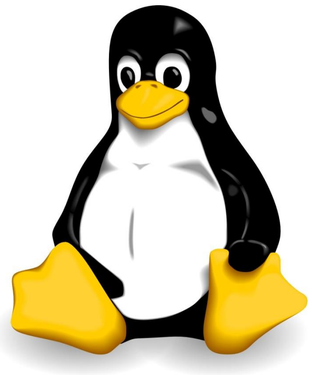
Web Cryptography API

W3C Candidate Recommendation *11 December 2014*



Still Prefixed



Using the API

window.crypto

window.crypto.getRandomValues

window.crypto.subtle

window.crypto.subtle.

- encrypt
- decrypt
- sign
- verify
- digest
- generateKey
- deriveKey
- deriveBits
- importKey
- exportKey
- wrapKey
- unwrapKey

Subtle?

It is named SubtleCrypto to reflect the fact that many of these algorithms have **subtle usage requirements** in order to provide the required algorithmic **security guarantees.**

(emphasis mine)

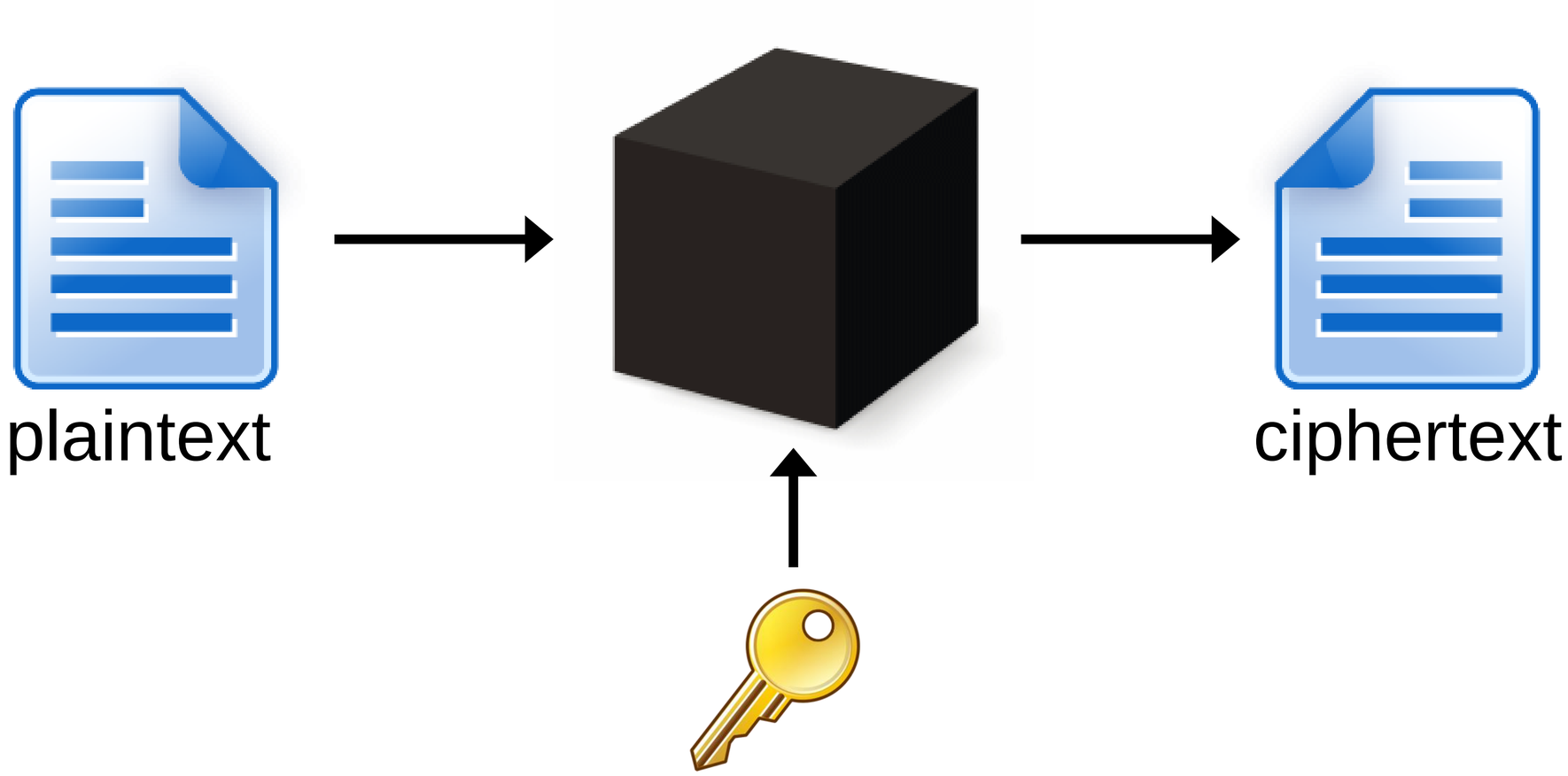
Unreadable W3 specs

Now it was suggested that the correct way of doing this can be found in the specification. That's entirely possible, but I am unable to make heads or tails of the spec.

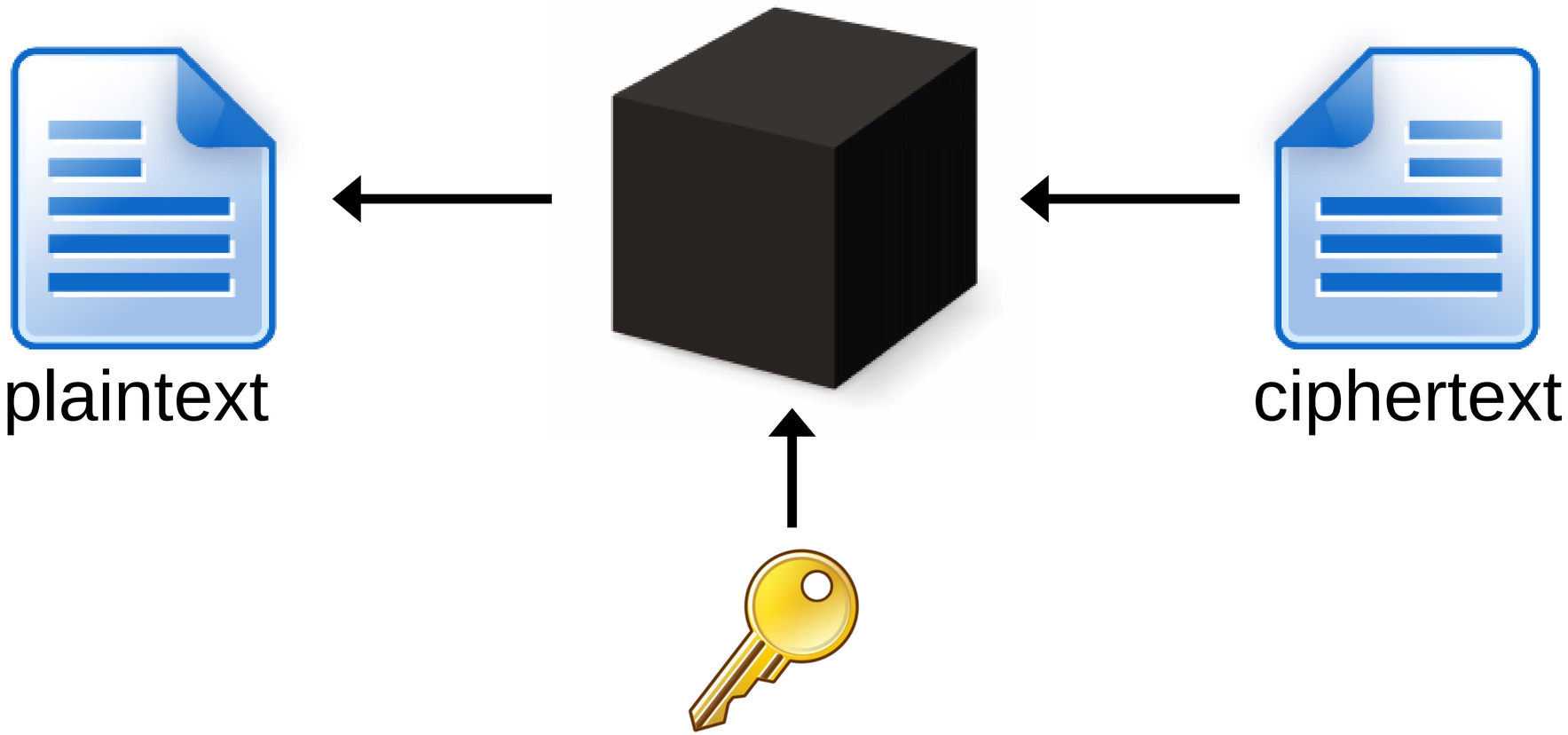
Peter-Paul Koch

Give it a try

Symmetric Cryptography



Works Either Way



```
window.crypto.subtle.encrypt(
      algorithmIdentifier,
      key,
      data);
```

First Try

```
var ciphertext =
window.crypto.subtle.encrypt(
    "AES",
    0x1234567890abcdef01234567890abcdef0,
    "This is really super secret!");
```

First Try

```
var ciphertext =
window.crypto.subtle.encrypt(
    "AES",
    0x1234567890abcdef01234567890abcdef0,
    "This is really super secret!");
```

```
var ciphertext =
window.crypto.subtle.encrypt(
```

Returns a Promise resolving to the ciphertext, not the ciphertext itself

```
"AES",
```

Not an AlgorithmIdentifier object

```
{
    name: "AES-CBC",
    iv: initVec //16 bytes
}
```

`0x1234567890abcdef01234567890abcdef0,`

Not a CryptoKey object

```
"This is really super secret!");
```

Must be BufferSource, not string.

Promises and ArrayBuffers

Using a Promise

```
p.then(function(result) {
    // It worked and yielded result
}, function(err) {
    // It failed. err is usually an Error object
});
```

Either parameter of **then** can be omitted:

```
p.then(resolve);
p.then(, reject);
```

`p.catch(reject)` is an alias for `p.then(, reject)`

`p.then()` always returns another Promise

Promises can be chained

ArrayBuffer

A contiguous block of memory

```
var buf = new ArrayBuffer(8);
```

Cannot access or manipulate contents directly.

ArrayBufferView

```
var view = new Uint8Array(buf);
```

or a shortcut:

```
var view = new Uint8Array(
    [1, 2, 3, 4, 5, 6, 7, 8]);
```

Do it right

```
var keyBuffer = new Uint8Array([
    0x12, 0x34, 0x56, 0x78, 0x9a, 0xbc, 0xde, 0xf0,
    0x12, 0x34, 0x56, 0x78, 0x9a, 0xbc, 0xde, 0xf0]);
var initVec = window.crypto.getRandomValues(
    new Uint8Array(16));
var plaintext = new TextEncoder("utf-8").
    encode("This is super secret!");
```

```
0x12, 0x34, 0x56, 0x78, 0x9a, 0xbc, 0xde, 0xf0]);
var initVec = window.crypto.getRandomValues(
    new Uint8Array(16));
var plaintext = new TextEncoder("utf-8").
    encode("This is super secret!");
window.crypto.subtle.importKey(
    'raw', keyBuffer, {name: "AES-CBC"},
    false, ["encrypt", "decrypt"]
).
```

```
window.crypto.subtle.importKey(
    'raw', keyBuffer, {name: "AES-CBC"},
    false, ["encrypt", "decrypt"]
).
then(function(key) {
    return window.crypto.subtle.encrypt(
        {name: "AES-CBC", iv: initVec},
        key, plaintext);
}).
```

```
then(function(key) {
    return window.crypto.subtle.encrypt(
        {name: "AES-CBC", iv: initVec},
        key, plaintext);
}).
then(function(ciphertext) {
    useCipherText(ciphertext, initVec);
}).
```

```
then(function(ciphertext) {
    useCipherText(ciphertext, initVec);
}).
catch(function(err) {
    alert("Error: " + err.message);
});
```

Finding info in the spec

Section 19: Algorithms

`encrypt` and **`decrypt`**:

`RSA-OAEP`, **`AES-CTR`**, **`AES-CBC`**, **`AES-GCM`**, **`AES-CFB`**

`sign` and **`verify`**:

`RSASSA-PKCS1-v1_5`, **`RSA-PSS`**, **`ECDSA`**, **`AES-CMAC`**, **`HMAC`**

`digest`:

`SHA-1`, **`SHA-256`**, **`SHA-384`**, **`SHA-512`**

`deriveKey` and **`deriveBits`**:

`ECDH`, **`DH`**, **`CONCAT`**, **`HKDF-CTR`**, **`PBKDF2`**

`wrapKey` and **`unwrapKey`**:

All encrypt and decrypt algorithms, plus **`AES-KW`**

There are **no** algorithms that conforming user agents are required to implement

Widely Supported Algorithms

RSASSA-PKCS1-v1_5 with SHA-1 or SHA-256

RSA-OAEP

AES-CBC

SHA-1, **SHA-256**, **SHA-512**

PBKDF2 with **SHA-1**

Section 14: SubtleCrypto interface

```
Promise<any> generateKey(AlgorithmIdentifier algorithm,
                         boolean extractable,
                         sequence<KeyUsage> keyUsages );
```

19. Algorithm Overview
20. RSASSA-PKCS1-v1_5
    20.1. Description
    20.2. Registration
    20.3. RsaKeyGenParams dictionary
    20.4. RsaHashedKeyGenParams dictionary
    20.5. RsaKeyAlgorithm dictionary
    20.6. RsaHashedKeyAlgorithm dictionary
    20.7. RsaHashedImportParams dictionary
    20.8. Operations

Summary of operations and parameters

The recognized algorithm name for this algorithm is `"RSASSA-PKCS1-v1_5"`.

Operation	Parameters	Result
sign	None	ArrayBuffer
verify	None	boolean
generateKey	RsaHashedKeyGenParams	CryptoKeyPair
importKey	RsaHashedImportParams	CryptoKey
exportKey	None	object

```
dictionary RsaHashedKeyGenParams : RsaKeyGenParams {
  // The hash algorithm to use
  required HashAlgorithmIdentifier hash;
};

dictionary RsaKeyGenParams : Algorithm {
  // The length, in bits, of the RSA modulus
  [EnforceRange] required unsigned long modulusLength;
  // The RSA public exponent
  required BigInteger publicExponent;
};
```

```
window.crypto.subtle.generateKey(
    {
        name: "RSASSA-PKCS1-v1_5", hash: "SHA-256",
        modulusLength: 2048,
        publicExponent: new Uint8Array([1, 0, 1])
    },
    false, // privateKey only
    ["sign", "verify"]
).
```

65537 (per RFC 6485)
as a 24-bit
big endian integer

```
window.crypto.subtle.generateKey(
    {
        name: "RSASSA-PKCS1-v1_5", hash: "SHA-256",
        modulusLength: 2048,
        publicExponent: new Uint8Array([1, 0, 1])
    },
    false,
    ["sign", "verify"]
).then(function(keyPair) {
    // use keyPair.publicKey and keyPair.privateKey
});
```

Winding Down

X.509 and CMS

PKIX standards like x.509 certificates and Cryptographic Message Syntax can be implemented on top of the Web Cryptography API.

PKIX adds standardized formatting to results, using ASN.1 and BER/DER encoding.

(1980's era ITU standards)

Can "roll your own" with JavaScript…

or:

Use PKIjs and ASN1js libraries

At **pkijs.org** and **asn1js.org**

See **github.com/infotechinc/create-x509-certificate**

Stanford

Cryptography I

Learn about the inner workings of cryptographic primitives and how to apply this knowledge in real-world applications!

Preview Lectures

engelke.com/fluent

github.com/infotechinc

blog.engelke.com/webcrypto