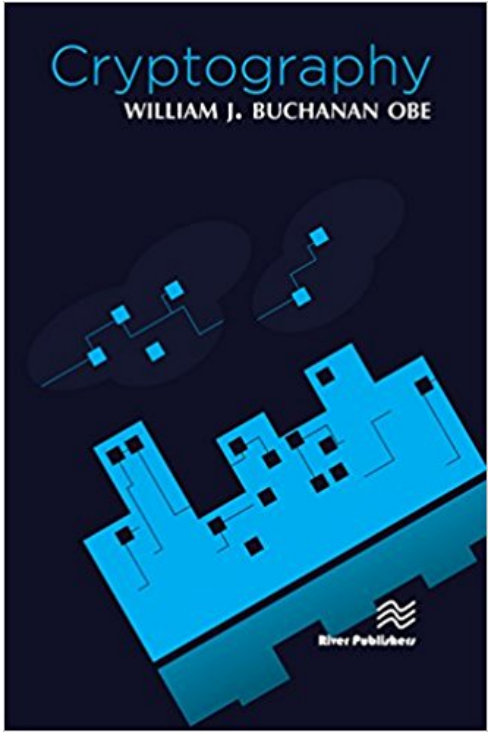
Tokens, Authorization and Docker

ERC-20 Tokens
JSON Web Tokens
OAuth 2.0
Fernet Tokens
PAKE
Docker

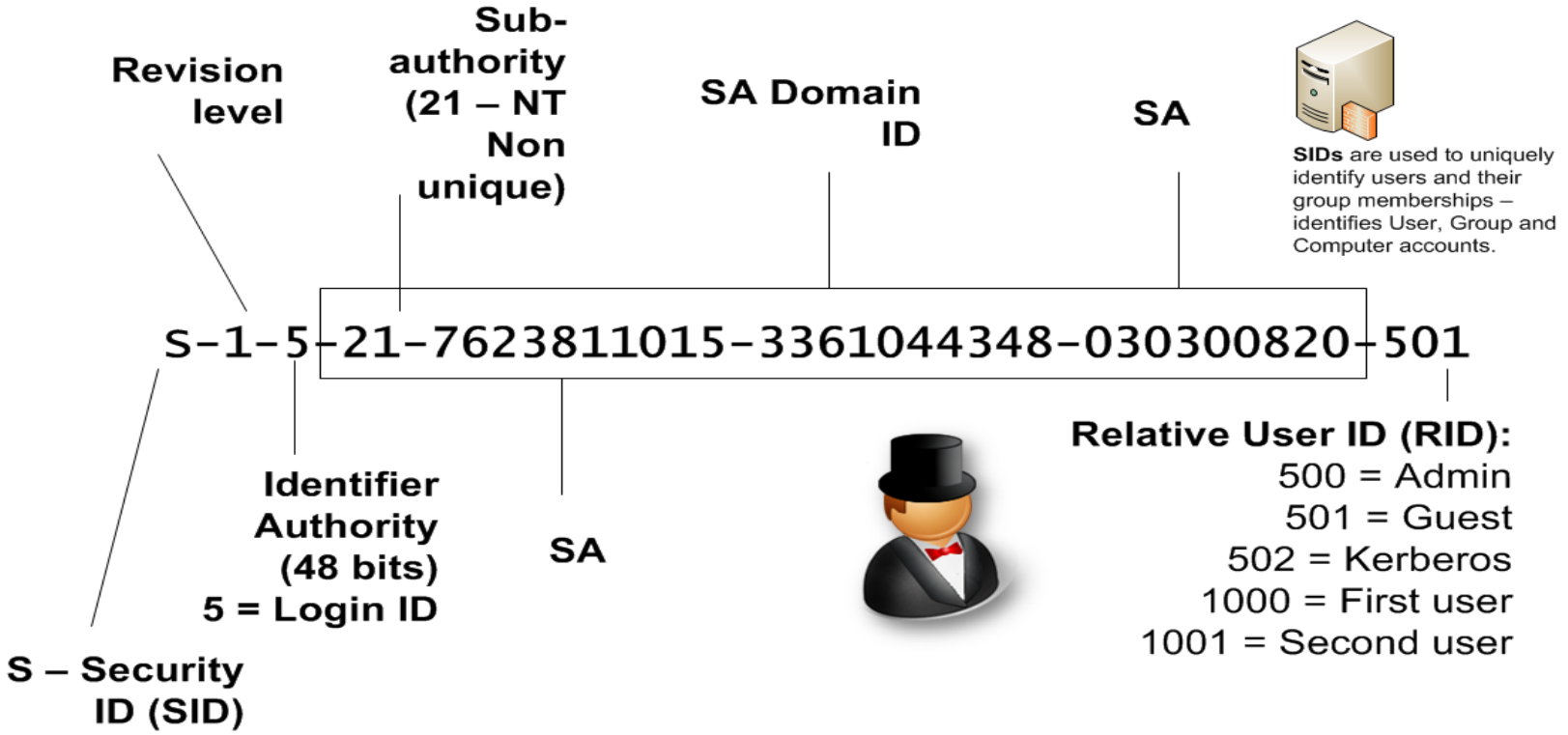
Prof Bill Buchanan OBE

http://asecuritysite.com/esecurity

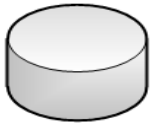


Authorization and Access

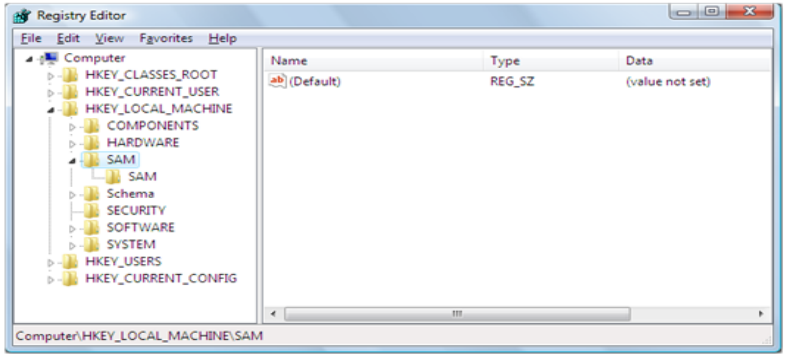
```
C:> user2sid \pluto guest
S-1-5-21-7623811015-3361044348-030300820-501
C:> sid2user 5 21 7623811015 3361044348 030300820 500
Name is Fred
Domain is PLUTO
```

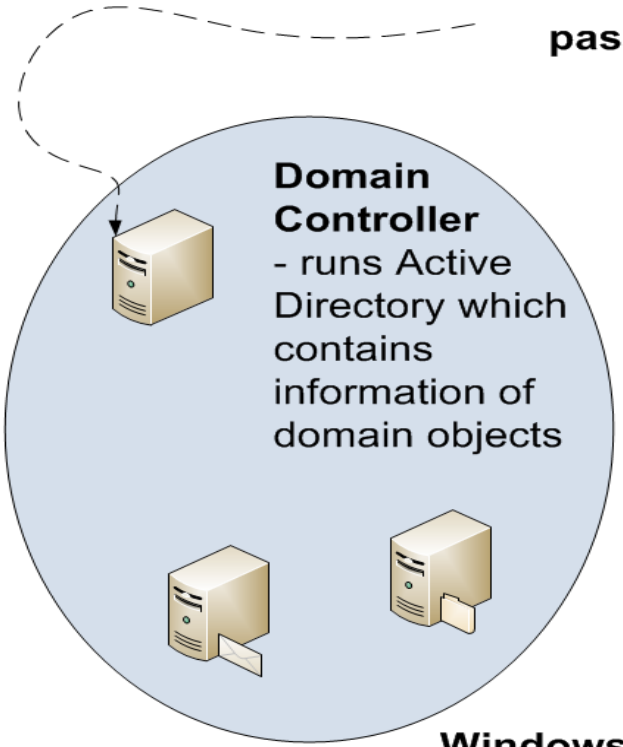


HKLM\SAM



SAM Database (stores usernames and passwords)



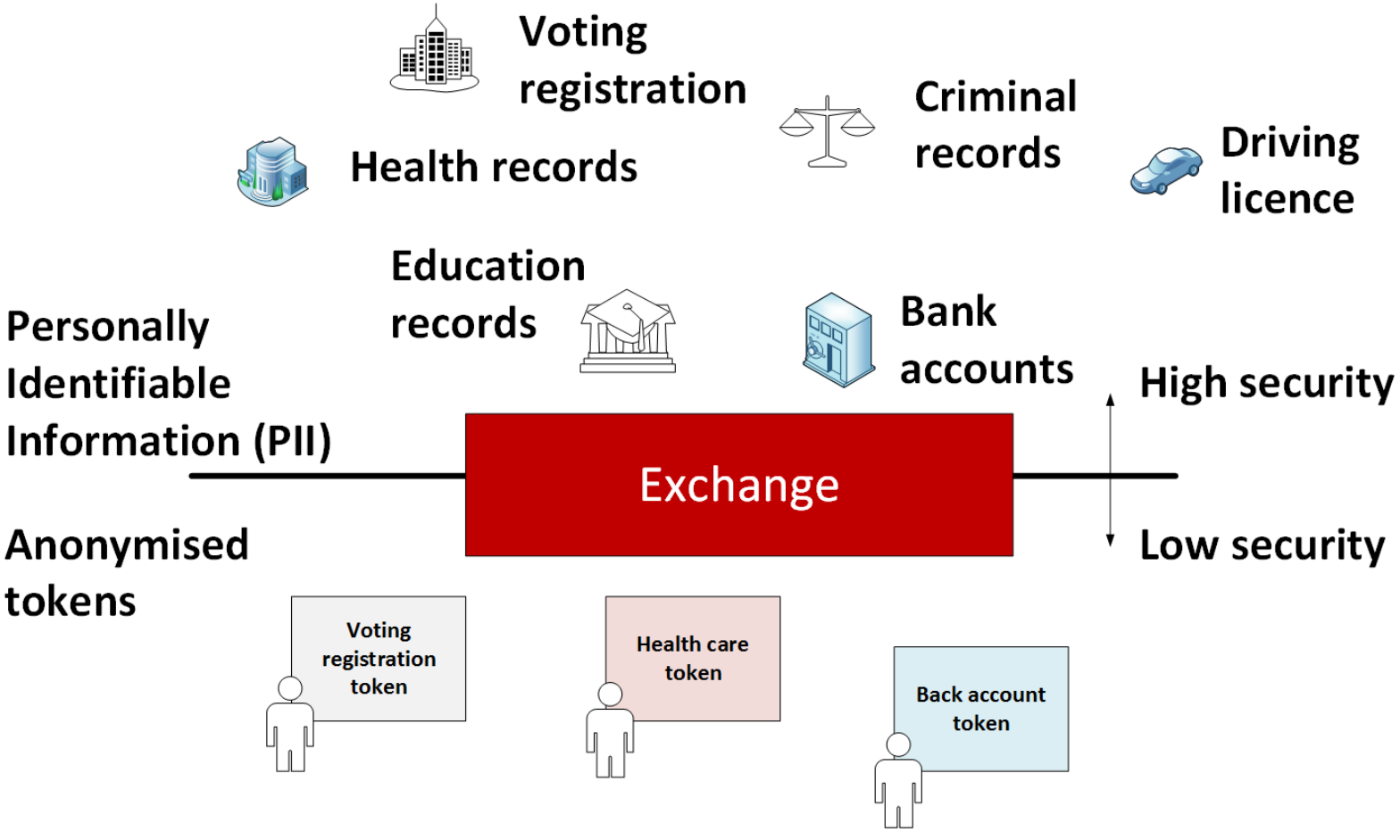


Local Authority Subsystem (Lsass) – Windows Security mechanism – Attached by Sasser Worm which exploited a buffer overflow

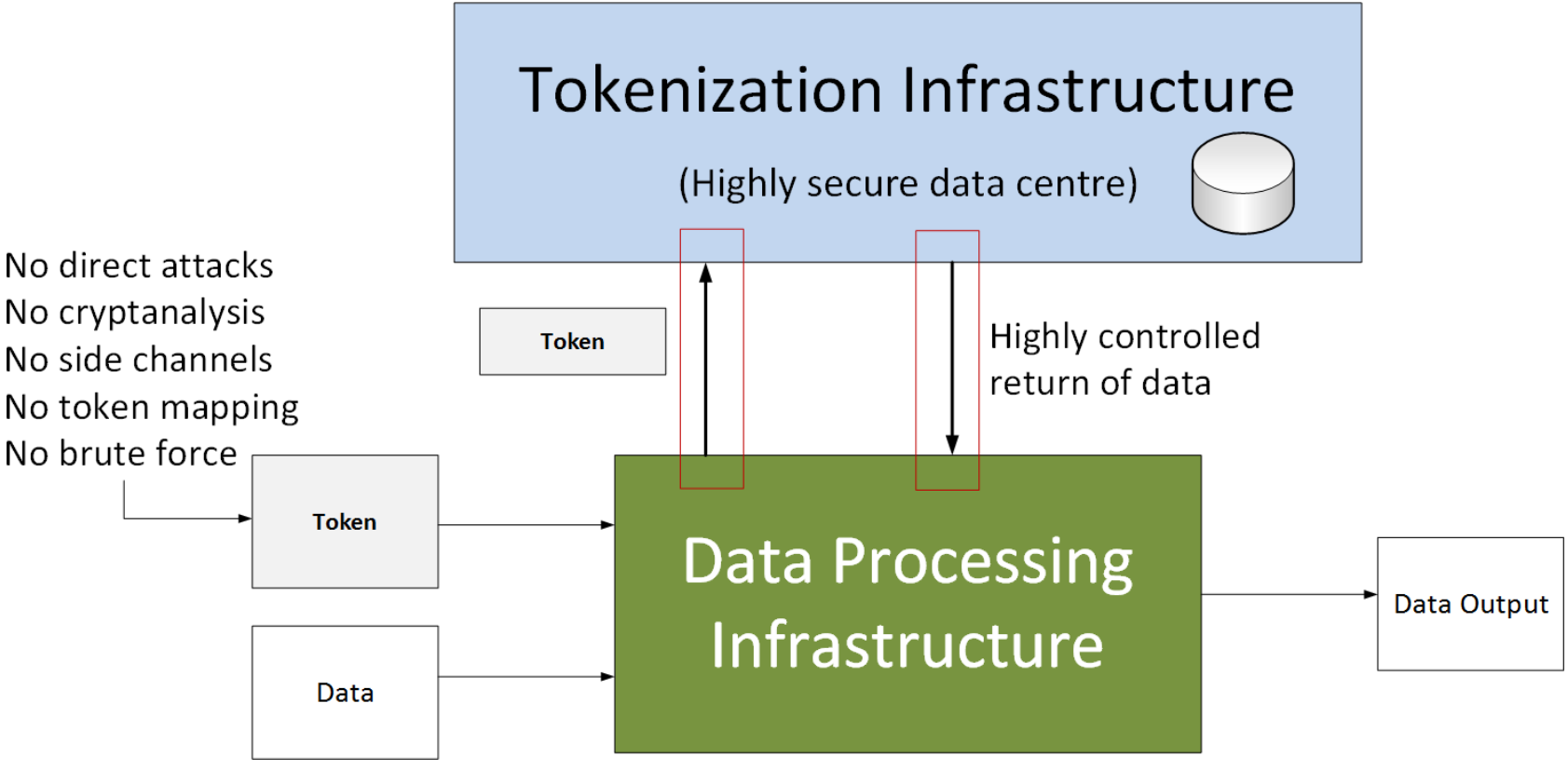
Responsible for local security policy

- Controls access.
- Managing password policies.
- User authentication.
- Audit messages.

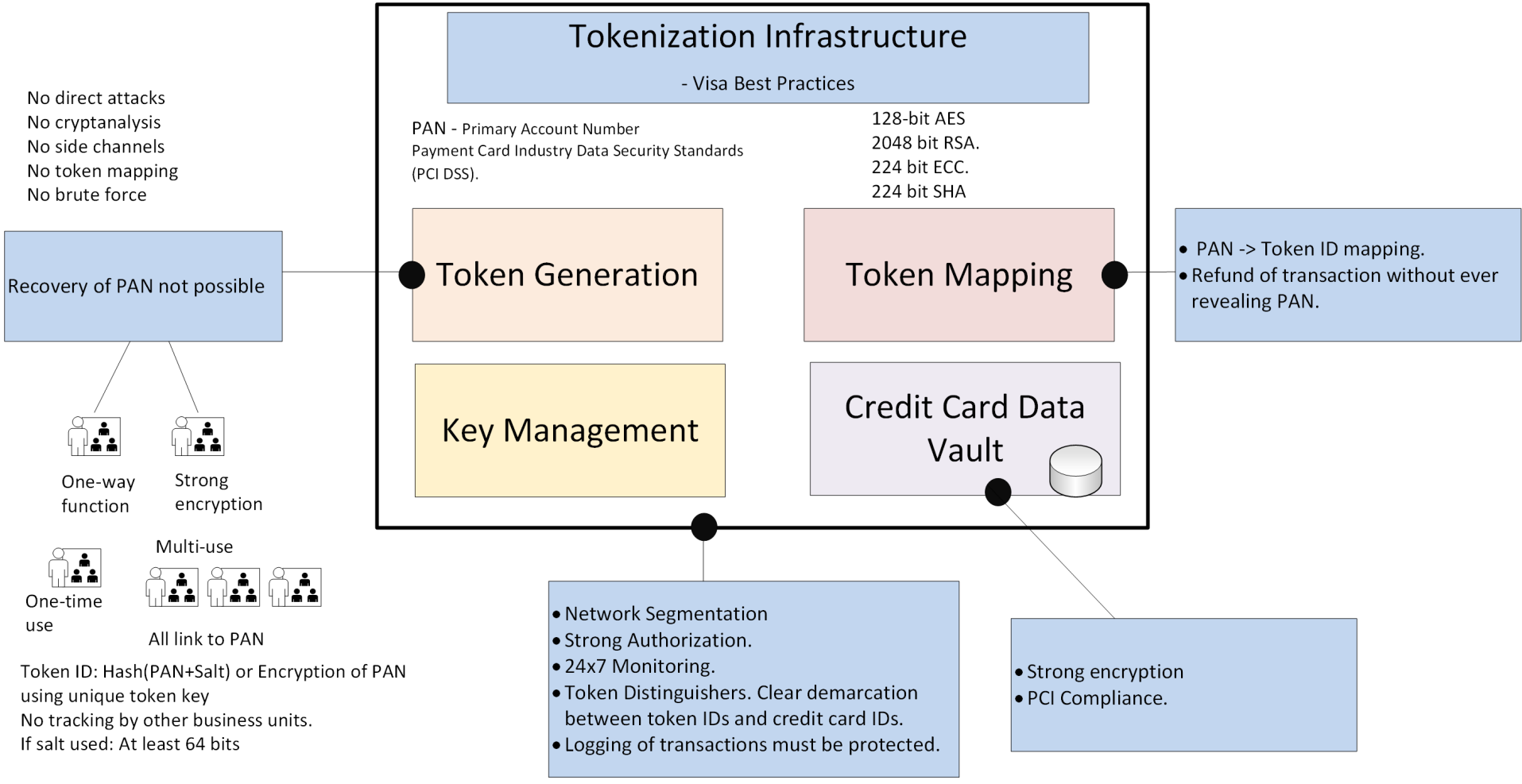
Tokenization with data



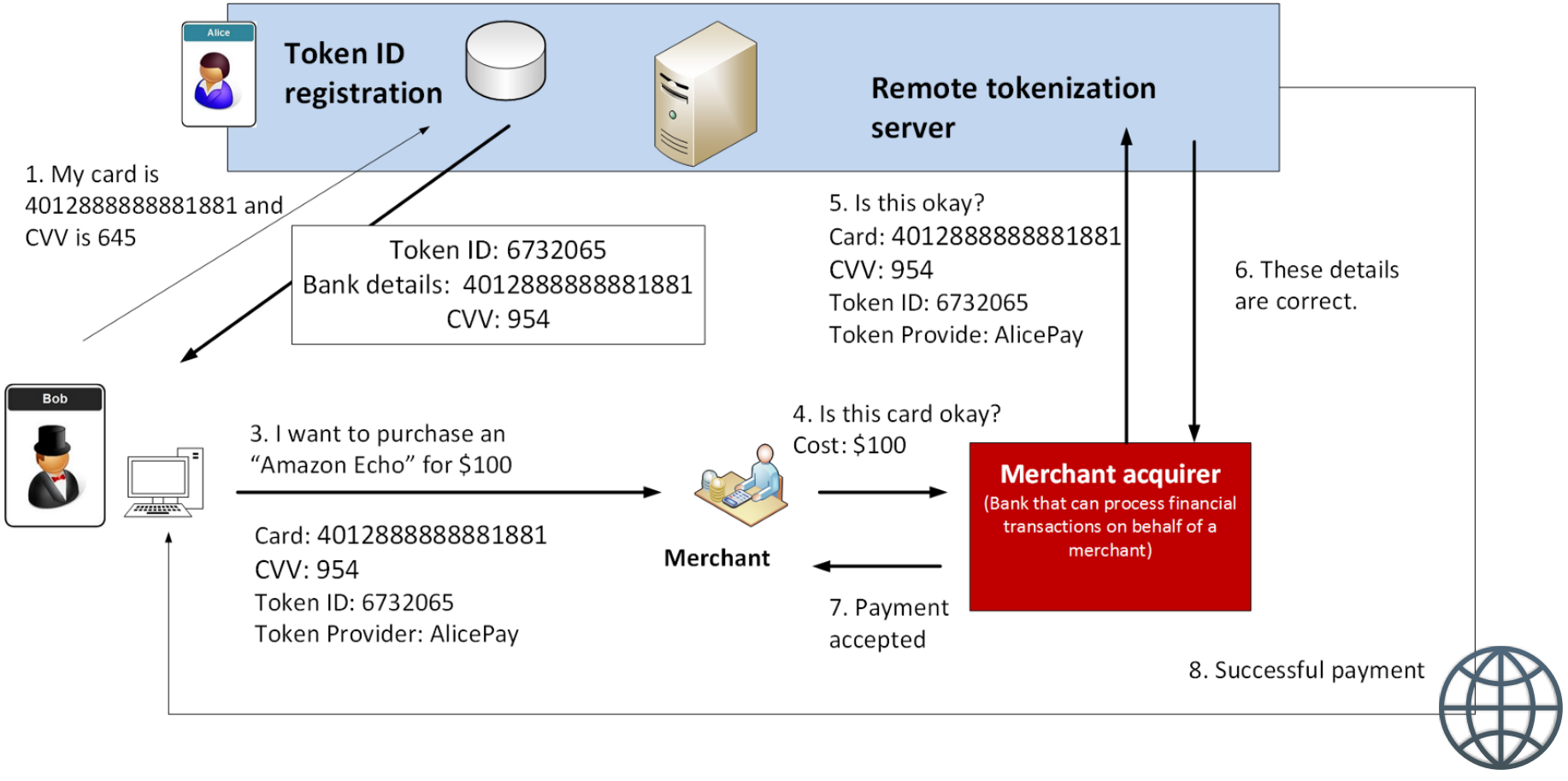
Tokenization with data



Visa Best Practice for Tokenization



Token Mapping

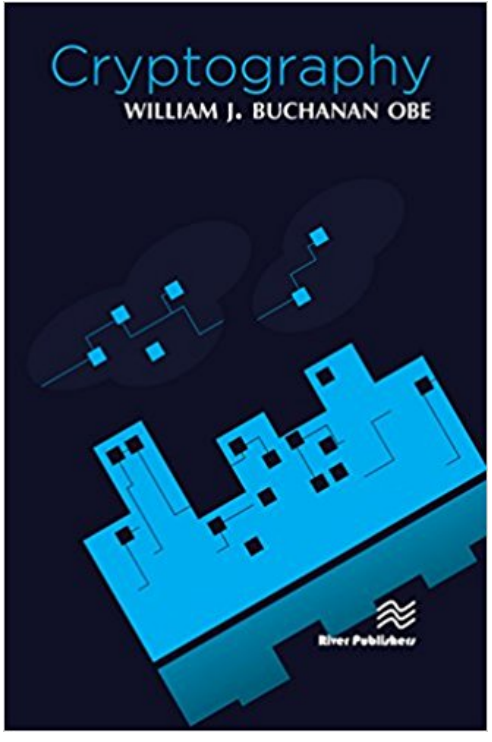


Tokens, Authorization and Docker

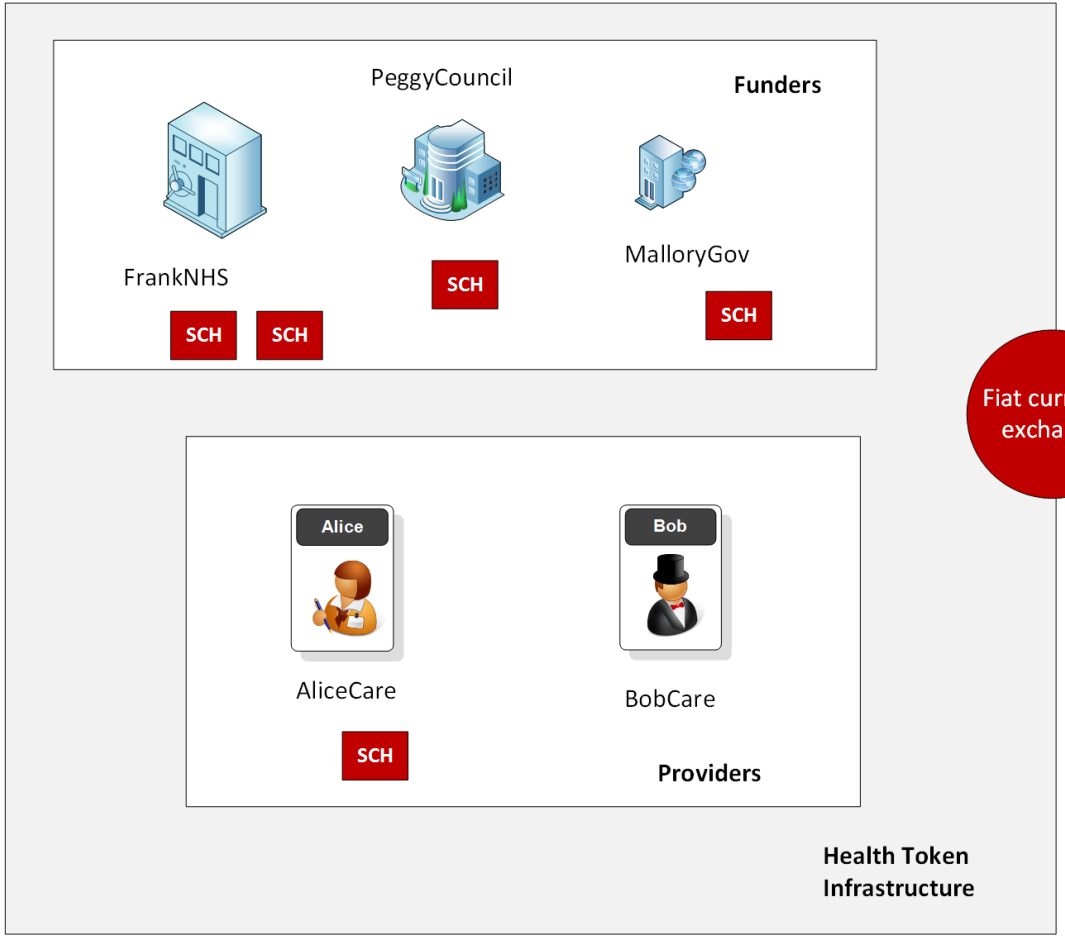
ERC-20 Tokens

Prof Bill Buchanan OBE

http://asecuritysite.com/encryption
http://asecuritysite.com/unit06



ERC-20 Tokens

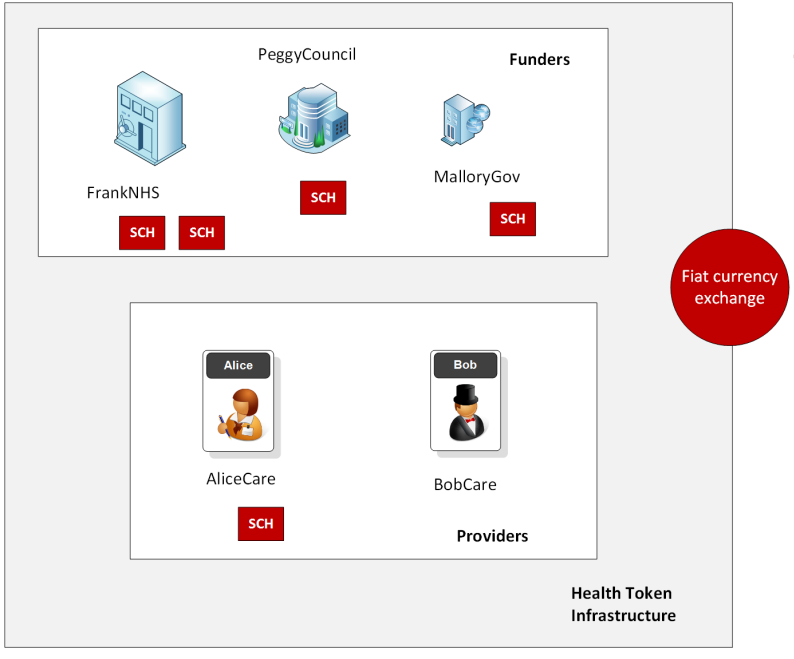


Within the Ethereum blockchain we can record transactions and run smart contracts. These things allow us to run DApps (decentralized applications) and which can support the running of the infrastructure in return for some payment (Ether).

A DApp can also **create tokens for new currencies**, shares in a company or to prove the ownership of an asset.

ERC-20 allow for the sharing, transfer and storage of tokens.

ERC-20 Tokens



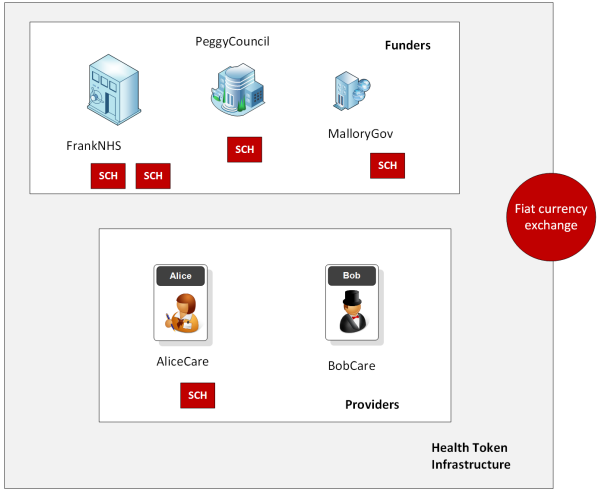
ERC-20 tokens are supported by the whole of the Ethereum infrastructure and can be easily traded. They support a number of mandatory functions:

- **totalSupply.** This function is the total number of ERC-20 tokens that have been created.
- **balanceOf. T**his function identifies the number of tokens that a given address has in its account.
- **transfer.** This function supports the transfer of tokens to a defined user address.
- **transferFrom.** This function supports a user to transfer tokens to another user.
- **approve.** This function checks that a transaction is valid, based on the supply of token.
- **allowance.** This function checks if a user has enough funds in their account for a transaction.

Optional:

- Token Name. This is the name that the token will be defined as.
- Symbol. This is the symbol that the token will use.
- Decimal. This is the number of decimal places to be used for any transactions.

ERC-20 Tokens



Details:

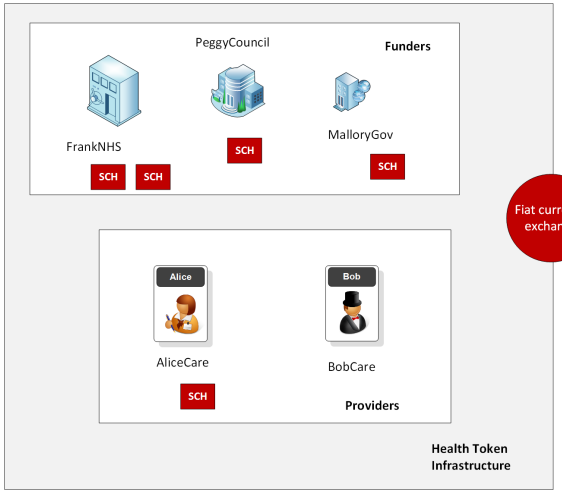
- Extends standard contract.
- 100,000,000,000 tokens.
- Name: BearToken

Ref: Build a smart contract that transfers ERC20 token from your wallet to other ERC20 compliant wallet- Part 1, Coinmonks, Medium.com

ERC-20 tokens are supported by the whole of the Ethereum infrastructure and can be easily traded. They support a number of mandatory functions:

- **totalSupply.** This function is the total number of ERC-20 tokens that have been created.
- **balanceOf. T**his function identifies the number of tokens that a given address has in its account.
- **transfer.** This function supports the transfer of tokens to a defined user address.
- **transferFrom.** This function supports a user to transfer tokens to another user.
- **approve.** This function checks that a transaction is valid, based on the supply of token.
- **allowance.** This function checks if a user has enough funds in their account for a transaction.

ERC-20 Tokens



Details:

- Extends standard contract.
- 100,000,000,000 tokens.
- Name: BearToken

Ref: Build a smart contract that transfers ERC20 token from your w to other ERC20 compliant wallet- Part 1, Coinmonks, Medium.com

```
pragma solidity ^0.4.19;

import "openzeppelin-solidity/contracts/token/ERC20
/StandardToken.sol";
import "openzeppelin-solidity/contracts/ownership/Ownable.sol";

/**
* @title BearToken is a basic ERC20 Token
*/
contract BearToken is StandardToken, Ownable{

  uint256 public totalSupply;
  string public name;
  string public symbol;
  uint32 public decimals;

  /**
 * @dev assign totalSupply to account creating this contract */
constructor() public {
  symbol = "BEAR";
  name = "BearToken";
  decimals = 5;
  totalSupply = 100000000000;

  owner = msg.sender;
  balances[msg.sender] = totalSupply;

  emit Transfer(0x0, msg.sender, totalSupply);
 }}
```

ERC-20 Tokens


```
truffle(development)> Bear.name()
'BearToken'
truffle(development)> Bear.totalSupply()
{ [String: '100000000000'] s: 1, e: 11, c: [ 100000000000 ] }
truffle(development)> Bear.balanceOf(web3.eth.accounts[0])
{ [String: '100000000000'] s: 1, e: 11, c: [ 100000000000 ] }
truffle(development)>
```

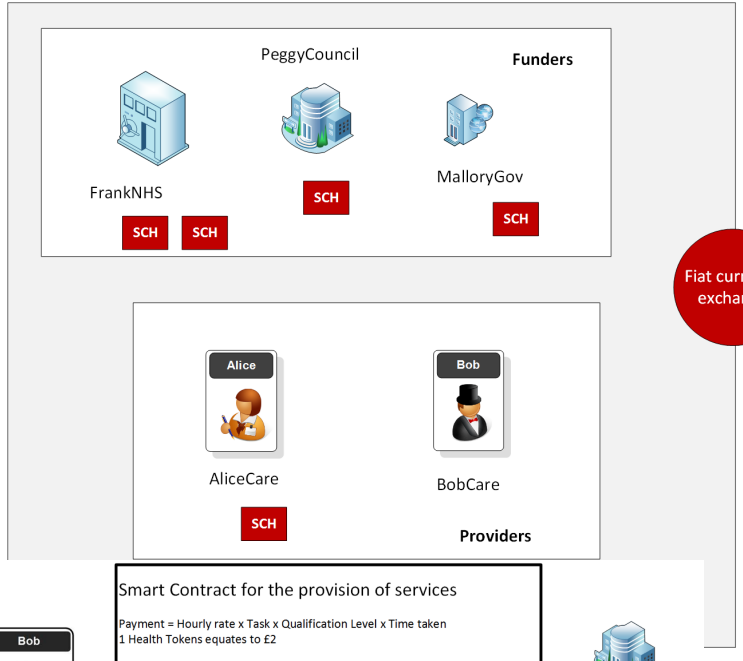
Details:

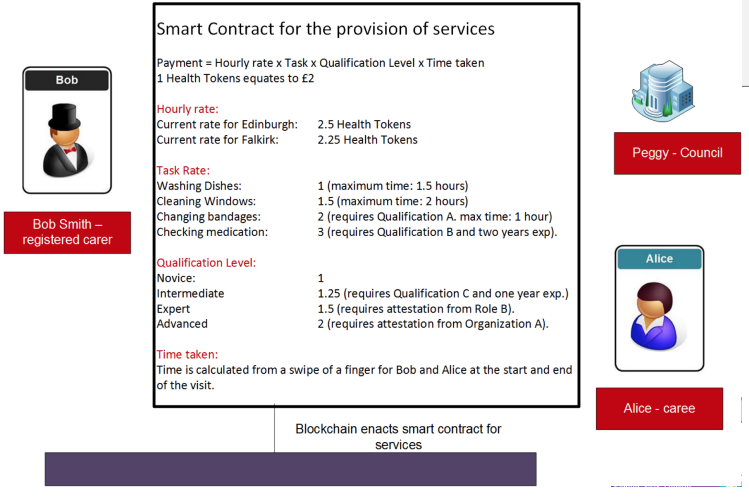
- Extends standard contract.
- 100,000,000,000 tokens.
- Name: BearToken

Ref: Build a smart contract that transfers ERC20 token from your w to other ERC20 compliant wallet- Part 1, Coinmonks, Medium.com

```
pragma solidity ^0.4.19;

                                                              ol";

  string public name;
  string public symbol;
  uint32 public decimals;

  /**
 * @dev assign totalSupply to account creating this contract */
constructor() public {
  symbol = "BEAR";
  name = "BearToken";
  decimals = 5;
  totalSupply = 100000000000;

  owner = msg.sender;
  balances[msg.sender] = totalSupply;

  emit Transfer(0x0, msg.sender, totalSupply);
 }}
```

ERC-20 Tokens





Let's say the budget for the year is to allocate 500 SCH tokens to NHS, 300 to Council and 200 to Government. We would then run the *transfer* function, and perform these allocations to the Ethereum addresses defined by them. When we conduct the *balanceOf* function we should see:

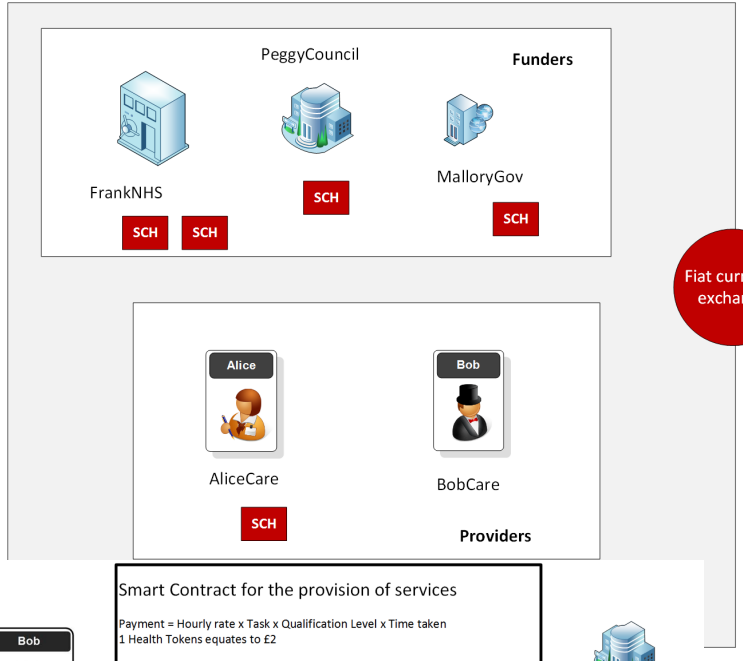
NHS: 500 SCH
Council: 300 SCH
Government: 200 SCH
BobCare: 0 SCH

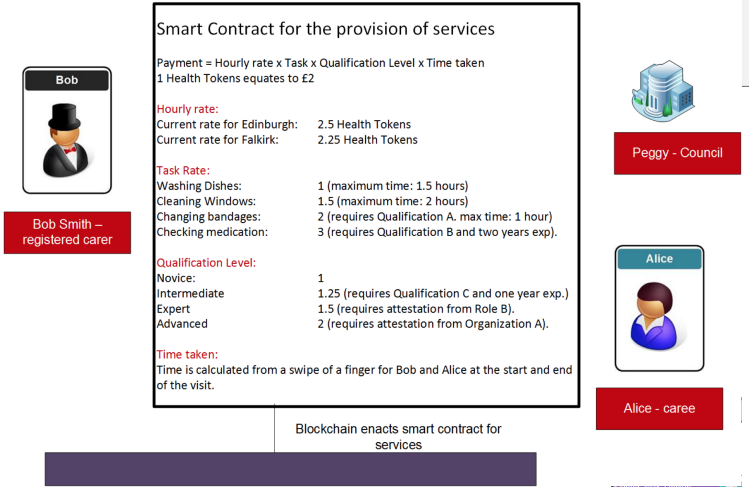
In this case BobCare now agrees a contract with PeggyCouncil to change bandages for Alice—who lives in Edinburgh—and at a rate of 2.5 SCH per hour. The staff at BobCare are at a novice level, so the payment will be 2.5 SCH for one hour's work. Now once BobCare proves that they have changed Alice's bandages, PeggCouncil then uses the *transferFrom* function to move the funds to BobCare.

When we conduct the *balanceOf* function we should see:

NHS: 500 SCH
Council: 297.5 SCH
Government: 200 SCH
BobCare: 2.5 SCH

ECR-20 Tokens





Let's say the budget for the year is to allocate 500 SCH tokens to NHS, 300 to Council and 200 to Government. We would then run the *transfer* function, and perform these allocations to the Ethereum addresses defined by them. When we conduct the *balanceOf* function we should see:

NHS: 500 SCH
Council: 300 SCH
Government: 200 SCH
BobCare: 0 SCH

In this case BobCare now agrees a contract with PeggyCouncil to change bandages for Alice—who lives in Edinburgh—and at a rate of 2.5 SCH per hour. The staff at BobCare are at a novice level, so the payment will be 2.5 SCH for one hour's work. Now once BobCare proves that they have changed Alice's bandages, PeggCouncil then uses the *transferFrom* function to move the funds to BobCare.

When we conduct the *balanceOf* function we should see:

NHS: 500 SCH
Council: 297.5 SCH
Government: 200 SCH
BobCare: 2.5 SCH

zkERC20: Confidential Token Standard #1724

Open **zac-williamson** opened this issue 15 days ago · 16 comments

zac-williamson commented 15 days ago • edited

```
eip: 1724
title: Confidential Token Standard
author: AZTEC
discussions-to: https://github.com/ethereum/EIPs/issues/1724
status: Draft
type: Standards Track
category: ERC
created: 2019-01-25
requires: 1723
```

Simple Summary

This EIP defines the standard interface and behaviours of a confidential token contract, where ownership values and the values of transfers are encrypted.

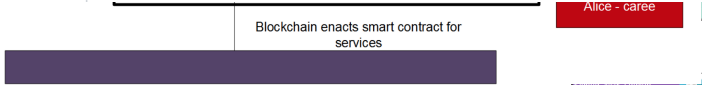
Abstract

This standard defines a way of interacting with a *confidential* token contract. Confidential tokens do not have traditional balances - value is represented by *notes*, which are composed of a public owner and an encrypted value. Value is transferred by splitting a note into multiple notes with different owners. Similarly notes can be combined into a larger note. Note splitting is analogous to the behaviour of Bitcoin UTXOs, which is a good mental model to follow.

r the year is to allocate 500 SCH tokens and 200 to Government. We would then n, and perform these allocations to the efined by them. When we conduct the should see:

ow agrees a contract with PeggyCouncil r Alice—who lives in Edinburgh—and at our. The staff at BobCare are at a novice ill be 2.5 SCH for one hour's work. Now hat they have changed Alice's bandages, the *transferFrom* function to move the

balanceOf function we should see:



Government: 200 SCH
BobCare: 2.5 SCH

zkERC20: Confidential Token Standard #1724

r the year is to allocate 500 SCH tokens
and 200 to Government. We would then

Open **zac-williamson** opened this issue 15 days ago · 16 comments

zac-williamson commented 15 days ago • edited

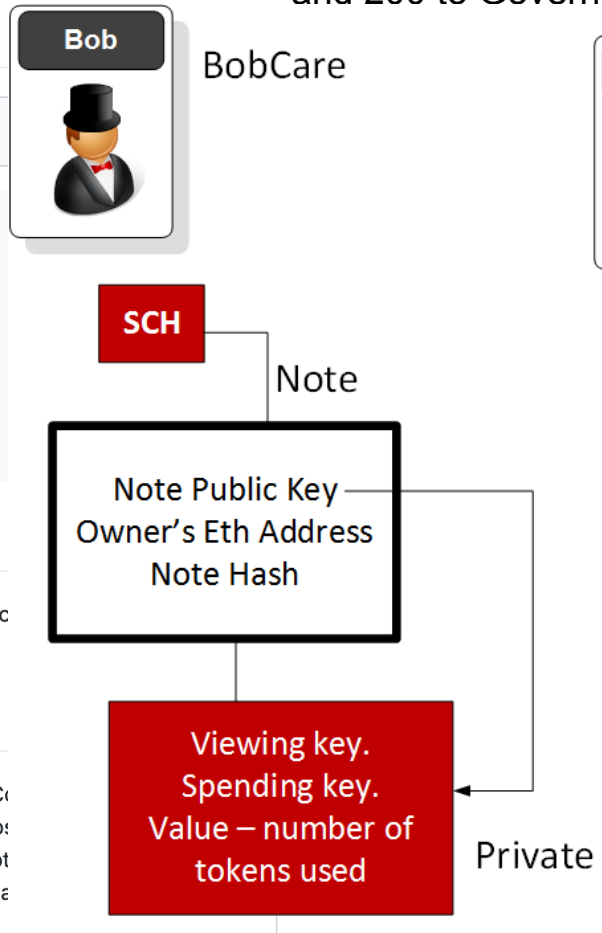
```
eip: 1724
title: Confidential Token Standard
author: AZTEC
discussions-to: https://github.com/ethereum/EIPs/issues/1724
status: Draft
type: Standards Track
category: ERC
created: 2019-01-25
requires: 1723
```

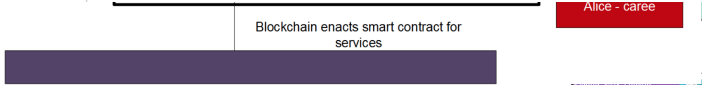
Simple Summary

This EIP defines the standard interface and behaviours of a confidential token c
ownership values and the values of transfers are encrypted.

Abstract

This standard defines a way of interacting with a *confidential* token contract. C
not have traditional balances - value is represented by *notes*, which are compos
and an encrypted value. Value is transferred by splitting a note into multiple not
owners. Similarly notes can be combined into a larger note. Note splitting is ana
of Bitcoin UTXOs, which is a good mental model to follow.





Government: 200 SCH
BobCare: 2.5 SCH

Saving ERC-20 Tokens

Ledger Nano S
(Supports 24 cryptocurrencies)

Trezor (12-digit code

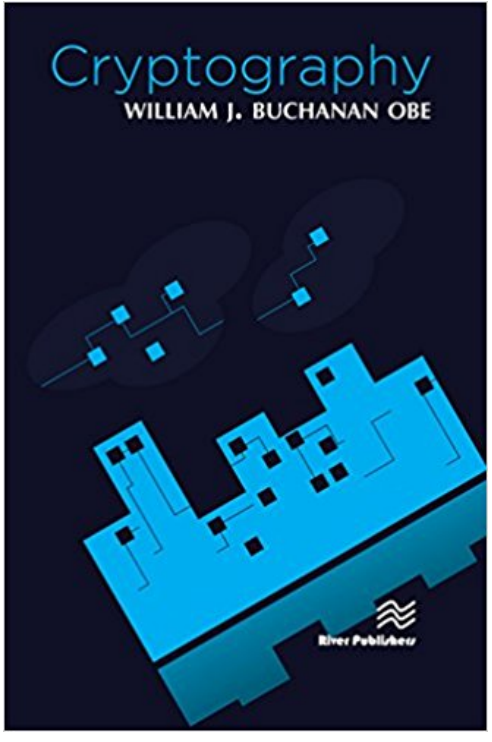
MyWallet (Never saves data on server!) … integrate with Ledger Nano

Tokens, Authorization and Docker

JSON Web Tokens

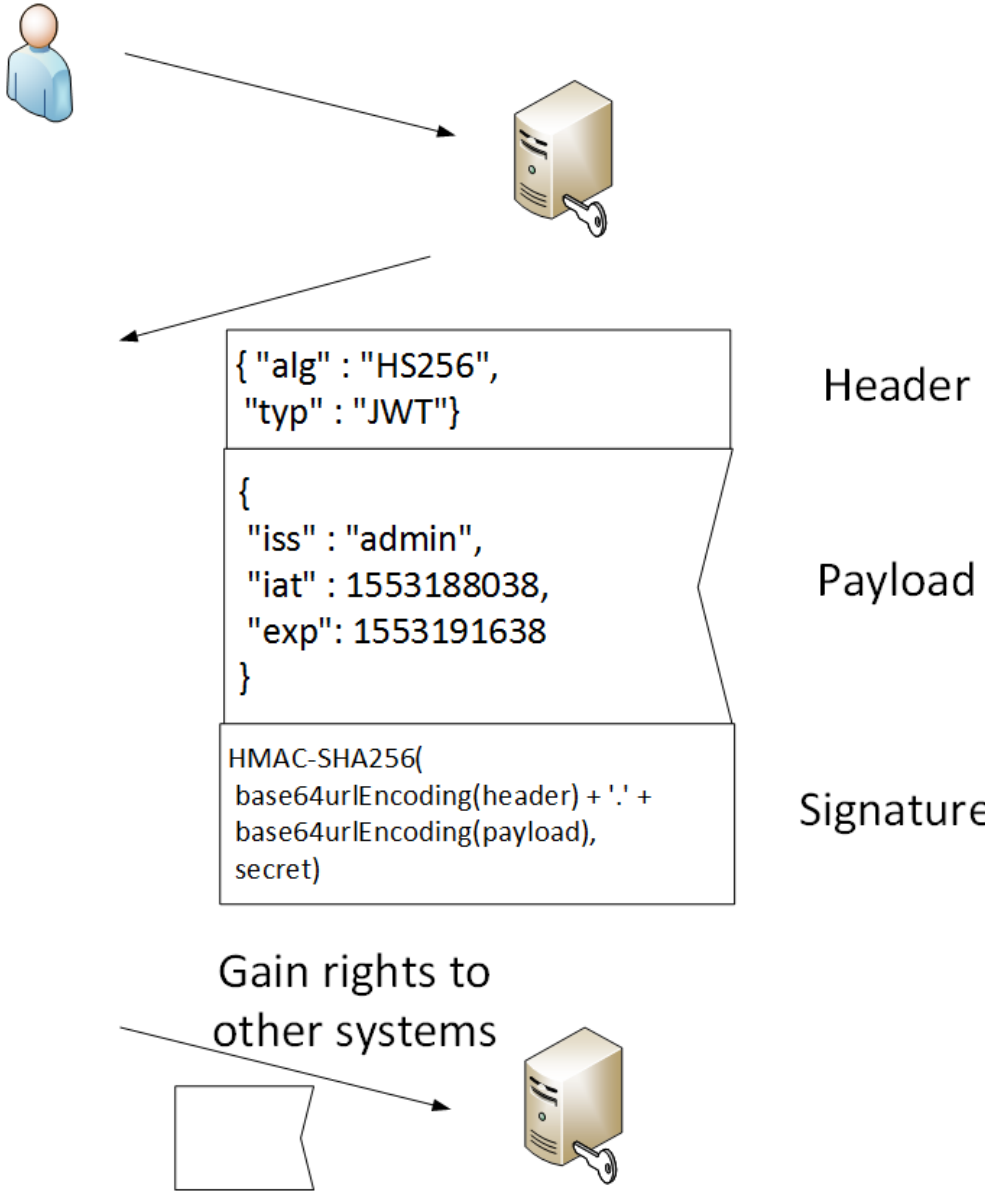
Prof Bill Buchanan OBE

http://asecuritysite.com/encryption
http://asecuritysite.com/unit06



JSON Web Token

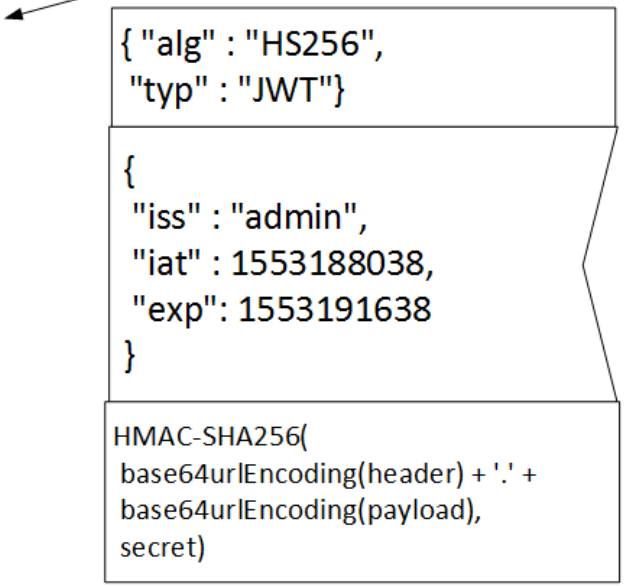
- alg: HS-256
- iss: Used to.
- iat: Issued at.
- exp: Expires at.

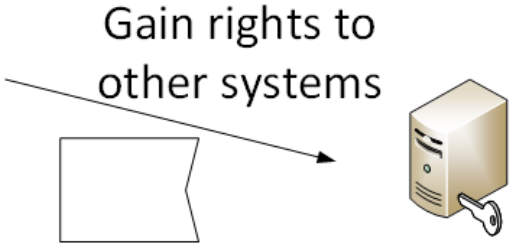


JSON Web Token

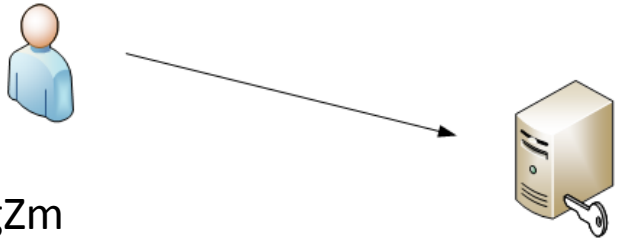
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.InsgZm9vOiAnYmFyJ30i.YysrHrgIR351UIruepwNCpwwOfH0feLlibPgASG-ljw

```
Header:
{
  "typ": "JWT",
  "alg": "HS256"
}

Payload:

"{ foo: 'bar'}"

Signature:

HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) secret base64 encoded
```





JSON Web Token



eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.InsgZm
9vOiAnYmFvJ30i.YvsrHrgIR351UIruepwNCpww

GET | https://auth.docker.io/token?service=registry.docker.io | Send

Params | Authorization | Headers (2) | Body | Pre-request Script | Tests | Cookies | Code

KEY	VALUE	DESCRIPTION	•••	Bulk Edit
Content-Type	application/x-www-form-urlencoded			
Authorization	Bearer eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCIsIng1...			

```
"{ foo: 'bar'}"
```

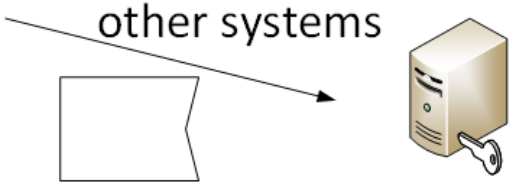
Signature:

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) secret base64 encoded
```

HMAC-SHA256(
base64urlEncoding(header) + '.' +
base64urlEncoding(payload),
secret)

Signature

Gain rights to other systems

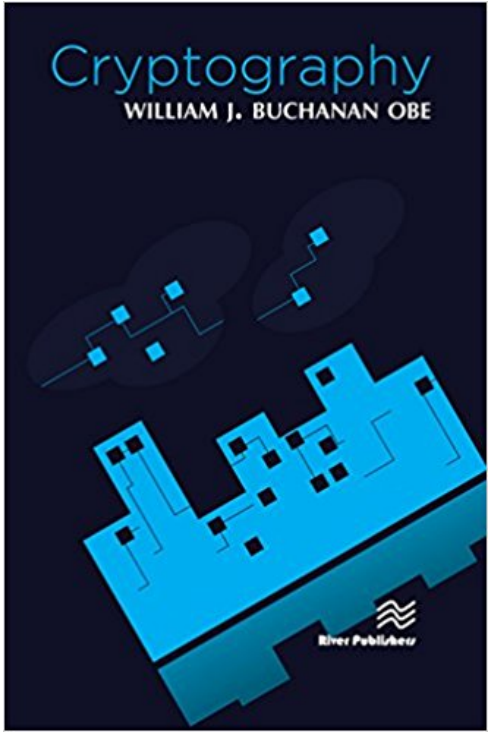


Tokens, Authorization and Docker

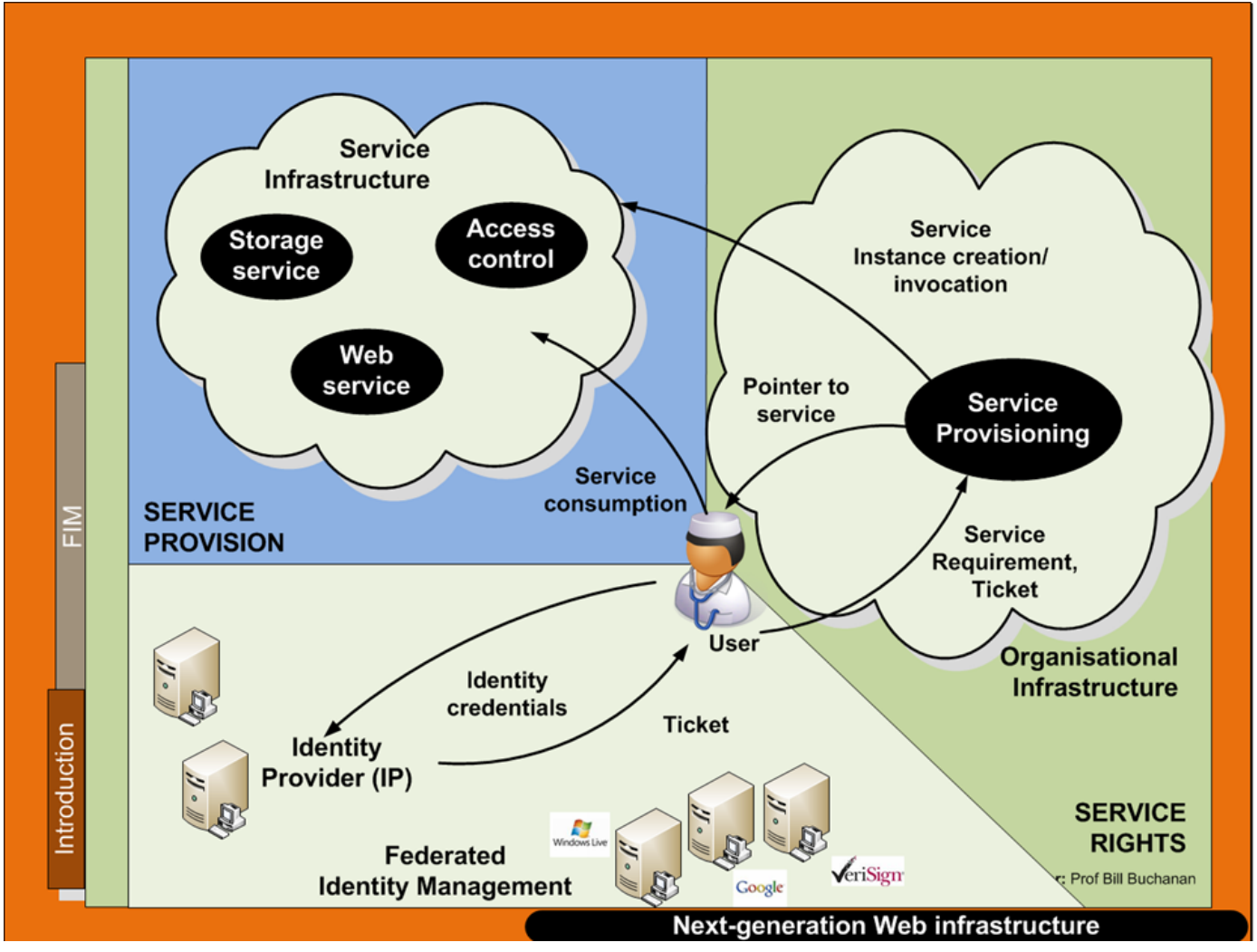
OAuth 2.0

Prof Bill Buchanan OBE

http://asecuritysite.com/encryption
http://asecuritysite.com/unit06



Federated ID

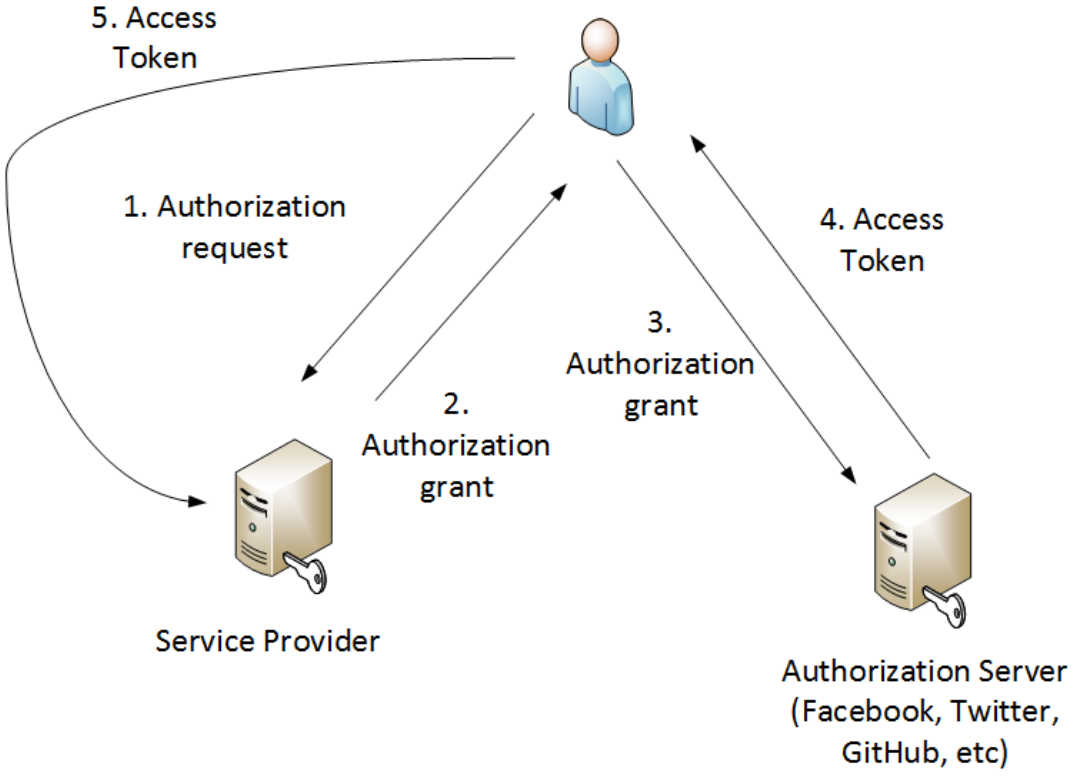


OAuth 2

- Application Name
- Application Website
- Redirect URI or Callback URL

client credentials"

client identifier
client secret.

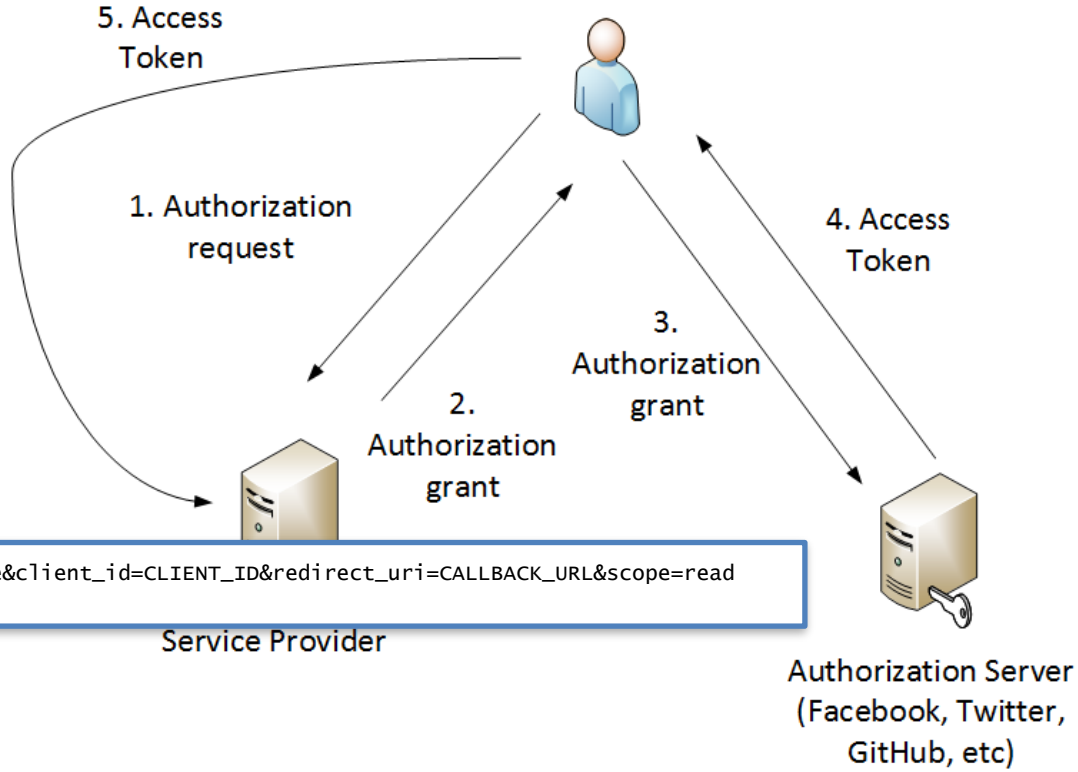


OAuth 2

- Application Name
- Application Website
- Redirect URI or Callback URL

client credentials"

client identifier
client secret.



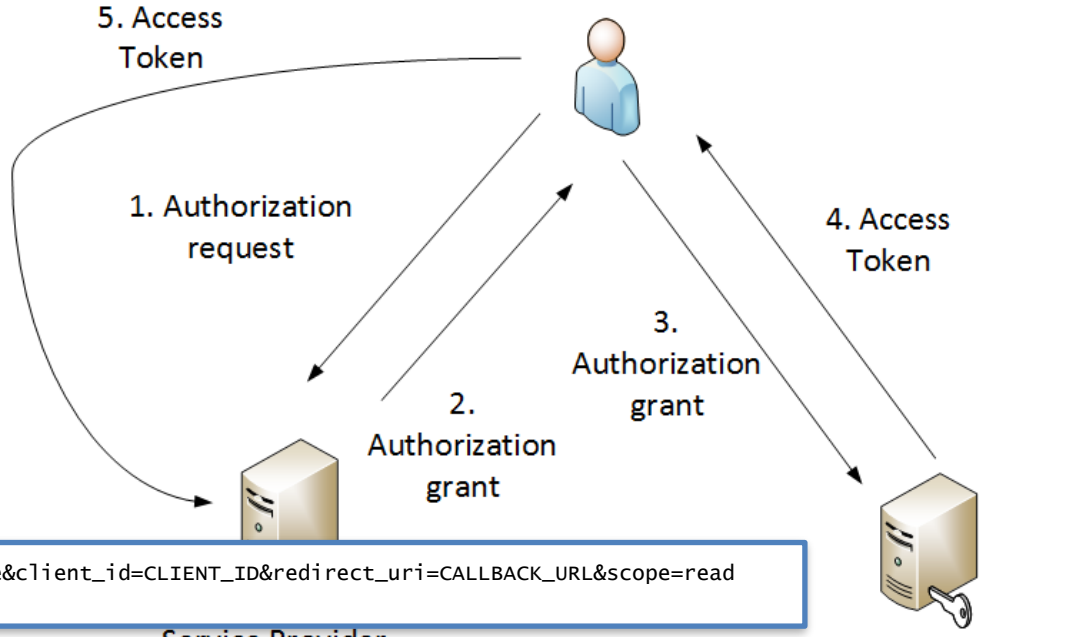

```
https://github.com/login/oauth/authorize?response_type=code&client_id=CLIENT_ID&redirect_uri=CALLBACK_URL&scope=read
```

OAuth 2

- Application Name
- Application Website
- Redirect URI or Callback URL

client credentials"

client identifier
client secret.




```
https://github.com/login/oauth/authorize?response_type=code&client_id=CLIENT_ID&redirect_uri=CALLBACK_URL&scope=read
```

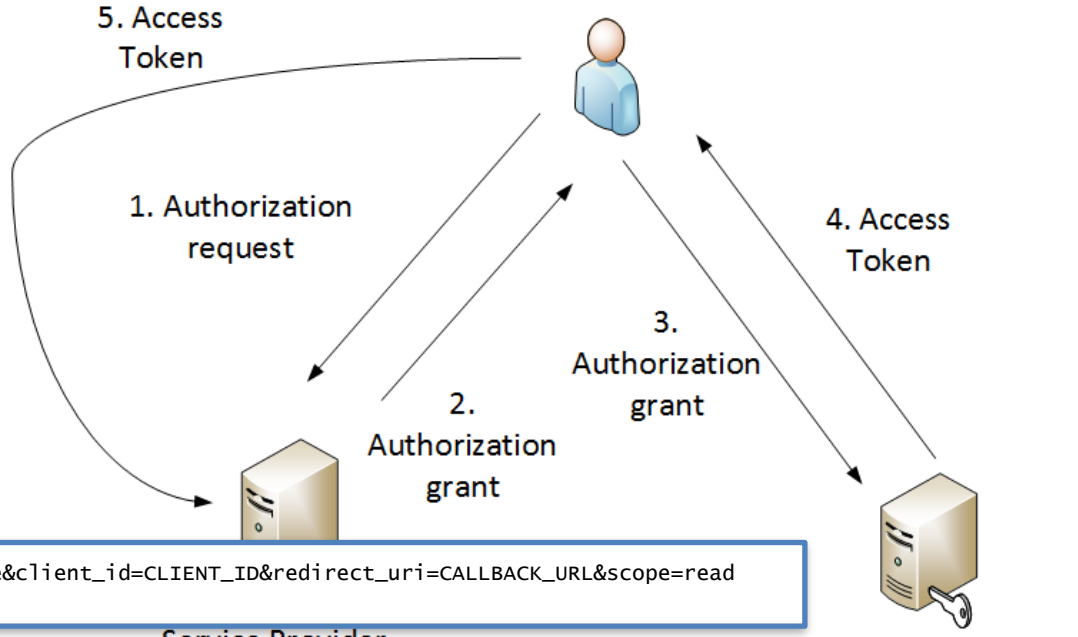
```
https://dropletbook.com/callback?code=AUTHORIZATION_CODE
```

OAuth 2

- Application Name
- Application Website
- Redirect URI or Callback URL

client credentials"

client identifier
client secret.




```
https://github.com/login/oauth/authorize?response_type=code&client_id=CLIENT_ID&redirect_uri=CALLBACK_URL&scope=read
```

```
https://dropletbook.com/callback?code=AUTHORIZATION_CODE
```

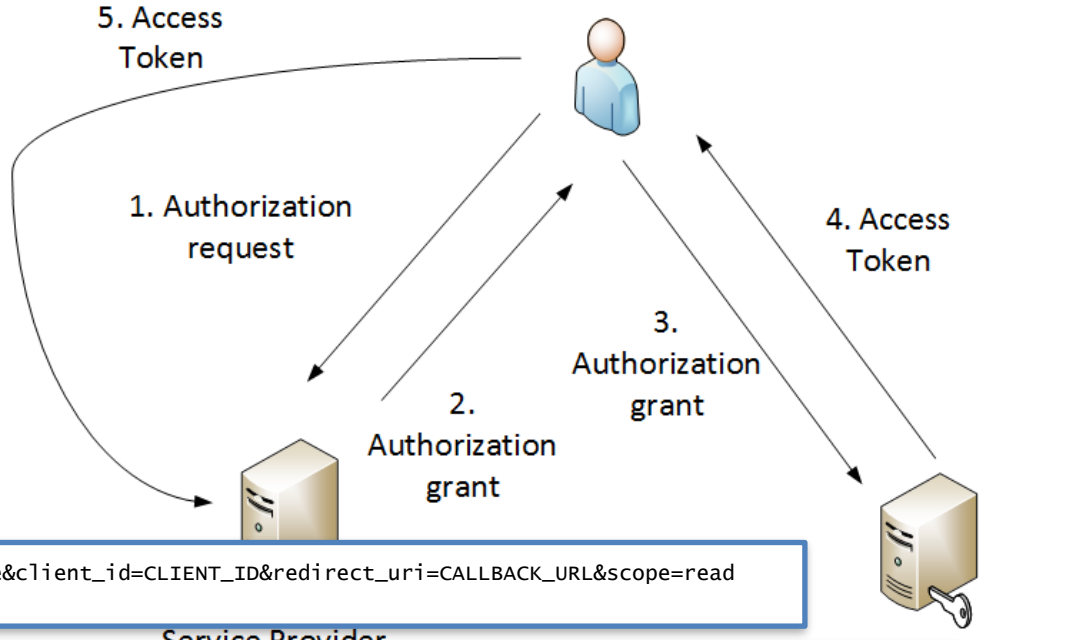
```
https://cloud.digitalocean.com/v1/oauth/token?
client_id=CLIENT_ID&client_secret=CLIENT_SECRET&grant_type=authorization_code&code=AUTHO
RIZATION_CODE&redirect_uri=CALLBACK_URL
```

OAuth 2

- Application Name
- Application Website
- Redirect URI or Callback URL

client credentials"

client identifier
client secret.




```
https://github.com/login/oauth/authorize?response_type=code&client_id=CLIENT_ID&redirect_uri=CALLBACK_URL&scope=read
```

```
https://dropletbook.com/callback?code=AUTHORIZATION_CODE
```

```
https://cloud.digitalocean.com/v1/oauth/token?
client_id=CLIENT_ID&client_secret=CLIENT_SECRET&grant_type=authorization_code&code=AUTHO
RIZATION_CODE&redirect_uri=CALLBACK_URL
```

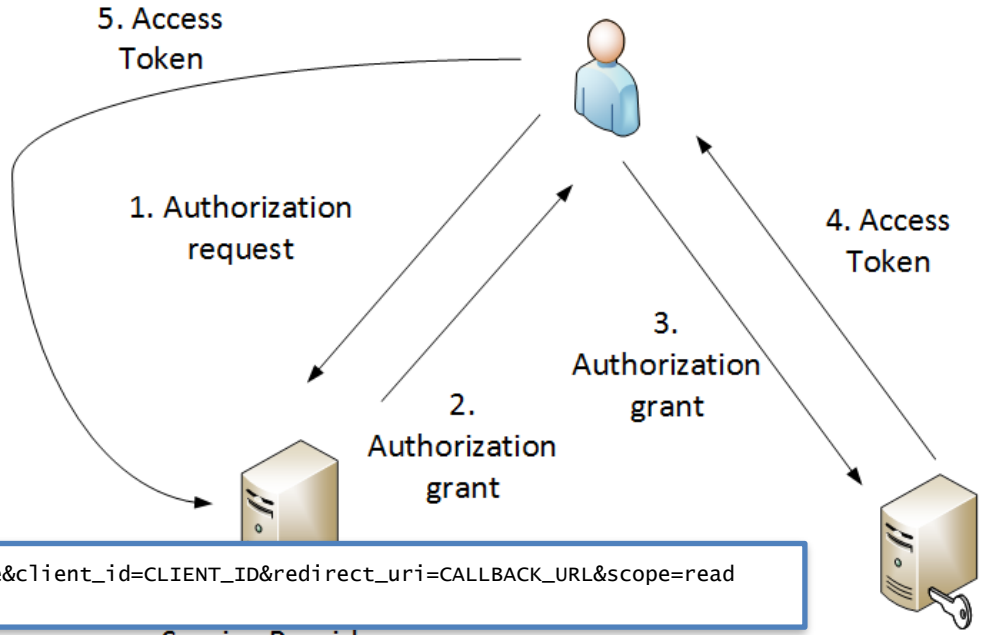
```
{"access_token":"ACCESS_TOKEN","token_type":"bearer","expires_in":
2592000,"refresh_token":"REFRESH_TOKEN","scope":"read","uid":100101,"info":{"name":"Mark E.
Mark","email":"mark@thefunkybunch.com"}}
```

OAuth 2

- Application Name
- Application Website
- Redirect URI or Callback URL

client credentials"

client identifier
client secret.




```
https://github.com/login/oauth/authorize?response_type=code&client_id=CLIENT_ID&redirect_uri=CALLBACK_URL&scope=read
```

```
https://dropletbook.com/callback?code=AUTHORIZATION_CODE
```

```
https://cloud.digitalocean.com/v1/oauth/token?
client_id=CLIENT_ID&client_secret=CLIENT_SECRET&grant_type=authorization_code&code=AUTHO
RIZATION_CODE&redirect_uri=CALLBACK_URL
```

```
{"access_token":"ACCESS_TOKEN","token_type":"bearer","expires_in":
2592000,"refresh_token":"REFRESH_TOKEN","scope":"read","uid":100101,"info":{"name":"Mark E.
Mark","
```

```
curl -X POST -H "Authorization: Bearer ACCESS_TOKEN""https://api.digitalocean.com/
v2/$OBJECT"
```

OAuth 2

- Application Name
- Application Website
- Redirect URI or Callback URL

client credentials"

client identifier
client secret.

```
https://github.com/login/oauth/authorize?
```

```
https://dropletbook.com/cal
```

```
https://cloud.digitalocean.
client_id=CLIENT_ID&client_
RIZATION_CODE&redirect_uri=
```

```
{"access_token":"ACCESS_TOKEN","t
2592000,"refresh_token":"REFRESH_
Mark","
```

```
curl -X POST -H "Auth
v2/$OBJECT"
```

Signing into One Billion Mobile App Accounts Effortlessly with OAuth2.0

Ronghai Yang Wing Cheong Lau Tianyu Liu
The Chinese University of Hong Kong

Abstract

OAuth2.0 protocol has been widely adopted by mainstream Identity Providers (IdPs) to support Single-Sign-On service. Since this protocol was originally designed to serve the authorization need for 3rd party websites, different pitfalls have been uncovered when adapting OAuth to support mobile app authentication. To the best of our knowledge, all the attacks discovered so far, including BlackHat USA'16 [3], CCS'14 [2] and ACSAC'15 [5], require to interact with the victim, for example via malicious apps or network eavesdropping, *etc*. On the contrary, we have discovered a new type of widespread but incorrect usages of OAuth by 3rd party mobile app developers, which can be exploited remotely and solely by the attacker to sign into a victim's mobile app account without any involvement/ awareness of the victim. To demonstrate the prevalence and severe impact of this vulnerability, we have developed an exploit to examine the implementations of 600 top-ranked US and Chinese Android Apps which use the OAuth2.0-based authentication service provided by three top-tier IdPs, namely Facebook, Google or Sina. Our empirical results are alarming: on average, 41.21% of these apps are vulnerable to this new attack. We have reported our findings to the affected IdPs, and received their acknowledgements/ rewards in various ways.

OAuth 2

- Application Name
- Application Website
- Redirect URI or Callback URL

client credentials"

client identifier
client secret.

```
https://github.com/login/oauth/authorize?
```

```
https://dropletbook.com/cal
```

```
https://cloud.digitalocean.
client_id=CLIENT_ID&client_
RIZATION_CODE&redirect_uri=
```

```
{"access_token":"ACCESS_TOKEN","t
2592000,"refresh_token":"REFRESH_
Mark","
```

```
curl -X POST -H "Auth
v2/$OBJECT"
```

Signing into One Billion Mobile App Accounts

```
Internet Engineering Task Force (IETF)
Request for Comments: 8471
Category: Standards Track
ISSN: 2070-1721

                                                          Kings

                    The Token Binding Protocol Version 1.0

Abstract

   This document specifies version 1.0 of the Token Bin
   The Token Binding protocol allows client/server appl
   create long-lived, uniquely identifiable TLS binding
   multiple TLS sessions and connections.  Applications
   to cryptographically bind security tokens to the TLS
   preventing token export and replay attacks.  To prot
   Token Binding identifiers are only conveyed over TLS
   by the user at any time.
```

OAuth 2

- Application Name
- Application Website
- Redirect URI or Callback URL

client credentials"

client identifier
client secret.

```
https://github.com/login/oauth/authorize?
```

```
https://dropletbook.com/cal
```

```
https://cloud.digitalocean.
client_id=CLIENT_ID&client_
RIZATION_CODE&redirect_uri=
```

```
{"access_token":"ACCESS_TOKEN","t
2592000,"refresh_token":"REFRESH_
Mark","
```

```
curl -X POST -H "Auth
v2/$OBJECT"
```

Signin

Interr
Reques
Categc
ISSN:

Abstra

Thi
The
cre
mul
to
pre
Tok
by

Popov, et al. Standards Track

RFC 8471 The Token Binding Protocol Version 1.0

7. Security Considerations

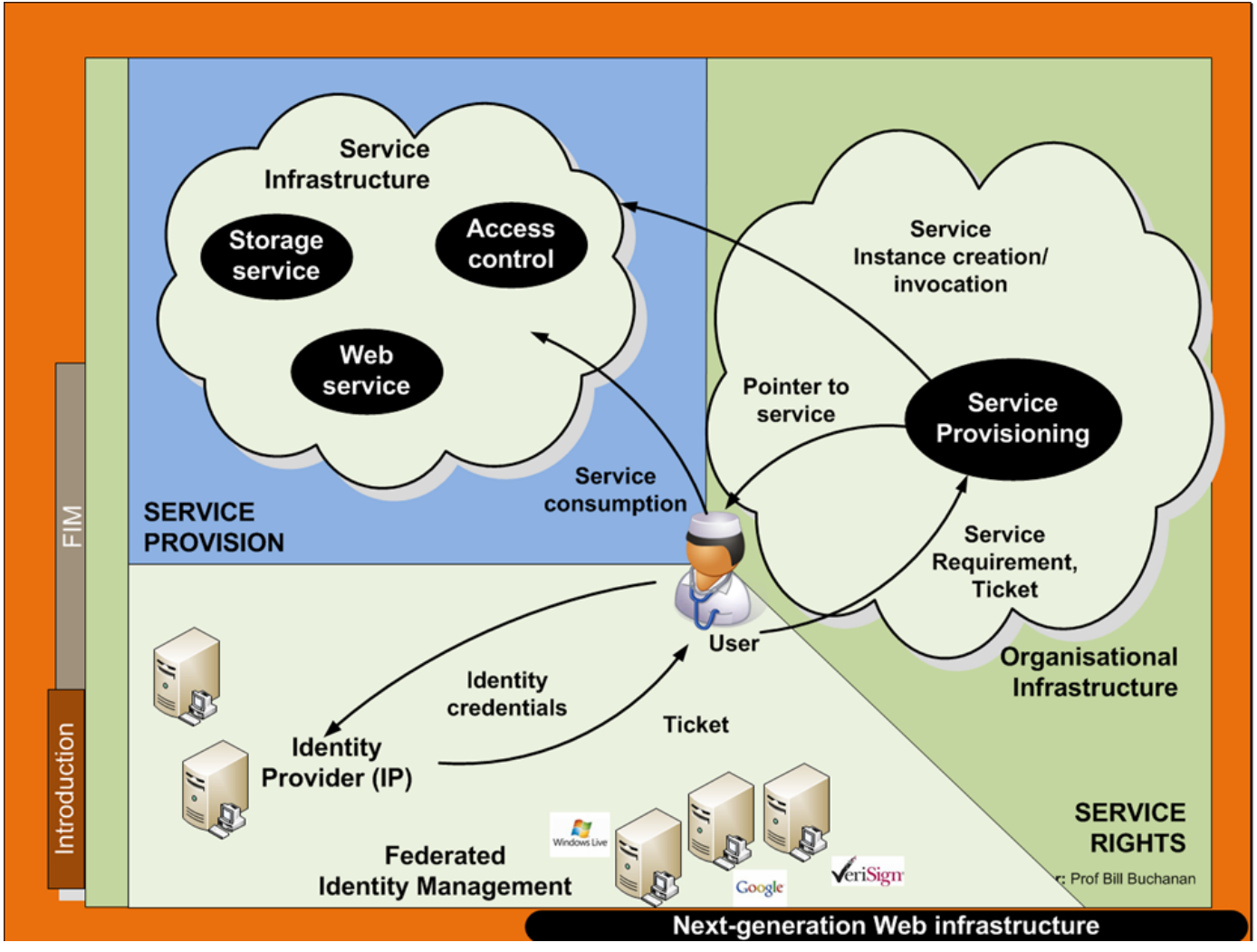
7.1. Security Token Replay

The goal of the Token Binding protocol is to prevent atta
exporting and replaying security tokens and from thereby
impersonating legitimate users and gaining access to prot
resources. Bound tokens can be replayed by malware prese
User Agents; this may be undetectable to a server. Howev
to export bound tokens to other machines and successfully
them, attackers also need to export corresponding Token B
private keys. Token Binding private keys are therefore h
assets and SHOULD be strongly protected, ideally by gener
in a hardware security module that prevents key export.

The manner in which a token is bound to the TLS layer is
the application and is beyond the scope of this document.
the resulting bound token needs to be integrity-protected
attacker cannot remove the binding or substitute a Token
of their choice without detection.

The Token Binding protocol does not prevent cooperating c
sharing a bound token. A client could intentionally expo
token with the corresponding Token Binding private key or
signatures using this key on behalf of another client.

Federated ID



Register a new OAuth application

Application name *

MyGit

Something users will recognize and trust.

Homepage URL *

https://asecuritysite.com

The full URL to your application homepage.

Application description

Application description is optional

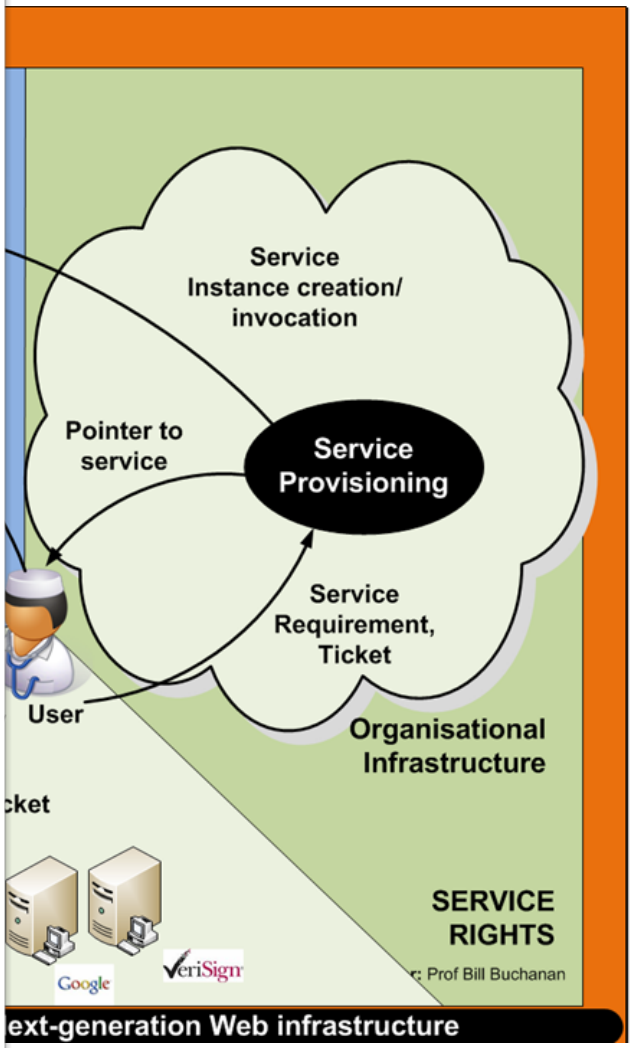
This is displayed to all users of your application.

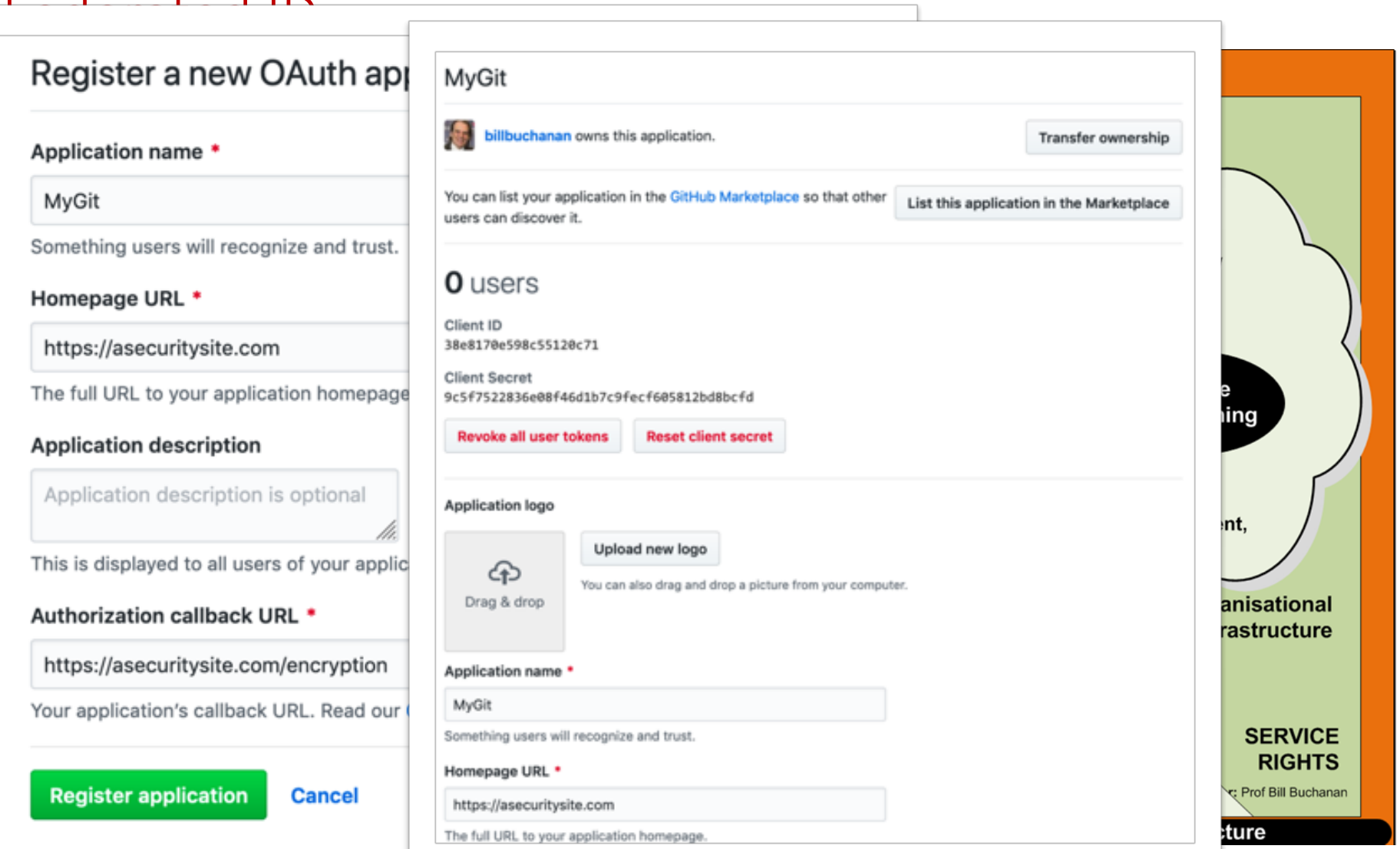
Authorization callback URL *

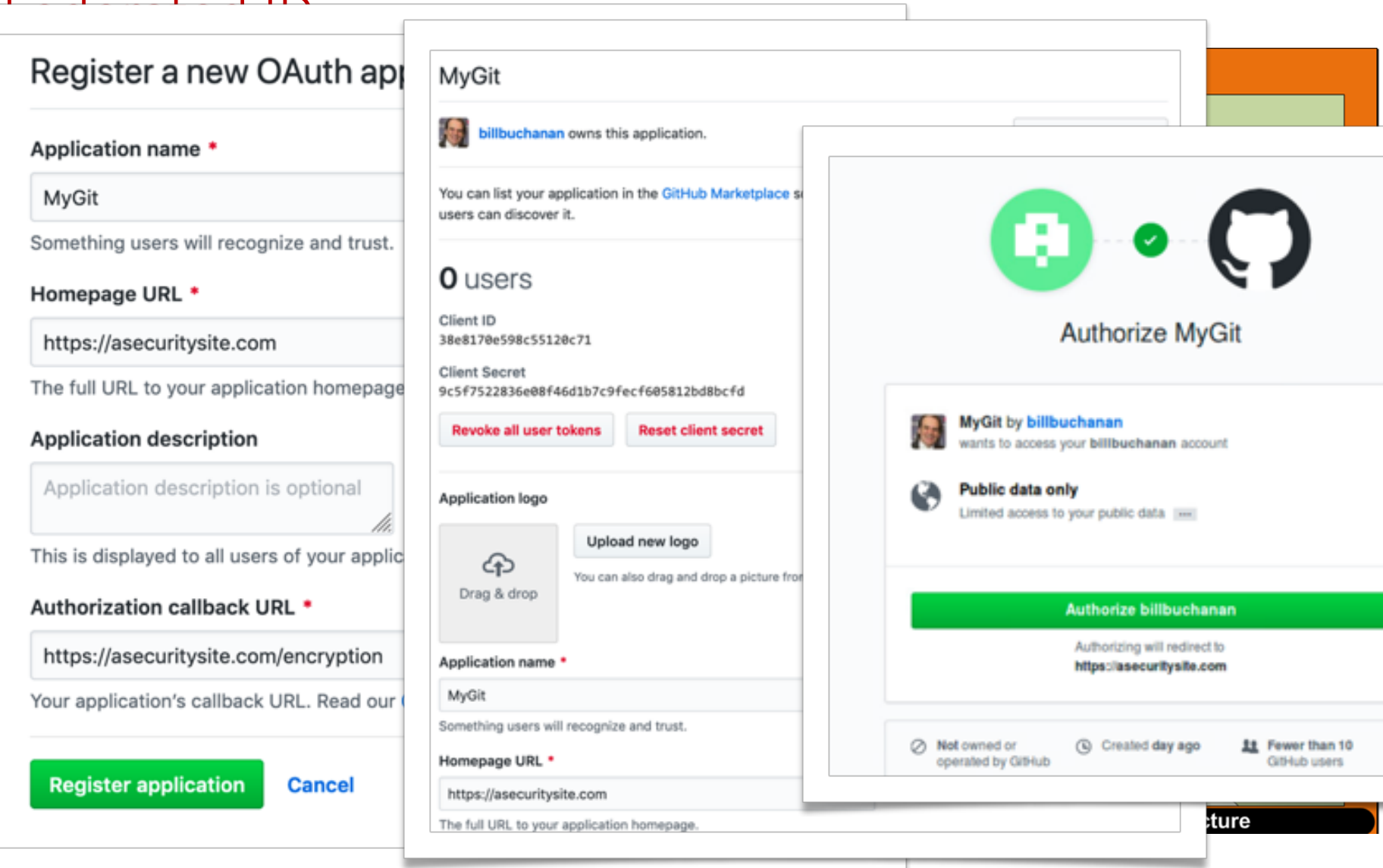
https://asecuritysite.com/encryption

Your application's callback URL. Read our OAuth documentation for more information.

Register application Cancel




Register a new OAuth app
Application name *
MyGit
Something users will recognize and trust.
Homepage URL *
https://asecuritysite.com
The full URL to your application homepage
Application description
Application description is optional
This is displayed to all users of your applic
Authorization callback URL *
https://asecuritysite.com/encryption
Your application's callback URL. Read our
Register application
Cancel
MyGit
billbuchanan owns this application.
Transfer ownership
You can list your application in the GitHub Marketplace so that other users can discover it.
List this application in the Marketplace
0 users
Client ID
38e8170e598c55120c71
Client Secret
9c5f7522836e08f46d1b7c9fecf605812bd8bcfd
Revoke all user tokens
Reset client secret
Application logo
Drag & drop
Upload new logo
You can also drag and drop a picture from your computer.
Application name *
MyGit
Something users will recognize and trust.
Homepage URL *
https://asecuritysite.com
The full URL to your application homepage.
SERVICE
RIGHTS
Prof Bill Buchanan


Register a new OAuth app
Application name *
MyGit
Something users will recognize and trust.
Homepage URL *
https://asecuritysite.com
The full URL to your application homepage
Application description
Application description is optional
This is displayed to all users of your applic
Authorization callback URL *
https://asecuritysite.com/encryption
Your application's callback URL. Read our
Register application
Cancel
MyGit
billbuchanan owns this application.
You can list your application in the GitHub Marketplace
users can discover it.
0 users
Client ID
38e8170e598c55120c71
Client Secret
9c5f7522836e08f46d1b7c9fecf605812bd8bcfd
Revoke all user tokens
Reset client secret
Application logo
Drag & drop
Upload new logo
You can also drag and drop a picture fro
Application name *
MyGit
Something users will recognize and trust.
Homepage URL *
https://asecuritysite.com
The full URL to your application homepage.
Authorize MyGit
MyGit by billbuchanan
wants to access your billbuchanan account
Public data only
Limited access to your public data
Authorize billbuchanan
Authorizing will redirect to
https://asecuritysite.com
Not owned or operated by GitHub
Created day ago
Fewer than 10 GitHub users

Tokens, Authorization and Docker

Fernet Tokens

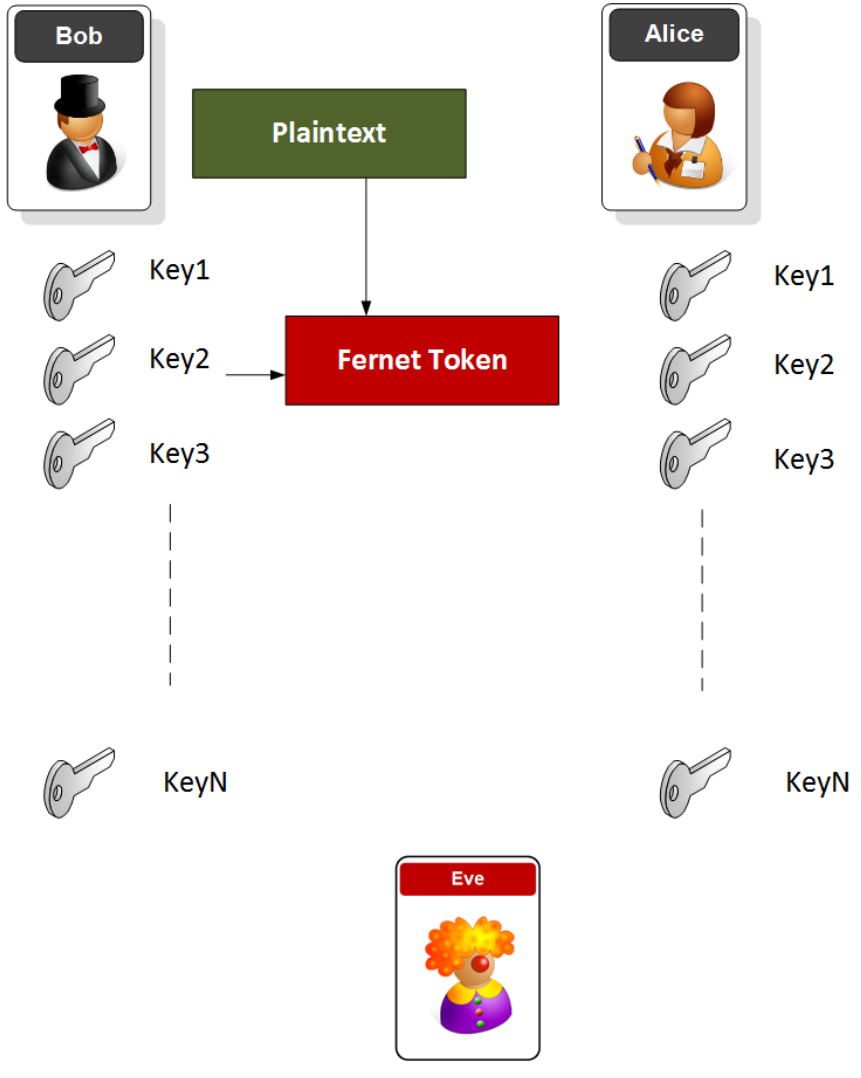
Prof Bill Buchanan OBE

http://asecuritysite.com/encryption
http://asecuritysite.com/unit06

Fernet Token

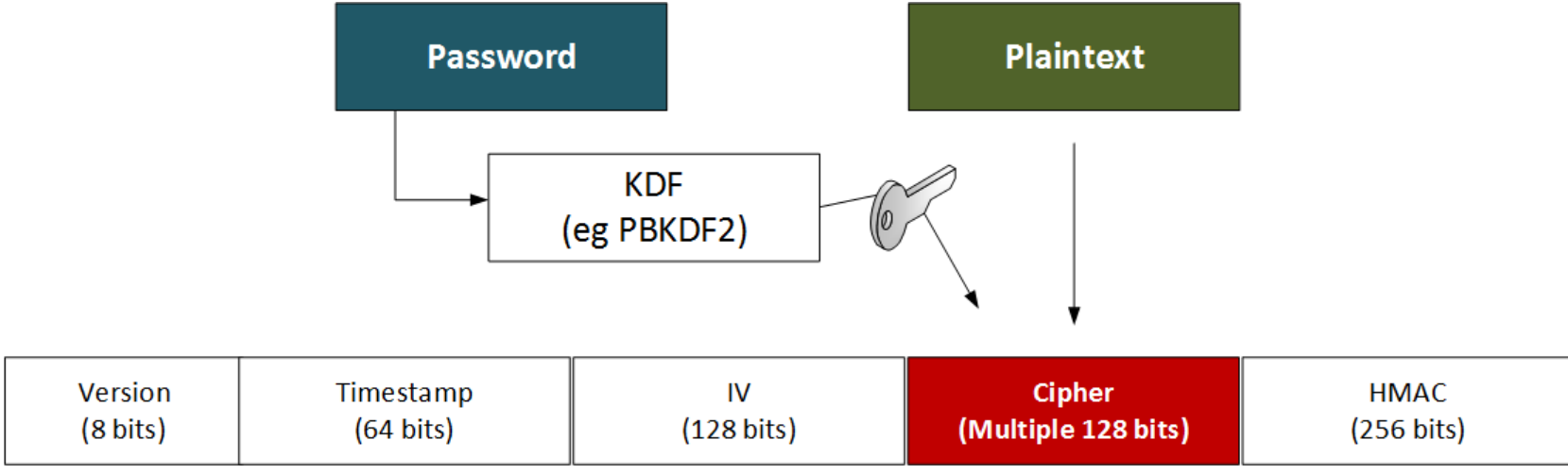
The token has a version number, a time stamp, the IV, the cipher text and an HMAC signature:

- Version: 8 bits
- Timestamp: 64 bits (the number of seconds since between January 1, 1970 UTC and the time of the encryption.
- IV: 128 bits
- Ciphertext - variable length: Multiple of 128 bits
- HMAC: 256 bits



Fernet

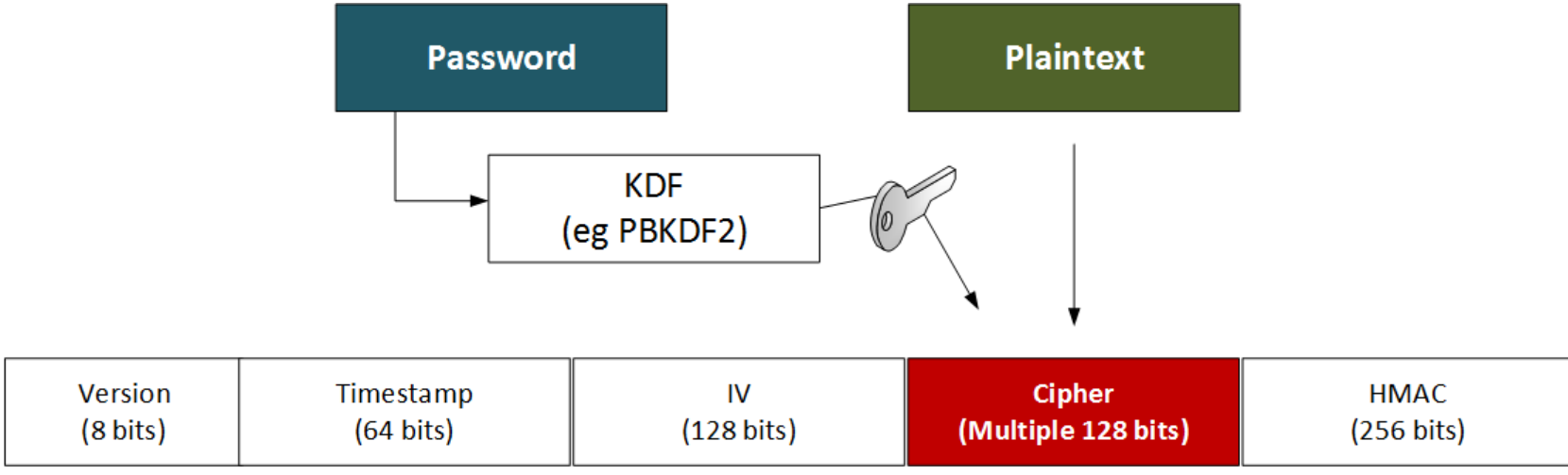
- AES 128-bit.
- 256-bit HMAC.
- AES CBC.
- PKCS7 padding [here].



Version: 8 bits
Timestamp: 64 bits
IV: 128 bits
Ciphertext - variable length: Multiple of 128 bits
HMAC: 256 bits

Fernet

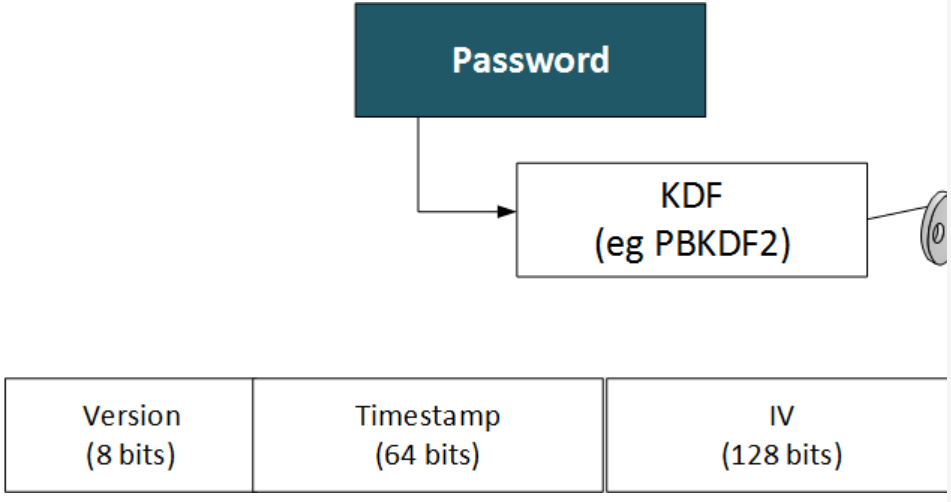
- AES 128-bit.
- 256-bit HMAC.
- AES CBC.




```
Key: 4c504a4e756c2d776f77346d3644737178626e696e687357486c776670304a656377517a59704f4c6d43513d
Cipher:
6741414141414263463846744a5a7935555f79334e50394b6d4c6f54523032614c533354774e6f6651324630732d4976456a674a67
7a765f386f70684a426f2d627a596e355a746776686b314e4e364d6a6a384b47545f4f6d6c4b384e6c793575513d3d

Version:        67
Time stamp:     4141414141426346
IV:             3846744a5a7935555f79334e50394b6d
Cipher:         4c6f54523032614c533354774e6f6651324630732d4976456a674a677a765f386f70684a426f2d627a596e
HMAC:           355a746776686b314e4e364d6a6a384b47545f4f6d6c4b384e6c793575513d3d

Plain text: hello world
```

Fernet

- AES 128-bit.
- 256-bit HMAC.
- AES CBC.

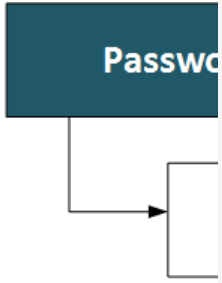



```
Key: 4c504a4e756c2d776f77346d3644737178626e696e687357486c776670304a
Cipher:
67414141414142634638467 44a5a7935555f79334e50394b6d4c6f54523032614c5
7a765f386f70684a426f2d627a596e355a746776686b314e4e364d6a6a384b47545

Version:        67
Time stamp:     4141414141426346
IV:             3846744a5a7935555f79334e50394b6d
Cipher:         4c6f54523032614c533354774e6f6651324630732d4976456a6
HMAC:           355a746776686b314e4e364d6a6a384b47545f4f6d6c4b384e6

Plain text: hello world
```

```
from cryptography.fernet import Fernet
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.backends import default_backend

import sys
import binascii
import base64

password="hello"
val="hello world"


def get_key(password):
    digest = hashes.Hash(hashes.SHA256(), backend=default_backend())
    digest.update(password)
    return base64.urlsafe_b64encode(digest.finalize())

if (len(sys.argv)>1):
        val=sys.argv[1]

if (len(sys.argv)>2):
        password=str(sys.argv[2])

if (len(password)>1):
        key = get_key(password)
else:
        key = Fernet.generate_key()


print "Key: "+binascii.hexlify(bytearray(key))


cipher_suite = Fernet(key)
cipher_text = cipher_suite.encrypt(val)
cipher=binascii.hexlify(bytearray(cipher_text))
print "Cipher: "+cipher

print "\nVersion:\t"+cipher[0:2]
print "Time stamp:\t"+cipher[2:18]
print "IV:\t\t"+cipher[18:50]
print "HMAC:\t\t"+cipher[-64:]

plain_text = cipher_suite.decrypt(cipher_text)
```

Fernet

- AES 128-bit.
- 256-bit HMAC.
- AES CBC.
- PKCS7 padding [here].

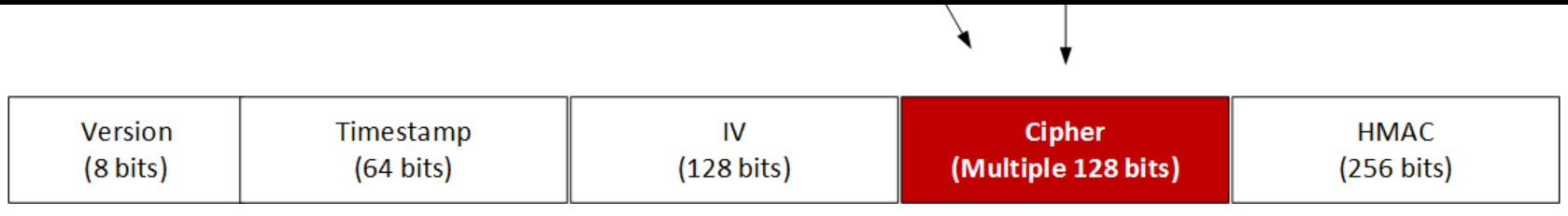


Version: 8 bits
Timestamp: 64 bits
IV: 128 bits
Ciphertext - variable length: Multiple of 128 b
HMAC: 256 bits

```
from cryptography.fernet import Fernet
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.kdf.pbkdf2 import PBKDF2HMAC

import sys
import binascii
import base64
import os

password="hello"
val="hello world"

def get_key(password):

    salt = os.urandom(16)
    kdf = PBKDF2HMAC(algorithm=hashes.SHA256(),length=32,salt=salt,iterations=100,backend=default_backend())
    key=base64.urlsafe_b64encode(kdf.derive(password))
    return (key,salt)

if (len(sys.argv)>1):
        val=sys.argv[1]

if (len(sys.argv)>2):
        password=str(sys.argv[2])

(key,salt) = get_key(password)


print "Password:\t"+password
print "Key: "+binascii.hexlify(bytearray(key))
print "Salt:\t"+binascii.hexlify(salt)

cipher_suite = Fernet(key)
cipher_text = cipher_suite.encrypt(val)
print "\nCipher: "+binascii.hexlify(bytearray(cipher_text))
print "\nVersion:\t"+cipher[0:2]
print "Time stamp:\t"+cipher[2:18]
print "IV:\t\t"+cipher[18:50]
print "HMAC:\t\t"+cipher[-64:]

plain_text = cipher_suite.decrypt(cipher_text)
```

URL Safe

```
Key:  LuIuBBZrVgnQI3DWbN4QlaHXLehOTfGkzxr_pd3SygY=
Token:  gAAAAABcF8VblJLZgwq3qmQfxUsqMyVAMQcsIatXLjmcFLOwWDrbSWjL45yUy62xGEpVuXlK2TBzSbs_xUv6KCrgumZtGHHLMQ==
Current time:   Mon Dec 17 15:48:43 2018

Token Details
==============
Decoded data:
80000000005c17c55b9492d9830ab7aa641fc54b2a33254031072c21ab572e399c14b3b0583adb4968cbe39c94cbadb1184a55b9794ad9307349bb3fc54bfa282ae0ba666d1871cb31
======Analysis====
Version:        80
Date created:   000000005c17c55b
IV:             9492d9830ab7aa641fc54b2a33254031
Cipher:         072c21ab572e399c14b3b0583adb4968
HMAC:           cbe39c94cbadb1184a55b9794ad9307349bb3fc54bfa282ae0ba666d1871cb31
======Converted====
Time stamp:     1545061723
Date created:   Mon Dec 17 15:48:43 2018
IV:             9492d9830ab7aa641fc54b2a33254031
Decoded:        password
```



URL Safe

Token:

```
gAAAAABWC9P7-9RsxTz_dwxh9-
O2VUB7Ih8UCQL1_Zk4suxnkCvb26Ie4i8HSUJ4caHZuiNtjLl3qfmCv_fS3_VpjL7HxCz7_Q==
```

Key:

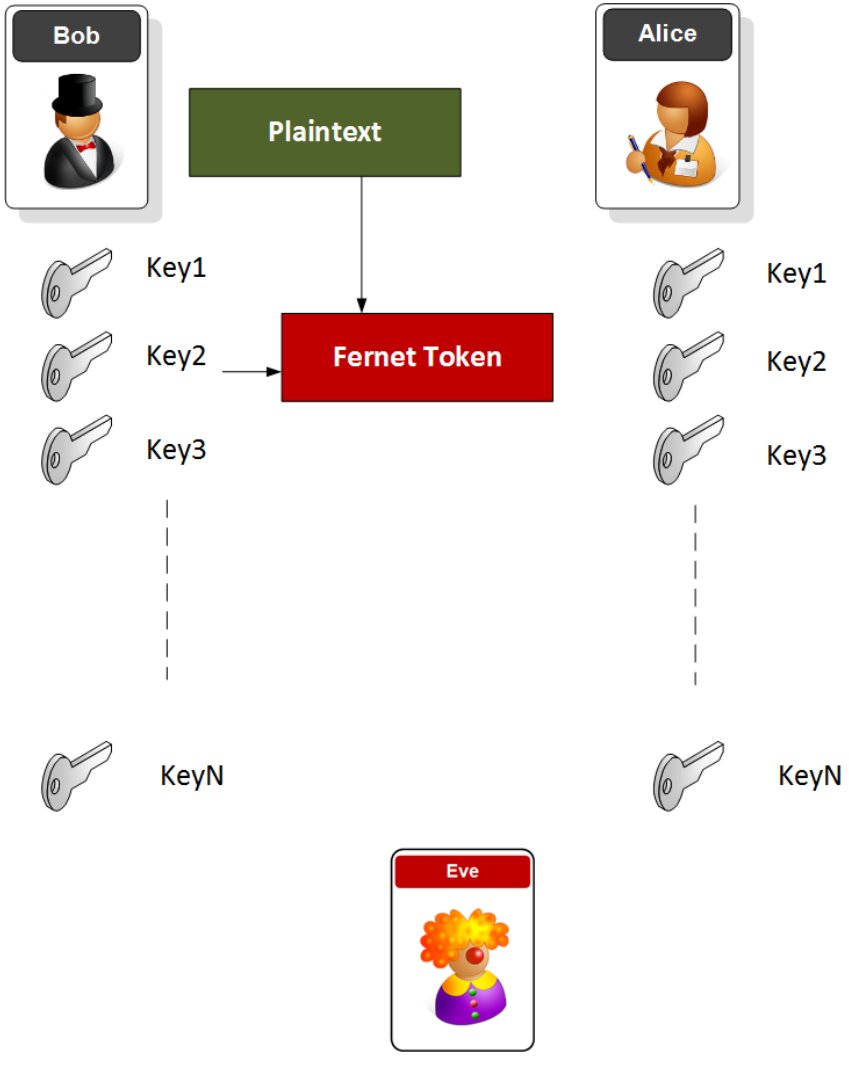
Determine

```
-s6eI5hyNh8liH7GqOurPC-vzPgNnxauKvRO4g03oYI=
```

```
Decoded:        password
Date created:   Wed Sep 30 13:22:19 2015
Current time:   Mon Dec 17 15:49:35 2018

======Analysis====
Decoded data:
8000000000560bd3fbfbd46cc53cff770c61f7e3b655407b221f140902f5fd9938b2ec67902bdbdba21ee22f0749427871a1d9ba236d8cb977a9f982bff7d2dff5698cbec7c42
cfbfd
Version:        80
Date created:   00000000560bd3fb
IV:             fbd46cc53cff770c61f7e3b655407b22
Cipher:         1f140902f5fd9938b2ec67902bdbdba2
HMAC:           1ee22f0749427871a1d9ba236d8cb977a9f982bff7d2dff5698cbec7c42cfbfd

======Converted====
IV:             fbd46cc53cff770c61f7e3b655407b22
Time stamp:     1443615739
Date created:   Wed Sep 30 13:22:19 2015
```

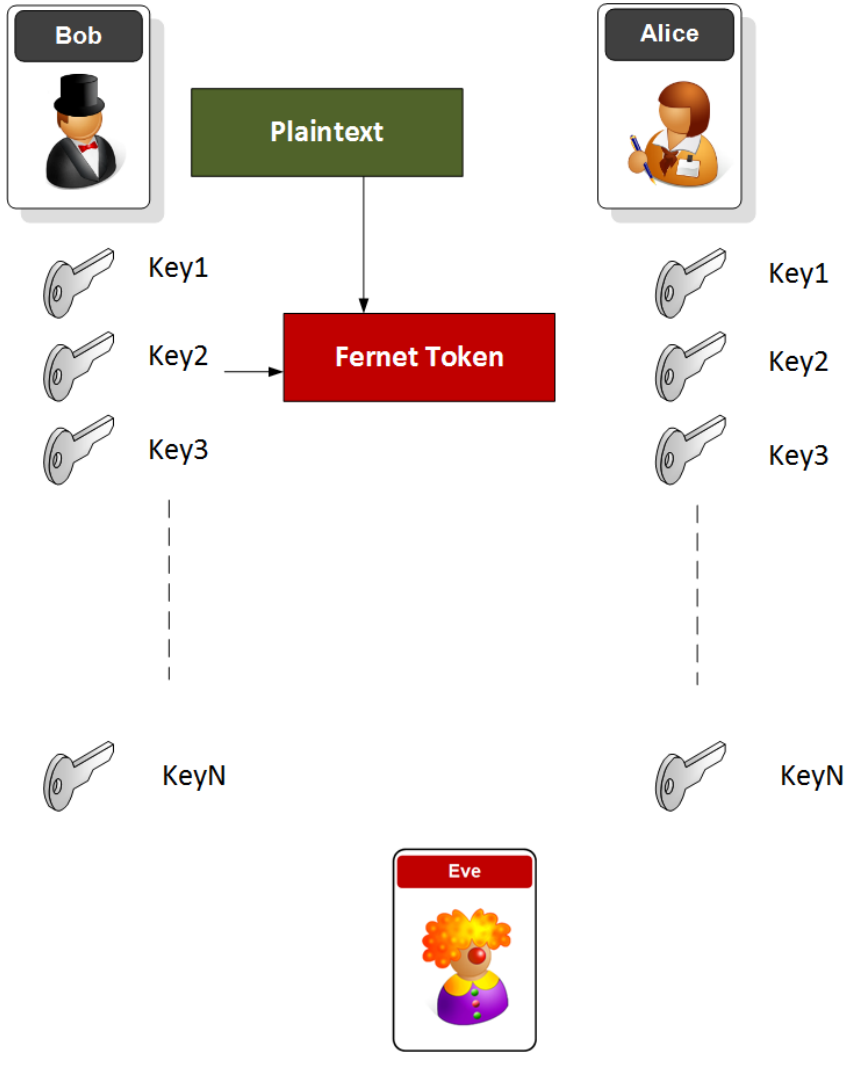
ba666d187

Key Rotation



Key Rotation

```
def get_key(password):

    salt = os.urandom(16)
    kdf = PBKDF2HMAC(algorithm=hashes.SHA256(),length=32,salt=salt,iterations=100,backend=default_backend())
    key=base64.urlsafe_b64encode(kdf.derive(password))
    return (key,salt)

if (len(sys.argv)>1):
        val=sys.argv[1]

if (len(sys.argv)>2):
        password=str(sys.argv[2])


(key1,salt) = get_key(password)
(key2,salt) = get_key(password)


print "Password:\t"+password
print "Key1: "+binascii.hexlify(bytearray(key1))
print "Key2: "+binascii.hexlify(bytearray(key2))
print "Key2: "+binascii.hexlify(bytearray(key3))


cipher_suite=MultiFernet([key1, key2])

cipher_suite = Fernet(key2)

cipher_text = cipher_suite.encrypt(val)
cipher=binascii.hexlify(bytearray(cipher_text))
print "Cipher: "+cipher


plain_text = cipher_suite.decrypt(cipher_text)
print "\nPlain text: "+plain_text
```

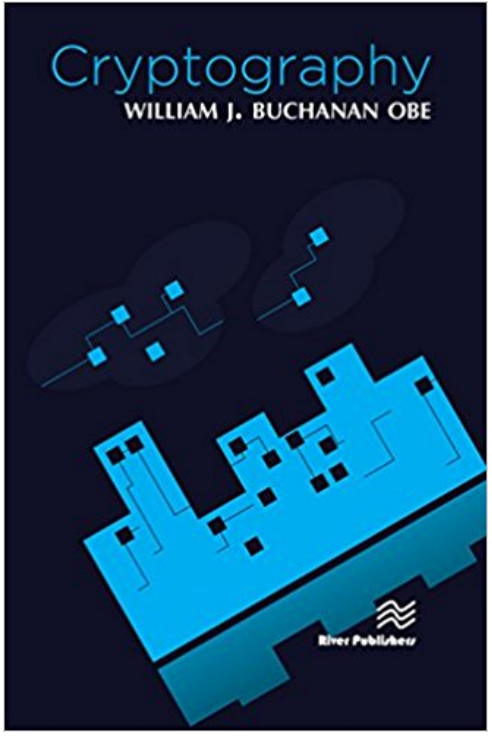


Tokens, Authorization and Docker

PAKE

Prof Bill Buchanan OBE

http://asecuritysite.com/encryption
http://asecuritysite.com/unit06



The problem with passwords, hashing and salt

User: alice
Password: qwerty

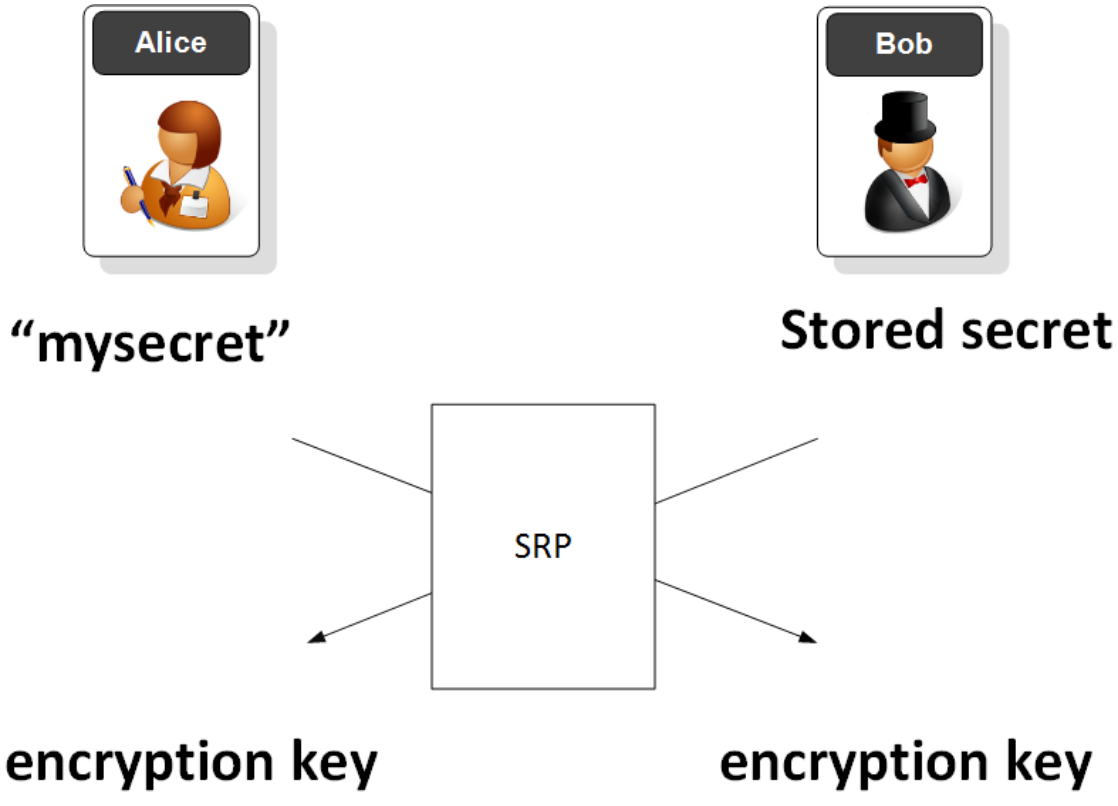
alice:$apr1$6Jda0$jkPR037bdd=

salt

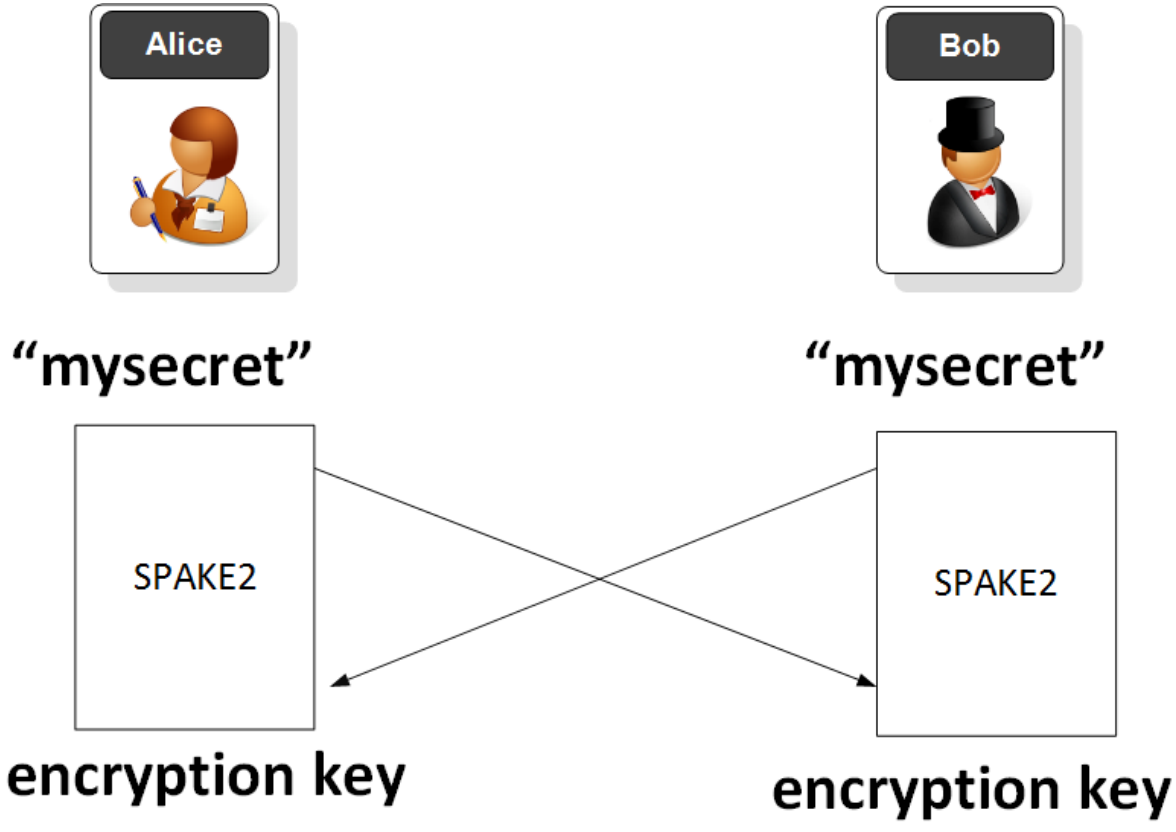
hash

Hash(“123456”+”6Jda0”)
Hash(“password”+”6Jda0”)
Hash(“qwerty”+”6Jda0”)

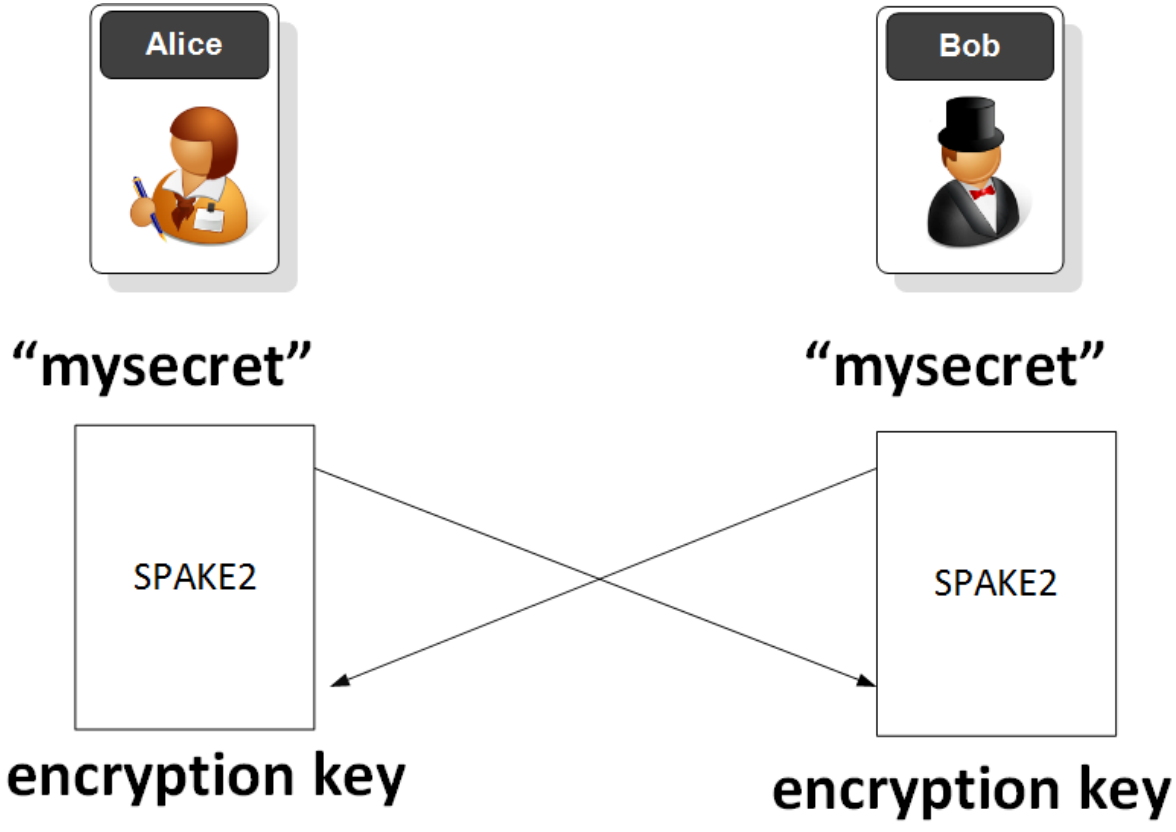
SRP


Alice
Bob
“mysecret”
Stored secret
SRP
encryption key
encryption key

SPAKE2 (Password-Authenticated Key Exchange)



SPAKE2 (Password-Authenticated Key Exchange)



SPAKE2 (Password-Authenticated Key Exchange)

$X = xG$ and $T = wM + X$

The value of T is sent to Bob. Bob also creates w, and picks a random number (y) from 0 to p (a prime number). He then calculates:

$Y = yG$ and $S = wN + Y$

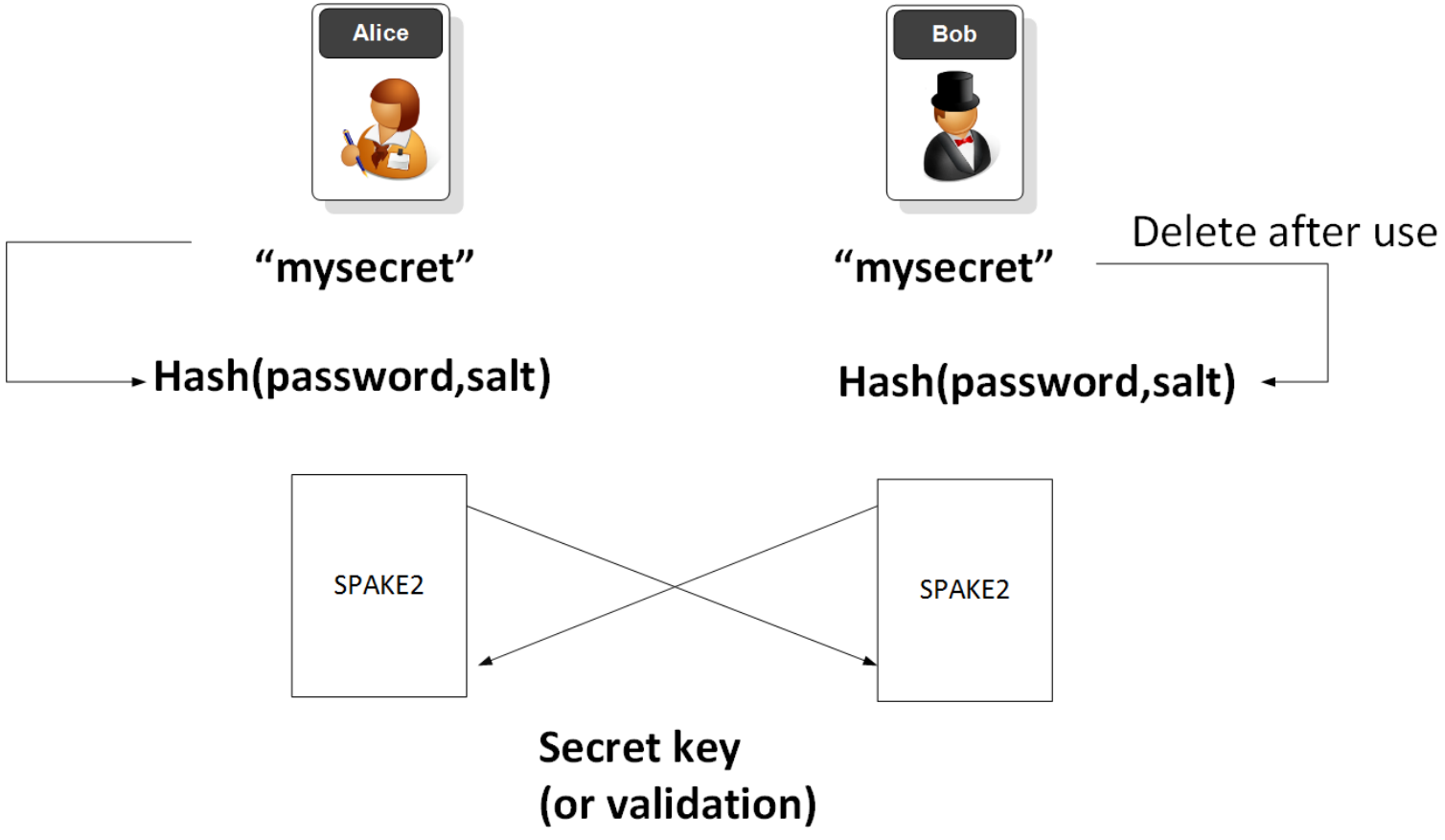
Bob sends the value of Y to Alice.

Alice calculates $K(Alice) = x(S - wN)$, and Bob calculates $K(Bob) = y(T - wM)$. These values are basically:

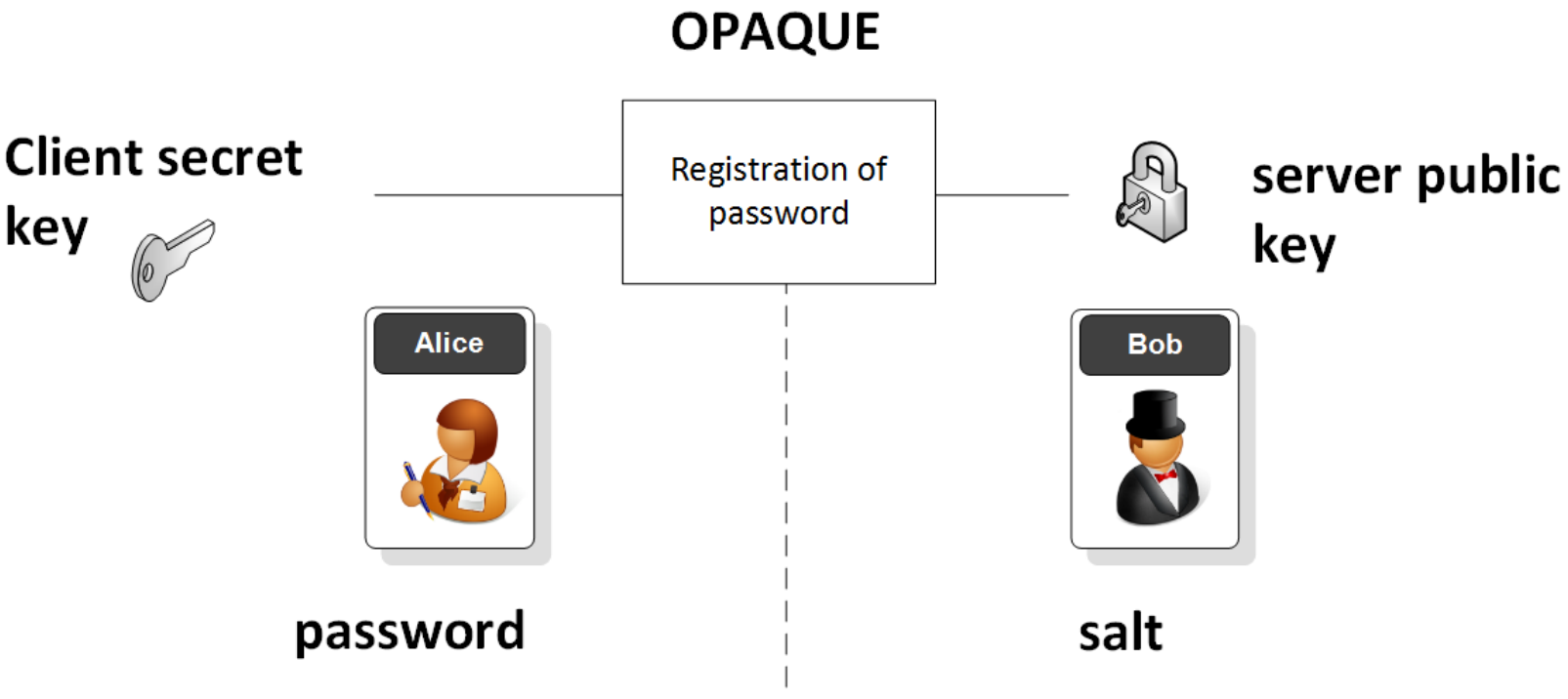
$K(Bob) = x(S - wN) = x(wN + Y - wN) = xY = xyG$

$K(Alice) = y(T - wM) = y(wM + X - wN) = yX = xyG$

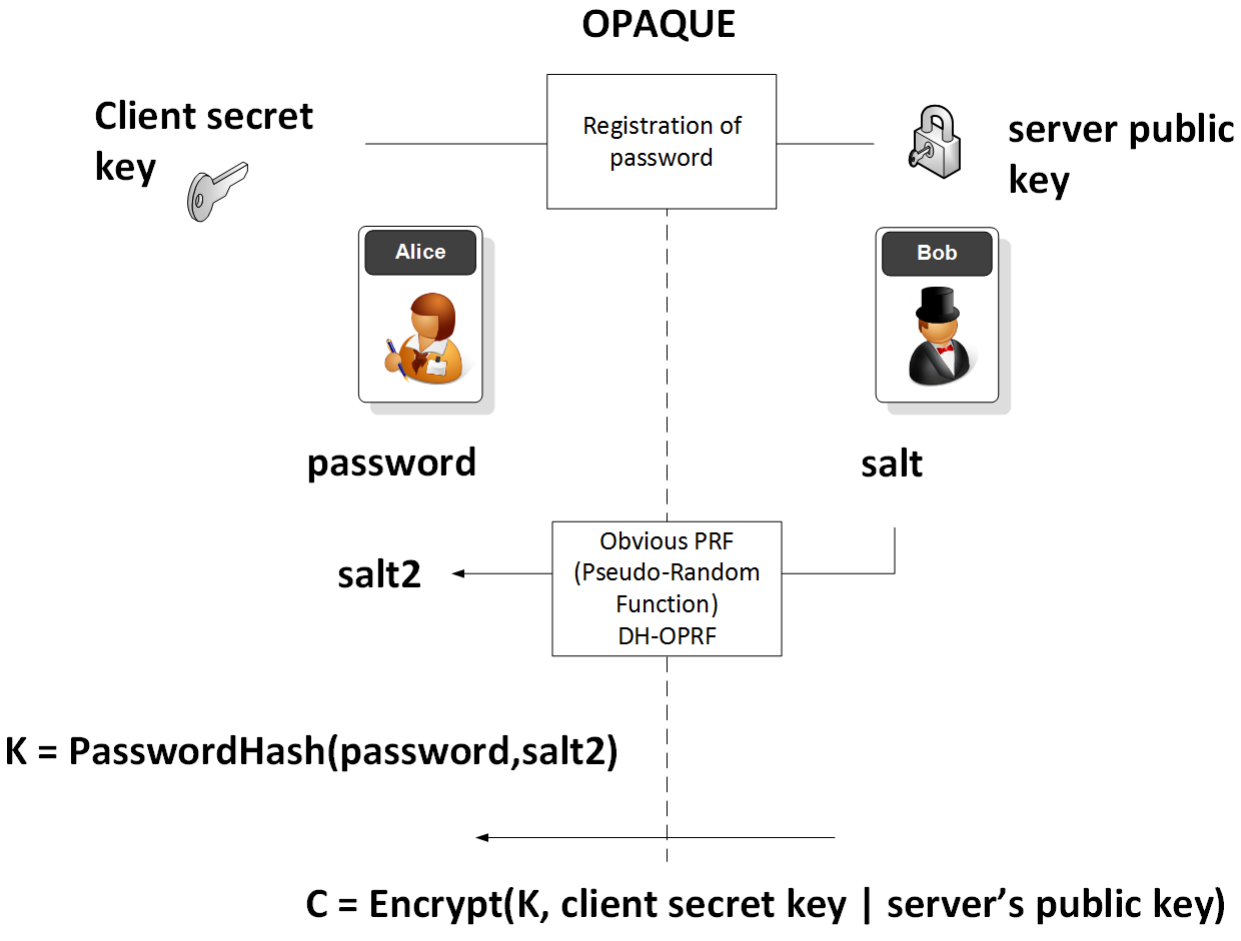
Proving Password



OPAQUE


OPAQUE
Client secret key
Registration of password
server public key
Alice
Bob
password
salt

OPAQUE

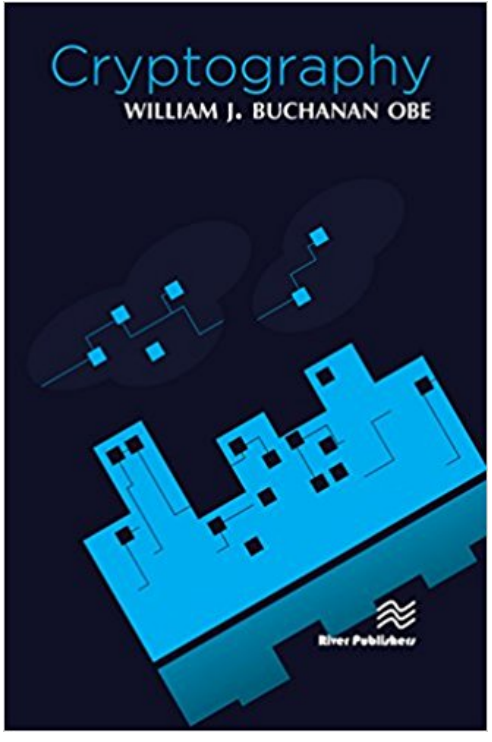

OPAQUE
Client secret key
Registration of password
server public key
Alice
Bob
password
salt
salt2
Obvious PRF (Pseudo-Random Function) DH-OPRF
K = PasswordHash(password,salt2)
C = Encrypt(K, client secret key | server's public key)

Tokens, Authorization and Docker

Docker

Prof Bill Buchanan OBE

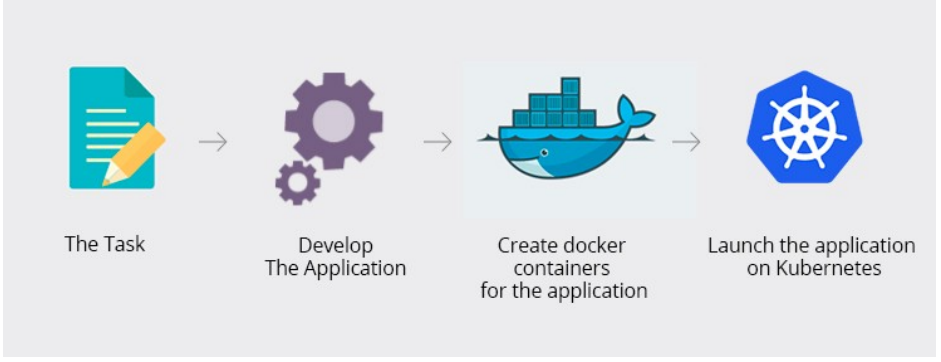
http://asecuritysite.com/encryption
http://asecuritysite.com/unit06



Docker

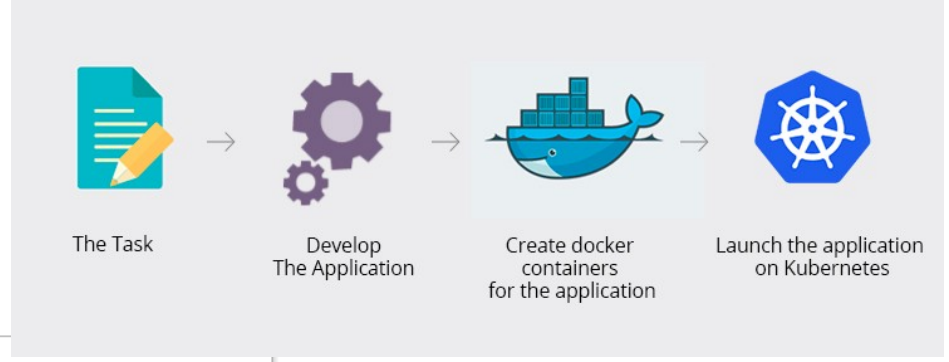
Hung, L. H., Kristiyanto, D., Lee, S. B., & Yeung, K. Y. (2016). GUIdock: using Docker containers with a common graphics user interface to address the reproducibility of research. *PloS one*, *11*(4), e0152686.

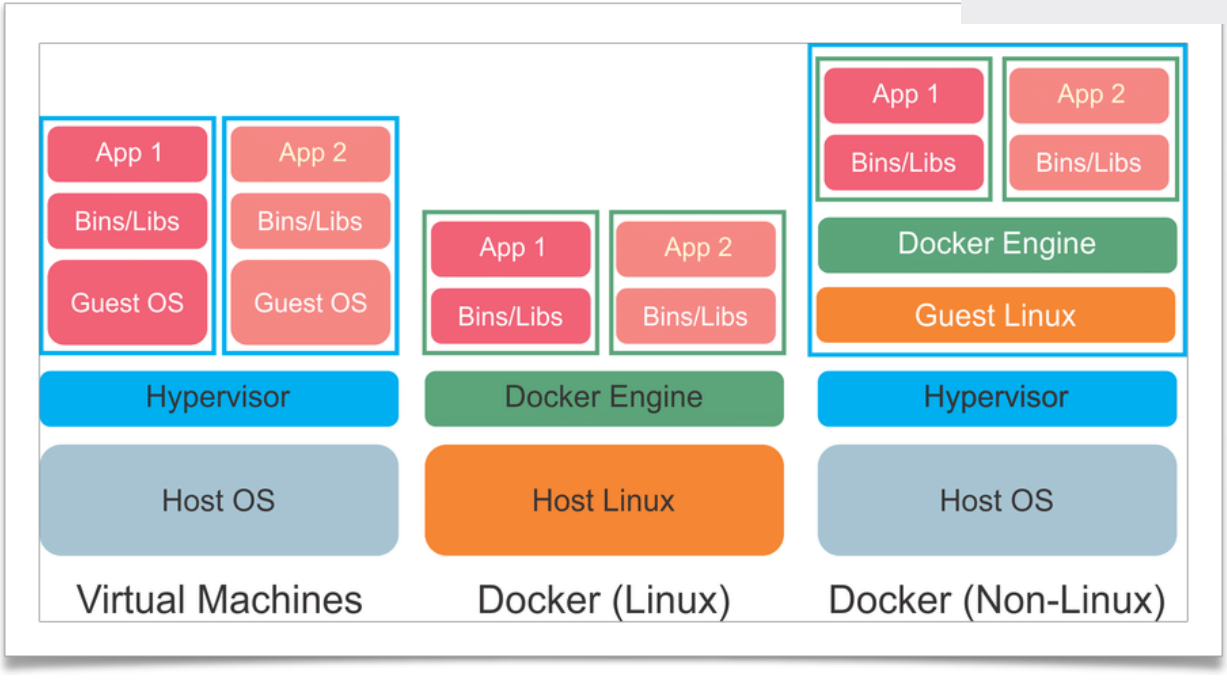
Docker



Hung, L. H., Kristiyanto, D., Lee, S. B., & Yeung, K. Y. (2016). GUIdock: using Docker containers with a common graphics user interface to address the reproducibility of research. *PloS one*, *11*(4), e0152686.

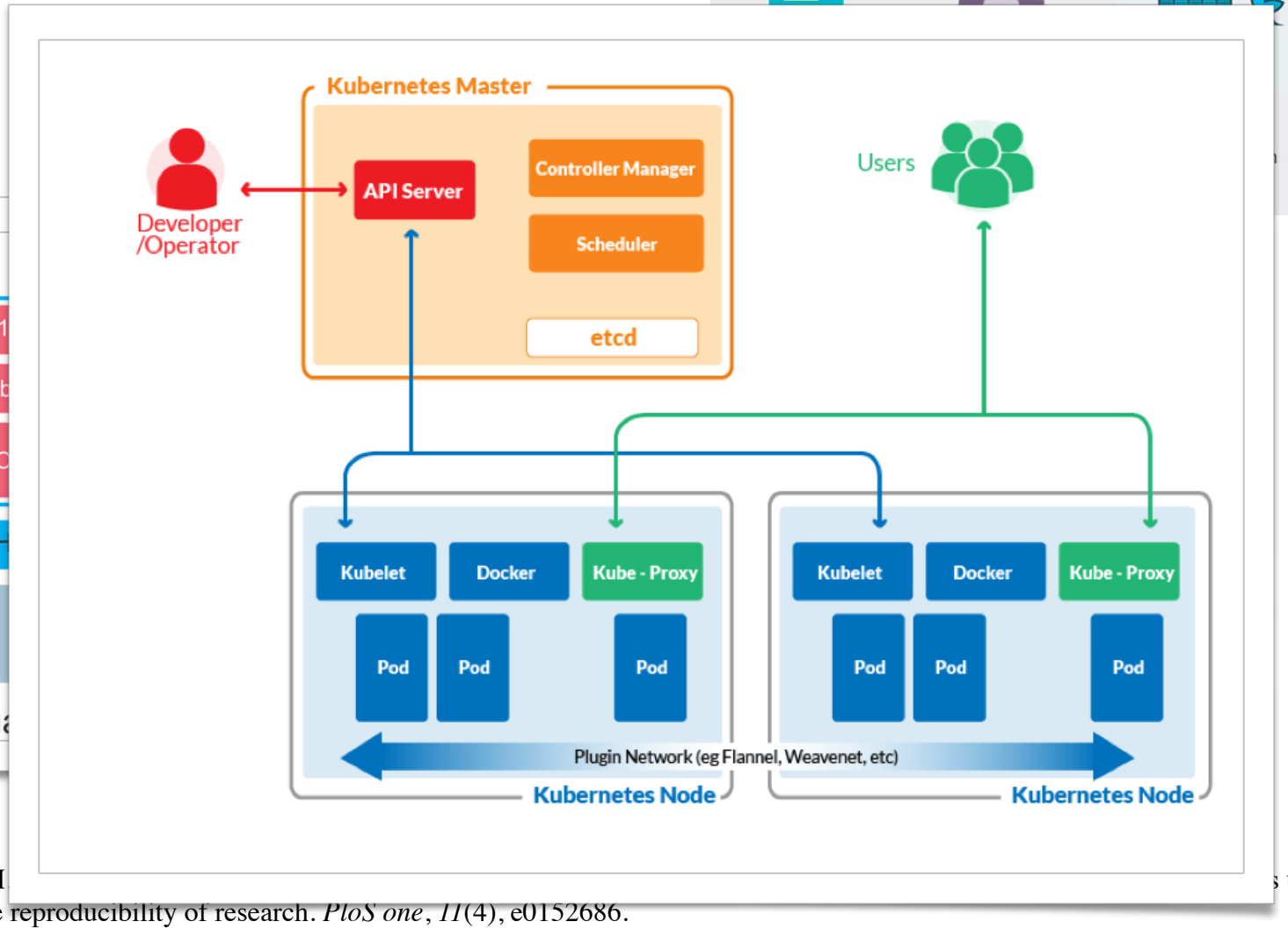
Docker

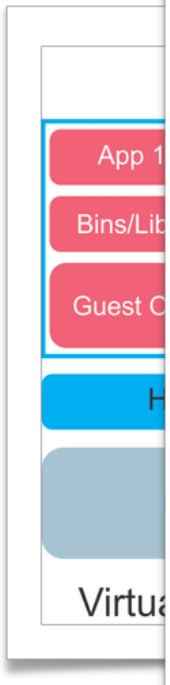




Hung, L. H., Kristiyanto, D., Lee, S. B., & Yeung, K. Y. (2016). GUIdock: using Docker containers with a common graphics user interface to address the reproducibility of research. *PloS one*, *11*(4), e0152686.

Docker





Hung, L. H … s user interface to address the reproducibility of research. *PloS one*, *11*(4), e0152686.

Docker

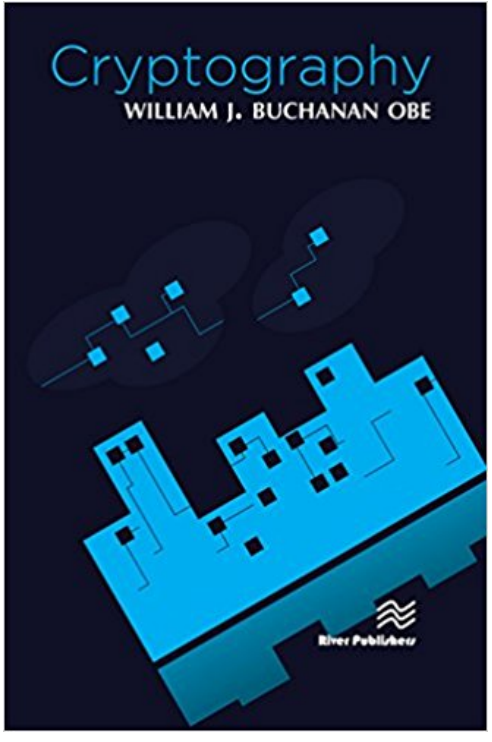
Kubernetes Master

```
FROM ubuntu:16.04

RUN apt-get update && apt-get install -y openssh-server
RUN mkdir /var/run/sshd
RUN echo 'root:THEPASSWORDYOUCREATED' | chpasswd
RUN sed -i 's/PermitRootLogin prohibit-password/PermitRootLogin
yes/' /etc/ssh/sshd_config

# SSH login fix. Otherwise user is kicked off after login
RUN sed 's@session\s*required\s*pam_loginuid.so@session optional
pam_loginuid.so@g' -i /etc/pam.d/sshd

ENV NOTVISIBLE "in users profile"
RUN echo "export VISIBLE=now" >> /etc/profile

EXPOSE 22
CMD ["/usr/sbin/sshd", "-D"]
```

Hung, L. H … s user interface to address the reproducibility of research. *PloS one*, *11*(4), e0152686.

Tokens, Authorization and Docker

ECR-20 Tokens
JSON Web Tokens
OAuth 2.0
Fernet Tokens
PAKE
Docker



Prof Bill Buchanan OBE

http://asecuritysite.com/encryption
http://asecuritysite.com/unit06