SeedQuest
Didery

Samuel M. Smith Ph.D.

Internet Identity Workshop

2018.10.24

sam@samuelsmith.org

Motivation

Decentralized Autonomic Data and the Three R's of Key Management

RWOT Spring 2018 https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust-spring2018/blob/master/final-documents/DecentralizedAutonomicData.pdf

Maintain a provenance chain for data flows that follows diffuse trust perimeter-less security principles

Minimally Sufficient Means

SeedQuest

https://github.com/reputage/seedQuest

Didery

https://github.com/reputage/didery

DAD: Decentralized Autonomic Data

Provenance for decentralized streaming data applications including transformations

D*ecentralized:* governance of the data may not reside with a single party, trust in the data provenance is diffuse, DID based.

A*utonomic:* self-managing or self-regulating. Self-managing includes cryptographic techniques for maintaining data provenance that make the data self-identifying, self-certifying, and self-securing.

- Autonomic implies the use of cryptographic signatures to provide a root of trust for data integrity and to maintain that trust over transformation of that data
- Key management is thus first order property of DAD items. 3 R's
- Reproduction, Rotation, and Recovery

Recovery Methods

Physical

- Shift exploit surface from network security to physical security
- Good backup method
- Memory burden is recall of physical location of physical backup

Secret Sharing

- Requires multi-party interactive infra-structure
- Recovery burden placed on third parties

Mnemonic

- Memory burden is lessened by using elaborative encoding techniques that humans are good at

Hybrid

- Minimally sufficient means

Memory Recovery

Human memory incompatible with long strings of random numbers.

Humans ill equipped for task of directly recalling cryptographic keys.

Want a “seduction” mnemonic.

Seduce users into effortless rehearsal

Leverage human innate ability to recall complex spatial temporal visual auditory paths with little or no rehearsal = elaborative encoding, rich encoding.

One time rehearsal.

Memorable experiences.

Dramatic, exiting, interesting, entertaining.

Cryptographic Security

information-theoretic security:

crypto-system that cannot be broken algorithmically even if the adversary has nearly unlimited computing power including quantum computing. It must be broken by brute force if at all. Brute force means that in order to guarantee success the adversary must search every combination of key or seed.

perfect security:

special case of *information-theoretic security* where cipher text provides no information about the key

Practical Perfect Security

Secret Sharing or Secret Splitting:

Simple N of N

Threshold M of N

One-time Pad:

Pad = string of random characters used to encode/decode message. Pad is as long as message.

Can be performed manually

Brute Force Cryptographic Strength

Systems with perfect security the critically parameter is the number of bits of entropy needed to resist any brute force attack

Entropy in bits per Shannon information theory

number of bits = $\log_2$(number of random possibilities)

Convention is that 128 bits is sufficient = 2^{128} possibilities or random choices

1 million super computers each performing 1 quadrillion trials per second will take 8,589,934,592 years for brute force exploit

One-time pad

Perfect security

Book cypher

Hiding in plain sight

Recovery task means no need to exchange one-time pad = self-contained

Complication with private key rotation history. Recover multiple keys

Modify to recovery of seed for CSPRNG that deterministically generates a one-time pad.

Strength bounded by seed entropy.

Rotation history much much shorter than CSPRNG period

Mnemonics

mnemonic = device or technique to aid human memory

memory task = recall 128-bit random number as a key or seed

need to recover set of private keys and/or private key rotation history not just a single private key
use a 128-bit random number as a seed (CSPRNG) to a crypto-system that hides and recovers the whole rotation history, such as, one-time pad.

Cryptographic strength governed by seed

elaborative encoding: use additional sensory cues to enhance recall, such as, story, path, spatial relationships, auditory, visual, color, fantastical juxtaposition

Example: method of loci or memory palace.

DiceWare

randomly select words from 7776 word list

log2(7776) = 12.9 bits per word

128 bits = 10 words in order

Non-trivial memory load. Requires rehearsal.

Recovery Process

Mneumonic to recall a single seed

Seed used with CSPRNG to generate one-time pad

One-time pad used to encrypt all the private keys

Because method employs *near* perfect security the encrypted private keys may be stored in the open. Depends on value of data controlled by keys

Recovery task is to recover seed, use seed to reproduce one-time pad, use pad to decrypt keys

SeedQuest

Virtual world role play game that generates seed through playing

Maximize elaborative encoding to Minimize mnemonic load

Entropy

16 sites = 2^4 = 4 bits of entropy per site selection

Each site interior has 32 spots = 2^5 = 5 bits of entropy per spot

Each spot has 4 actions = 2^2 = 2 bits of entropy per task

Each spot-action (task) choice = 5 + 2 = 7 bits of entropy

Sequence of 4 tasks gives 4*7 = 28 bits of entropy

Sequence of 4 sites each with 4 tasks = 4*(4 +28) = 128 bits of entropy

Comparable to the easiest level in a game like Legend of Zelda

Are you smarter than a ten year old?

SeedQuest

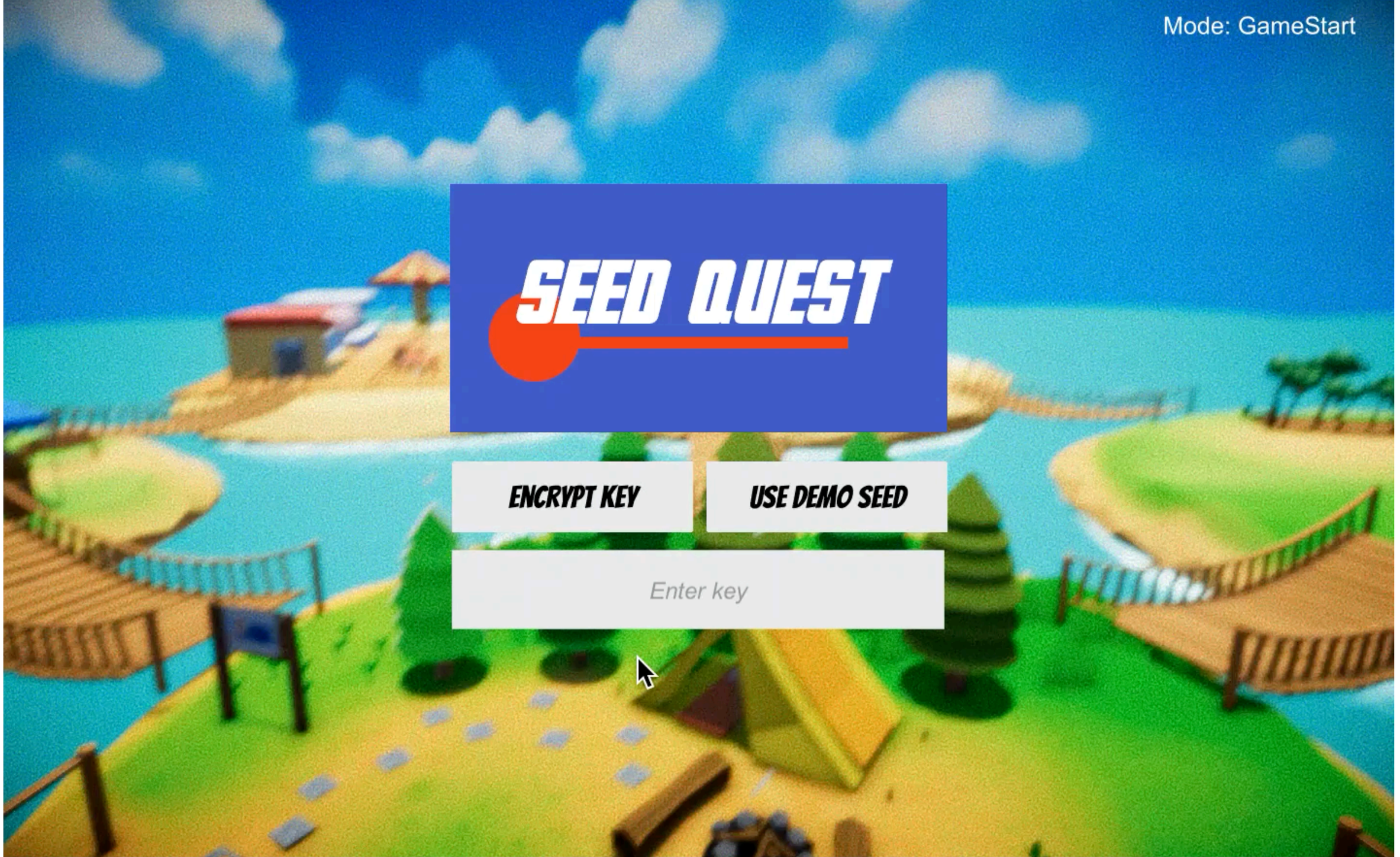
Two Game Play Modes:

Rehearsal

- Random seed selects path and tasks in sequence

Recovery

- Play game to recall path and tasks in sequence


Mode: GameStart
SEED QUEST
ENCRYPT KEY
USE DEMO SEED
Enter key

Rotation

Rotation = Revoke and Replace

Revoke only = Revoke and replace with null key

Replace only = security vulnerability

In general revocation and be implemented as special case of rotation

Why Rotate?

Public key vulnerable to attack due to time exposure

Continued use of private key increases vulnerability to exploit

Challenge is rotation after suspected exploit, exposure, or capture of private key.

Rotation operation that is signed by exploited private key is problematic.

Seek minimally sufficient means.

Using another key for rotation adds complexity and in a sense is just displacing the rotation problem without solving it.

Pre-Rotation

Likelihood of exploit is a function of exposure to probing or continued monitoring.

Narrowly restricting the the opportunity for exploit in time, place and method, such as, one time only, minimizes the vulnerability to exploit.

Exploiter has to either predict the one-time event or has to continuously universally monitor all events.

Create and declare the next “rotated” key (pre-rotated key) at the one-time only inception event for the original key. The private key of the pre-rotated key is then stored securely and is not exposed until a rotation event is needed.

The rotation event is now another one-time event determined by the key owner

Exploit of the original key cannot change the pre-rotated key or spoof a rotation event.

Rotation is self-contained

Pre-Rotation Requirements

Inception event is one-time only. Contains two public keys signed by original key.

Rotation event contains three public keys. Original, Pre-rotated, next pre-rotated. Signed by original and pre-rotated.

Private: pair-wise interaction requires recipient maintain log database of inception and rotation events. Can detect any replay attacks or attempts to spoof inception event.

Public: redundant, immutable, persistent, time-stamped log database of rotation event history for a given original (inception) key on replicant servers

Examples with DAD

Inception Event

```
{
    "id": "did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=",
    "changed" : "2000-01-01T00:00:00+00:00",
    "ensuer": "Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148="
}
\r\n\r\n
u72j9aKHgz99f0K8pSkMnyqwvEr_3rpS_z2034L99sTWrMIIJGQPbVuIJ1cupo6cfIf_KCB5ecVRYoFRzAPnAQ==
```

Rotation Event

```
{
    "id": "did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=",
    "changed" : "2000-01-01T00:00:00+00:00",
    "erster": "Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=",
    "signer": "Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=",
    "ensuer": "dZ74MLZXD-1QHoa73w9pQ9GroAvxqFi2RTZWlkC0raY="
}
\r\n\r\n
jc3ZXMA5GuypGWFEsxrGVOBmKDtd0J34UKZyTIYUMohoMYirR8AgH5O28PSHyUB-UlwfWaJlibIPUmZVPTG1DA==
\r\n\r\n
efIU4jplMtZzjgaWc85gLjJpmmay6QoFvApMuinHn67UkQZ2it17ZPebYFvmCEKcd0weWQONaTO-ajwQxJe2DA==
```

Listed Pre-rotation Structure

Inception Event

```
{
    "id": "did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=",
    "changed" : "2000-01-01T00:00:00+00:00",
    "signer": 0,
    "signers":
    [
        "Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=",
        "Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=",
    ]
}
\r\n\r\n
jc3ZXMA5GuypGWFEsxrGVOBmKDtd0J34UKZyTIYUMohoMYirR8AgH5O28PSHyUB-UlwfWaJlibIPUmZVPTG1DA==
```

Rotation Event

```
{
    "id": "did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=",
    "changed" : "2000-01-01T00:00:00+00:00",
    "signer": 1,
    "signers":
    [
        "Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=",
        "Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=",
        "dZ74MLZXD-1QHoa73w9pQ9GroAvxqFi2RTZWlkC0raY="
    ]
}
\r\n\r\n
jc3ZXMA5GuypGWFEsxrGVOBmKDtd0J34UKZyTIYUMohoMYirR8AgH5O28PSHyUB-UlwfWaJlibIPUmZVPTG1DA==
\r\n\r\n
efIU4jplMtZzjgaWc85gLjJpmmay6QoFvApMuinHn67UkQZ2it17ZPebYFvmCEKcd0weWQONaTO-ajwQxJe2DA==
```

Listed Pre-rotation Continued

Rotation Event


```
{
    "id": "did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=",
    "changed" : "2000-01-01T00:00:00+00:00",
    "signer": 2,
    "signers":
    [
        "Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=",
        "Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=",
        "dZ74MLZXD-1QHoa73w9pQ9GroAvxqFi2RTZWlkC0raY=",
        "3syVH2woCpOvPF0SD9Z0bu_OxNe2ZgxKjTQ961LlMnA="
    ]
}
\r\n\r\n
AeYbsHot0pmdWAcgTo5sD8iAuSQAfnH5U6wiIGpVNJQQoYKBYrPPxAoIc1i5SHCIDS8KFFgf8i0tDq8XGizaCg==
\r\n\r\n
o9yjuKHHNJZFi0QD9K6Vpt6fP0XgXlj8z_4D-7s3CcYmuoWAh6NVtYaf_GWw_2sCrHBAA2mAEsml3thLmu50Dw==
```

Multi-signature Pre-rotation

Inception Event

```
{
    "id": "did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=",
    "changed" : "2000-01-01T00:00:00+00:00",
    "signer": [0,1],
    "signers":
    [
        "Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=:blue",
        "Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=:red",
        "dZ74MLZXD-1QHoa73w9pQ9GroAvxqFi2RTZWlkC0raY=:blue",
        "3syVH2woCpOvPF0SD9Z0bu_OxNe2ZgxKjTQ961LlMnA=:red"
    ]
}
\r\n\r\n
AeYbsHot0pmdWAcgTo5sD8iAuSQAfnH5U6wiIGpVNJQQoYKBYrPPxAoIc1i5SHCIDS8KFFgf8i0tDq8XGizaCg==
\r\n\r\n
o9yjuKHHNJZFi0QD9K6Vpt6fP0XgXlj8z_4D-7s3CcYmuoWAh6NVtYaf_GWw_2sCrHBAA2mAEsml3thLmu50Dw==
```

Rotation Event

```
{
    "id": "did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=",
    "changed" : "2000-01-01T00:00:00+00:00",
    "signer": [2,3],
    "signers":
    [
        "Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=:blue",
        "Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=:red",
        "dZ74MLZXD-1QHoa73w9pQ9GroAvxqFi2RTZWlkC0raY=:blue",
        "3syVH2woCpOvPF0SD9Z0bu_OxNe2ZgxKjTQ961LlMnA=:red"
        "rTkep6H-4HA8tr54sHON1vWl6FEQt27fThWoNZsa88V=:blue",
        "7IUhL0JRaU2_RxFP0AL43wYn148Xq5YqaL6L48pf0fu=:red",
    ]
}
\r\n\r\n
AeYbsHot0pmdWAcgTo5sD8iAuSQAfnH5U6wiIGpVNJQQoYKBYrPPxAoIc1i5SHCIDS8KFFgf8i0tDq8XGizaCg==
\r\n\r\n
o9yjuKHHNJZFi0QD9K6Vpt6fP0XgXlj8z_4D-7s3CcYmuoWAh6NVtYaf_GWw_2sCrHBAA2mAEsml3thLmu50Dw==
\r\n\r\n
GpVNJQQoYKBYrPPxAoIc1i5SHCIDS8KFFgf8i0tDq8XGizaCgAeYbsHot0pmdWAcgTo5sD8iAuSQAfnH5U6wiI==
\r\n\r\n
8z_4D-7s3CcYmuoWAh6NVtYaf_GWw_2sCrHBAA2mAEsml3thLmu50Dwo9yjuKHHNJZFi0QD9K6Vpt6fP0XgXlj==
```

Didery

Service with two functions:

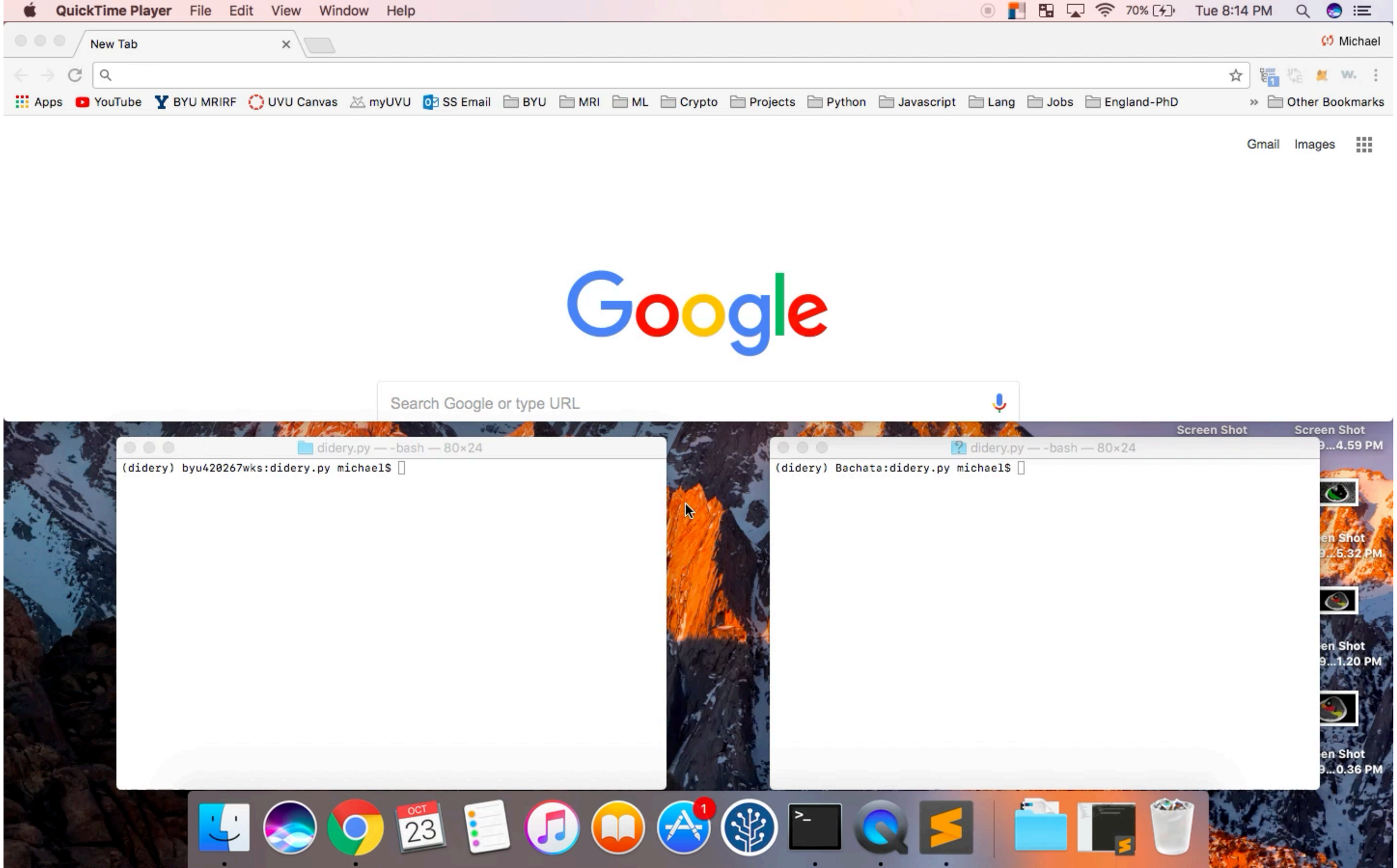
One Time Pad Encrypted Blob Storage

DID Pre-Rotation Histories

Server: Python

Web Client Javascript (npm)

Command Line Client, Python (pip)


QuickTime Player File Edit View Window Help
70% Tue 8:14 PM
New Tab
Michael
Apps YouTube BYU MRIRF UVU Canvas myUVU SS Email BYU MRI ML Crypto Projects Python Javascript Lang Jobs England-PhD Other Bookmarks
Gmail Images
Google
Search Google or type URL
didery.py — -bash — 80×24
(didery) byu420267wks:didery.py michael$
didery.py — -bash — 80×24
(didery) Bachata:didery.py michael$
Screen Shot
Screen Shot

Conclusion & Discussion

Backup Slides

DDID Management

DDID Database

```
{

    "did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=":
"did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=?chain=0\1\2",

   ...

}
```

DDID NameSpacing

```
did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=:blue?chain=0/1
did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=:red?chain=0/1
```

DDID Sequencing

```
did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=/10057
```

Change Detection

Prevent replay attacks:

- sequence number in DDID
- *changed* field with monotonically increasing sequence number or date time


```
{
    "id": "did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=/10057",
    "changed" : "2000-01-01T00:00:00+00:00",
    "data":
    {
        "temp": 50,
        "time": "12:15:35"
    }
}
\r\n\r\n
u72j9aKHgz99f0K8pSkMnyqwvEr_3rpS_z2034L99sTWrMIIJGQPbVuIJ1cupo6cfIf_KCB5ecVRYoFRzAPnAQ==
```

DDID Generation

On the fly DDIDs:

Data source is not identified so receiver generates DDID that is later correlated to or claimed by the data source

Public Derivation:

Client communicates with large number of public services

DDID is derived from root private key and public service identifier

Client does not need to store DDID but can re-derive on demand

Reproduction

Simple Privacy via unique cryptonym (DDID) per pair-wise interaction context

More sophisticated methods such as zero knowledge proofs may not minimally sufficient

DDIDS derived via some type of hierarchically deterministic algorithm allow for simple method to generated large numbers public DDIDS without having to store the associated private keys only the root private key

```
did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=?chain=0\1\2
```

```
did:dad:Qt27fThWoNZsa88VrTkep6H-4HA8tr54sHON1vWl6FE=
```

DID and DDID

DID = Decentralized Identifier
https://w3c-ccg.github.io/did-spec/

```
did:*method*:*idstring*
```

```
did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=
did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=:blue
did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=?who=me
```

DDO = DID Document, provides meta-data about DID
DAD Streaming = Multiplicity of data items and associated identifiers
DID/DDO pair per DAD item may not be practical
DDID = Derived DID = Unique DID format identifier derived from one root
DID/DDO that provides meta-data for a large number of DDIDs

Example Signed DAD


```
{
    "id": "did:dad:Xq5YqaL6L48pf0fu7IUhL0JRaU2_RxFP0AL43wYn148=",
    "data":
    {
        "name": "John Smith",
        "nation": "USA"
    }
}
\r\n\r\n
u72j9aKHgz99f0K8pSkMnyqwvEr_3rpS_z2034L99sTWrMIIJGQPbVuIJ1cupo6cfIf_KCB5ecVRYoFRzAPnAQ==
```

Minimally Sufficient Means

Streaming data applications may impose significant performance demands on the processing of the associated data

Desire efficient mechanisms for providing the autonomic properties of DADs

Find minimally sufficient means for managing cryptographic integrity

Key Reproduction, Rotation, and Recovery

Representation and Processing

Threshold Secret Sharing

Perfect security

M of N shards to recover

Requires multi-party interaction to recover shards

Not self-contained

Good backup method

Simple Secret-Splitting

Create multiple random shards

XOR shards together to generate/recover secret

Order independent combination of shards. Don't have to recall order

Recall task is N shards in any order

Cryptographic strength is now the entropy in bits due to the number of combinations of N possibilities per shard taken K shards at a time.

$$C(N,K) = \left(\begin{array}{c} N \\ K \end{array} \right) = \frac{N!}{K!(N-K)!}$$

$$\left(\frac{N}{K} \right)^K \leq \frac{N!}{K!(N-K)!} \leq \left(\frac{Ne}{K} \right)^K$$

Example

Suppose each shard has 35 bits of entropy = 2^{35} possibilities = billions

Four shards combined with secret splitting (XOR) means combinations of 2^{35} taken 4 at a time

$$\left(\frac{N}{K}\right)^{K}, N = 2^{35}, K = 4 = 2^{2}$$

$$\left(\frac{2^{35}}{2^{2}}\right)^{4} = \left(2^{33}\right)^{4} = 2^{132} = 132 \text{ bits of entropy}$$

Hybrid Recovery

Mneumonic for each of multiple shards.

In combination shards provide 128 bits of entropy for seed

Seed used with CSPRNG to generate one-time pad

One-time pad used to encrypt private key rotation history

Because method employs *near* perfect security the encrypted private key rotation history may be stored in the open. Depends on value of data controlled by key

Recovery task is to recall shards, combine shards to recover seed, use seed to reproduce one-time pad, use pad to decrypt key rotation history

Remaining challenge is finding sources of mnemonically recoverable sources of seed material.

Hiding seed shards in plain sight

Github.com repository commit hashes. (20 byte SHA-1 hash)

- 80 million repositories * 1000 commits per repository = 80 billion choices. Four choices > 128 bits
- Recall project name, repository name, commit date, which commit on a given data
- Mnemonic is GitHub search tags

Flickr.com Image Database (hash of display image scraped from site)

- 10 billion images Four choices > 124 bits
- Mnemonic is Flickr search tags and image contents

FamilySearch.org genelogical database (hash of primary record fields)

- 6 billion records each with 7 event types. Four choices > 128 bits
- Mnemonic is name, event type, event place, event date

Google Maps world database (hash of latitude, longitude)

- Inhabited land area at 4 decimal places = 10 m resolution = 40 bits of entropy. 4 choices > 128 bits
- Inhabited land area at 5 decimal places = 1 m resolution = 47 bits of entropy. 3 choices > 128 bits
- Mnemonic is country, region, city, neighborhood and geospatial details