



TALER

Taxable Anonymous Libre Electronic Reserves

Project number: Horizon Europe 101135475

D6.1

Educational materials

Due date of deliverable: 31.8.2024

Actual submission date: 31.8.2024

WP contributing to the deliverable: WP6

Start date of project: 1. December 2023

Duration: 3 years

Coordinator:

Eindhoven University of technology

www.taler.net/eurotaler

Revision 1.0

Project co-funded by the European Commission within Horizon Europe		
Dissemination Level		
PU	Public	<input checked="" type="checkbox"/>
PP	Restricted to other programme participants (including the Commission services)	<input type="checkbox"/>
RE	Restricted to a group specified by the consortium (including the Commission services)	<input type="checkbox"/>
CO	Confidential, only for members of the consortium (including the Commission services)	<input type="checkbox"/>

HISTORY OF CHANGES		
VERSION	PUBLICATION DATE	CHANGE
0.9	30.7.2024	First internally circulated version
1.0	31.8.2024	Minor edits based on internal review

Educational materials

Christian Grothoff Emmanuel Benoist

31.8.2024
Revision 1.0

The work described in this report has been funded (in part) by the European Union in the HORIZON-CL4-2023-HUMAN-01-CNECT call in project 101135475 TALER. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Abstract

This report is a collection of the lecture materials for integrating GNU Taler and related technologies into curricula on **Web programming with REST APIs** and/or **Applied cryptography**. The lectures use GNU Taler as a running example, relating theory to its applications in the context of the GNU Taler payment system.

The sources and Web sites with links to additional materials can be found in our public Git repository at <https://git.taler.net/lectures.git/>. This document is thus merely one possible rendering of the provided slides and exercises. Additionally, a small collection of video-tutorials can be found at <https://tutorials.taler.net/>.

Keywords: lectures, cryptography, RESTful, blockchain, privacy, anonymity, payments, applied cryptography

NEXT GENERATION INTERNET

REST API - HyperText Transfer Protocol

Emmanuel Benoist

29.08.2024

HyperText Transfer Protocol

Web

Headers

Methods

Principles

Request

Response

Caching

HTTP is the Cornerstone of World Wide Web (I)

HyperText Transfer Protocol

Used to download HTML pages

Used also to download other types of documents

- ▶ Cascading style sheets (CSS),
- ▶ JavaScript (JS),
- ▶ Images (JPEG, PNG, GIF, SVG, ...)
- ▶ ...

HTTP is the Cornerstone of World Wide Web (II)

Used to upload information onto the server,

- ▶ formulars containing data,
- ▶ files uploaded to the server.

HTTP for Client Server Communication (I)

HTTP is “stateless”

- ▶ The server processes each request independently of other requests from the same client.

Client sends a request

- ▶ to access some information / to send some information to the server
- ▶ contains information about the client itself (type of browser, accepted formats or accepted languages),
- ▶ info about the requested document,
- ▶ if information is sent, meta info about this (type and size for instance).

HTTP for Client Server Communication (II)

Server gives a response

- ▶ Contains the requested document (HTML / CSS / JS),
- ▶ meta information about the document (type, size, ...),
- ▶ meta information about the server (date, server, programs used, ...).

Example of a typical HTTP exchange

HTTP Request

- ▶ is sent by the browser,
- ▶ opens a TCP IP Connection to the port 80 of the server¹,
- ▶ asks for a resource (a page for instance).

HTTP Response

- ▶ says that it is an HTML document and its size,
- ▶ provides some optional details about the server (which server is used, or which version of PHP has been used for instance), and finally
- ▶ gives the main document back (the HTML document).

¹Or more realistically, a secure connection to port 443

Another example

HTTP Request

- ▶ Browser needs an image,
- ▶ an previous version of the image is already in its cache (locally stored from a previous request),
- ▶ the browser asks for the image, but only if newer.

HTTP Response

- ▶ Server responds that the image in the cache is still valid.

Syntax HTTP Request

HTTP Request Header

- ▶ First line (Query string) = Method URI Protocol
- ▶ Header lines = header: value
- ▶ End of header = an empty line

HTTP Request Body

- ▶ Depending on the method² can contain information sent to the server

²details on methods presented later

Example: HTTP Request for a page

GET /fr/ HTTP/2

Host: www.taler.net

User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:121.0) Gecko/20100

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/

Accept-Language: fr-FR,fr-CH;q=0.9,fr;q=0.7,en-US;q=0.6,en;q=0.4,de-DE;q=0.3,de

Accept-Encoding: gzip, deflate, br

DNT: 1

Sec-GPC: 1

Connection: keep-alive

Upgrade-Insecure-Requests: 1

Sec-Fetch-Dest: document

Sec-Fetch-Mode: navigate

Sec-Fetch-Site: none

Sec-Fetch-User: ?1

TE: trailers

HTTP Methods (I)

GET to download resource

- ▶ Access to a page or
- ▶ Download any resource
- ▶ Form sending small information
- ▶ URL contains URL-encoded parameters
- ▶ HTTP body SHOULD remain empty
- ▶ Result can be cached.

HTTP Methods (II)

POST to send information

- ▶ Used by forms to send info to the server (for large information)
- ▶ Data is written in the body of the request
- ▶ Can not be cached.

HTTP is Request/Response oriented

Request

- ▶ Request for a page (giving its URL)
- ▶ for an image or any file
- ▶ contains the input of a form
- ▶ contains some settings of the browser

Response

- ▶ The file (html or any file)
- ▶ Contains properties of the document
- ▶ Can lead to another URL

Request

Syntax

```
METHODE URI PROTOCOL  
HEADER1: VALUE  
...  
HEADERn: VALUE  
  
BODY OF THE MESSAGE
```

Example

Very minimalistic: the only header that is obligatory is host name.

```
GET / HTTP/1.1  
host: www.taler.net
```


Request

A not so simple example

```
GET / HTTP/2
Host: www.taler.net
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:124.0) Gecko/20100101 Firefox/124.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: fr-FR,fr-CH;q=0.9,fr;q=0.7,en-US;q=0.6,en;q=0.4,de-DE;q=0.3,de-CH;q=0.1
Accept-Encoding: gzip, deflate, br
DNT: 1
Sec-GPC: 1
Connection: keep-alive
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: none
Sec-Fetch-User: ?1
```

Request Headers (I)

Description of the client

- ▶ **User-Agent**: browser and OS description
- ▶ **Accept**: which documents mime-types are accepted
- ▶ **Accept-Language** idem for the languages
- ▶ **Accept-Encoding** which encoding can be used content
- ▶ ...

Request Headers (II)

Description of the request

- ▶ Host usefull for virtual servers
- ▶ Proxy-Connection: keep-alive Allows more than one request in one connexion
- ▶ Keep-Alive: 300 set the time-out
- ▶ Referer Which page contains the link that created the request

Send information to the server

Forms in HTML

Example:

```
<form method="POST" action="http://localhost/test.php">  
<input type="text" name="text1" >  
<input type="hidden" name="text2" value="80">  
<input type="submit" value="OK">  
</form>
```

```
<form method="GET" action="http://localhost/test.php">  
<input type="text" name="text1" >  
<input type="hidden" name="text2" value="80">  
<input type="submit" value="OK">  
</form>
```

Method for forms: GET

Method = GET

- ▶ For GETTING a page
- ▶ Used for URL typed in the address bar
- ▶ Used for links
- ▶ Can send a small set of information
- ▶ The information **MUST** not reach the server
- ▶ It can be cached

Method for forms: POST

Method = POST

- ▶ For POSTING information to the server
- ▶ Can contain large data
- ▶ Must arrive at the server
- ▶ Can not be cached

The GET Method

The GET request

- ▶ The following is a GET request
- ▶ There is no content,
- ▶ The values are sent in the URL (they are URLEncoded)

```
GET /tmp/example-forms.html?text1=Hello+Taler&text2=80 HTTP/3
```

```
Host: www.taler.net
```

```
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:124.0) Gecko/20100
```

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/
```

```
Accept-Language: fr-FR,fr-CH;q=0.9,fr;q=0.7,en-US;q=0.6,en;q=0.4,de-DE;q=0.3,de
```

```
Accept-Encoding: gzip, deflate, br
```

```
Referer: https://www.taler.net/tmp/example-forms.html
```

```
...
```

URL encoding (I)

Coding

A URL can only contain alphanumerical characters plus the following ones:

- ▶ ? marks the end of the resource and the start of parameters
- ▶ = to separate a parameter name and its value
- ▶ & to separate parameters
- ▶ + to represent spaces

Automatically encoded in the FORM

- ▶ Couple: (variable, value).

URL encoding (II)

Links in a page

- ▶ A link in a page executes a `GET` method
- ▶ You can set values for links too. You can insert any parameter in the URL: ``

Any GET URL

- ▶ Any GET request may contain some parameters
- ▶ Generated from a form
- ▶ Implemented statically inside the document
- ▶ Generated in JavaScript inside the DOM
- ▶ Generated by any program.

The POST request

The following is the request generated by Firefox on a GNU-Linux Platform.

The content type is “urlencoded”

The values are sent in the body of the request

There is a description of the content (Content-type:, Content-length:)

The POST request

```
POST /tmp/example-forms.html HTTP/3
Host: www.taler.net
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10.15; rv:124.0) Gecko/20100101 Firefox/124.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
Accept-Language: fr-FR,fr-CH;q=0.9,fr;q=0.7,en-US;q=0.6,en;q=0.4,de-DE;q=0.3
Accept-Encoding: gzip, deflate, br
Referer: https://www.taler.net/tmp/example-forms.html
Content-Type: application/x-www-form-urlencoded
Content-Length: 26
Origin: https://www.taler.net
DNT: 1
Sec-GPC: 1
Connection: keep-alive
```

Response

From the server to the client

- ▶ Is a response for the question contained in the request

Contains a status

- ▶ Document OK, moved permanently, does not exist (404), the version in cache is still ok, ...

And the desired document or information

- ▶ The body contains the document (html, gif, jpeg,...)
- ▶ The header contains meta information (date of production, validity, language, ...)

Response, Example

```
HTTP/2 200
server: nginx
date: Wed, 17 Apr 2024 05:27:59 GMT
content-type: text/html
last-modified: Mon, 15 Apr 2024 20:18:46 GMT
vary: Accept-Encoding
etag: W/"661d8ba6-3e21"
...
content-encoding: gzip
X-Firefox-Spdy: h2
```

```
<!DOCTYPE html>
<html lang="fr">
  <head>
    <title >GNU Taler </title >
```

Response (Syntax)

Syntax

STATUS-LINE

HEADER1: value

HEADER2: value

HEADER3: value

BODY OF THE DOCUMENT

Status Line (I)

Format

Status-Line = HTTP-Version SP Status-Code SP Reason-Phrase CRLF

Status-Code:

The Status-Code element is a 3-digit integer result code of the attempt to understand and satisfy the request.

Reason-Phrase

The Reason-Phrase is intended to give a short textual description of the Status-Code. The Status-Code is intended for use by automata and the Reason-Phrase is intended for the human user.

Status Line (II)

Examples

HTTP/1.1 200 OK

HTTP/1.1 404 Not Found

HTTP/1.1 501 Method Not Implemented

Status Code

- ▶ **1xx: Informational** - Request received, continuing process
- ▶ **2xx: Success** - The action was successfully received, understood, and accepted
- ▶ **3xx: Redirection** - Further action must be taken in order to complete the request
- ▶ **4xx: Client Error** - The request contains bad syntax or cannot be fulfilled
- ▶ **5xx: Server Error** - The server failed to fulfill an apparently valid request

Status Code (Cont.)

Success

200 : OK
201 : Created
202 : Accepted
203 : Non-Authoritative Information
204 : No Content
205 : Reset Content

Informational

100 : Continue
101 : Switching Protocols

Status Code (Cont.)

Redirection

- 300 : Multiple Choices
- 301 : Moved Permanently
- 302 : Found
- 303 : See Other
- 304 : Not Modified
- 305 : Use Proxy
- 307 : Temporary Redirect

Status Code (Cont.)

Client Error

400 : Bad Request
401 : Unauthorized
402 : Payment Required
403 : Forbidden
404 : Not Found
405 : Method Not Allowed
406 : Not Acceptable
407 : Proxy Authentication Required
408 : Request Time-out
409 : Conflict
410 : Gone
411 : Length Required
412 : Precondition Failed
413 : Request Entity Too Large
414 : Request-URI Too Large

Status Code (Cont.)

Server Error

- 500 : Internal Server Error
- 501 : Not Implemented
- 502 : Bad Gateway
- 503 : Service Unavailable
- 504 : Gateway Time-out
- 505 : HTTP Version not supported

Example: Access a resource

Request to the site `www.taler.net`

```
GET /en/ HTTP/2  
Host: www.taler.net
```

Status 200 OK

```
HTTP/2 200  
server: nginx  
date: Wed, 17 Apr 2024 05:33:08 GMT  
content-type: text/html  
last-modified: Mon, 15 Apr 2024 20:18:46 GMT  
vary: Accept-Encoding  
etag: W/"661d8ba6-3d01"  
strict-transport-security: max-age=63072000; includeSubDomains; preload  
x-xss-protection: 1; mode=block  
x-frame-options: SAMEORIGIN  
x-content-type-options: nosniff
```

...

Access a resource in different formats

Client can indicate which format it prefers

- ▶ Header `Accept` contains a list of preferred mime types.
- ▶ Example:

```
Accept: text/html,application/xhtml+xml,application/xml;q=0.9
```

Server will deliver the resource in the right format.

- ▶ The same resource can be delivered in different forms,
- ▶ XML, JSON, PDF documents for instance.

Example of a resource in different formats (I)

Terms and conditions for a Taler exchange can be accessed in different languages and formats

- ▶ Clients must set the two headers `Accept` and `Accept-Language`
- ▶ Server will deliver the document in the right language and the right format.

Request for a pdf in German

```
curl --request GET --url https://exchange.demo.taler.net/terms \  
  --header 'Accept: application/pdf' --header 'Accept-Language: de' \  
  --output 'terms.pdf'
```


Example of a resource in different formats (II)

Request for a HTML document in English

The same resource (<https://exchange.demo.taler.net/terms>) will deliver a different document (english and html) but containing the same information.

```
curl --request GET \  
  --url https://exchange.demo.taler.net/terms \  
  --header 'Accept: text/html' \  
  --header 'Accept-Language: en'
```

Caching (I)

Content of a response to a GET request can be cached (and should be)

- ▶ The document is based on “information” that has a versioning.
- ▶ This information can be cached to save bandwidth and time.
- ▶ Cache can be local (inside the browser) but also inside reverse proxies or Content Delivery Networks (CDNs).

Caching (II)

The response contains following headers concerning caching:

- ▶ `etag`: version number for the “information”
- ▶ `last-modified`: the date of last modification of the “information”
- ▶ `vary`: shows the parameters for which the cached response will not be valid any more.

For instance `vary : accept` means that if `accept` changes from `text/html` to `text/csv` the page in cache is not valid anymore.

Caching information using Etag

Etag is a version number.

- ▶ It does not have to change for each small modification,
- ▶ for instance, adding a comma may not change the tag.
- ▶ It should change with significant revisions,
- ▶ it is included in a response to describe the body.

If a cached version is available, the client will ask *“Did it change?”*

- ▶ The request includes the header `If-None-Match` with the value of Etag of the version in cache.
- ▶ If this is still valide, response code is 304 Not Modified (and body is empty).

Example of use of Etag (I)

Request a page:

```
curl --request GET --verbose \  
  --url https://exchange.demo.taler.net/terms \  
  --header 'Accept: text/html'
```

Response:

```
HTTP/2 200  
date: Fri, 07 Jun 2024 12:51:27 GMT  
content-type: text/html  
content-length: 20381  
vary: Accept-Encoding  
access-control-allow-origin: *  
access-control-expose-headers: *  
avail-languages: de,en  
cache-control: public,max-age=864000  
content-language: en  
etag: MOF6HBHJ4GXQ5YNTDT5J2AM5C22JPM9ZTRNQ3485HM8K5CVM65W0  
taler-terms-version: exchange-tos-v0  
expires: Sat, 08 Jun 2024 12:51:27 GMT  
vary: Accept-Language,Accept,Accept-Encoding  
strict-transport-security: max-age=63072000; includeSubDomains; preload
```

Example of use of Etag (II)

We request again the page that we have in cache

```
curl --request GET --verbose \  
  --url https://exchange.demo.taler.net/terms \  
  --header 'Accept: text/html\' \  
  --header 'If-None-Match: MOF6HBJ4CXQ5YNTDT5J2AM5C22JPM9ZTRNQ3485HM8K5CVM65W0'
```

Response just says the version is still valid (contains no body).

```
HTTP/2 304  
server: nginx  
date: Fri, 07 Jun 2024 12:56:32 GMT  
access-control-allow-origin: *  
access-control-expose-headers: *  
etag: MOF6HBJ4CXQ5YNTDT5J2AM5C22JPM9ZTRNQ3485HM8K5CVM65W0  
expires: Sat, 08 Jun 2024 12:56:32 GMT  
strict-transport-security: max-age=63072000; includeSubDomains; preload
```

Caching of information using the date

We ask for the page in cache using the date we received.

```
curl --request GET \  
  --verbose \  
  --url https://exchange.demo.taler.net/terms \  
  --header 'Accept: text/html\' \  
  --header 'If-Modified-since: Fri, 07 Jun 2024 12:51:27 GMT '
```

Response: Cache is still valid, Status 304 Not Modified

Conclusion (I)

HTTP is a Request Response protocol

- ▶ A request is sent by the client,
- ▶ A response is generated by the server.

HTTP is sessionless

- ▶ Unlike SSH or SFTP for instance

Conclusion (II)

HTTP GET requests can be cached

- ▶ To save time and money, the same GET request should return the same value.

These features will be used to build a Computer to Computer communication architecture: REST API's

More details in the RFC 2616

<https://datatracker.ietf.org/doc/html/rfc2616>

Examples for this course

Examples are taken out of GNU-Taler implementation

- ▶ Main web site
`https://www.taler.net`
- ▶ Demo merchant back-end
`https://backend.demo.taler.net`
- ▶ Demo book-store
`https://shop.demo.taler.net/`
- ▶ Documentation of the Core API:
`https://docs.taler.net/core/index.html`

Acknowledgements

Funded by the European Union (Project 101135475).



**Co-funded by
the European Union**

Funded by SERI (HEU-Projekt 101135475-TALER).

Project funded by



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs,
Education and Research EAER
**State Secretariat for Education,
Research and Innovation SERI**

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union.
Neither the European Union nor the granting authority can be held responsible for them.

00-HTTP : Exercises

E. Benoist

August 27, 2024

1 Install CURL and WGET

Install `wget` and `curl` in your system. Something like `sudo apt install wget curl` should do the job.

Read the manpages of the two softwares to learn how to generate GET and POST requests.

2 Send a Request for a page

Download the page `https://www.taler.net` first using `wget` and then using `curl`.

3 Fill out a form

3.1 GET Method

We have the page `https://www.benoist.ch/ngi-taler/GET-TEST.php`. That page contains a form that is sent to the page `https://www.benoist.ch/ngi-taler/GET-TEST-2.php`. Use the *Developer Mode* of your browser to study the request that is generated by this form.

Use `wget` and `curl` to send the same values of the form. Send your first name as input.

3.2 POST Method

We have the page `https://www.benoist.ch/ngi-taler/POST-TEST.php`. That page contains a form that is sent to the page `https://www.benoist.ch/ngi-taler/POST-TEST2.php`. Use the *Developer Mode* of your browser to study the request that is generated by this form.

Use `wget` and `curl` to send the same values of the form. Send your first name as input.

3.3 Script

Write a script that contacts the server by sending `GET` and `POST` requests to the previous URLs.

NEXT GENERATION INTERNET

REST API - Introduction

Emmanuel Benoist

29.08.2024

Introduction

Web Architecture

The REST architecture

JSON

Parsing JSON

Connection to a Server

Direct Connection

Conclusion

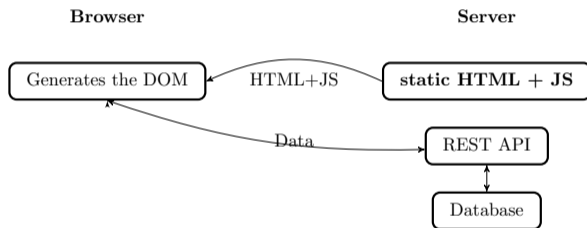
Classical Web Architecture



Web pages rendered on the server

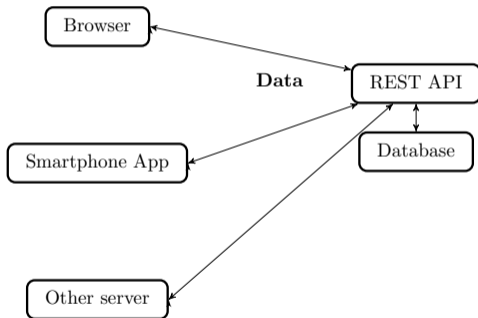
Many existing frameworks

Single Page Application



Runs inside the client application (browser) and communicate with an API.

REST API



**Why not use the same interface for other means?
RESTful Application Programming Interface**

REST: REpresentational State Transfer (I)

software architectural style

- ▶ REST defines a set of constraints for how the architecture of a distributed system should work.

Developed by Roy Fielding

- ▶ For his PhD Thesis: *Architectural Styles and the Design of Network-based Software Architectures*[1]

REST: REpresentational State Transfer (II)

Goals:

“Throughout the HTTP standardization process, I was called on to defend the design choices of the Web. That is an extremely difficult thing to do within a process that accepts proposals from anyone on a topic that was rapidly becoming the center of an entire industry. I had comments from well over 500 developers, many of whom were distinguished engineers with decades of experience, and I had to explain everything from the most abstract notions of Web interaction to the finest details of HTTP syntax. That process honed my model down to a core set of principles, properties, and constraints that are now called REST.”

From Web Services to RESTful services (I)

Web Services

- ▶ At the beginning: Ad hoc protocols for each service
 - ▶ parsing necessary for each of them
- ▶ First standards: Web Services using SOAP use HTTP as underlying protocol
 - ▶ SOAP works with XML
 - ▶ Parsing is done by an XML parser

From Web Services to RESTful services (II)

RESTful services

- ▶ Enable interoperability of heterogenous servers,
- ▶ Lightweight integration using HTTP,

Main Ideas

- ▶ Use URIs (Uniform Resource Identifier) to name resources
- ▶ Access resources using HTTP
- ▶ Data can be represented in JavaScript Object Notation (JSON), XML, DER and CBOR

From Web Services to RESTful services (III)

Formats for Data

Format	JSON	XML	DER	CBOR
<i>Human readable</i>	Yes	Yes	No	No
<i>Size</i>	Large	extra large	small	small
<i>Validation</i>	No	Possible	Yes	No
<i>Widely used</i>	Yes	Old	No	Just starting

REST Principles (I)

Uniform Interface

- ▶ All requests for one resource are the same,
- ▶ independant from the origin of the request,
- ▶ app, single page application, server, ...

Client-server decoupling.

- ▶ Application on the server does know nothing about the client.

REST Principles (II)

State-less

- ▶ Any request is self-contained. Does not need to refer to a previous interaction.
- ▶ It contains everything needed to treat the request

Caching

- ▶ Caching must be possible,
- ▶ Response contains information about caching (if possible, how long, what, ...)

REST Principles (III)

Layered architecture

- ▶ In REST APIs, calls and responses pass through different layers.
- ▶ Don't assume that client and server applications connect directly to each other.

On demand code (optional)

- ▶ REST APIs usually send static resources,
- ▶ They can also send executable code (JavaScript for instance).

Anatomy of a RESTful API (I)

Resource-Based Architecture

- ▶ RESTful APIs are centered around resources, which represent entities or data objects.
- ▶ Resources are identified by unique URIs (Uniform Resource Identifiers) and can be accessed via standard HTTP methods.
- ▶ Each resource can have multiple representations (e.g., JSON, XML) and can be manipulated using CRUD operations (Create, Read, Update, Delete).

Anatomy of a RESTful API (II)

HTTP Methods

- ▶ HTTP methods (also known as verbs) define the actions that can be performed on resources in a RESTful API.
- ▶ The most commonly used HTTP methods in RESTful APIs are:
 - ▶ GET: Retrieve a resource or collection of resources.
 - ▶ POST: Create a new resource.
 - ▶ PUT: Update an existing resource.
 - ▶ PATCH: Partially update an existing resource.
 - ▶ DELETE: Delete a resource.
- ▶ Additional methods such as HEAD, and OPTIONS may also be used for specific purposes.

Anatomy of a RESTful API (III)

Uniform Interface

- ▶ A key principle of RESTful architecture is the uniform interface, which promotes simplicity, consistency, and scalability.
- ▶ The uniform interface is achieved through standardization of resource URIs, HTTP methods, and representations (e.g., JSON, XML).
- ▶ This standardization allows clients and servers to communicate effectively without the need for custom interfaces or protocols.

Anatomy of a RESTful API (IV)

Stateless Communication

- ▶ RESTful APIs are stateless, meaning that each request from the client to the server must contain all the information necessary to understand and process the request.
- ▶ The server does not store any client state between requests, which improves scalability, reliability, and performance.
- ▶ Stateless communication allows RESTful APIs to be highly cacheable and resilient to failures and network interruptions.

Anatomy of a RESTful API (V)

Layered System

- ▶ RESTful APIs are designed as layered systems, allowing intermediaries such as proxies, gateways, and caches to be inserted between clients and servers.
- ▶ Intermediaries can improve scalability, reliability, security, and performance by handling requests and responses at different layers of the architecture.
- ▶ The layered system architecture enables RESTful APIs to be flexible, extensible, and adaptable to changing requirements and environments.

JavaScript Object Notation - JSON

Designed to generate JavaScript objects and lists

- ▶ Lists: sequences of elements
- ▶ Objects: mappings key - value

Transfer format

- ▶ Between an App and a server
- ▶ Between a Web Client side application (JavaScript in the browser) and a server
- ▶ Between two servers

JSON syntax (I)

Strings are enclosed in double quotes

- ▶ "Hello world"

Lists are represented with square brackets

- ▶ ["A", "B", "C", "D"]

Objects are represented with curly brackets

- ▶ {}

JSON syntax (II)

Objects contain key: value pairs

- ▶ `{"color": "red", "taste": "sweet", "name": "orange"}`

Objects and lists can be mixed

Example 1 JSON ¹

```
{  
  "name": "Kudos",  
  "currency": "KUDOS",  
  "num_fractional_input_digits": 2,  
  "num_fractional_normal_digits": 2,  
  "num_fractional_trailing_zero_digits": 2  
}
```

¹Source: <https://exchange.demo.taler.net/keys>

Example 2 JSON ²

```
[
  {
    "stamp_expire": {
      "t_s": 1732808492
    },
    "stamp_end": {
      "t_s": 1795880492
    },
    "master_sig": "71W0CGF543A9... 6 ECTCWEXCXY3JGP10",
    "key": "CN8SCGF6Z4D2M549DNNM... 5 HYAY7ZP9Z1SP0QFT5JCNK0"
  },
  {
    "stamp_expire": {
      "t_s": 1718293892
    },
    "stamp_end": {
      "t_s": 1781365892
    },
    "master_sig": "1T8XFH64HC2SK2... VDQCDN9D9MBVJ00",
    "key": "397M3FFX9F41P4TY... 0 1WY804QXPW9W3W6BB6G"
  }
]
```

Parsing JSON (I)

PHP function `json_decode`³

```
json_decode(  
    string $json,  
    ?bool $associative = null,  
    int $depth = 512,  
    int $flags = 0  
): mixed
```

³Source: <https://www.php.net/manual/en/function.json-decode.php>

Parsing JSON (II)

Example

```
<?php
$json = ' {"a":1,"b":2,"c":3} ';
var_dump(json_decode($json));
var_dump(json_decode($json, true));
?>
```

Parsing JSON (III)

Output:

```
object(stdClass)#1 (5) {  
    ["a"] => int(1)  
    ["b"] => int(2)  
    ["c"] => int(3)  
}  
array(5) {  
    ["a"] => int(1)  
    ["b"] => int(2)  
    ["c"] => int(3)  
}
```

Examples (I)

Access objects

```
<?php
$json = '{"foo-bar": 12345}';

$obj = json_decode($json);
print $obj->{'foo-bar'}; // 12345
?>
```


Examples (II)

Access an array

```
<?php
$json = '[12345, 5678, 9012]';

$arr = json_decode($json);
print $arr[0]; // 12345
?>
```

Principle of URI (I)

URI = Uniform Resource Identifier

- ▶ Identify uniquely a resource
- ▶ Can be a URL (Uniform Resource *Locator*),
- ▶ or a URN (Uniform Resource *Name*).

Principle of URI (II)

Examples

- ▶ URL `https://docs.taler.net/index.html`
`ftp://ftp.is.co.za/rfc/rfc1808.txt` Location of the RFC 1808 (defining the FTP protocol).
- ▶ URN `https://docs.taler.net/taler-merchant-pos-terminal.html#apis-and-data-formats`
`urn:ietf:rfc:2396` is the identifier for the RFC 2396 (defining the URIs).

URI Schema

Syntax of a URI

`scheme:[//authority]path[?query][#fragment]`

For `https://docs.taler.net/taler-merchant-pos-terminal.html#apis-and-data-formats`

- ▶ scheme = `https`
- ▶ authority = `docs.taler.net`
- ▶ path = `taler-merchant-pos-terminal.html`
- ▶ we do not have any query
- ▶ fragment = `apis-and-data-formats`

URIs for REST APIs

Accessing a resource

- ▶ A resource is one single element
- ▶ `https://merchant.taler.net/products/1234`
- ▶ Is the locator of the product 1234

Access a collection

- ▶ A set of resources
- ▶ `https://merchant.taler.net/products/`
- ▶ Is the locator of the set of all products

Access a Resource using PHP

Simple case: no authentication / authorization

```
<?php
    $url = 'https://exchange.demo.taler.net/keys';
    $response = file_get_contents($url);
?>
```

Use a CURL object to generate a Request (I)

Example

```
<?php
    $url = 'https://example.com/api';
    $curl = curl_init($url);
    curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
    $response = curl_exec($curl);
    curl_close($curl);
?>
```

Use a CURL object to generate a Request (II)

Details

- ▶ Set options: we use the option `CURLOPT_RETURNTRANSFER` to download the returned string instead of printing it out
- ▶ We can configure any header or method for this request.

Example:

Print out the keys of a server.

```
<?php
$url = 'https://exchange.demo.taler.net/keys';
$curl = curl_init($url);
curl_setopt($curl, CURLOPT_RETURNTRANSFER, true);
$response = curl_exec($curl);
$keys = json_decode($response, true);
curl_close($curl);
var_dump(json_decode($keys));
?>
```

Direct Connection to a REST API

Wget

- ▶ to download content of a resource / collection

```
wget https://exchange.demo.taler.net/keys
```

CURL

- ▶ to read / write data in a resource or a collection
- ▶ headers and body can be manipulated.

```
curl https://exchange.demo.taler.net/keys
```

Requests using CURL (I)

GET request

```
curl --request GET \  
  --url https://backend.demo.taler.net/private/orders \  
  --header 'Authorization: Bearer secret-token:sandbox' \  
  --header 'User-Agent: insomnia/8.6.1 '
```

Requests using CURL (II)

POST Request

```
curl --request POST \  
  --url https://backend.demo.taler.net/private/orders \  
  --header 'Authorization: Bearer secret-token:sandbox' \  
  --header 'Content-Type: application/json' \  
  --header 'User-Agent: insomnia/8.6.1' \  
  --data '{ "order" :{  
            "amount": "KUDOS:3",  
            "summary": "purchase for testing",  
            "fullfilment_message": "Thank you for your order"  
          },  
  "create_token": false }'
```

Using a Software (I)

Insomnia⁴ is a software to generate REST requests

- ▶ Can send request to a server,
- ▶ can also generate the corresponding code in different languages.

Can be configure to send complex requests

- ▶ using all HTTP methods,
- ▶ with or without body (depending on method),
- ▶ different authentication methods.

⁴More info: <https://docs.insomnia.rest/>

Using a Software (II)

The screenshot displays the Insomnia application window. The main interface shows a REST client configuration for a GET request to the endpoint `https://backend.demo.taler.net/private/orders`. The request is configured with a Bearer token authentication, where the token is `secret-token:sandbox`. The response status is `200 OK`, with a response time of `58.5 ms` and a body size of `4.9 KB`. The response body is shown in a JSON format in the preview pane:

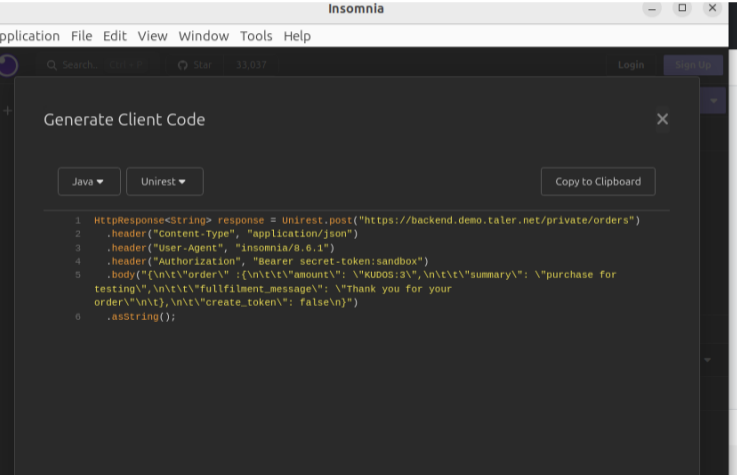
```
1 {  
2   "orders": [  
3     {  
4       "order_id": "2024.109-03R5P5X7Z2PRP",  
5       "row_id": 33650,  
6     }  
7   ]  
8 }
```

Using a Software (III)

Can generate code in many languages

- ▶ From C to JavaScript including Java or Shell
- ▶ Includes variants for many frameworks

Using a Software (IV)



Insomnia

Application File Edit View Window Tools Help

Search... Star 33,037 Login Sign Up

Generate Client Code

Java Unirest Copy to Clipboard

```
1  HttpResponse<String> response = Unirest.post("https://backend.demo.taler.net/private/orders")
2  .header("Content-Type", "application/json")
3  .header("User-Agent", "insomnia/8.6.1")
4  .header("Authorization", "Bearer secret-token:sandbox")
5  .body("{\"order\": {\"amount\": \"KUDOS:3\", \"summary\": \"purchase for
testing\", \"fulfillment_message\": \"Thank you for your
order\", \"create_token\": false}}")
6  .asString();
```


Conclusion

RESTfull architecture allows reuse of software

- ▶ Same piece of software can be used many times
- ▶ from an app, from a single page application,
- ▶ or from another software!

Lightweight development and deployment

- ▶ Install a web server,
- ▶ define your endpoints,
- ▶ develop the methods you want to implement.
- ▶ Straightforward to integrate from the client point of view.

References

Taler Demo Merchant Backend

<https://backend.demo.taler.net/>

Insomnia

<https://www.insomnia.rest/>

Bibliography

-  R. T. Fielding and R. N. Taylor.
Architectural styles and the design of network-based software architectures, volume 7.
University of California, Irvine Doctoral dissertation, 2000.

Acknowledgements

Funded by the European Union (Project 101135475).



**Co-funded by
the European Union**

Funded by SERI (HEU-Projekt 101135475-TALER).

Project funded by



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs,
Education and Research EAER
**State Secretariat for Education,
Research and Innovation SERI**

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union.
Neither the European Union nor the granting authority can be held responsible for them.

01-Introduction : Exercises

E. Benoist

August 29, 2024

1 Install the GNU Taler Merchant API

For this exercise, we use the GNU Taler Merchant backend API demo server.

The server has the following URL : <https://backend.demo.taler.net/>

The prices on this server should be provided in KUDOS which is an imaginary currency that is only used for GNU Taler demonstration servers.

Alternatively, you can install your own instance of this server. Instructions can be found at the following address : <https://docs.taler.net/taler-merchant-manual.html#installation>

2 Send GET Requests

For all the following examples, you will need to use a secure token. If you are using the demo Taler Merchant backoffice, you should use `secret-token:sandbox` as a Bearer token.

2.1 Insomnia

Install and start insomnia

```
# sudo apt install insomnia
# insomnia &
```

If you use only the local version of Insomnia, you can use this software for free.

- Method : GET
- Endpoint URL : <https://backend.demo.taler.net/private/products>
- Authentication Type : Bearer token with value `secret-token:sandbox` (need to be adapted if you are using your own installation of the GNU Taler Merchant backend).
- You do not need to specify a specific “Accept” header, since this server uses per default JSON.

Download all the elements of items in a big JSON file.

2.2 cURL

Install cURL.

```
# sudo apt install curl
```

Use cURL to generate a GET request downloading a JSON with all the products.

In Insomnia, you have a specific option to generate this command. Instead of clicking on the Send button, select the arrow. Then **Generate client code**, and then **Shell** and you terminate by selecting **cURL**

2.3 Add a product to the inventory

In the web interface (<https://backend.demo.taler.net/>), insert a new product in the inventory.

Use cURL to fetch the JSON list of all the products. The new product should be in the list.

If you installed your own instance of the merchant locally, connect to the SQL server and verifies that the new product is stored in the database.

The documentation of this endpoint can be found at the following address.

```
https://docs.taler.net/core/api-merchant.html#adding-products-to-the-inventory
```

2.4 GET for a specific resource

Generate a request to access the product itself. You need to combine the endpoint for viewing the products and the identifier received in the list of all products.

3 Programming

This part of the exercise can be done in any language : Java, PHP, Python, etc.

3.1 Read data

Write a program that connects to the Taler Merchant backoffice. For this exercise, we switch to the endpoint for the orders. The documentation for the orders endpoint can be found at the following address:

```
https://docs.taler.net/core/api-merchant.html#creating-orders
```

Your programm will print out the list of all the orders. For each of the orders, send another GET request to show the details of the order.

NEXT GENERATION INTERNET

REST API - Principles

Emmanuel Benoist

29.08.2024

Rest Principles

Architectural Constraints

URIs and Methods

GET

POST

PUT

PATCH

DELETE

Example of use

Conclusion

Architectural Constraints (I)

Here are the 6 architectural constraints seen in our previous course.

Client-server architecture

- ▶ User interface is separated from back-end.

State-less protocol

- ▶ The server must not store data corresponding to the session of the client.

Cacheability

- ▶ Clients can cache response values.
- ▶ Values must be marked as cacheable or non cacheable.
- ▶ Saves a lot of bandwidth and time.

Architectural Constraints (II)

Code on demand (optional)

Layered system

- ▶ For scalability reason, one could use proxy or a load balancer,
- ▶ without changing the code of the client or the server.
- ▶ Servers can send executable code (scripts for instance) to the clients.

Uniform interface

- ▶ Makes the architecture interoperable,
- ▶ Any client can connect any server.

Uniform interface (I)

Resource identification in requests

- ▶ Each resource is identified in the requests,
- ▶ for instance using a URI

Resource manipulation through representations

- ▶ Client has enough information in the representation to manipulate the resource (CRUD).

Self-descriptive messages

- ▶ Message contains enough information to be interpreted;
- ▶ for instance MIME-Type for choosing how to display the resource.

Uniform interface (II)

Hypermedia as the engine of application state (HATEOAS)

- ▶ Once a resource is downloaded, it contains hyperlinks to related resources;
- ▶ no need to hard code this in the client code.

RESTful Web Services

A REST API is defined with:

A Base URI

- ▶ For instance `https://bitcoin.ice.bfh.ch/keys/`
- ▶ called end-point or entry-point.

HTTP methods

- ▶ GET, POST, PUT, PATCH and DELETE

Encoding of the resource

- ▶ To know how to change data state.
- ▶ Examples: Atom, microformats, `application/vnd.collection+json`, etc.

URL for accessing to data

Collection corresponds to a URI that are pointing to a endpoint directly

- ▶ For instance `https://backend.demo.taler.net/private/orders;`
- ▶ refers to a collection of items.

Access directly to one member

- ▶ URL contains the address of the endpoint plus the identifier of the element.
- ▶ Example: `https://backend.demo.taler.net/private/orders/ABB2GG33453`

Method GET to access information (I)

Method GET for a collection

- ▶ Gets the URI of the resources in the collection,
- ▶ returns a list of the resources.

Method GET for a member resource

- ▶ Contains the identifier of the resource in the URL.

Method GET to access information (II)

Access to the representation of the resource

- ▶ Depending on the `Accept` header of the request,
- ▶ Can generate different formats for the same resource (JSON / XML / CSV / HTML / ...).

GET APIs should be idempotent

- ▶ Making multiple identical requests must produce the same result every time
- ▶ until another API-call (POST, PUT, PATCH) has changed the state of the resource on the server.

Method GET to access information(III)

Returned Resource

- ▶ If the Request-URI refers to a data-producing process,
- ▶ it is the produced data that shall be returned as the entity in the response,
- ▶ and not the source text of the process,
- ▶ unless that text happens to be the output of the process.

GET examples: Collection (I)

Read a collection

```
GET /private/orders HTTP/1.1
```

```
User-Agent: yourApp/0.0.1
```

```
Authorization: Bearer secret-token:sandbox
```

```
Host: backend.demo.taler.net
```

This will download the list of all the elements in the collection.

Authorization token

For all the following examples, we will use this http header for the authorization on the server (more details in the next course):

```
Authorization: Bearer secret-token:sandbox
```

GET examples: Collection (II)

JSON returned

```
{  "orders": [
  {
    "order_id": "2024.113-03R7NSPTFOYQ6",
    "row_id": 34830,
    "timestamp": {
      "t_s": 1713799282
    },
    "amount": "KUDOS:3",
    "summary": "payment after refund", ...
  },
  {
    "order_id": "2024.113-02G8P0KQT52R2",
    "row_id": 34829,
    "timestamp": {
      "t_s": 1713799273
    },
    "amount": "KUDOS:7",
    "summary": "order that will be refunded", ...
  },
  ...
]
```

GET examples: Element (I)

Read a collection

GET /private/orders/2024.113-03R7NSPTFOYQ6 HTTP/1.1

User-Agent: yourApp/0.0.1

Authorization: Bearer secret-token:sandbox

Host: backend.demo.taler.net

This will download the details concerning the specific instance of the collection.

GET examples: Element (II)

JSON returned

```
{
  "wire_reports": [],
  "exchange_code": 0,
  "exchange_http_status": 0,
  "exchange_ec": 0,
  "exchange_hc": 0,
  "deposit_total": "KUDOS:0",
  "contract_terms": {
    "summary": "payment after refund",
    "fulfillment_url": "taler://fulfillment-success/thx",
    "minimum_age": 0,
    "products": [],
    "h_wire": "WYEMPXPTA7....BFTJQ660GS1TG",
    "wire_method": "iban",
    "order_id": "2024.111-03429C3QC89DG",
    "timestamp": {
      "t_s": 1713576293
    }
  },
  ...
}
```

POST request (I)

Creates a new resource

- ▶ Creation uses the instructions in the body of the request

URI of resource automatically generated

- ▶ URI is returned in the `location` header of response,
- ▶ Can also be part of the body of the response.

POST is not idempotent

- ▶ Running POST a second time with the same arguments, do not expect the result to be consistent.

POST request (II)

Do not use a POST on a URL of an element

- ▶ It must create a new resource, and should not modify an existing one.
- ▶ Use PATCH or PUT instead.

POST to add a product (HTTP)

```
POST /private/products HTTP/1.1
Content-Type: application/json
User-Agent: yourApp/0.0.1
Authorization: Bearer secret-token:sandbox
Host: backend.demo.taler.net
Content-Length: 125
```

```
{
  product_id: "0001-1";
  description: "REST course";
  unit: "File";
  price: "KUDOS:2.0";
  total_stock: -1;
}
```


POST to add an order (Java)

```
HttpRequest request = HttpRequest.newBuilder()  
    .uri(URI.create("https://backend.demo.taler.net/private/orders"))  
    .header("Content-Type", "application/json")  
    .header("User-Agent", "yourApp/0.0.1")  
    .header("Authorization", "Bearer secret-token:sandbox")  
    .method("POST", HttpRequest.BodyPublishers.ofString("{\n\t\"order\  
    :{\n\t\t\"amount\": \"KUDOS:3\", \n\t\t\"summary\": \"purchase for  
    testing\", \n\t\t\"fullfilment_message\": \"Thank you for your  
    order\"\n\t}, \n\t\"create_token\": false\n}"))  
    .build();  
HttpResponse<String> response = HttpClient.newHttpClient().send(request,  
    HttpResponse.BodyHandlers.ofString());  
System.out.println(response.body());
```

PUT request on a Collection

Replaces ALL the instances of the collection

- ▶ Takes the data in the body to replace all the elements.

or create a new collection

- ▶ If the collection is empty;
- ▶ If it does not exist yet;
- ▶ uses the data in the body to initialize the collection.

PUT request on an instance

Replaces ALL the values of an element

- ▶ Takes the data in the body to replace the element.

or create a new element

PUT vs PATCH

- ▶ you CAN use PUT to *CREATE* or *REPLACE* an element, the key point is that you are basically expected to upload the new resource "verbatim" to its final destination,
- ▶ while with PATCH you express that you want to just modify it and could provide a delta (change set).

PATCH request (I)

Partial update on a resource

- ▶ PUT is used to modify all the entity
- ▶ PATCH will just make a delta between the existing resource and the new value.
- ▶ PATCH will not errase the rest of the element that is not contained in the body of the request.

PATCH request (II)

Suppose we have the following product at this URL

<https://backend.demo.taler.net/private/products/0001-1>

Element

```
{
  "description": "REST course",
  "description_i18n": {},
  "unit": "File",
  "price": "KUDOS:2",
  "image": "",
  "taxes": [],
  "total_stock": -1,
  "total_sold": 0,
  "total_lost": 0,
  "address": {},
  "minimum_age": 0
}
```

PATCH request (III)

We patch the product

```
PATCH /private/products/0001-1 HTTP/1.1
Content-Type: application/json
User-Agent: yourApp/0.0.1
Authorization: Bearer secret-token:sandbox
Host: backend.demo.taler.net
Content-Length: 101
```

```
{
  "description": "REST course in PDF",
  "unit": "File",
  "price": "KUDOS:2.5 ",
  "total_stock": -1
}
```

PATCH request (IV)

The patched product is changed

```
{  
  "description": "REST course in PDF",  
  "description_i18n": {},  
  "unit": "File",  
  "price": "KUDOS:2.5",  
  "image": "",  
  "taxes": [],  
  "total_stock": -1,  
  ...  
}
```

PATCH request (V)

Update all the instances of the collection

- ▶ According to values given in the body of the request;
- ▶ updates the existing instances in the collection.

Create the collection if it does not exist

Like PUT, PATCH is not intended primarily for collections

- ▶ One should avoid PUT and PATCH on collections.

Method DELETE (I)

Deletes the resource (if a resource)

- ▶ If you DELETE a resource,
- ▶ it's removed from the collection of resources.

Delete a collection

- ▶ Erases all the resources contained in the collection.

Method DELETE (II)

Return codes

- ▶ Return codes 200 (OK): resource has been removed and response contains an entity
- ▶ 202 (Accepted): deletion has been queued
- ▶ 204 (No Content): resource removed, response does not contain any body.
- ▶ 404 (Not found): resource has already been removed.

DELETE request

We patch the product

```
DELETE /private/products/0001-1 HTTP/1.1
Content-Type: application/json
User-Agent: yourApp/0.0.1
Authorization: Bearer secret-token:sandbox
Host: backend.demo.taler.net
Content-Length: 101
```

```
{
  "description": "REST course in PDF",
  "unit": "File",
  "price": "KUDOS:2.5 ",
  "total_stock": -1
}
```

Example of use

Create a new product

- ▶ List the existing products
- ▶ Send a POST request for generating a new product

Verify the status of the product

- ▶ View the list of all products
- ▶ View the details of our product
- ▶ Change the value of our product

Delete the generated product

List of all existing products (I)

GET request on the following resource:

`https://backend.demo.taler.net/private/products/`

```
curl --request GET \  
  --url https://backend.demo.taler.net/private/products \  
  --header 'Authorization: Bearer secret-token:sandbox' \  
  --header 'User-Agent: yourApp/0.0.1'
```

List of all existing products (II)

Response body contains the following JSON

`https://backend.demo.taler.net/private/products/`

HTTP/1.1

server: nginx

date: Tue, 25 Jun 2024 12:57:50 GMT

content-type: application/json

content-length: 158

vary: Accept-Encoding

access-control-allow-origin: *

access-control-expose-headers: *

strict-transport-security: ~~max~~-age=63072000; includeSubDomains; preload

```
{ "products": [  
  { "product_serial": 1, "product_id": "asas" },  
  { "product_serial": 2, "product_id": "0001-1" }  
]
```

Add a new product (I)

POST request to the same resource

The body contains the definition of the new product.

```
curl --request POST \  
  --url https://backend.demo.taler.net/private/products \  
  --header 'Authorization: Bearer secret-token:sandbox' \  
  --header 'Content-Type: application/json' \  
  --header 'User-Agent: yourApp/0.0.1' \  
  --data '{"address":{},"description_i18n":{} ,  
        "taxes":[] ,"next_restock":{"t_s":"never"} ,  
        "price":"KUDOS:5" ,"product_id":"0001-2" ,  
        "description":"Cryptography course" ,  
        "unit":"piece" ,"total_stock":-1}'
```

Add a new product (II)

Response

204 No Content

Read the new inserted product

GET request on the following resource:

```
https://backend.demo.taler.net/private/products/0001-2
```

```
curl --request GET \  
  --url https://backend.demo.taler.net/private/products/0001-2 \  
  --header 'Authorization: Bearer secret-token:sandbox' \  
  --header 'User-Agent: yourApp/0.0.1'
```

Shows the content of the new product

```
{  
  "description": "Cryptography course",  
  "description_i18n": {},  
  "unit": "piece",  
  "price": "KUDOS:5",  
  "image": "",  
  ...  
}
```

Change the price of the product (from 5 to 6 Kudos)

PATCH request on the product 0001-2 resource:

`https://backend.demo.taler.net/private/products/0001-2`

```
curl --request PATCH \  
  --url https://backend.demo.taler.net/private/products/0001-2 \  
  --header 'Authorization: Bearer secret-token:sandbox' \  
  --header 'Content-Type: application/json' \  
  --header 'User-Agent: insomnia/9.2.0' \  
  --data '{  
    "description": "Cryptography course",  
    "unit": "piece",  
    "price": "KUDOS:6",  
    "total_stock": -1 }'
```

Delete our product from the database

DELETE request on the product 0001-2 resource:

`https://backend.demo.taler.net/private/products/0001-2`

```
curl --request DELETE \  
  --url https://backend.demo.taler.net/private/products/0001-2 \  
  --header 'Authorization: Bearer secret-token:sandbox' \  
  --header 'User-Agent: myApp/0.0.1'
```

Response: 204 No Content

Means everything OK, the element has been deleted.

DELETE is not idempotent: repeating it will generate a 404 NOT FOUND error.

Conclusion

CRUD operations are backed by HTTP Methods

- ▶ Creation with POST
- ▶ Read with GET and HEAD
- ▶ Update with PUT (full update) and PATCH (partial update)
- ▶ Delete with DELETE
- ▶ return information is contained in the status code and the body of the response.

Authorizations

- ▶ Are possible in various means,

References (I)

Wikipedia

- ▶ <https://en.wikipedia.org/wiki/REST>

Taler Merchant Backend API description

- ▶ <https://docs.taler.net/core/api-merchant.html>

Taler Demo Merchant Backend

- ▶ <https://backend.demo.taler.net/>

REST API Tutorial

- ▶ <https://restfulapi.net/http-methods/>

References (II)

The 4 most used REST authentication methods

- ▶ <https://blog.restcase.com/4-most-used-rest-api-authentication-methods/>

The 5 essential HTTP methods in RESTful API development

- ▶ <https://www.techtarget.com/searchapparchitecture/tip/The-5-essential-HTTP-methods-in-RESTful-API-development>

RFC8959: The "secret-token" URI Scheme

- ▶ <https://datatracker.ietf.org/doc/html/rfc8959>

Acknowledgements

Funded by the European Union (Project 101135475).



**Co-funded by
the European Union**

Funded by SERI (HEU-Projekt 101135475-TALER).

Project funded by



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs,
Education and Research EAER
**State Secretariat for Education,
Research and Innovation SERI**

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union.
Neither the European Union nor the granting authority can be held responsible for them.

02-Operations : Exercises

E. Benoist

August 27, 2024

1 Installation for this exercise

As a preamble to this exercise, you need to install a Taler Wallet on your phone.

Download the GNU Taler wallet from the F-droid, Android play or the Apple store.

Go to the demo bank server : <https://bank.demo.taler.net/webui/#/login>. Create a bank account (click on Register, you need not register with your legal identity, this is a demo server accepting any identity).

The demo bank is running the KUDOS currency. This is a virtual fake currency that does not exist outside this demonstration.

Transfer 100 KUDOS from your bank account to your Taler-Wallet.

2 Insert Item

For this exercise, you contact your local Taler Merchant backend instance (or alternatively the server <https://backend.demo.taler.net/>) as explained in the previous exercise sheet [01-introduction-exercises.pdf](#).

2.1 Generate a POST request

Use the software *Insomnia* to generate the POST request used to insert a new Item inside the list of items.

The Endpoint is the same as for the GET used in the previous exercise. `/private/products`. The documentation for this endpoint is at the following URL.

<https://docs.taler.net/core/api-merchant.html#adding-products-to-the-inventory>)

The POST request must be configured :

- The method is POST
- The secure token is
- Body type is JSON (mime-type : `application/json`)

The body of the POST is an object containing at least the members: "product_id", "description", "unit", "price", and "total_stock". Since our demo servers are configured to work with KUDOS, price must be in KUDOS, for instance : "KUDOS:2.0", if you configured your server to work with a different currency you should use it.

2.2 Integrate the POST request in a program

Write a program to insert a new item in the database of items to sell. This

2.3 Integrate the POST request in a program

Write a program to insert a new item in the database of items to sell. This program will generate the same request as the previous exercise.

3 Modify item

Write a program that changes the price of your item.

4 Create a transaction

For this exercise, we use another endpoint. It is the `orders` endpoint (`/private/orders`). A transaction is generated by the merchants when they want clients to buy a product from the list of product. The merchant must create a new `PostOrder-Request` (using a POST request), it must at least contain an `order` object and a member `create_token` that we set to *false*. The `order` object contains at least an `amount` (the form is like "KUDOS:2.0", i.e. `currency:price`), a `summary` text and a `fullfilment_message` containing the text that will be displayed after the payment to the client.

You can find the details of all the optional fields in the API documentation :

<https://docs.taler.net/core/api-merchant.html#payment-processing>

As a response to your request, you get a unique order-ID in JSON.

As a response to your request, you get a unique order-ID in JSON.

Then you need to generate a payment URL to give to your client wallet. Your URL will have the form :

`https://backend.demo.taler.net/orders/ <your order ID>`

5 Pay the transaction

Visit the page with the URL you generated. You receive a QRCode.

Scan the QR Code inside your Wallet. Follow the instructions to pay the bill.

6 Terminate the transaction

From the point of view of the seller, verify that the payment occurred.

The status of an order can be checked on the orders endpoint.

Generate a GET request to the orders endpoint with the orderID. The URL should look like :

`https://backend.demo.taler.net/private/orders/2024.022-0202020WEW`

Change your program such that it checks the status returned. The system works with long polling. So the response may be delayed.

NEXT GENERATION INTERNET

REST API - Security

Emmanuel Benoist

29.08.2024

REST Security

API authorization methods

OAuth 2.0

OAuth for Taler Exchange

Deploy a Secure REST-API

- Secure Communication

- Validate inputs

- Log Errors

- Access Validation

API authorization methods

- ▶ Authorization or authentication?
- ▶ HTTP authentication schemes
- ▶ API Key
- ▶ OAuth

Authorization / Authentication (I)

Authentication vs Authorization

- ▶ Authentication = who you are?
- ▶ Authorization = what you can do?

On Web pages

- ▶ User authenticate once using a username / password;
- ▶ A session is started;
- ▶ A cookie is sent for remembering that session;
- ▶ Authorization is based on the right of the authenticated user.
- ▶ Session authentication using a session-cookie is not possible for REST (since stateless requirement)

Authorization / Authentication (II)

Stateless authorization / authentication

- ▶ User gets a token,
- ▶ Token is resent inside each request.

Different Tokens

- ▶ Basic Authentication
- ▶ Bearer Token
- ▶ API-Key

HTTP Authentication Schemes:

Basic authentication (I)

Basic Authentication

- ▶ the sender places a username:password into the request header (base64 encoded).
- ▶ It is rarely recommended due to its inherent security vulnerabilities.

Vulnerabilities of Basic Authentication

- ▶ Client must hold username and password.
- ▶ Credentials are transmitted clear text.
- ▶ Credentials can not easily be revoked.
- ▶ Replay attacks are easy to conduct.

HTTP Authentication Schemes: Basic authentication (II)

Here's an example of a Basic Auth in a request header:

Authorization: Basic ZW1tYW51ZWw6cGFzc3dvcnQ=

Correspond to the string `emmanuel:password`

Password authentication (I)

Problems with password authentication

- ▶ **Brute force:** attacker will try many passwords for one account;
- ▶ attacker may try many pairs (account, password) stolen from another service;
- ▶ need to limit the number of attempts in a given period;
- ▶ Assume that someone who is performing hundreds of failed input validations per second is up to no good.

Password authentication (II)

Store password in a secure way

- ▶ Storing password clear text is a bad idea (in case of a data leakage);
- ▶ Password must be hashed and salted;
- ▶ one should use a specifically designed hash function that is difficult to brute force:
argon2id, scrypt, bcrypt, or PBKDF2.
- ▶ In case hashed salted passwords are exposed, obtaining the passwords them will at least be expensive (in terms of memory and/or of CPU time).

HTTP Authentication Schemes: Bearer token (I)

Bearer token

- ▶ The token is sent in each request,
- ▶ Provided normally by the server as answer to an authentication (username / password for instance)

Here's an example of a Bearer Token in a request header:

```
Authorization: Bearer <token>
```

HTTP Authentication Schemes:

Bearer token (II)

Risk with bearer in the source code

- ▶ Token should not be hard coded in the source code
- ▶ Could be read by anybody accessing the code (`git clone` for instance)

Secret Token should be compliant with RFC8959

- ▶ <https://datatracker.ietf.org/doc/html/rfc8959>
- ▶ Idea: every secret token has prefix "secret-token:",
- ▶ Easy to detect and resolve hard-coded tokens!

HTTP Authentication Schemes: Bearer token (III)

Bearer token in Taler

- ▶ Taler REST APIs sometimes uses bearer tokens to authenticate clients.
- ▶ Token is configured globally for a service or per user of the service.
- ▶ Token can be copied into the client.

Example

POST /private/products HTTP/1.1

...

Authorization: Bearer secret-token:sandbox

Host: backend.demo.taler.net

...

API Key (I)

One unique key is given to one user

- ▶ Key is generated at the first visit of the user
- ▶ Same key will be reused in each request.

Example of a header containing an API key:

Authorization: Apikey 1234567890abcdef

API Key (II)

Key can be in various places

- ▶ Authorization header (c.f. hereover);
- ▶ basic auth (c.f. previous slides);
- ▶ body data;
Contains for instance:

```
{"api-key" : "1234-567890-1234"}
```

- ▶ Custom header;
- ▶ Query string (Has a lot of security problems, should be avoided).

Problems with token in the query string (I)

Query string can be logged and cached

- ▶ Every proxy or reverse proxy can store the values of GET requests (and hence the URL)
- ▶ History and cache of the browser will store the URL

Anybody that can read those files can access to all the query strings

- ▶ If the query string does contain the token,
- ▶ they can access this token.

Problems with token in the query string (II)

URL can also be transferred to third parties

- ▶ For instance inside the URL referer header of a request.
- ▶ System analyzing log files of the third party will have access to your system.

JWT: JSON Web Tokens (I)

Standard for the exchange of secure tokens

- ▶ Defined in the RFC7519
- ▶ <https://datatracker.ietf.org/doc/html/rfc7519>

Structure

- ▶ Header: describes the token (a JSON object);
- ▶ payload: The informations stored inside this object;
- ▶ digital signature (for security purpose).

JWT: JSON Web Tokens (II)

Example

Header

```
{"typ": "jwt", "alg": "HS512"}
```

Payload

```
{"name": "Benoist", "iat": 121345543298}
```

JWT: JSON Web Tokens (III)

Encoding

- ▶ We encode header and payload using Base64url,
- ▶ concatenate the two strings (adding a "." between the two items),
- ▶ hash the concatenation plus your secret key (and encode in Base64url),
- ▶ concatenate the message and the hash into your token.

Verification

- ▶ Decode the first part of the message,
- ▶ hash it together with your secret key,
- ▶ verify the hash!

JWT: JSON Web Tokens (IV)

Security issue: Replay attacks

- ▶ JWT must contain a validity limit (not mandatory),
- ▶ If a session is terminated (i.e. logout): Token must be added to a revocation list.

Advantages of JWT

- ▶ Session less,
- ▶ easy to revoke,
- ▶ can be fine tuned (possible to separate authentication from authorization),
- ▶ if well configured, no risk of replay attacks.

OAuth2.0

OAuth2.0 is a protocol for delegating authorizations

- ▶ Your client requires access to a resource owned by someone else
- ▶ The resource owner logs into resource provider
- ▶ They selects which resource you are to be given access to
- ▶ You receive a token for accessing the resource
- ▶ Token can be used to talk to the REST API

Owner delegates authorization to the bearer of the token

- ▶ Token means: *“the holder of the token is authorized to access (or modify) this set of resources.”*

History: Password sharing anti-pattern

A user wants to delegate access to service to the App

- ▶ User gives username and password to the app
- ▶ The app logs on the resource server using user's credentials
- ▶ The app acts on behalf of the user.

Problems (some of them at least)

- ▶ Risk of leakage (hacking or rogue administrators)
- ▶ Not possible to repudiate some delegations
- ▶ Not limited in time
- ▶ Not limited in depth (impossible to share only **some** elements)

OAuth (2.0)

The actors:

- ▶ *Resource owner* (RO) the user that will delegate the access;
- ▶ *Client Application* (Client) that wants to consume a resource;
- ▶ *Authorization server* (AS) that makes the authentication and delivers the tokens;
- ▶ *Resource server* (RS) that delivers the service and consumes the tokens.

Sharing a resource

Applied to REST APIs

- ▶ Client wants to access a resource on the RS
- ▶ App requests to the RO to authenticate by the AS via a redirect (typically in a browser).
- ▶ Once RO is authenticated, AS asks if RO grants the requested authentication. If yes, the AS generates an **Authorization Token**.
- ▶ Authorization Token is transferred by the RO (or its browser) to the Client
- ▶ The Client uses this Authorization Token to generate an **Access Token** on the AS
- ▶ This Access Token is used to access resources on the RS

OAuth is for access delegation

It is not intended for:

- ▶ Traditional access control
- ▶ Authentication
- ▶ Federated authority

Idea: Get only necessary authorization (I)

The user wants to delegate some access

- ▶ RO uses the Client app or visits the site RS
- ▶ RO is asked for delegation of rights on resource RS.
- ▶ RO is redirected to AS for authorizing Client to access resource on a given scope (is a subset of the resources RO owns on RS).

RO must get a token for accessing RS

- ▶ Is redirected to AS where it logs in (using their credentials).
- ▶ RO validates the scope of access granted to Client.
- ▶ Receives an authorization token

Idea: Get only necessary authorization (II)

Authorization code is transferred to Client

- ▶ Client receives the authorization token.
- ▶ Client uses the authorization token to authenticate to AS.
- ▶ Client asks for the access token.
- ▶ Client uses the access token to use resources on RS.

Why two tokens? (I)

Access Token and Refresh Token

- ▶ Access Token is provided by AS to the client, that uses it for accessing a resource on the RS.
- ▶ Refresh Token can be provided by the AS to get another Access Token (e.g. when it has expired).

Tokens are flexible

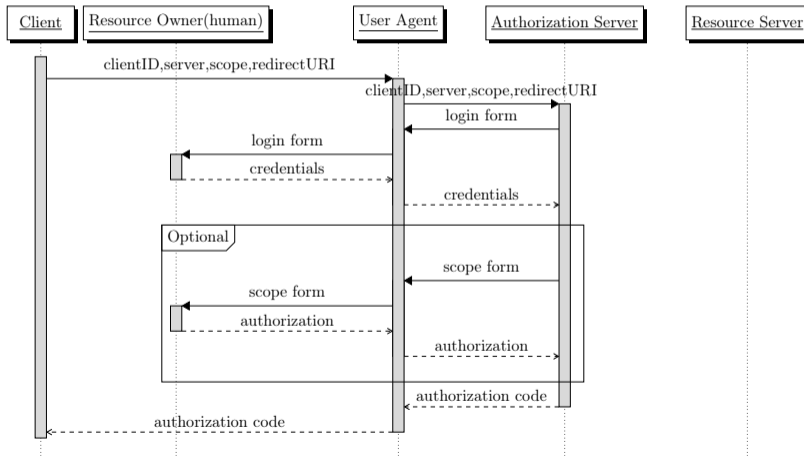
- ▶ Access or refresh tokens can have a limited scope (only access one item or one property of the user).

Why two tokens? (II)

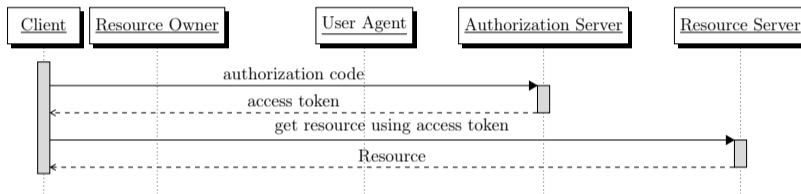
Access token should have a short duration

- ▶ Access token is used to authorize all the requests.
- ▶ No need of a revocation mechanism, since it has a short validity period.
- ▶ Needs to be refreshed regularly. Revocation is done on the refresh token.

OAuth Flow (I)



OAuth Flow (II)



OAuth for KYC on Taler Exchange (I)

Taler Exchange

- ▶ Transforms wire-transfer into coins stored in Taler Wallet.
- ▶ Could be operated by a central bank (for Central Bank Digital Currency).
- ▶ Receives FIAT money, delivers corresponding digital coins.

The KYC processus for Exchange

- ▶ Banks must fight money laundering and fraud.
- ▶ They need to know their customers: what is the identity of people having accounts?
- ▶ Procedure known as Know Your Customer (KYC).

OAuth for KYC on Taler Exchange (II)

First part of Anti-Money Laundering (AML) procedures

- ▶ Know Your Customer (verify the identity of your client).
- ▶ Compare with lists of sensitive persons (politically exposed or under an embargo).
- ▶ Set limits according to the law (per day/week/month/year).

OAuth Roles (I)

The User

- ▶ Wants to use a new exchange server for payments,
- ▶ Has a phone and an email,
- ▶ Will have to prove having access to a device.

The Challenger

- ▶ Tests if the user can read messages sent to a phone,
- ▶ or an email address.
- ▶ Validate that the user's phone number or mail address are valid.

OAuth Roles (II)

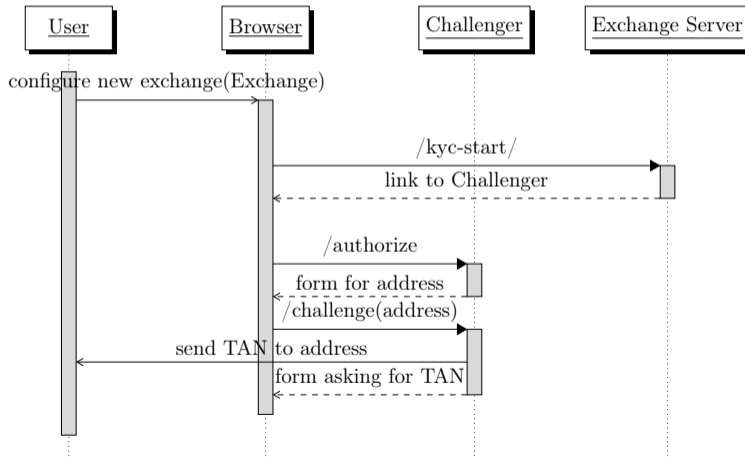
Exchange

- ▶ Transforms wire transfer (classical bank transfer) into digital money (coins).
- ▶ Is required by law to know their customers.

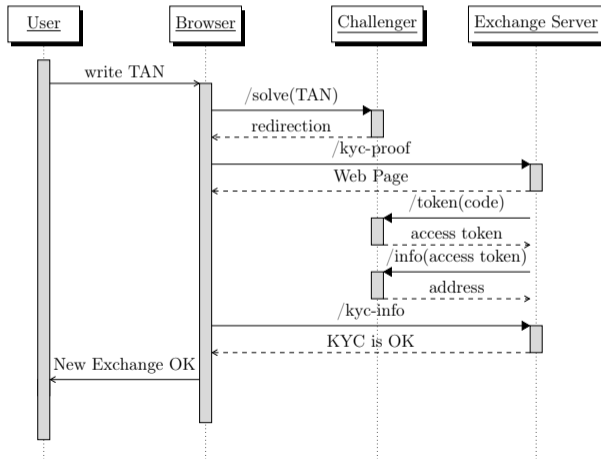
Wallet

- ▶ The software needs to access the Exchange server
- ▶ Owner needs to be known by the exchange to receive digital cash.

OAuth for Taler (I)



OAuth for Taler (II)



Deploy a secure REST-API

- ▶ Use HTTPS only
- ▶ Never expose any information in URL's
- ▶ Validate inputs
- ▶ Access validation

Use HTTPS only (I)

HTTP is not secure

- ▶ Is sent clear text (including credentials)
- ▶ Everybody can read everything
- ▶ Prone to Man in The Middle attacks

Encrypt connexions also inside your firm network

- ▶ LAN can be attacked from within,
- ▶ a computer/router can be compromised,
- ▶ an attacker could insert hardware inside your network.

Use HTTPS only (II)

HTTPS with client authentication useful but complex

- ▶ In HTTPS clients can be authenticated using a certificate;
- ▶ Can be expensive (depending on the type of certificate);
- ▶ Cumbersome to configure.
- ▶ Normally, only server authentication is necessary for security, user authentication will be done using other means.

Never expose information in URL's

URL could contain sensitive information

- ▶ Secure tokens
- ▶ Sensitive data (username or passwords)

URL can (and should) be stored in cache

- ▶ URL will appear in browser history and server logs
- ▶ If contains sensitive information: it makes it accessible

Manage HTTP

Deny methods by default

- ▶ Restrict to the used methods GET, POST, PUT for instance.
- ▶ Return 405 Method not allowed for all methods that you did not implement. You must list the allowed methods.

Use appropriate HTTP Return Codes

For instance:

- ▶ 401 Unauthorized: wrong or no authentication provided
- ▶ 403 Forbidden: authentication succeeded but user lacks access permission
- ▶ 404 Not Found if the resource does not exist.

Validate Inputs (I)

Do not trust input parameters or objects

- ▶ Validate input:
- ▶ length
- ▶ range
- ▶ format
- ▶ type (string, numbers, positive/negative, floats, integers, ...);

Achieve an implicit input validation by using strong types

- ▶ like number, booleans, dates, times
- ▶ or fixed data ranges in API parameters

Validate Inputs (II)

Example:

<https://docs.taler.net/core/api-taldir.html#tsref-type-VersionResponse>

```
interface VersionResponse {
    version: string;
    name: "taler-directory";
    methods: TaldirMethod[];
    monthly_fee: Amount;
}

interface TaldirMethod {
    name: string;
    challenge_fee: Amount;
}
```

Validate Inputs (II)

Inputs: Accept only known good

- ▶ Do not use black list of inputs
- ▶ Always use white lists
- ▶ Constrain text inputs with Regular Expressions
- ▶ Make use of validation libraries in your specific language.

Reject unexpected and illegal content

- ▶ Do not try to sanitize inputs,
- ▶ User friendly feedback is not the job of an API.

Validate Inputs (III)

Define a maximum input size limit

- ▶ If the request is too large: 413 Request Entity Too Large

Use a secure parser

- ▶ outdated or badly configured parsers are subjects to attacks that are well documented,
- ▶ for instance if working in XML, make sure not to be vulnerable to XML eXternal Entity attack (XXE) and similar attacks.

Log errors

Log all errors

- ▶ Password Errors
- ▶ HTTP method not allowed
- ▶ Access denied
- ▶ ...

Monitor your log files

- ▶ Regularly control errors files
- ▶ Automatically send alarms
- ▶ React automatically to repeated attempts (brute force or DoS)

Access Validation (I)

Each API endpoint must perform access control (if not public)

- ▶ Do not expect a endpoint to be known only by your applications,

Access decision must be taken locally by each endpoint

- ▶ For the function,
- ▶ for the list integrated in a collection
- ▶ for controlling the access to each of the elements
- ▶ the server must verify that the user has the right to access to any element they want to access (Create, Read, Update or Delete)

Access Validation (II)

No validation of the access rights is expected on the client side

- ▶ Client may validate the rights for its own purpose (usability);
- ▶ Server must validate rights for security;
- ▶ If client is rogue, no problem should occur.

Validation at two levels:

- ▶ First validate that user has access to the endpoint, (authenticated user, authorised to access this endpoint)
- ▶ then validate that user has access to the requested data (has been authorised to read/write/update that specific data).

Conclusion (I)

Authorization or authentication

- ▶ Delegation of authorization is often sufficient, sometime is authentication also necessary.
- ▶ Be carefull with authentication of a client that is not on your control (Single Page Applications for instance). Data will be available to third parties.

OAuth2.0

- ▶ Can be used for authorization delegation,
- ▶ with OpenID Connect it is used for authentication also.

Conclusion (II)

Validate Inputs / Implement all security requirements

- ▶ A REST API is open to the world,
- ▶ it can be attacked by anybody,
- ▶ do not expect that only your client will connect your server,
- ▶ it should be protected accordingly.

References (I)

OWASP REST Security: REST Security Cheat Sheet

- ▶ https://cheatsheetseries.owasp.org/cheatsheets/REST_Security_Cheat_Sheet.html

Taler Merchant back end tutorial

- ▶ <https://tutorials.taler.net/merchant/introduction>

OWASP Top 10 API Security Risks – 2023

- ▶ <https://owasp.org/API-Security/editions/2023/en/0x11-t10/>

Acknowledgements

Funded by the European Union (Project 101135475).



**Co-funded by
the European Union**

Funded by SERI (HEU-Projekt 101135475-TALER).

Project funded by



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs,
Education and Research EAER
**State Secretariat for Education,
Research and Innovation SERI**

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union.
Neither the European Union nor the granting authority can be held responsible for them.

03-Security : Exercises

E. Benoist

August 29, 2024

1 Install and configure the Taler Challenger

Follow the instructions given in the Challenger operator manual:

<https://docs.taler.net/taler-challenger-manual.html#>

You can install the software based on source code or using package managers for Debian, Trisquel and Ubuntu.

Use the email validation. For SMS validation, an account on a gateway is necessary and may generate some costs.

2 Write a web page

Write a simple web page for registering a new user (you create a new account using email as an identifier).

You should verify the email. This is where you use Challenger to validate the email used for the account.

You can find the documentation of the API here:

<https://docs.taler.net/core/api-challenger.html>

Blockchains

Introduction to Blockchains

Christian Grothoff

Bern University of Applied Sciences

31.05.2024

2024-08-26

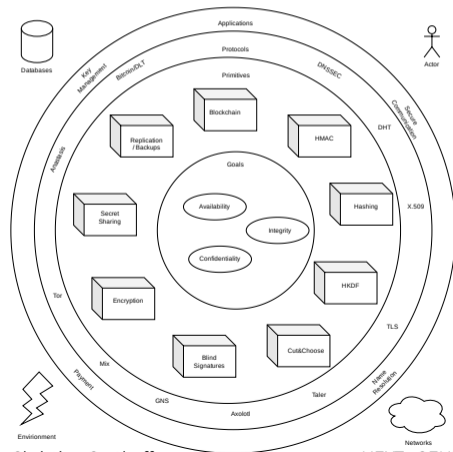
Blockchains

Blockchains
Introduction to Blockchains

Christian Grothoff
Bern University of Applied Sciences
31.05.2024

1. In this lecture, we will give you a first brief introduction to Blockchains.
2. Blockchains, like AI, are currently a hype topic in computing.
3. It is thus important to understand what they can or cannot do.

Blockchains on our map



Christian Grothoff

NEXT, GENERATION, INTERNET

2024-08-26

Blockchains

└─ Blockchains on our map



1. Blockchains are a construction used primarily to achieve availability and integrity.
2. Key applications include payment and name resolution.
3. Related topics are replication, backups, HMACs, hashing, and DHTs as an example for a different approach to distributed storage.

Learning Objectives

What is a Blockchain?

What properties are Blockchains claimed to have?

How does Proof-of-Work solve the Byzantine consensus problem?

Bitcoin and Payments: A good match?

What are other applications for Blockchains?

2024-08-26

Blockchains

Learning Objectives

1. One difficulty with Blockchains is that the term is not well defined. So we will begin with a high-level illustration of the concept, omitting details that do not universally apply.
2. Then we will look at what Blockchain proponents claim as the properties of Blockchains.
3. The consensus problem is central to all blockchains, so we will explore this in more detail.
4. Payments are seen as one big application domain for Blockchains, so we will see how suitable Blockchains are for this key application.
5. Finally, we will see what other applications may benefit from Blockchains.

What is a Blockchain?

What properties are Blockchains claimed to have?

How does Proof-of-Work solve the Byzantine consensus problem?

Bitcoin and Payments: A good match?

What are other applications for Blockchains?



2024-08-26

What is a Blockchain?

Blockchain¹



¹Illustrations by Alexandra Dirksen, IAS, TUBS [2]

1. We will begin our exploration with a simple transaction as a starting point: Bob wants to buy a phone from Alice.
2. We'll assume Bob already has some "money" in the system, let's not yet worry where it came from.

Blockchain



2024-08-26

Blockchains

└─ What is a Blockchain?

└─ Blockchain

1. To make the transaction “real”, Alice and Bob take a snapshot of their transaction data.
2. They then make it public by posting it on a public bulletin board.
3. Everyone in the world can then see that Alice sold her phone to Bob.



1. Next, let's assume Peter wants to buy a car from Charlie.
2. Now, in this case, it probably doesn't matter that this happens after Alice sold her phone to Bob.
3. But, sometimes the order of transactions matters.
4. Just imagine Alice buying the car from Peter with the money from Bob.



1. Next, let's assume Peter wants to buy a car from Charlie.
2. Now, in this case, it probably doesn't matter that this happens after Alice sold her phone to Bob.
3. But, sometimes the order of transactions matters.
4. Just imagine Alice buying the car from Peter with the money from Bob.

Blockchain



2024-08-26

Blockchains

└─ What is a Blockchain?

└─ Blockchain

1. Charlie and Peter can show that Charlie sells the car after Alice sold her phone by putting the snapshot of Alice and Bob's transaction into their background when producing evidence of their own transaction.
2. This both affirms Alice and Bob's transaction and establishes a transaction order.
3. Cryptographically, it is of course enough to put the hash of the original transaction into the new snapshot.

Blockchain

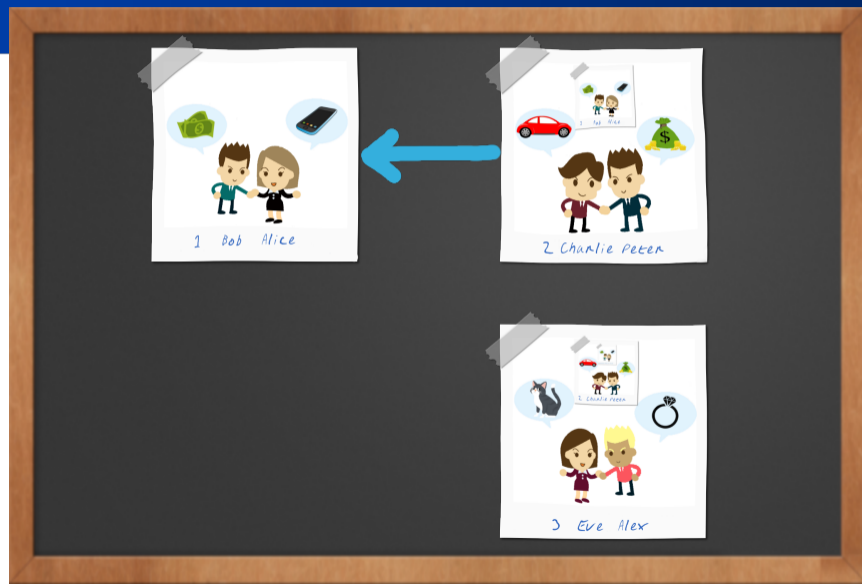


2024-08-26

Blockchains
└─ What is a Blockchain?
└─ Blockchain

1. As before, the snapshot of the new transaction is put up on the public bulletin board.

Blockchain



2024-08-26

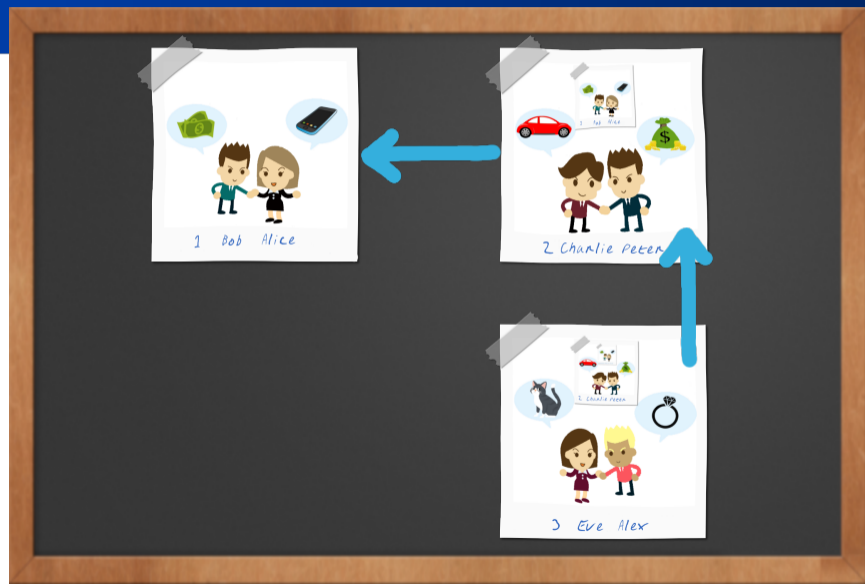
Blockchains

└─ What is a Blockchain?

└─ Blockchain

1. The hash of the previous transaction in the background “chains” the two transactions.
2. If somebody wanted to now alter the transaction between Alice and Bob, they would also have to alter the snapshot posted by Charlie and Peter to ensure consistency of the chain.

Blockchain



2024-08-26

Blockchains

└─ What is a Blockchain?

└─ Blockchain

1. Further transactions follow the same pattern.

Blockchain

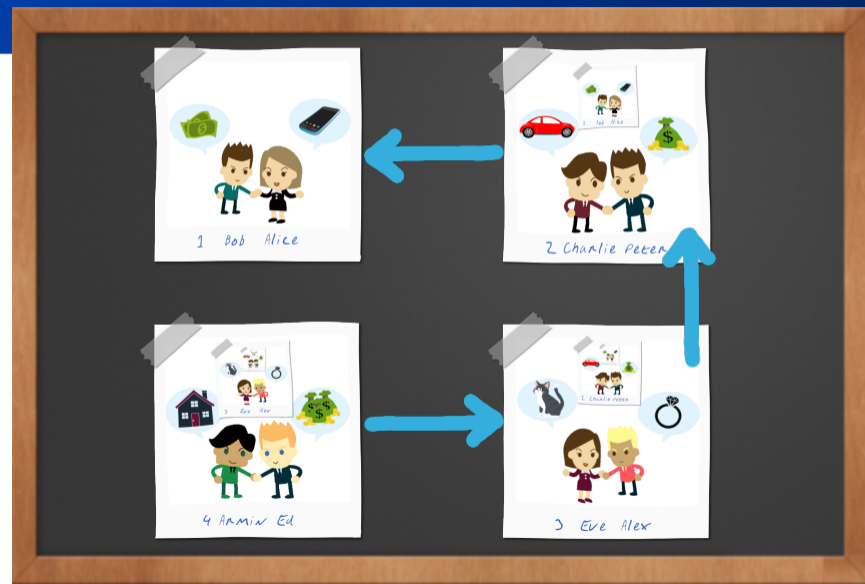


2024-08-26

Blockchains
└─ What is a Blockchain?
└─ Blockchain

1. Note that each new snapshot does not only affirm its immediate predecessor, but transitively all previous transactions.

Blockchain



2024-08-26

Blockchains

└─ What is a Blockchain?

└─ Blockchain

1. Due to the use of a hash, the actual size of each block (snapshot) is pretty constant: the reference to the previous block always has the same size, regardless of how long the chain has gotten.
2. However, to fully understand the balances of people involved, we do need the full chain, and not just the last snapshot.
3. So a Blockchain grows linearly as more transactions are added, so space consumption is a key concern.

What properties are Blockchains claimed to have?

Advertised Blockchain “properties”



2024-08-26

Blockchains

└ What properties are Blockchains claimed to have?

└ Advertised Blockchain “properties”

1. Now, let's talk about the properties Blockchains are frequently proclaimed to have.
2. I'm saying “proclaimed” here, as each of these properties kind-of holds.
3. And the kind-of is critical as the limitations are serious sources of problems.

Immutability



2024-08-26

Blockchains

└ What properties are Blockchains claimed to have?

└ Immutability

1. Immutability means that once a transaction has been posted, it is final and cannot be changed anymore.
2. The claim does not arise from transactions being posted in public, as there could be conflicting pictures posted in public.
3. Instead, the idea is that *old* transactions cannot be modified because one would need to update all of the newer transaction records as well.
4. So this is a cost-based argument: to modify the transaction between Charlie and Peter, we would need to re-do the work for subsequent transactions between Eve and Alex and Armin and Ed.
5. But, at least in principle, it is of course always possible to re-do that work.

Transparency



2024-08-26

Blockchains

└ What properties are Blockchains claimed to have?

└ Transparency

1. The transparency property is that everyone sees everything that is going on with the Blockchain.
2. This is because all transactions are public on a bulletin board.
3. Sure, honest participants in the peer-to-peer network will share their view of the Blockchain.
4. However, the Internet doesn't exactly have a bulletin board, and downloading large Blockchains (some have grown to many Terrabytes) can be prohibitive for most users.
5. Also, network outages can still prevent timely visibility.
6. Finally, malicious participants can hide new blocks.
7. There are attacks where hiding new blocks from competitors can yield economic benefits.

Decentralization



2024-08-26

Blockchains

└ What properties are Blockchains claimed to have?

└ Decentralization

1. The decentralization claim is that anyone can participate in the system on an equal footing.
2. There is no operator, the system operates as a *permissionless* peer-to-peer network.
3. In reality, few users can afford to download the entire Blockchain, and to effectively participate requires specialized resources.
4. Thus, very few entities end up dominating the process.
5. Finally, not all “Blockchains” are open decentralized peer-to-peer networks. Sometimes closed proprietary systems with a well-defined restricted set of participants are still called “Blockchains”. These are often called *permissioned* or *private* Blockchains; an well-known example is Hyperledger Fabric.

Autonomy



2024-08-26

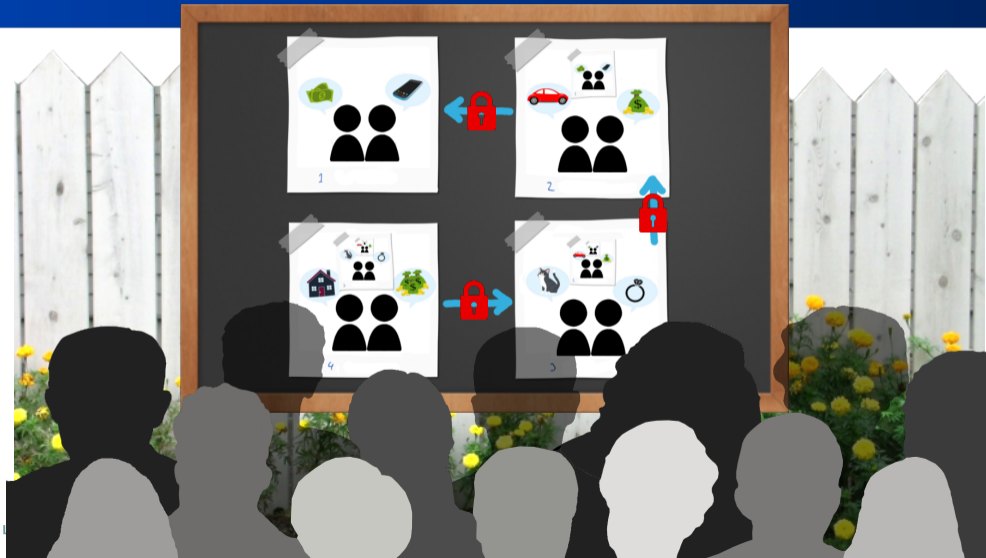
Blockchains

└ What properties are Blockchains claimed to have?

└ Autonomy

1. Autonomy is about the Blockchain continuing to work even if some participants drop out.
2. This ignores the issue that the participants that dropped out may have the resources to create a competing fork of the Blockchain, thus threatening immutability.
3. The possibility that anyone *could* find the resources to run a peer at any time makes the system very hard to shut down legally.

Anonymity



2024-08-26

Blockchains

└ What properties are Blockchains claimed to have?

└ Anonymity

1. Anonymity technically is about ensuring that a transaction cannot be linked to the individuals involved or other transactions (by the same individuals).
2. On a typical Blockchain, transactions are not actually between individuals but between cryptographic keys.
3. Anonymity is claimed as the individual in control of a private key may not be known.
4. However, multiple transactions by the same key are linkable, so often Blockchains at best achieve pseudonymity which is a weaker form of anonymity where transactions can still be linked.
5. Pseudonymity can be difficult to maintain, as one transaction that exposes the link to one's identity may expose many other transactions linked to the same key.
6. Furthermore, few Blockchains include anonymization at the IP layer, so merely accessing the overlay network to add a new transaction has the potential to break anonymity.

Summary: Blockchain “properties”



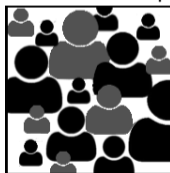
Im-
mutability



Trans-
parency



Anonymity



Decentralization



Autonomy



Irreversibil-
ity

2024-08-26

Blockchains

└ What properties are Blockchains claimed to have?

└ Summary: Blockchain “properties”

1. Irreversibility or finality are a variant of immutability, which stresses that transactions also cannot later be deleted. The same caveats as with immutability apply.
2. Again, all of these **only** hold with **significant caveats!**
3. Immutability, autonomy, decentralization and anonymity are the key reasons why Blockchains can be seen as “censorship-resistant”.
4. For example, some CSAM posted on some Blockchains cannot be effectively removed.

How does Proof-of-Work solve the Byzantine consensus problem?
(Who gets to append the next block?)

1. The most critical operation of any Blockchain is adding new transactions.
2. In practice, the process is not done for an individual transaction, but for a set of transactions.
3. A *block* is thus simply a new set of transactions (with a hash chaining it to its predecessor) that is being appended to the Blockchain.
4. Alice can use her private key to sign two conflicting transactions: one to send all her money to Bob, and another to send all her money to Carol.
5. All transaction systems need to ensure consistency: agreement about who owns what.
6. Thus, it is critical to determine which block is valid to decide: Should the “money” go to Bob or to Carol?

Proof of Work



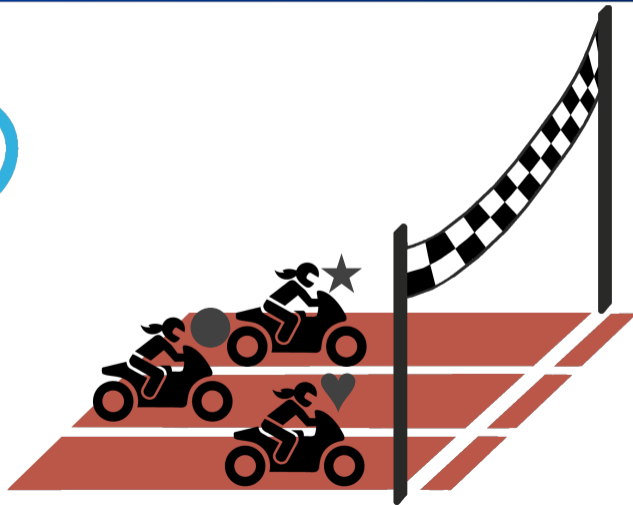
2024-08-26

Blockchains

- └ How does Proof-of-Work solve the Byzantine consensus problem?
 - └ Proof of Work

1. Proof of work is an attempt to solve the Byzantine consensus problem.
2. Here, we have a cloud of conflicting possible future realities.
3. Each of these futures has a stakeholder (miner) that would primarily benefit from this future.
4. These miners are *competitors*, each competing for their version of reality to become consensus.

Proof of Work



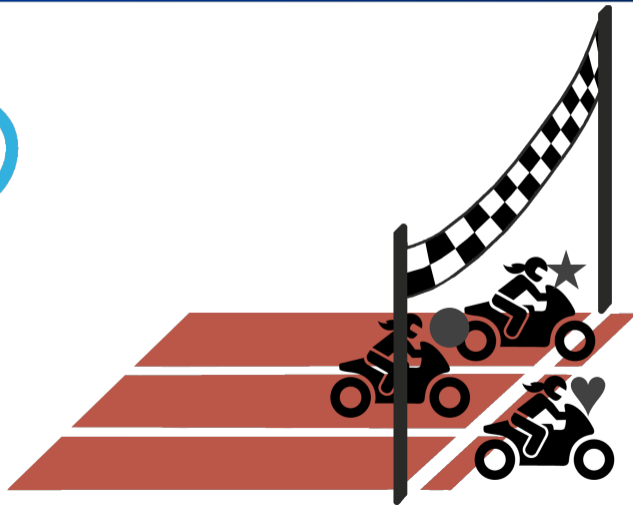
2024-08-26

Blockchains

- └ How does Proof-of-Work solve the Byzantine consensus problem?
 - └ Proof of Work

1. Proof of Work can be seen as a race.
2. The miners are challenged to solve a computational puzzle.
3. A typical puzzle involves finding an input that results in a (partial) hash collision.
4. For example, given a future reality represented by block B , find an I such that $H(B, I) \bmod N \leq M$ for a given value of N and M .
5. Given a cryptographic hash function, the chance of winning is $\frac{M}{N}$ per random input I .

Proof of Work



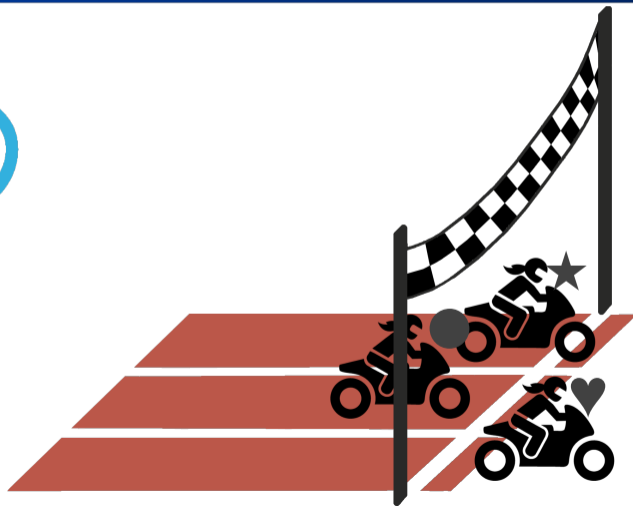
2024-08-26

Blockchains

- └ How does Proof-of-Work solve the Byzantine consensus problem?
 - └ Proof of Work

1. The block proposed by the first miner to solve the puzzle wins.
2. Miners can improve their chances to finish the computational puzzle first by putting in more computational power.
3. Usually, a monetary award is made to the miner who wins.
4. The choice of puzzle may provide advantages to miners using general-purpose CPUs, GPUs or specialized ASICs.
5. Some Blockchains use puzzles that are not about computational power but about storage space.

Proof of Work



2024-08-26

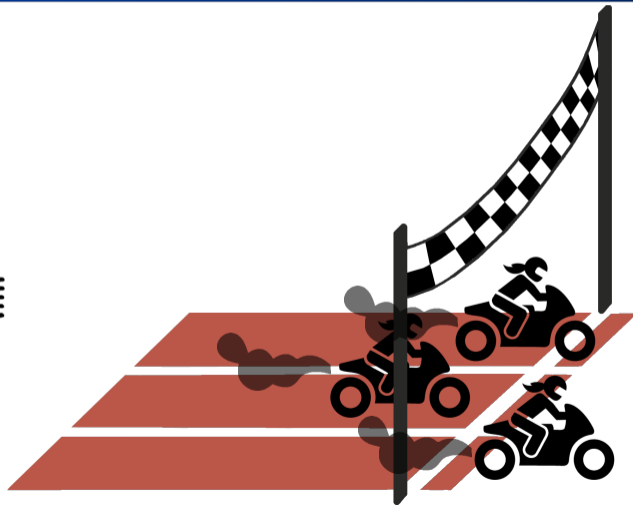
Blockchains

└ How does Proof-of-Work solve the Byzantine consensus problem?

└ Proof of Work

1. Usually, a monetary award is made to the miner who wins.
2. For example, in Bitcoin a special transaction that increases the money supply is made to the miner's account.
3. Outside of Blockchains, monetary policy issues are usually decided by experts at central banks behind closed doors.
4. Money supply policies programmed into Blockchains are detached from economic realities and used as a primary differentiator when advertising new shitcoins to unsavvy investors.
5. Algorithms designed to only ever create a finite amount of cryptocurrency create artificial scarcity and thus embody the cryptocurrency with a sense of value.

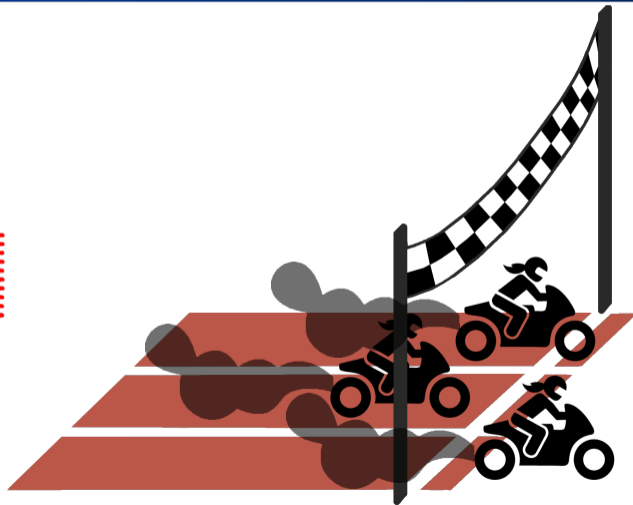
Proof of Work



Blockchains
└ How does Proof-of-Work solve the Byzantine consensus problem?
└ Proof of Work

1. After the race is before the race.
2. Once a new Block was mined, all miners are supposed to start with solving the next puzzle.
3. The general rule is that mining should be done on the longest chain.
4. The longest chain, that had the most puzzles solved, is considered "valid".

Proof of Work



└ How does Proof-of-Work solve the Byzantine consensus problem?

└ Proof of Work

1. Solving proof of work is a major contributor to global energy consumption. Bitcoin alone clocks **137 TWh/year**. (<https://www.statista.com/statistics/881472/worldwide-bitcoin-energy-consumption/>)
2. Globally, we produce $\approx 29,000$ TWh/year (<https://www.statista.com/statistics/270281/electricity-generation-worldwide/>). So Bitcoin uses 0.5%.

Bitcoin and Payments: A good match?

Bitcoin claims to be a *payment system* using a Blockchain:

- ▶ Public keys identify accounts, private keys used to send money from the account into other accounts.
- ▶ Set of internally consistent transactions form each block
- ▶ Each block includes a transaction creating fresh coins and transferring applicable fees to block creator
- ▶ Computational difficulty adjusts to mining power. A new block is mined in ≈ 10 minutes
- ▶ Amount of bitcoin money supply created per block is exponentially decreasing

Bitcoin claims to be a payment system using a Blockchain:

- ▶ Public keys identify accounts, private keys used to send money from the account into other accounts.
- ▶ Set of internally consistent transactions form each block
- ▶ Each block includes a transaction creating fresh coins and transferring applicable fees to block creator
- ▶ Computational difficulty adjusts to mining power. A new block is mined in ≈ 10 minutes
- ▶ Amount of bitcoin money supply created per block is exponentially decreasing

1. Miners make two types of profits from each block they mine.
2. One source of income are the new bitcoins created, effectively inflating the money supply.
3. The second source are fees paid by users wanting their transactions to be included in the block.
4. Transactions with higher fees are thus more likely to be included in a block by miners.

Rational Forking

Imagine:

- ▶ The previous block had a transaction from X to Y over 100 BTC with a fee of 0.001 BTC, a block reward of 7.5 BTC and total transaction fees of 5 BTC.
- ▶ The next consistent blocks can be assumed to again have block rewards of 7.5 BTC and transaction fees of 5 BTC.
- ▶ The issuer X of the 100 BTC transaction now signs a conflicting transaction where 50 BTC go to Z with a 25 BTC transaction fee.

What is the **rational** behavior for a miner M ?

2024-08-26

Blockchains

└ Bitcoin and Payments: A good match?

└ Rational Forking

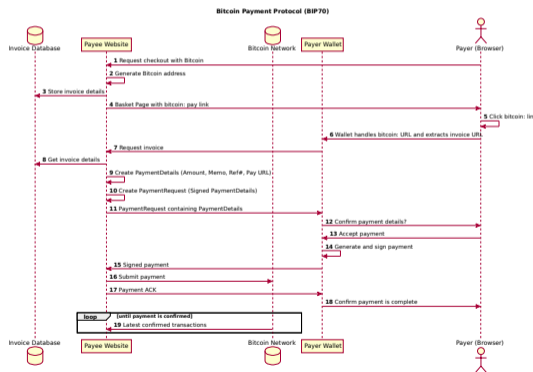
Imagine:

- ▶ The previous block had a transaction from X to Y over 100 BTC with a fee of 0.001 BTC, a block reward of 7.5 BTC and total transaction fees of 5 BTC.
- ▶ The next consistent blocks can be assumed to again have block rewards of 7.5 BTC and transaction fees of 5 BTC.
- ▶ The issuer X of the 100 BTC transaction now signs a conflicting transaction where 50 BTC go to Z with a 25 BTC transaction fee.

What is the **rational** behavior for a miner M ?

1. Re-mine the previous block P to produce a fork is clearly financially more beneficial than mining the next block.
2. But, you want the other miners to then also move to your fork.
3. Thus, M should post another dummy transaction from M with say a 5 BTC transaction fee that is only valid if M 's fork becomes the new longest chain, effectively rewarding other miners to switch.
4. In reality, the game theory involved gets quite complex. Note that especially the immutability and durability properties of the Blockchain depend on the game-theoretic results of the case.

Bitcoin Payment flow (by W3C Payment Interest Group)



2024-08-26

Blockchains

└ Bitcoin and Payments: A good match?

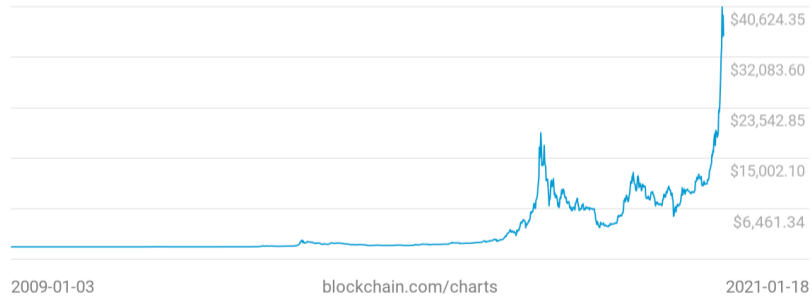
└ Bitcoin Payment flow (by W3C Payment Interest Group)



1. Here is an example for a W3C proposal for using Bitcoin for online payments.
2. Most of the details do not matter too much for us here.
3. But the critical step is the **loop** at the bottom in step **19**.
4. Here, after the payer was given a payment confirmation, the receiver is expected to loop **until payment is confirmed**.
5. But what does that mean? Satoshi's original paper suggests to wait for 6 blocks after the block containing the transaction in question to avoid issues with forks. At 10 minutes per block, this would take **one hour**.

The Value of Bitcoin

Market Price (USD)
\$35,793.01



2024-08-26

Blockchains

└ Bitcoin and Payments: A good match?

└ The Value of Bitcoin



1. The market value of Bitcoin has been extremely volatile.
2. Reasons include hype-driven speculation and limited liquidity.
3. While this type of fluctuation inherently benefits some and bankrupts others, we need to remember the golden rule of pure Pyramid schemes: This is a zero-sum game, so what one person gains, another must lose.
4. Central bankers define “money” as an asset with 3 properties: it can be used to **purchase** goods and services by entities other than the issuer, serves as a **store of value**, and as a **unit of account**. [?]
5. To serve as a **unit of account** requires reasonable stability. Bitcoin is not useful in that respect.

Mining requires:

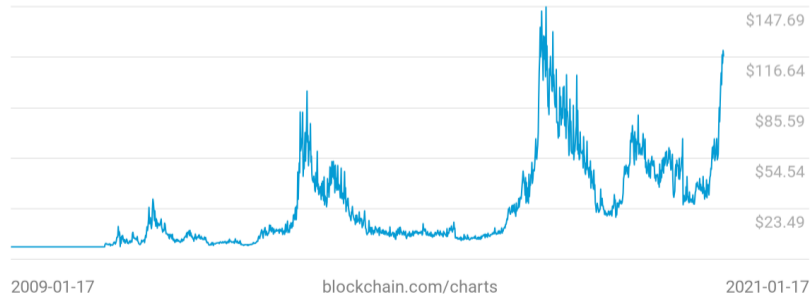
- ▶ Learning pending transactions from peers
- ▶ Selecting a subset of transactions which is valid (no double spending) by computing current account balances against the entire history
- ▶ Finding a hash collision (with adaptive difficulty)
- ▶ Propagating the new block to other miners

Usually specialized systems are used for finding hash collisions.

1. Checking that transactions are valid requires checking a cryptographic signature and an account balance.
2. To know all account balances, one must compute it from the (entire) history of the Blockchain.
3. A single wallet can use many keys (= accounts) to hold its assets. Using more keys may improve privacy.
4. Thus, there can be many more accounts than there are actual users on a Blockchain!

Mining cost

Cost per Transaction
\$117.47



Current average transaction value: \approx 1000 USD

2024-08-26

Blockchains

└ Bitcoin and Payments: A good match?

└ Mining cost



1. This chart is based on the mining rewards from computing each block, that is both the new money created as well as the transaction fees.
2. This total is then divided by the number of transactions in the block and converted from BTC to USD using the current exchange rate.
3. It does thus not reflect the actual fee paid, but includes the cost all Bitcoin owners are indirectly bearing via inflation of the money supply.
4. This is the **rational** limit of resources (hardware, electricity, bandwidth) a miner would spend on proof-of-work per transaction in a perfect market.
5. In reality, miners of course also compete by going for the cheapest available electricity supply (= poor countries where governments provide subsidies to make electricity affordable). Miners have of course also directly been stealing electric power.
(<https://www.bbc.com/news/uk-england-birmingham-57280115>)
6. At this cost, using Bitcoin to transact is primarily rational for very large and/or illegal transactions.

Bitcoin performance

- ▶ Privacy: all transactions happen in the clear in public view
- ▶ Latency: transactions take 1h to kind-of be confirmed
- ▶ Storage: grows linearly forever, no garbage collection
- ▶ Power: Bitcoin mining consumes more than the Netherlands today
- ▶ Rate: Network handles at most about 7 transactions per second
- ▶ Accountability: use of public keys as addresses enables criminal use

⇒ Bitcoin fever lasting for years. Why?

2024-08-26

Blockchains

└ Bitcoin and Payments: A good match?

└ Bitcoin performance

- ▶ Privacy: all transactions happen in the clear in public view
- ▶ Latency: transactions take 1h to kind-of be confirmed
- ▶ Storage: grows linearly forever, no garbage collection
- ▶ Power: Bitcoin mining consumes more than the Netherlands today
- ▶ Rate: Network handles at most about 7 transactions per second
- ▶ Accountability: use of public keys as addresses enables criminal use

→ Bitcoin fever lasting for years. Why?

1. Privacy: So as a user, you have no good privacy assurances as you have to assume that someone might be able to link the public key of your account(s) to you eventually.
2. Latency: The EU now requires instant payments to be done in about 10 seconds between European banks. So 1 hour is not exactly fast.
3. Storage: This is a forever-cost. Even just 10 year retention that is common in banking, storage costs can be a major IT cost driver.
4. Power consumption by country:
<https://www.statista.com/statistics/1260553/eu-power-demand-country/>
5. Rate: This is a theoretical limit, in practice Bitcoin has barely crossed over 4 TPS, despite having always over 100k transactions in the pool waiting to be included in the block chain.
6. Accountability: Blockchains with stronger privacy assurances of course do even worse here.
7. Why? Investors have an interest in pumping their investment!

- ▶ Dogecoin: same as Bitcoin, just named after a dog meme (an idea that is obviously worth billions!)
- ▶ Zcash: uses ZKSNARKs³ to hide transactions (criminal activity on Bitcoin was too low)
- ▶ Ethereum: run Turing-complete virtual machine logic in the blockchain to enable “smart” contracts and arbitrary applications, not just payments (is “Accelerando” an utopia or dystopia?)
- ▶ Polkadot: use side-chains to improve scalability

³ ≈ 1-15 minutes CPU time to create new transaction needed!

- ▶ Dogecoin: same as Bitcoin, just named after a dog meme (an idea that is obviously worth billions!)
- ▶ Zcash: uses ZKSNARKs³ to hide transactions (criminal activity on Bitcoin was too low)
- ▶ Ethereum: run Turing-complete virtual machine logic in the blockchain to enable “smart” contracts and arbitrary applications, not just payments (is “Accelerando” an utopia or dystopia?)
- ▶ Polkadot: use side-chains to improve scalability

³ ≈ 1-15 minutes CPU time to create new transaction needed!

1. Pumping and dumping is easier in illiquid markets. Thus, cryptocurrencies proliferate.
2. Tons of possible design variations: different puzzles, privacy, programmability, scalability
3. Proof-of-work is rather egalitarian; proof-of-stake more efficiently distributes wealth to those already wealthy by giving those with the biggest amount of coins the power to mine.
4. The elimination of private money issued by private banks and the centralization of credit at a national (central) bank was one of the core demands from the Communist Manifesto (1848).

What are other applications for Blockchains?

James Mickens on Blockchains

James W. Mickens is an American computer scientist and the Gordon McKay Professor of Computer Science at Harvard John A. Paulson School of Engineering and Applied Sciences at Harvard University. His research focuses on distributed systems, such as large-scale services and ways to make them more secure.

At the Digital Initiative's Future Assembly on April 6, 2018, he presented "Blockchains Are a Bad Idea: More Specifically, Blockchains Are a Very Bad Idea."

2024-08-26

Blockchains

└─What are other applications for Blockchains?

└─James Mickens on Blockchains

James W. Mickens is an American computer scientist and the Gordon McKay Professor of Computer Science at Harvard John A. Paulson School of Engineering and Applied Sciences at Harvard University. His research focuses on distributed systems, such as large-scale services and ways to make them more secure.

At the Digital Initiative's Future Assembly on April 6, 2018, he presented "Blockchains Are a Bad Idea: More Specifically, Blockchains Are a Very Bad Idea."

1. So maybe payments are not actually a strength of Blockchains. But maybe there are other applications for Blockchains?
2. For this, we will now watch an opinionated talk by James Mickens.
3. He is probably qualified to speak on the subject, after all he explained at USENIX Security that a core tenet of technological manifest destiny is that history is uninteresting — which is key given that Blockchain proponents seem to have “forgotten” the historical issues that led to the creation of central banks. So what does he think of the Blockchain religion?

2024-08-26

Blockchains

└─ What are other applications for Blockchains?

<https://www.youtube.com/watch?v=15RTC22Z2xI> (2018)

<https://www.youtube.com/watch?v=15RTC22Z2xI> (2018)

Security Goals for Time Stamping Services

- ▶ Document must have existed at the timestamp
- ▶ Modifications must be detected
- ▶ Document must have been created after the timestamp
- ▶ Validation of timestamp proof possible forever
- ▶ Non-repudiation
- ▶ No trusted third party (see [1, 3] for protocols with trusted third party)
- ▶ Availability

2024-08-26

Blockchains

└─ What are other applications for Blockchains?

└─ Security Goals for Time Stamping Services

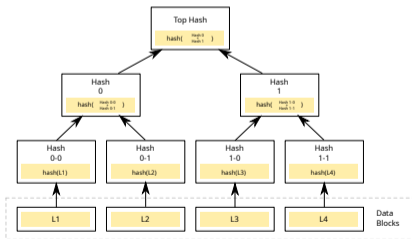
- ▶ Document must have existed at the timestamp
- ▶ Modifications must be detected
- ▶ Document must have been created after the timestamp
- ▶ Validation of timestamp proof possible forever
- ▶ Non-repudiation
- ▶ No trusted third party (see [1, 3] for protocols with trusted third party)
- ▶ Availability

1. Applications for time stamping include establishing ownership, for example of prior art in patent or copyright disputes.
2. Timestamping services using a trusted third party have existed online and offline for a very long time.
3. But as always, we do not like trusted third parties in information security.

Blockchain-based Time Stamping Services

- ▶ <https://originastamp.com/>: Bitcoin&Ethereum, 100 timestamps \$10
- ▶ <https://blockchainsign.io/>: Ethereum, 1 timestamp \$5
- ▶ <https://guardtime.com/>: private KSI Blockchain (!?)

Key idea:



2024-08-26

Blockchains

└ What are other applications for Blockchains?

└ Blockchain-based Time Stamping Services

- ▶ <https://originastamp.com/>: Bitcoin&Ethereum, 100 timestamps \$10
- ▶ <https://blockchainsign.io/>: Ethereum, 1 timestamp \$5
- ▶ <https://guardtime.com/>: private KSI Blockchain (!?)



1. There are various commercial providers offering timestamping solutions using Blockchains.
2. They differ in pricing and Blockchains used. I do not see a tangible benefit of using a private Blockchain over just using some trusted third parties.
3. As putting data onto a Blockchain is expensive, the various documents to be timestamped are hashed, and the hashes combined with each other in a Merkle tree.
4. The root of the Merkle tree is then included in a block on the Blockchain.

Security Goals for Name Systems

- ▶ Query origin anonymity
- ▶ Data origin authentication and integrity protection
- ▶ Zone confidentiality
- ▶ Query and response privacy
- ▶ Censorship resistance
- ▶ Traffic amplification resistance
- ▶ Availability

2024-08-26

Blockchains

└─ What are other applications for Blockchains?

└─ Security Goals for Name Systems

- ▶ Query origin anonymity
- ▶ Data origin authentication and integrity protection
- ▶ Zone confidentiality
- ▶ Query and response privacy
- ▶ Censorship resistance
- ▶ Traffic amplification resistance
- ▶ Availability

1. QOA: We do not know who asked for name resolution. In traditional DNS, the origin is only exposed to the ISP running the recursive resolver.
2. DOA: We want to make sure that the answer is correct.
3. ZC: The zone publisher does not want all records to be public, only given the label one should be able to determine the record set. Not all labels are public.
4. QRP: The name resolved (question) and the record set returned (answer) are themselves private and not disclosed to the infrastructure.
5. CR: Authorities cannot selectively block access to some record sets.
6. TAR: The system cannot be abused as a traffic multiplier for DDoS attacks.
7. A: New zones can be added, and new record sets published and resolved.

Approaches Adding Cryptography to DNS

- ▶ DNSSEC
- ▶ DNSCurve
- ▶ DNS-over-TLS (DoT)
- ▶ DNS-over-HTTPS (DoH)
- ▶ RAINS
- ▶ GNU Name System (GNS)

2024-08-26

Blockchains

└─What are other applications for Blockchains?

└─Approaches Adding Cryptography to DNS

- ▶ DNSSEC
- ▶ DNSCurve
- ▶ DNS-over-TLS (DoT)
- ▶ DNS-over-HTTPS (DoH)
- ▶ RAINS
- ▶ GNU Name System (GNS)

1. These are all mostly cryptographic proposals making different political trade-offs.
2. Except for GNS, they all assume some ICANN-like authority to manage the root zone.
3. Even GNS *supports* (and would benefit from) trustworthy registration authorities.
4. But can we do without a trusted third party like ICANN?

No need for a trusted third party: put the records into the Blockchain!

Or rather, put the public key of the owner and signed updates into it.

Plus, expiration rules.

└─ What are other applications for Blockchains?

└─ Namecoin

No need for a trusted third party: put the records into the Blockchain!

Or rather, put the public key of the owner and signed updates into it.

Plus, expiration rules.

1. And of course use our new **utility token** to pay for registration.
2. Big issues: trademark infringement, use by malware, etc.

Ethereum Name System⁴

Let's have a smart contract in the Blockchain manage naming!

Blockchain contains smart contract and data who controls which name.

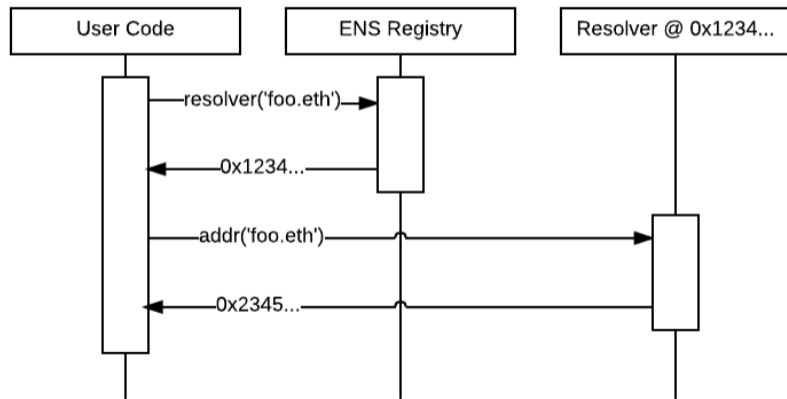
Contract allocates names under .eth using auctions.

⁴<https://ens.domains/>

1. Why write another Blockchain? Ethereum is programmable!

Let's have a smart contract in the Blockchain manage naming!
Blockchain contains smart contract and data who controls which name.
Contract allocates names under .eth using auctions.
<https://ens.domains/>

Ethereum Name System⁶

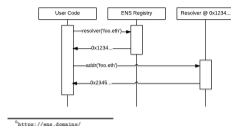


2024-08-26

Blockchains

└─ What are other applications for Blockchains?

└─ Ethereum Name System^a



1. Actual lookup is indirect: first lookup resolver code
2. Then ask resolver code for actual resolution result(s)
3. Issue: Ethereum chain is **huge**

Handshake Name System⁸

Incremental improvements over Namecoin and ENS:

- ▶ New blockchain with “HNS” utility tokens
- ▶ Compact proofs: resolvers do not need the full chain
- ▶ Pre-reserved names (ICANN TLDs, top-100k Alexa domains)
- ▶ Air-drop to “stakeholders” to boost adoption

⁸<https://handshake.org/>



└─ What are other applications for Blockchains?

└─ Handshake Name System^a

1. Cryptographic improvement: secure name resolution without full chain
2. Social consideration: do not allow anybody to register any name immediately, instead give existing DNS-owners grace period to register “\$TRADEMARK.hns”.
3. Presumes trademark owners care enough about potential success of Handshake to spend money to buy HNS and register trademarks in yet another (unofficial) TLD.

- ▶ New blockchain with “HNS” utility tokens
- ▶ Compact proofs: resolvers do not need the full chain
- ▶ Pre-reserved names (ICANN TLDs, top-100k Alexa domains)
- ▶ Air-drop to “stakeholders” to boost adoption


References I


-  C. Adams, P. Cain, D. Pinkas, and R. Zuccherato.
Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP).
RFC 3161 (Proposed Standard), August 2001.
Updated by RFC 5816.
-  Alexandra Dirksen.
A blockchain picture book.
https://media.ccc.de/v/35c3-9573-a_blockchain_picture_book, 12
2018.

2024-08-26

Blockchains
└─References

└─References

 C. Adams, P. Cain, D. Pinkas, and R. Zuccherato.
Internet X.509 Public Key Infrastructure Time-Stamp Protocol (TSP).
RFC 3161 (Proposed Standard), August 2001.
Updated by RFC 5816.

 Alexandra Dirksen.
A blockchain picture book.
https://media.ccc.de/v/35c3-9573-a_blockchain_picture_book, 12
2018.

References II

-  D. Pinkas, N. Pope, and J. Ross.
Policy Requirements for Time-Stamping Authorities (TSAs).
RFC 3628 (Informational), November 2003.

2024-08-26

Blockchains

└─References

└─References

 D. Pinkas, N. Pope, and J. Ross.
Policy Requirements for Time-Stamping Authorities (TSAs).
RFC 3628 (Informational), November 2003.

Acknowledgements

Co-funded by the European Union (Project 101135475).



Co-funded by
the European Union

Co-funded by SERI (HEU-Projekt 101135475-TALER).

Project funded by



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs,
Education and Research EAER
**State Secretariat for Education,
Research and Innovation SERI**

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Blockchains

Christian Grothoff

Using Bitcoin

1. Installing bitcoind

You can find the installation instructions at <https://bitcoin.org/en/full-node>. Download and compile the sources from <https://bitcoin.org/en/download>. You may need to install various dependencies. I needed at least:

```
# apt install qttools5-dev-tools build-essential libtool \  
  autotools-dev automake pkg-config libssl-dev libevent-dev \  
  bsdmaintils libboost-system-dev libboost-filesystem-dev \  
  libboost-chrono-dev libboost-program-options-dev \  
  libboost-test-dev libboost-thread-dev libminiupnpc-dev \  
  libzmq3-dev jq libqt5opengl5-dev
```

Details may differ for your system depending on what you already had installed and which exact operations system version you are using. Read the output of `configure` carefully.

In general, compilation merely requires

```
$ ./autogen.sh  
$ ./configure --prefix=$HOME/btc  
$ make # patience  
$ make install
```

After compiling, use

```
$ $HOME/btc/bin/bitcoind -server -testnet
```

to launch your daemon on TESTNET. You will need about 50 GB of disk space in `$HOME/.bitcoin/`. Wait a few hours to become synchronized.

2. Create a wallet

Stop `bitcoind` and use

```
$ $HOME/btc/bin/bitcoin-qt -testnet
```

to launch the graphical client. Create a new wallet and select “Receive”. Find a testnet Faucet (from <https://en.bitcoin.it/wiki/Testnet>) and request free TESTNET BTC. It may take some time for your TESTNET BTC to arrive.

3. Making a payment

Send some funds to `tb1qen3qwrzza6vt3fs43ufzz0m635jv8secqdpshg`. Use the TESTNET blockchain explorer at <https://live.blockcypher.com/btc-testnet> to check that your transaction worked. Patience may be required.

4. Ethical Case Study: DoH

DNS is known to suffer from a lack of end-to-end integrity protections. As a result, Chinese “great firewall” DNS manipulation has been shown to impact name resolution even in Europe.

“The IETF is standardizing DNS over HTTPS (DOH), where all DNS queries are sent over the HTTPS protocol to some well-known HTTPS server (such as Google’s 8.8.8.8 or Cloudflare’s 1.1.1.1). This will prevent local governments from manipulating DNS traffic and improve the user’s privacy with respect to their ISPs and governments. However, Google or Cloudflare will see the DNS queries and replies of the users; they must be expected to have weak privacy policies and are subject to US law which includes secret rules and court orders. The NSA has a history of snooping on (MORECOWBELL) and manipulating (QUANTUMDNS) DNS traffic.”

Should we develop and deploy technologies like DoH?

5. Ethical Case Study: RAINS

DNS is known to suffer from a lack of end-to-end integrity protections. As a result, Chinese “great firewall” DNS manipulation has been shown to impact name resolution even in Europe.

“The ETH Zurich is developing a new name system called RAINS with a new trust anchor operated by the regional Internet service providers, aka the local Isolation Service Domain (ISD). RAINS does not change the privacy of DNS (providers can continue to monitor traffic, all zone data becomes public) and allows the local authorities to block Web sites to improve public safety and enforce local laws (see also: “Glücksspielgesetz in Switzerland”). At the same time, foreign censorship efforts are less likely to be effective (unless they foreign government forces the DNS authority to alter the authoritative records).”

Should we develop and deploy technologies like RAINS?

6. Ethical Case Study: Namecoin

DNS is known to suffer from a lack of end-to-end integrity protections. As a result, Chinese “great firewall” DNS manipulation has been shown to impact name resolution even in Europe.

“Namecoin establishes a new name system on the blockchain (where thus zone data is also public), but where public authorities cannot block information. Queries are performed against a local copy of the blockchain and thus also private. There is no WHOIS, so the owner of a name can also be anonymous. However, Namecoin uses much more bandwidth and energy as blockchain payments are used for registration and name resolution. Names are registered on a first-come, first-served basis. Trademarks, copyrights anti-fraud or anti-terrorism judgements cannot be used to force owners of names to relinquish names.”

Should we develop and deploy technologies like Namecoin?

NEXT GENERATION INTERNET

Secure Integration

Christian Grothoff

31.05.2024

2024-08-26

NEXT GENERATION INTERNET

NEXT
GENERATION
INTERNET

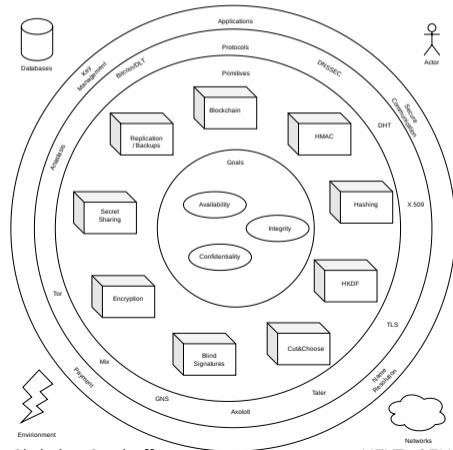
Secure Integration

Christian Grothoff

31.05.2024

1. In this lecture, we will talk about security issues when trying to integrate applications.
2. As motivation, you can take the sPACE attack on the German passport application.
3. Here, we had a secure digital passport (the nPA) that was to be accessed via an App on a smart phone.
4. Now some legitimate Web site or Internet service wants to use the nPA for authentication, the user is asked to launch the App, and ...
5. ... some attacker has installed a different App, launches a man-in-the-middle attack, and opens a bank account in the user's name.

Integration on our map



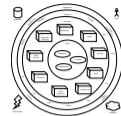
Christian Grothoff

NEXT , GENERATION , INTERNET

2024-08-26

NEXT GENERATION INTERNET

Integration on our map



1. Integration is a cross-cutting concern that is about achieving our security goals while ensuring usability under real-world environmental constraints.
2. It is primarily a concern at the edge: if we have different applications that are individually secure, how can we make them work together and maintain security?
3. A related security topic is standardization and standardization processes, as for good security is recommended to use standards where possible, but also to contribute to the development of standards where necessary.

Learning Objectives

How can we securely integrate services?

Terminology

How to register a URI scheme?

Example: RFC 8905

Example: LSD 0006

2024-08-26

NEXT GENERATION INTERNET

Learning Objectives

1. At a high level, the goal is to understand the options for integrating services...
2. ... and the security risks and what could be done about them — or not.

How can we securely integrate services?

Terminology

How to register a URI scheme?

Example: RFC 8905

Example: LSD 0006

How can we securely integrate services?

Integration: Problem Statement

Isolation is a key paradigm in security:

- ▶ processes (address spaces!)
- ▶ users (quotas, access rights)
- ▶ departments (accounting, controlling, revision)
- ▶ organizations (auditors)

2024-08-26

NEXT GENERATION INTERNET

└ How can we securely integrate services?

└ Integration: Problem Statement

Isolation is a key paradigm in security:

- ▶ processes (address spaces!)
- ▶ users (quotas, access rights)
- ▶ departments (accounting, controlling, revision)
- ▶ organizations (auditors)

1. We don't usually build the one mega application that integrates everything.
2. Just look at many e-commerce sites, which redirect you to some third party page for payment.
3. This can be a good thing: only the specialized logic has access to sensitive data, reducing the attack surface.
4. But, if we isolate functions such as login, authentication, payment or data access into dedicated components, we still need to facilitate access to key results from those functions to other parts of the system.

Integration: Problem Statement

Isolation is a key paradigm in security:

- ▶ processes (address spaces!)
- ▶ users (quotas, access rights)
- ▶ departments (accounting, controlling, revision)
- ▶ organizations (auditors)

How to ensure good user experience across application boundaries?

2024-08-26

NEXT GENERATION INTERNET

└─ How can we securely integrate services?

└─ Integration: Problem Statement

Isolation is a key paradigm in security:

- ▶ processes (address spaces!)
- ▶ users (quotas, access rights)
- ▶ departments (accounting, controlling, revision)
- ▶ organizations (auditors)

How to ensure good user experience across application boundaries?

1. We don't usually build the one mega application that integrates everything.
2. Just look at many e-commerce sites, which redirect you to some third party page for payment.
3. This can be a good thing: only the specialized logic has access to sensitive data, reducing the attack surface.
4. But, if we isolate functions such as login, authentication, payment or data access into dedicated components, we still need to facilitate access to key results from those functions to other parts of the system.

Solution domains

► Fax

2024-08-26

NEXT GENERATION INTERNET

└ How can we securely integrate services?

└ Solution domains

► Fax

1. So for inter-departmental communication, the classical solution is of course the use of Fax machines. Yes, those still exist and are not exactly “secure”.
2. How do we do this with software? (Let students answer, list above is incomplete.)
3. Today, we will focus on **deep links**, a common approach and basically also the only one (!) available on iOS.

Solution domains

- ▶ Fax
- ▶ Inter-process communication (UNIX Domain Sockets, Shared Memory, Networking)
- ▶ Intents (Android-only!)
- ▶ **Deep links**

2024-08-26

NEXT GENERATION INTERNET

└─ How can we securely integrate services?

└─ Solution domains

- ▶ Fax
- ▶ Inter-process communication (UNIX Domain Sockets, Shared Memory, Networking)
- ▶ Intents (Android-only)
- ▶ **Deep links**

1. So for inter-departmental communication, the classical solution is of course the use of Fax machines. Yes, those still exist and are not exactly “secure”.
2. How do we do this with software? (Let students answer, list above is incomplete.)
3. Today, we will focus on **deep links**, a common approach and basically also the only one (!) available on iOS.

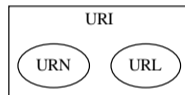


2024-08-26

Terminology



URI Uniform Resource Identifier
URL Uniform Resource Locator — object identification tied to location
URN Uniform Resource Name — namespace independent of location



URI Uniform Resource Identifier

URL Uniform Resource Locator — object identification tied to location

URN Uniform Resource Name — namespace independent of location

1. This is just a quick review, you've probably seen this before.
2. When we talk about links, we usually talk about URIs, which can be URLs or URNs.

Structure of a URI

URI = scheme “:” hierarchical-part [“?” query] [“#” fragment]

scheme Defines the method of identification

hierarchical part Hierarchical access path of the URI

query Search function

fragment access to a part of the document

2024-08-26

NEXT GENERATION INTERNET

└ Terminology

└ Structure of a URI

URI = scheme ":" hierarchical-part ["?" query] ["#" fragment]
scheme Defines the method of identification
hierarchical part Hierarchical access path of the URI
query Search function
fragment access to a part of the document

1. This is the general structure of a URI.
2. Note that the “//” of “://” is not actually required syntactically, it arose from an older convention that separated directories and filenames with “/” and used “//” to indicate that the first part was a hostname.
3. These days, “//” is technically indicative that the next element is the *authority* consisting of an optional username (possibly with passport) at a host and an optional port.
4. However, because **users** recognize URIs by the “scheme://”-prefix, it can be reasonable to use “://” syntactically even without an authority following as the next element.

URI/URN examples

2024-08-26

NEXT GENERATION INTERNET

└ Terminology

└ URI/URN examples



foo://example.com:8042/over/there?name=ferret#nose

└ scheme └ authority └ path └ query └ fragment

urn:example:animal:ferret:nose

The Internet Assigned Numbers Authority (IANA)

- ▶ Responsible for unique assignment of parameters and numbers in Internet protocols
- ▶ Operates the DNS root zone
- ▶ Performs the administrative work *on behalf of* ICANN and IETF
- ▶ Web site: <https://iana.org/>
- ▶ Used to be just Jon Postel



2024-08-26

NEXT GENERATION INTERNET

└ Terminology

└ The Internet Assigned Numbers Authority (IANA)

- ▶ Responsible for unique assignment of parameters and numbers in Internet protocols
- ▶ Operates the DNS root zone
- ▶ Performs the administrative work *on behalf of* ICANN and IETF
- ▶ Web site: <https://iana.org/>
- ▶ Used to be just Jon Postel



1. When we have protocol parameters in protocols, we must ensure that the same value has the same meaning for all applications.
2. To avoid conflicts, it thus is important to establish a **registry**. This way, we can ensure that parameters have a unique use.
3. For the Internet, the official registry for protocol parameters is the IANA.
4. IANA is operated as a public service by the Internet Society (ISOC). Thus, registering protocol parameters does not cost you any money, only some effort.

Common URI scheme

- [ftp](#) File Transfer Protocol [1]
- [http](#) Hyper Text Transfer Protocol [2]
- [https](#) HTTP Secure [6]
- [mailto](#) Electronic mail address [3]
- [file](#) Host-specific file names [1]
- [imap](#) Internet Message Access Protocol [4]
- [ldap](#) Lightweight Directory Access Protocol [7]
- [urn](#) Uniform Resource Names¹ [5]

Excerpt from <http://www.iana.org/assignments/uri-schemes>.

¹<http://www.iana.org/assignments/urn-namespaces>

1. For URIs, the registry is the `uri-schemes` registry; with some well-known schemes listed above.
2. A key point to note is that such a scheme does **not** identify a specific application. For example, “http” does not say “Chromium” or “Firefox”. In fact, it doesn’t even say “browser” (“wget” works, too!).
3. So a URI identifies a resource, but it does not specify which software should interact with the resource.

URI examples

`http://prof.hti.bfh.ch/index.php?id=1403&L=2#howto`

`http://[2001:620:500:ff80::80]/owncloud`

`ftp://ftp.rfc-editor.org/in-notes/rfc3986.txt`

`ftp://user:geheim@ftp.bfh.ch/`

`file:///C:/WINDOWS/system32/drivers/etc/services`

`mailto:firstname.lastname@bfh.ch`

`urn:ietf:rfc:3986`

`urn:ISBN:1-56592-862-8`

2024-08-26

NEXT GENERATION INTERNET

└ Terminology

└ URI examples

```
http://prof.hti.bfh.ch/index.php?id=1403&L=2#howto
http://[2001:620:500:ff80::80]/owncloud
ftp://ftp.rfc-editor.org/in-notes/rfc3986.txt
ftp://user:geheim@ftp.bfh.ch/
file:///C:/WINDOWS/system32/drivers/etc/services
mailto:firstname.lastname@bfh.ch
urn:ietf:rfc:3986
urn:ISBN:1-56592-862-8
```

1. Here are some specific examples of URIs for the various schemes.
2. Note the 2nd example for “ftp”, where a password is encoded as part of the URI.
3. So URIs frequently carry sensitive credentials. For Web resources, the credentials are also often encoded in the path.
4. Hence, obviously we need to be concerned with security when using URIs.

Security Considerations

- ▶ Link hijacking: malicious or competing apps can register for your scheme
- ▶ Data interception: links with sensitive data may be transmitted by users over insecure channels
- ▶ Access control bypass: ensure checking access when handling links
- ▶ Insecure parameter handling: strings are the source of all eval

2024-08-26

NEXT GENERATION INTERNET

└ Terminology

└ Security Considerations

- ▶ Link hijacking: malicious or competing apps can register for your scheme
- ▶ Data interception: links with sensitive data may be transmitted by users over insecure channels
- ▶ Access control bypass: ensure checking access when handling links
- ▶ Insecure parameter handling: strings are the source of all eval

1. We cannot be sure which app will be used to open a link. On many OSes, there is a scheme registry where applications can register themselves as able to open links of certain schemes. A secure OS at least gives the user the list of choices and asks for confirmation before passing a link to some random application. But in the end, the user could of course always choose the wrong app. We cannot really stop a user from passing a link with some credentials to malware.
2. URIs can be passed via QR code, NFC or some random messaging application — and those mechanisms may be subject to data interception by malicious parties. Again, very hard to stop users from doing this.
3. Now, when handling a URI that accesses some logic within some application, a key issue is that we must not forget access control checks. Deep links can open new control paths, so check access control.
4. URIs are strings from an untrusted source. So we need to worry about our parsing logic and input validation.

How to register a URI scheme?

How to register a URI scheme? [8]

There are *permanent* and *provisional* registrations. Always start with *provisional*, but largely follow *permanent* guidelines:

1. Write and publish citable specification (ideally, RFC-style) explaining the use-case, syntax, semantics and security considerations
2. Follow syntactic requirements and ensure name is not taken
3. Send a registration request to `uri-review@ietf.org` and possibly other relevant lists for discussion.
4. Respond to comments, address in specification where reasonable (wait a few weeks for discussion to conclude).
5. Submit updated registration request to `iana@iana.org` with pointer to the discussion.

You can always “upgrade” to *permanent* status later!

2024-08-26

NEXT GENERATION INTERNET

└─How to register a URI scheme?

└─How to register a URI scheme? [8]

1. The process is not that hard, it mostly requires writing a reasonably clear specification, an open ear for community feedback, and some patience.
2. Going for “permanent” immediately is setting yourself up for failure, and to avoid conflicts “provisional” is perfectly sufficient.

There are permanent and provisional registrations. Always start with provisional, but largely follow permanent guidelines:

1. Write and publish citable specification (ideally, RFC-style) explaining the use-case, syntax, semantics and security considerations
2. Follow syntactic requirements and ensure name is not taken
3. Send a registration request to `uri-review@ietf.org` and possibly other relevant lists for discussion.
4. Respond to comments, address in specification where reasonable (wait a few weeks for discussion to conclude).
5. Submit updated registration request to `iana@iana.org` with pointer to the discussion.

You can always “upgrade” to permanent status later!



2024-08-26

Example: RFC 8905

RFC 8905: payto: Uniform Identifiers for Payments and Accounts

Like `mailto:`, but for bank accounts instead of email accounts!

```
payto://<PAYMENT-METHOD>/<ACCOUNT-NR>  
?subject=InvoiceNr42  
&amount=EUR:12.50
```

Default action: Open app to review and confirm payment.

SEPA Überweisung/Conversion
SEPA Bank (Zahlungsbank) (Teil 1) ist
Name und für die Überweisung/Conversion
Konten- und Zahlungsinformationen (Teil 2) sind für die Überweisung/Conversion
SEPA Überweisung/Conversion

SEPA Überweisung/Conversion	SEPA Überweisung/Conversion
SEPA Bank (Zahlungsbank) (Teil 1) ist Name und für die Überweisung/Conversion Konten- und Zahlungsinformationen (Teil 2) sind für die Überweisung/Conversion SEPA Überweisung/Conversion	SEPA Bank (Zahlungsbank) (Teil 1) ist Name und für die Überweisung/Conversion Konten- und Zahlungsinformationen (Teil 2) sind für die Überweisung/Conversion SEPA Überweisung/Conversion

SEPA Überweisung/Conversion
SEPA Bank (Zahlungsbank) (Teil 1) ist
Name und für die Überweisung/Conversion
Konten- und Zahlungsinformationen (Teil 2) sind für die Überweisung/Conversion
SEPA Überweisung/Conversion

MICR line: 01 5720 8007 2033 4550 00124

Large red watermark: MUSTER

2024-08-26

NEXT GENERATION INTERNET

Example: RFC 8905

RFC 8905: `payto:` Uniform Identifiers for Payments and Accounts

Like `mailto:`, but for bank accounts instead of email accounts!
`payto://<PAYMENT-METHOD>/<ACCOUNT-NR>`
`?subject=InvoiceNr42`
`&amount=EUR:12.50`

Default action: Open app to review and confirm payment.



1. The idea of “`payto`” is pretty simple: it identifies an address where somebody could send money.
2. The “`PAYMENT-METHOD`” determines all the details about the payment system used for the transfer, and in particular determines what the “`ACCOUNT-NR`” must look like
3. Optionally, meta-data such as the wire transfer subject or the amount to wire can be passed.

Benefits of payto://

- ▶ Standardized way to represent financial resources (bank account, bitcoin wallet) and payments to them
- ▶ Useful on the client-side on the Web and for FinTech backend applications
- ▶ Payment methods (such as IBAN, ACH, Bitcoin) are registered with GANA ²

²<https://gana.gnunet.org/>

1. Previously, various banking systems had defined their own methods, often for QR-code payments, all non-interoperable and often without even using a proper scheme — and never with IANA registration.
2. On some platforms applications can register for a prefix like “schema://text/” instead of just a schema. On those platforms, applications could register for “payto://iban/” and only get invoked for European IBAN transfers and never for say American ACH payment methods.
3. GANA is an alternative protocol registry to IANA, as for unclear reasons IANA refused to run a sub-registry for “payto://”. So do consider setting up your own (sub)registry for protocol parameters if IANA is not applicable.

Security Considerations for `payto://`

- ▶ Interactive applications handling the 'payto' URI scheme MUST NOT initiate any financial transactions without confirmation from the user and MUST take measures to prevent click-jacking.
- ▶ Unless a 'payto' URI is received over a trusted, authenticated channel, a user might not be able to identify the target of a payment. A payment target type SHOULD NOT use human-readable names in combination with Unicode in the target account specification.
- ▶ The authentication/authorization mechanisms used to process a payment encoded in a 'payto' URI are handled by the application and are not in scope of this document.
- ▶ Payment target types SHOULD NOT include personally identifying information about the sender of a payment that is not essential to conduct a payment.

2024-08-26

NEXT GENERATION INTERNET

└ Example: RFC 8905

└ Security Considerations for `payto://`

1. Using Unicode is a problem, as it could give the user the illusion of being able to identify the target account from the URI due to homographs.
2. The last point is privacy-by-design: only put information that is strictly required. This also helps keep QR codes small and thus faster to scan.

- ▶ Interactive applications handling the 'payto' URI scheme MUST NOT initiate any financial transactions without confirmation from the user and MUST take measures to prevent click-jacking.
- ▶ Unless a 'payto' URI is received over a trusted, authenticated channel, a user might not be able to identify the target of a payment. A payment target type SHOULD NOT use human-readable names in combination with Unicode in the target account specification.
- ▶ The authentication/authorization mechanisms used to process a payment encoded in a 'payto' URI are handled by the application and are not in scope of this document.
- ▶ Payment target types SHOULD NOT include personally identifying information about the sender of a payment that is not essential to conduct a payment.



2024-08-26

NEXT GENERATION INTERNET
└ Example: LSD 0006

Example: LSD 0006

Example: LSD 0006

LSD 0006: taler: wallet triggers

<https://lsd.gnunet.org/lsd0006/>

Syntax:

```
taler-URI = ("taler://" / "TALER://" / "taler+http://"
            / "TALER+HTTP://" )
           action path-abempty [ "?" opts ]
action = ALPHA *( ALPHA / DIGIT / "-" / "." )
opts = opt *( "&" opt )
opt = opt-name "=" opt-value
opt-name = ALPHA *( ALPHA / DIGIT / "-" / "." / ":" )
opt-value = *pchar
```

Example:

`taler://pay-push/exchange.taler.grothoff.org/D83MG3W7WKVH3C9...`

```
Syntax:
taler-URI = ("taler://" / "TALER://" / "taler+http://"
            / "TALER+HTTP://" )
           action path-abempty [ "?" opts ]
action = ALPHA *( ALPHA / DIGIT / "-" / "." )
opts = opt *( "&" opt )
opt = opt-name "=" opt-value
opt-name = ALPHA *( ALPHA / DIGIT / "-" / "." / ":" )
opt-value = *pchar

Example:
taler://pay-push/exchange.taler.grothoff.org/D83MG3W7WKVH3C9...
```

1. Unlike “payto”, the “taler” scheme is used to trigger specific operations in a Taler wallet. So this scheme is a typical example of a “deep link” that other applications can use to integrate with the payment functionality offered by the Taler wallet.
2. The “action” here specifies the specific operation of the Taler wallet that should be triggered.
3. The action is followed by an action-specific mandatory path and optional arguments.

taler:// actions

- `withdraw` bank-initiated withdrawal
- `pay` merchant-initiated payment
- `refund` merchant-initiated refund
- `pay-push` P2P payment
- `pay-pull` P2P invoice
- `pay-template` merchant offline payment
- `restore` restore from backup
- `withdraw-exchange` wallet-initiated withdrawal

2024-08-26

NEXT GENERATION INTERNET


└ Example: LSD 0006

└ `taler:// actions`

`withdraw` bank-initiated withdrawal
`pay` merchant-initiated payment
`refund` merchant-initiated refund
`pay-push` P2P payment
`pay-pull` P2P invoice
`pay-template` merchant offline payment
`restore` restore from backup
`withdraw-exchange` wallet-initiated withdrawal

1. This is a list of the actions currently supported by the Taler wallet.
2. Note that opening a Taler wallet with the respective URL must not automatically perform anything, the user MUST be asked to consent to a particular operation.
3. An exception could be “refunds” — who would refuse to get money back?

References I

-  T. Berners-Lee, L. Masinter, and M. McCahill.
Uniform Resource Locators (URL).
RFC 1738 (Proposed Standard), December 1994.
Obsoleted by RFCs 4248, 4266, updated by RFCs 1808, 2368, 2396,
3986, 6196, 6270, 8089.

2024-08-26



NEXT GENERATION INTERNET

└References



└References

 T. Berners-Lee, L. Masinter, and M. McCahill.
Uniform Resource Locators (URL).
RFC 1738 (Proposed Standard), December 1994.
Obsoleted by RFCs 4248, 4266, updated by RFCs 1808, 2368, 2396,
3986, 6196, 6270, 8089.

References II

-  R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee.
Hypertext Transfer Protocol – HTTP/1.1.
RFC 2616 (Draft Standard), June 1999.
Obsoleted by RFCs 7230, 7231, 7232, 7233, 7234, 7235, updated by RFCs 2817, 5785, 6266, 6585.
-  P. Hoffman, L. Masinter, and J. Zawinski.
The mailto URL scheme.
RFC 2368 (Proposed Standard), July 1998.
Obsoleted by RFC 6068.

References III

-  A. Melnikov and C. Newman.
IMAP URL Scheme.
RFC 5092 (Proposed Standard), November 2007.
Updated by RFC 5593.
-  R. Moats.
URN Syntax.
RFC 2141 (Proposed Standard), May 1997.
Obsoleted by RFC 8141.

2024-08-26

NEXT GENERATION INTERNET



└References

└References

 A. Melnikov and C. Newman.
IMAP URL Scheme.
RFC 5092 (Proposed Standard), November 2007.
Updated by RFC 5593.

 R. Moats.
URN Syntax.
RFC 2141 (Proposed Standard), May 1997.
Obsoleted by RFC 8141.

References IV

-  E. Rescorla.
HTTP Over TLS.
RFC 2818 (Informational), May 2000.
Updated by RFCs 5785, 7230.
-  M. Smith and T. Howes.
Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator.
RFC 4516 (Proposed Standard), June 2006.

2024-08-26

NEXT GENERATION INTERNET

└References

└References

 E. Rescorla.
HTTP Over TLS.
RFC 2818 (Informational), May 2000.
Updated by RFCs 5785, 7230.

 M. Smith and T. Howes.
Lightweight Directory Access Protocol (LDAP): Uniform Resource Locator.
RFC 4516 (Proposed Standard), June 2006.

References V

-  D. Thaler, T. Hansen, and T. Hardie.
Guidelines and Registration Procedures for URI Schemes.
RFC 7595 (Best Current Practice), June 2015.

2024-08-26

NEXT GENERATION INTERNET

└References

└References

D. Thaler, T. Hansen, and T. Hardie.
Guidelines and Registration Procedures for URI Schemes.
RFC 7595 (Best Current Practice), June 2015.

Acknowledgements


Co-funded by the European Union (Project 101135475).



Co-funded by
the European Union

Co-funded by SERI (HEU-Projekt 101135475-TALER).

Project funded by

 Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs,
Education and Research EAER
**State Secretariat for Education,
Research and Innovation SERI**

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Secure Integration

Christian Grothoff

Using URIs

1. Write a tokenizer for RFC 8905 URIs

- The tokenizer must reject malformed URIs
- Output the payment method and target account
- Output standardized optional arguments (amount, message, instruction, receiver, sender)
- Validate target accounts in at least one registered schema (ach, bic, iban, etc., but not void)

Test vectors:

```
payto://iban/DE75512108001245126199?amount=EUR:200.0&message=hello
payto://iban/SANDBOXX/DE75512108001245126198?receiver-name=Dude
payto://bitcoin/12A1MyfXbW6RhdRAZEqofac5jCQQjwEPBu?amount=BTC:0,42
payto://bitcoin/tb1qxzjp3xmdk6ghyddpjlfmj06d9pwp9jptq5c6zt
```

Which one(s) of the above are valid?

2. `payto://` URIs are not always normalized!

For some applications (like duplicate detection) it is desirable for URIs to be normalized, that is byte-by-byte unique for the same target. Ignoring the optional arguments (which are obviously going to vary for different transactions), point out where `payto://` URIs are not normalized.

NEXT GENERATION INTERNET

Anonymity

Christian Grothoff

31.05.2024

2024-08-26

NEXT GENERATION INTERNET

NEXT
GENERATION
INTERNET
Anonymity

Christian Grothoff

31.05.2024

Learning Objectives

What is Anonymity?

How can we achieve anonymity on the Internet?

How does onion routing work?

2024-08-26

NEXT GENERATION INTERNET

└ Learning Objectives

[What is Anonymity?](#)

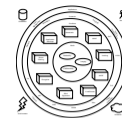
[How can we achieve anonymity on the Internet?](#)

[How does onion routing work?](#)

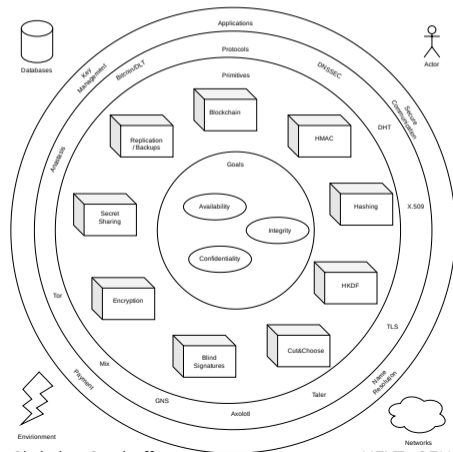
Anonymity on our map

2024-08-26

NEXT GENERATION INTERNET



└ Anonymity on our map



1. Anonymity is a special case of confidentiality, where the identity needs to remain confidential. Mix networks and blind signatures are the main constructions used to achieve anonymity.
2. Key applications are in the domains of communication (Tor) and payment (Taler).
3. Related topics are secure multi-party computation and cut&choose.

What is Anonymity?

Motivation



Suppose Alice and Bob communicate using encryption.

What can Eve still learn here?

2024-08-26

NEXT GENERATION INTERNET

└ What is Anonymity?

└ Motivation



Suppose Alice and Bob communicate using encryption.
What can Eve still learn here?

1. Eve cannot read the data Alice and Bob are sending
2. Eve knows that Alice and Bob are communicating.
3. Eve knows the amount of data they are sending and can observe patterns.
4. \Rightarrow Patterns may even allow Eve to figure out the data!
5. For example, Bob the Webserver may send a page which is fingerprinted in having 13kB of data, and 13 included objects with size from 2kB to 117kB.

How much does TLS leak?

“We present a traffic analysis attack against over 6000 webpages spanning the HTTPS deployments of 10 widely used, industry-leading websites in areas such as healthcare, finance, legal services and streaming video. Our attack **identifies individual pages** in the same website with 89% accuracy, exposing personal details including **medical conditions**, financial and **legal affairs** and **sexual orientation**. We examine evaluation methodology and reveal accuracy variations as large as 18% caused by assumptions affecting caching and cookies.” [11]

2024-08-26

NEXT GENERATION INTERNET

└─What is Anonymity?

└─How much does TLS leak?

“We present a traffic analysis attack against over 6000 webpages spanning the HTTPS deployments of 10 widely used, industry-leading websites in areas such as healthcare, finance, legal services and streaming video. Our attack **identifies individual pages** in the same website with 89% accuracy, exposing personal details including **medical conditions**, financial and **legal affairs** and **sexual orientation**. We examine evaluation methodology and reveal accuracy variations as large as 18% caused by assumptions affecting caching and cookies.” [11]

1. This is a research paper from 2014.
2. The more unique external resources a site includes, the worse the attacks get.
3. Caches and click-streams can be simulated, enhancing the efficacy of the attack.
4. ⇒ TLS is basically useless for confidentiality when reading public sites.

Anonymity definitions

Merriam-Webster:

1. not named or identified: “an anonymous author”, “they wish to remain anonymous”
2. of unknown authorship or origin: “an anonymous tip”
3. lacking individuality, distinction, or recognizability: “the anonymous faces in the crowd”, “the gray anonymous streets” – William Styron

2024-08-26

NEXT GENERATION INTERNET

└ What is Anonymity?

└ Anonymity definitions

Merriam-Webster:
1. not named or identified: “an anonymous author”, “they wish to remain anonymous”
2. of unknown authorship or origin: “an anonymous tip”
3. lacking individuality, distinction, or recognizability: “the anonymous faces in the crowd”, “the gray anonymous streets” – William Styron

1. This gives us an idea of use-cases, but not a technically operational definition.

Anonymity definitions

Andreas Pfitzmann et. al.:

“Anonymity is the state of being not identifiable within a set of subjects, the anonymity set.”

2024-08-26

NEXT GENERATION INTERNET

└─What is Anonymity?

└─Anonymity definitions

Andreas Pfitzmann et. al.:
“Anonymity is the state of being not identifiable within a set of subjects, the anonymity set.”

1. Pfitzmann’s definition hints at a core issue: anonymity requires a group
2. The EFF definition points to a related concept: Pseudonymity
3. This is a better definition for information security: we have an adversary, and the adversary has an objective.
4. Also, the set of people that could have performed the action gives us the group: the anonymity set

Anonymity definitions

Andreas Pfitzmann et. al.:

“Anonymity is the state of being not identifiable within a set of subjects, the anonymity set.”

EFF:

“Instead of using their true names to communicate, (...) people choose to speak using pseudonyms (assumed names) or anonymously (no name at all).”

2024-08-26

NEXT GENERATION INTERNET

└─What is Anonymity?

└─Anonymity definitions

Andreas Pfitzmann et. al.:

“Anonymity is the state of being not identifiable within a set of subjects, the anonymity set.”

EFF:

“Instead of using their true names to communicate, (...) people choose to speak using pseudonyms (assumed names) or anonymously (no name at all).”

1. Pfitzmann’s definition hints at a core issue: anonymity requires a group
2. The EFF definition points to a related concept: Pseudonymity
3. This is a better definition for information security: we have an adversary, and the adversary has an objective.
4. Also, the set of people that could have performed the action gives us the group: the anonymity set

Anonymity definitions

Andreas Pfitzmann et. al.:

“Anonymity is the state of being not identifiable within a set of subjects, the anonymity set.”

EFF:

“Instead of using their true names to communicate, (...) people choose to speak using pseudonyms (assumed names) or anonymously (no name at all).”

Our definition:

A user's action is anonymous if the adversary cannot link the action to the user's identity

2024-08-26

NEXT GENERATION INTERNET

└ What is Anonymity?

└ Anonymity definitions

Andreas Pfitzmann et. al.:

“Anonymity is the state of being not identifiable within a set of subjects, the anonymity set.”

EFF:

“Instead of using their true names to communicate, (...) people choose to speak using pseudonyms (assumed names) or anonymously (no name at all).”

Our definition:

A user's action is anonymous if the adversary cannot link the action to the user's identity

1. Pfitzmann's definition hints at a core issue: anonymity requires a group
2. The EFF definition points to a related concept: Pseudonymity
3. This is a better definition for information security: we have an adversary, and the adversary has an objective.
4. Also, the set of people that could have performed the action gives us the group: the anonymity set

The user's identity

includes personally identifiable information, such as:

- ▶ real name
- ▶ fingerprint
- ▶ passport number
- ▶ IP address
- ▶ MAC address
- ▶ login name
- ▶ ...

2024-08-26

NEXT GENERATION INTERNET

└ What is Anonymity?

└ The user's identity

includes personally identifiable information, such as:

- ▶ real name
- ▶ fingerprint
- ▶ passport number
- ▶ IP address
- ▶ MAC address
- ▶ login name
- ▶ ...

1. What is considered “personally identifiable information” ultimately depends on the specific application.

include:

- ▶ Internet access
- ▶ speech
- ▶ participation in demonstration
- ▶ purchase in a store
- ▶ walking across the street
- ▶ ...

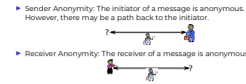
1. Similarly, the “action” depends on the specific application.

Anonymity: Terminology

- ▶ Sender Anonymity: The initiator of a message is anonymous. However, there may be a path back to the initiator.



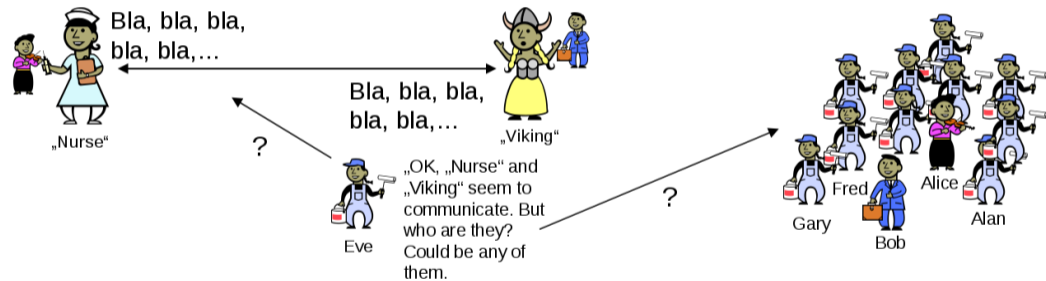
- ▶ Receiver Anonymity: The receiver of a message is anonymous.



1. The most common type of “action” considered is sending a message.
2. In this case, the question is who is anonymous: the sender or the receiver?
3. If you wonder *how* a receiver could be anonymous, think about a global broadcast!

Pseudonymity

A pseudonym is an alternative name for an entity in the system.



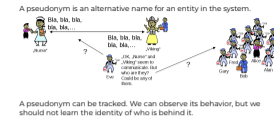
A pseudonym can be tracked. We can observe its behavior, but we should not learn the identity of who is behind it.

2024-08-26

NEXT GENERATION INTERNET

└ What is Anonymity?

└ Pseudonymity



1. “Nurse” is always “Nurse” while using the system.
2. Nobody, but (maybe) a trusted party may be able to link a pseudonym to the true identity of the holder of the pseudonym.
3. Pseudonymity is very hard to maintain: over time, an attacker can build up an extensive profile of linked actions that improves their ability to link the pseudonym to the user’s real identity.

Evaluating anonymity

How much anonymity does a given system provide?

- ▶ Number of known attacks?
- ▶ Lowest complexity of successful attacks?
- ▶ Number of users?
- ▶ Information leaked through messages and maintenance procedures?

2024-08-26

NEXT GENERATION INTERNET

└─What is Anonymity?

└─Evaluating anonymity

How much anonymity does a given system provide?

- ▶ Number of known attacks?
- ▶ Lowest complexity of successful attacks?
- ▶ Number of users?
- ▶ Information leaked through messages and maintenance procedures?

1. Counting attacks would give us the illusion that an unpopular system is secure.
2. Comparing attacks only works if we know successful attacks, which kind-of means the system is already not secure.
3. Given the need for a group, the number of users is actually a good indicator, but it unduly focuses on usability over cryptographic security; clear-text Web/E-mail have the most users!
4. This is usually a metric favored by cryptographers, but it unduly focuses on cryptographic security over usability; a messaging system with zero information leakage but only two users still provides no privacy at all!

Anonymity: Basics

- ▶ **Anonymity Set** is the set of suspects
- ▶ Attacker computes a **probability distribution** describing the likelihood of each participant to be the responsible party.
- ▶ The larger the anonymity set is and the more evenly distributed the subjects within that set are, the stronger the anonymity is.

2024-08-26

NEXT GENERATION INTERNET

└ What is Anonymity?

└ Anonymity: Basics

- ▶ **Anonymity Set** is the set of suspects
- ▶ Attacker computes a **probability distribution** describing the likelihood of each participant to be the responsible party.
- ▶ The larger the anonymity set is and the more evenly distributed the subjects within that set are, the stronger the anonymity is.

1. The Anonymity Set formalizes the notion of the number of users of the system.
2. Moreover, it introduces us to a key concept, namely that the adversary tries to improve a probability distribution. After all, usually suspicion is not binary or perfectly equally distributed.

Anonymity metric: Anonymity Set Size

Let \mathcal{U} be the attacker's probability distribution and $p_u = \mathcal{U}(u)$ describing the probability that user $u \in \Psi$ is responsible.

$$ASS := \sum_{\substack{u \in \Psi \\ p_u > 0}} 1 \quad (1)$$

1. The ASS is the simplest metric we have for anonymity.
2. We simply count the number of suspects.
3. Note that individuals with a probability of exactly zero are not suspects.

Large anonymity sets

Examples of large anonymity sets:

- ▶ Any human

2024-08-26

NEXT GENERATION INTERNET

└ What is Anonymity?

└ Large anonymity sets

Examples of large anonymity sets:

- ▶ Any human

1. That's still not perfect, ideally your dog actually ate your homework. So the best privacy would be if aliens are also legitimate suspects.
2. Realistically speaking, in the application domains we care about for cryptography, we kind-of require humans with Internet access.
3. Now, most interactions require processing text in some language, so this is another common restriction.
4. In practice, our anonymity set is going to be an intersection of various hard filters that often arise from the action. Note that there could be technical solutions. For example, "at 3am CEST" might not be exposed if the system has really high latency or if we teach users to not use the system when most other suspects are asleep. Similarly, using translation software may lift restrictions on the language spoken by the user.

Large anonymity sets

Examples of large anonymity sets:

- ▶ Any human
- ▶ Any human with Internet access

2024-08-26

NEXT GENERATION INTERNET

└ What is Anonymity?

└ Large anonymity sets

Examples of large anonymity sets:
▶ Any human
▶ Any human with Internet access

1. That's still not perfect, ideally your dog actually ate your homework. So the best privacy would be if aliens are also legitimate suspects.
2. Realistically speaking, in the application domains we care about for cryptography, we kind-of require humans with Internet access.
3. Now, most interactions require processing text in some language, so this is another common restriction.
4. In practice, our anonymity set is going to be an intersection of various hard filters that often arise from the action. Note that there could be technical solutions. For example, "at 3am CEST" might not be exposed if the system has really high latency or if we teach users to not use the system when most other suspects are asleep. Similarly, using translation software may lift restrictions on the language spoken by the user.

Large anonymity sets

Examples of large anonymity sets:

- ▶ Any human
- ▶ Any human with Internet access
- ▶ Any human speaking German

2024-08-26

NEXT GENERATION INTERNET

└ What is Anonymity?

└ Large anonymity sets

Examples of large anonymity sets:

- ▶ Any human
- ▶ Any human with Internet access
- ▶ Any human speaking German

1. That's still not perfect, ideally your dog actually ate your homework. So the best privacy would be if aliens are also legitimate suspects.
2. Realistically speaking, in the application domains we care about for cryptography, we kind-of require humans with Internet access.
3. Now, most interactions require processing text in some language, so this is another common restriction.
4. In practice, our anonymity set is going to be an intersection of various hard filters that often arise from the action. Note that there could be technical solutions. For example, "at 3am CEST" might not be exposed if the system has really high latency or if we teach users to not use the system when most other suspects are asleep. Similarly, using translation software may lift restrictions on the language spoken by the user.

Large anonymity sets

Examples of large anonymity sets:

- ▶ Any human
- ▶ Any human with Internet access
- ▶ Any human speaking German
- ▶ Any human speaking German with Internet access awake at 3am CEST

2024-08-26

NEXT GENERATION INTERNET

└ What is Anonymity?

└ Large anonymity sets

Examples of large anonymity sets:

- ▶ Any human
- ▶ Any human with Internet access
- ▶ Any human speaking German
- ▶ Any human speaking German with Internet access awake at 3am CEST

1. That's still not perfect, ideally your dog actually ate your homework. So the best privacy would be if aliens are also legitimate suspects.
2. Realistically speaking, in the application domains we care about for cryptography, we kind-of require humans with Internet access.
3. Now, most interactions require processing text in some language, so this is another common restriction.
4. In practice, our anonymity set is going to be an intersection of various hard filters that often arise from the action. Note that there could be technical solutions. For example, "at 3am CEST" might not be exposed if the system has really high latency or if we teach users to not use the system when most other suspects are asleep. Similarly, using translation software may lift restrictions on the language spoken by the user.

Anonymity metric: Maximum Likelihood

Let \mathcal{U} be the attacker's probability distribution describing the probability that user $u \in \Psi$ is responsible.

$$ML := \max_{u \in \Psi} p_u \quad (2)$$

2024-08-26

NEXT GENERATION INTERNET

└ What is Anonymity?

└ Anonymity metric: Maximum Likelihood

Let \mathcal{U} be the attacker's probability distribution describing the probability that user $u \in \Psi$ is responsible.

$$ML := \max_{u \in \Psi} p_u \quad (2)$$

1. The anonymity set size obviously ignores the specific probabilities in the distribution.
2. With ML, we focus on the main suspect, the one with the highest probability.
3. That is useful, as especially in court, you want your probability to be low.

Anonymity metric: Maximum Likelihood

- ▶ For successful criminal prosecution in the US, the law requires ML close to 1 (“beyond reasonable doubt”)
- ▶ For successful civil prosecution in the US, the law requires $ML > \frac{1}{2}$ (“more likely than not”)
- ▶ For a given anonymity set, the best anonymity is achieved if

$$ML = \frac{1}{ASS} \quad (3)$$

2024-08-26

NEXT GENERATION INTERNET

└ What is Anonymity?

└ Anonymity metric: Maximum Likelihood

- ▶ For successful criminal prosecution in the US, the law requires ML close to 1 (“beyond reasonable doubt”)
- ▶ For successful civil prosecution in the US, the law requires $ML > \frac{1}{2}$ (“more likely than not”)
- ▶ For a given anonymity set, the best anonymity is achieved if

$$ML = \frac{1}{ASS}$$

(3)

1. Note that what “close” to 1 is, is not precisely defined. It could be 90% or 95% or 99.9% depending on what the court thinks is reasonable.

Anonymity metric: Entropy

Let \mathcal{U} be the attacker's probability distribution describing the probability that user $u \in \Psi$ is responsible. Define the effective size S of the anonymity distribution \mathcal{U} to be:

$$S := - \sum_{u \in \Psi} p_u \log_2 p_u \quad (4)$$

where $p_u = \mathcal{U}(u)$.

└ What is Anonymity?

└ Anonymity metric: Entropy

Let \mathcal{U} be the attacker's probability distribution describing the probability that user $u \in \Psi$ is responsible. Define the effective size S of the anonymity distribution \mathcal{U} to be:

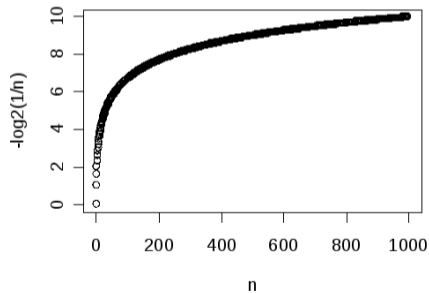
$$S := - \sum_{u \in \Psi} p_u \log_2 p_u \quad (4)$$

where $p_u = \mathcal{U}(u)$.

1. This is the *expected* number of bits of additional information that the attacker needs to definitely identify the user (with absolute certainty).
2. Comparing the entropy before and after an operation allows us to quantify precisely the amount of information disclosed in that operation: How much did the attacker learn?
3. In a perfect system, the information disclosed would be zero and so the entropy of the probability distribution would remain the same. If the entropy drops to zero, a suspect was proven to be guilty.

Interpretation of entropy

$$S = - \sum_{u \in \Psi} p_u \log_2 p_u \quad (5)$$

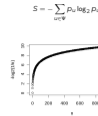


2024-08-26

NEXT GENERATION INTERNET

└ What is Anonymity?

└ Interpretation of entropy



(5)

1. The plot shows the maximum entropy possible for a given anonymity set size, that is the entropy of a perfectly balanced distribution.
2. We can also read it as an equivalence: if your entropy is 10, the attacker would need as much information as they would need to determine an individual out of 1024 without any other information, in other words, 10 bits of “pure” information.

Entropy calculation example

Suppose we have 101 suspects including Bob. Furthermore, suppose for Bob the attacker has a probability of 0.9 and for all the 100 other suspects the probability is 0.001.

What is S ?

2024-08-26

NEXT GENERATION INTERNET

└ What is Anonymity?

└ Entropy calculation example

Suppose we have 101 suspects including Bob. Furthermore, suppose for Bob the attacker has a probability of 0.9 and for all the 100 other suspects the probability is 0.001.

What is S ?

1. This example shows how entropy might be misleading.
2. Here, we have an entropy of above 1, so one might think that there are two suspects with about equal probability.
3. However, in a criminal system, the one suspect with 90% probability stands out so much that they would prosecute her and she might be found guilty — assuming the courts are willing to risk putting 10% of innocent people behind bars.

Entropy calculation example

Suppose we have 101 suspects including Bob. Furthermore, suppose for Bob the attacker has a probability of 0.9 and for all the 100 other suspects the probability is 0.001.

What is S ?

▶ For 101 nodes $H_{max} = 6.7$

▶

$$S = -\frac{100 \cdot \log_2 0.001}{1000} - \frac{9 \cdot \log_2 0.9}{10} \quad (6)$$

$$\approx 0.9965 + 0.1368 \quad (7)$$

$$= 1.133... \quad (8)$$

2024-08-26

NEXT GENERATION INTERNET

└ What is Anonymity?

└ Entropy calculation example

Suppose we have 101 suspects including Bob. Furthermore, suppose for Bob the attacker has a probability of 0.9 and for all the 100 other suspects the probability is 0.001.

What is S ?

▶ For 101 nodes $H_{max} = 6.7$

▶

$$S = -\frac{100 \cdot \log_2 0.001}{1000} - \frac{9 \cdot \log_2 0.9}{10} \quad (6)$$
$$\approx 0.9965 + 0.1368 \quad (7)$$
$$= 1.133... \quad (8)$$

1. This example shows how entropy might be misleading.
2. Here, we have an entropy of above 1, so one might think that there are two suspects with about equal probability.
3. However, in a criminal system, the one suspect with 90% probability stands out so much that they would prosecute her and she might be found guilty — assuming the courts are willing to risk putting 10% of innocent people behind bars.

Attacks to avoid

Hopeless situations include:

- ▶ All nodes collaborate against the user
- ▶ All directly adjacent nodes collaborate
- ▶ All non-collaborating adjacent nodes are made unreachable from the user
- ▶ The user is required to prove her innocence

2024-08-26

NEXT GENERATION INTERNET

└─What is Anonymity?

└─Attacks to avoid

Hopeless situations include:

- ▶ All nodes collaborate against the user
- ▶ All directly adjacent nodes collaborate
- ▶ All non-collaborating adjacent nodes are made unreachable from the user
- ▶ The user is required to prove her innocence

1. As with any secure protocol, we also need a threat model for systems that are to provide anonymity.
2. When designing such threat models, we must avoid threat models that are impossible to address.
3. After all, the user must have actually *done* the action that they want privacy for, and the goal is to hide within a group. So a group to hide in must exist.
4. In the real world actual users may face such threats, but here their case is *technically* hopeless.

There are hard issues in *the Economics of Anonymity* [1]:

- ▶ Providing anonymity services has economic disincentives (DoS, legal liability)
- ▶ Anonymity requires introducing inefficiencies!
- ⇒ Who pays for that?

1. The anonymizing server that has the best reputation (performance, most traffic) is presumably operated by an adversary.
2. P. Syverson, one of the authors of [1], works for the “Naval Research Lab”. Is that the real employer?
3. Paying for anonymity is also tricky, as the payment itself may leave a stronger trail of evidence compared to not using an anonymization service at all.

How can we achieve anonymity on the Internet?

Anonymity: Dining Cryptographers

“Three cryptographers are sitting down to dinner. The waiter informs them that the bill will be paid anonymously. One of the cryptographers maybe paying for dinner, or it might be the NSA. The three cryptographers respect each other’s right to make an anonymous payment, but they wonder if the NSA is paying.” – David Chaum

2024-08-26

NEXT GENERATION INTERNET

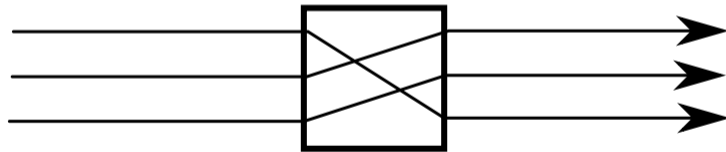
└ How can we achieve anonymity on the Internet?

└ Anonymity: Dining Cryptographers

“Three cryptographers are sitting down to dinner. The waiter informs them that the bill will be paid anonymously. One of the cryptographers maybe paying for dinner, or it might be the NSA. The three cryptographers respect each other’s right to make an anonymous payment, but they wonder if the NSA is paying.” – David Chaum

1. All cryptographers then perform a pair-wise secret coin flip.
2. They then publicly announce whether the two coin flips they witnessed matched, except if one of them paid, they lie about the match.
3. The XOR sum of all announcements is odd, a cryptographer paid; if even, the NSA paid.
4. The protocol can be extended to more participants, offers perfect anonymity for the given anonymity set, but does not scale at all.

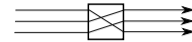
David Chaum's mix (1981) and cascades of mixes are the traditional basis for destroying linkability:



└ How can we achieve anonymity on the Internet?

└ Mixing

David Chaum's mix (1981) and cascades of mixes are the traditional basis for destroying linkability:



1. The traffic on all incoming and outgoing links must be encrypted and use fixed-sized messages.
2. The mix itself must re-encrypt the traffic, so the ciphertext looks completely different on both ends.
3. This way, an observer cannot correlate incoming and outgoing data.

David Chaum's mix (1981) and cascades of mixes are the traditional basis for destroying linkability:



└ How can we achieve anonymity on the Internet?

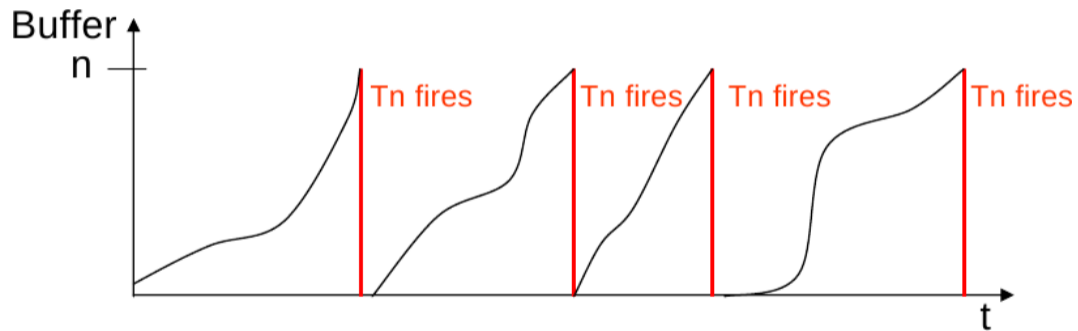
└ Mixing

David Chaum's mix (1981) and cascades of mixes are the traditional basis for destroying linkability:



1. Most critically, the mix must operate as a black box: how it shuffles the messages must be secret!
2. In practice, not all messages will arrive at exactly the same time.
3. There are various ways to group messages and to decide when to forward.

Threshold Mix

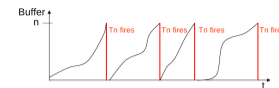


2024-08-26

NEXT GENERATION INTERNET

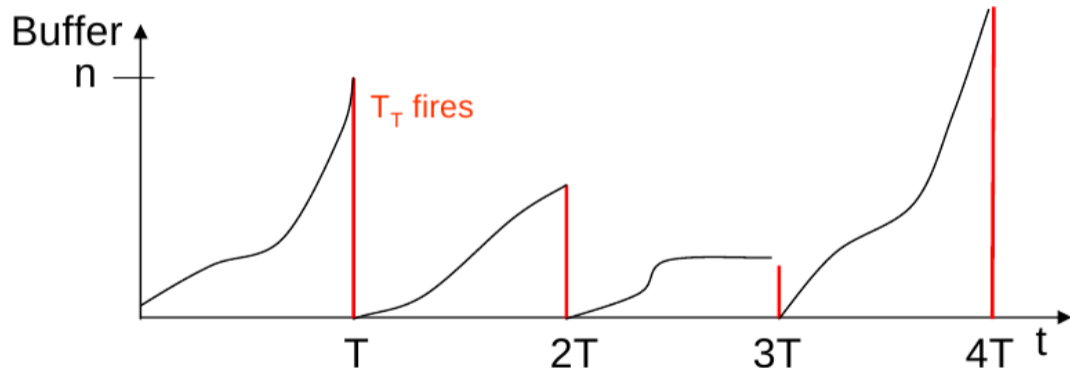
└ How can we achieve anonymity on the Internet?

└ Threshold Mix



1. A Threshold Mix simply waits until N messages have been received, then randomizes their order, and then sends all of them.
2. This guarantees mixing with N other messages and thus a certain anonymity set size.
3. Practical problem: unpredictable latency
4. Attack: Adversary sends $N - 1$ messages first with known destinations, thus the remaining message's route is revealed.

Timed Mix

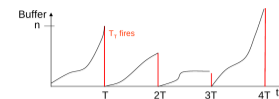


2024-08-26

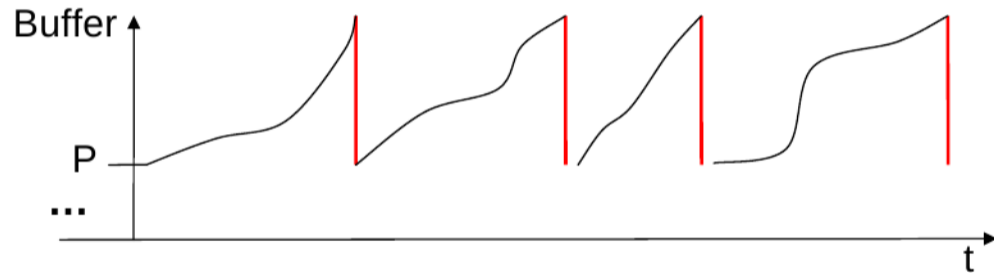
NEXT GENERATION INTERNET

How can we achieve anonymity on the Internet?

Timed Mix

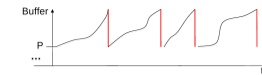


1. A Timed Mix simply waits until T time has passed, then randomized the messages received so far, and then sends all of them.
2. This guarantees timely delivery.
3. Practical problem: storage usage by the Mix only limited by bandwidth and T
4. Attack: anonymity set size could be really small, $N - 1$ attack also still applies.



└ How can we achieve anonymity on the Internet?

└ Pool mix



1. A Pool Mix waits until some maximum number of messages have been buffered, then randomizes the messages, and sends all but P of them.
2. This makes it much harder to mount a $N - 1$ attack as some messages from previous iterations are always in the pool.
3. Message delivery times are even more unpredictable, as a message may remain in the pool for many iterations by chance.
4. More variations exist,
5. A single mix could be compromised, so practical designs often combine multiple mix nodes into a mix network or a mix cascade.
6. The entire concept is deceptively simple: it has many, many caveats and implementing an actually reasonably secure and performant mix network remains an open challenge.

G. Danezis, R. Dingledine, D. Hopwood and N. Mathewson describe Mixminion [3]:

- ▶ builds on the idea of remailers: Mixes for E-mail
- ▶ possibility to reply
- ▶ directory servers to evaluate participating remailers (reputation system)
- ▶ exit policies
- ▶ dummy traffic

└ How can we achieve anonymity on the Internet?

└ Mixminion

G. Danezis, R. Dingledine, D. Hopwood and N. Mathewson describe Mixminion [3]:

- ▶ builds on the idea of remailers: Mixes for E-mail
- ▶ possibility to reply
- ▶ directory servers to evaluate participating remailers (reputation system)
- ▶ exit policies
- ▶ dummy traffic

1. As we have seen, Mixes inherently introduce high and variable latencies. Thus, messaging is a reasonable application domain for Mix networks.
2. The core function is to anonymize the sender. This could be a whistleblower contacting a journalist.
3. While Mixminion was never widely used, the key design ideas are still interesting.
4. Key features include the ability to respond to the anonymous sender, the use of a reputation system to keep remailers honest, exit policies to limit abuse, as well as dummy traffic and link padding to hide volume.

Mixminion: key ideas

When a message traverses Mixminion, each node must decrypt the message using its (ephemeral) private key.

The key idea behind **replies** is splitting the path into two legs:

- ▶ the first half is chosen by the responder to hide the responder's identity
- ▶ the second half was communicated by the receiver to hide the receiver identity
- ▶ a crossover-node in the middle is used to switch the headers specifying the path

2024-08-26

NEXT GENERATION INTERNET

└ How can we achieve anonymity on the Internet?

└ Mixminion: key ideas

When a message traverses Mixminion, each node must decrypt the message using its (ephemeral) private key.

The key idea behind **replies** is splitting the path into two legs:

- ▶ the first half is chosen by the responder to hide the responder's identity
- ▶ the second half was communicated by the receiver to hide the receiver identity
- ▶ a crossover-node in the middle is used to switch the headers specifying the path

1. The key feature is the ability to send replies.
2. In a traditional Mix network, the sender chooses a path through the network with layered encryption to hide their identity.
3. With Mixminion, an anonymous receiver makes a **reply block** available that allows other to send messages to them. The reply block specifies a path back to the receiver that is also protected by layered encryption, forcing a message through a specific sequence of mixes to reach the receiver.
4. Reply blocks are distributed as part of a previous message or via "Nym" servers.

Mixminion: replay?

Replay attacks were an issue in previous mixnet implementations.

- ▶ Mixes are vulnerable to replay attacks
- ▶ Mixminion: servers keep hash of previously processed messages until the server key is rotated
- ⇒ Bounded amount of state in the server, no possibility for replay attack due to key rotation

2024-08-26

NEXT GENERATION INTERNET

└ How can we achieve anonymity on the Internet?

└ Mixminion: replay?

Replay attacks were an issue in previous mixnet implementations.

- ▶ Mixes are vulnerable to replay attacks
- ▶ Mixminion: servers keep hash of previously processed messages until the server key is rotated
- ⇒ Bounded amount of state in the server, no possibility for replay attack due to key rotation

1. In a replay attack, an attacker sends the same message (such as a reply using a replay block) repeatedly and observes the resulting messages on the Internet.
2. As the replayed message will always use the same path, they can discard other traffic that follows different patterns, and repeat the process until they have fully identified the path to the anonymous recipient.
3. To prevent replay attacks, mixes must keep track of all messages they have seen previously, and never transmit such a message a second time.
4. This creates a conflict between not being able to process old messages because keys were already rotated, and keeping excessive state.

Mixminion: Directory Servers

- ▶ Inform users about servers
- ▶ Probe servers for reliability
- ▶ Allow a partitioning attack unless the user always queries all directory servers for everything

2024-08-26

NEXT GENERATION INTERNET

└ How can we achieve anonymity on the Internet?

└ Mixminion: Directory Servers

- ▶ Inform users about servers
- ▶ Probe servers for reliability
- ▶ Allow a partitioning attack unless the user always queries all directory servers for everything

1. If a Mix drops a message, the user is forced to either eventually send the message again (building a new path, possibly by chance picking more adversary-controlled mixes on the path) or even to abandon the mix network and trading privacy for availability.
2. Which Mix dropped the message is not clear to the user. In fact, that a Mix dropped a message (instead of the recipient merely not answering) is undetectable.
3. Mixminion directory servers send probing messages to identify unreliable mixes and provide the statistics to clients doing path selection.
4. Directory servers may disagree on mix performance, so to ensure all users choose paths with the *same* bias, all users must always query all directory servers to get the same statistical inputs into their path selection.

Mixminion: Nymserver

- ▶ Nymserver keep list of use-once reply blocks for a user
- ▶ Vulnerable to DoS attacks (deplete reply blocks)
- ▶ Nymserver could also store mail (use one reply block for many messages).

2024-08-26

NEXT GENERATION INTERNET

└ How can we achieve anonymity on the Internet?

└ Mixminion: Nymserver

- ▶ Nymserver keep list of use-once reply blocks for a user
- ▶ Vulnerable to DoS attacks (deplete reply blocks)
- ▶ Nymserver could also store mail (use one reply block for many messages).

1. Reply blocks are use-once, as we need to protect against replay attacks.
2. Nymserver allow (anonymous!) users to obtain reply blocks for *pseudonymous* recipients.
3. Storing mail can address the depletion issue, but may then create a storage issue.

Mixminion: obvious problems

- ▶ no benefits for running a mixmailer for the operator
- ▶ quite a bit of public key cryptography
- ▶ trustworthiness of directory servers questionable
- ▶ servers must keep significant (but bounded) amount of state
- ▶ limited to E-mail (high latency)

2024-08-26

NEXT GENERATION INTERNET

└ How can we achieve anonymity on the Internet?

└ Mixminion: obvious problems

- ▶ no benefits for running a mixmailer for the operator
- ▶ quite a bit of public key cryptography
- ▶ trustworthiness of directory servers questionable
- ▶ servers must keep significant (but bounded) amount of state
- ▶ limited to E-mail (high latency)

1. The general issue with economics of anonymity remains an open issue.
2. At every hop every message needs a public key operation. For small chat messages, this can be a bottleneck.
3. The security of the system is rooted in that the directory server operators are honest and do not direct the user towards adversary-controlled mixes.

Mixminion: open problems

- ▶ exit nodes are fair game for legal actions
 - ▶ no accounting to defend against abuse / DoS attacks
 - ▶ statistical correlation of entities communicating over time possible (observe participation)
- ⇒ bridging between an anonymous network and a traditional protocol is difficult

Subsequent remailer research has focused on improving the cryptography [4, 12] and integrating economic incentives for operators [5].

<https://nymtech.com/> and <https://github.com/katzenpost/katzenpost> are modern examples.

2024-08-26

NEXT GENERATION INTERNET

└ How can we achieve anonymity on the Internet?

└ Mixminion: open problems

▶ exit nodes are fair game for legal actions
▶ no accounting to defend against abuse / DoS attacks
▶ statistical correlation of entities communicating over time possible (observe participation)
⇒ bridging between an anonymous network and a traditional protocol is difficult
Subsequent remailer research has focused on improving the cryptography [4, 12] and integrating economic incentives for operators [5].
<https://nymtech.com/> and <https://github.com/katzenpost/katzenpost> are modern examples.

1. Exit policies allow a Mix to limit where they will deliver traffic, including only local delivery. In practice, this means only Mix operators themselves are reachable, as open exits that relay everywhere quickly end up on spam lists and are then filtered before reaching most recipients.
2. As the sender is anonymous, there is no accounting, and malicious users could send arbitrary amounts of messages.
3. You cannot deploy privacy-enhancing technology only at the time of need, as joining the network is observable and might in itself make you a suspect. Similarly, if a pair of users communicate repeatedly, their traffic entering and exiting the Mix network may stand out (unless using only local delivery).

How does onion routing work?

Onion Routing

- ▶ Multiple mix servers
- ▶ Path of mix servers chosen by initiator
- ▶ Chosen mix servers create “circuit”
 - ▶ Initiator contacts first server S_1 , sets up symmetric key K_{S_1}
 - ▶ Then asks first server to connect to second server S_2 ; through this connection sets up symmetric key with second server K_{S_2}
 - ▶ ...
 - ▶ Repeat with server S_i until circuit of desired length n constructed

2024-08-26

NEXT GENERATION INTERNET

└ How does onion routing work?

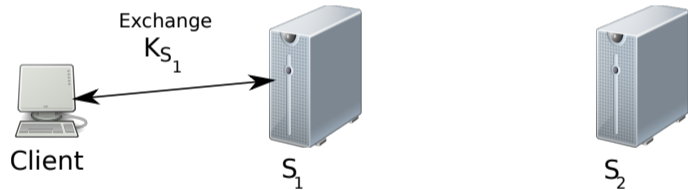
└ Onion Routing

1. Using a single mix is insecure, so have many and let the initiator choose a path.
2. Mixminion's use of public-key cryptography at every hop is expensive.
3. Onion routing instead uses public-key cryptography to establish a circuit, and then afterwards symmetric cryptography for the actual payload.

- ▶ Multiple mix servers
- ▶ Path of mix servers chosen by initiator
- ▶ Chosen mix servers create “circuit”
 - ▶ Initiator contacts first server S_1 , sets up symmetric key K_{S_1}
 - ▶ Then asks first server to connect to second server S_2 ; through this connection sets up symmetric key with second server K_{S_2}
 - ▶ ...
 - ▶ Repeat with server S_i until circuit of desired length n constructed

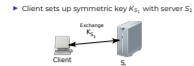
Onion Routing Example

- ▶ Client sets up symmetric key K_{S_1} with server S_1



2024-08-26

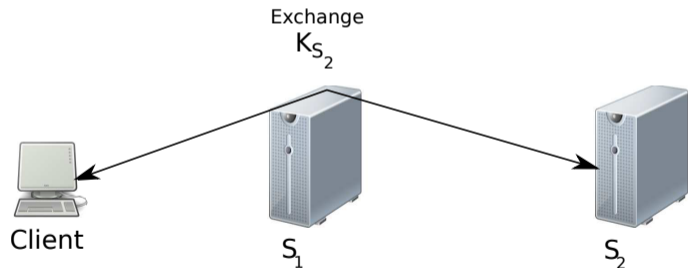
NEXT GENERATION INTERNET
└ How does onion routing work?
└ Onion Routing Example



1. This step uses public key cryptography to establish K_{S_1} .

Onion Routing Example

- ▶ Via S_1 , the client sets up symmetric key K_{S_2} with server S_2



2024-08-26

NEXT GENERATION INTERNET

└ How does onion routing work?

└ Onion Routing Example

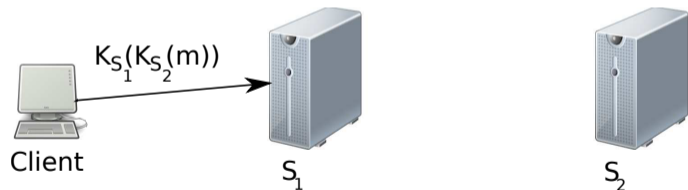
▶ Via S_1 , the client sets up symmetric key K_{S_2} with server S_2



1. S_1 proxies the traffic, thus hiding the IP of the client.
2. S_1 does not learn K_{S_2} due to use of public-key cryptography.
3. While this example only shows two servers, there is no theoretical limit on the length of the chain (but there is one in practice [7])

Onion Routing Example

- ▶ Client encrypts m as $E(K_{S_1}, E(K_{S_2}, (m)))$ and sends to S_1



2024-08-26

NEXT GENERATION INTERNET
└ How does onion routing work?
└ Onion Routing Example

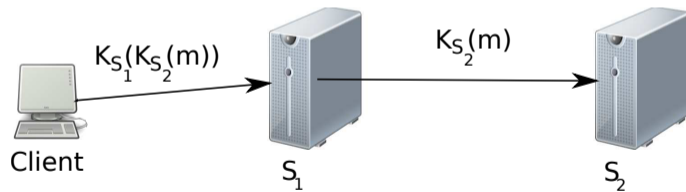
▶ Client encrypts m as $E(K_{S_1}, E(K_{S_2}, (m)))$ and sends to S_1



1. The actual payload is then encrypted using layered symmetric encryption.

Onion Routing Example

- ▶ Server S_1 decrypts and forwards $E(K_{S_2}, (m))$ to S_2 .

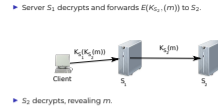


- ▶ S_2 decrypts, revealing m .

2024-08-26

NEXT GENERATION INTERNET
└ How does onion routing work?

└ Onion Routing Example



1. Each server effectively “peels” (decrypts) one layer of the “onion” around the payload.
2. Additional meta-data in each message contains information that allows each server to determine the circuit the message belongs to, allowing the server to determine the correct decryption key and the next hop.

- ▶ Tor is a P2P network of **low-latency** mixes which use onion routing to provide anonymous communication between parties on the Internet.
- ▶ Tor works for any TCP-based protocol and is designed for interactive traffic (https, ssh, etc.)
- ▶ TCP traffic enters the Tor network via a SOCKS proxy
- ▶ **Common usage:** client anonymity for Web browsing

- ▶ Tor is a P2P network of **low-latency** mixes which use onion routing to provide anonymous communication between parties on the Internet.
- ▶ Tor works for any TCP-based protocol and is designed for interactive traffic (https, ssh, etc.)
- ▶ TCP traffic enters the Tor network via a SOCKS proxy
- ▶ **Common usage:** client anonymity for Web browsing

1. Very similar to Mixminion, except no dummy traffic, no artificial delays.

Tor - How it Works

- ▶ "Directory Servers" store list of participating servers
 - ▶ Contact information, public keys, statistics
 - ▶ Directory servers are replicated for security
- ▶ Clients choose servers randomly with bias towards high BW/uptime
- ▶ Clients build long lived Onion routes "circuits" using these servers
- ▶ Circuits are bi-directional
- ▶ Circuits are of length three

2024-08-26

NEXT GENERATION INTERNET

└ How does onion routing work?

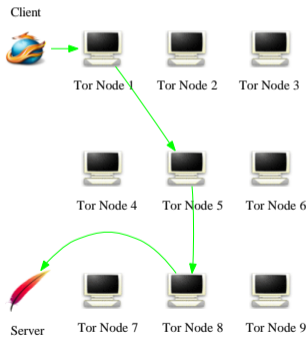
└ Tor - How it Works

- ▶ "Directory Servers" store list of participating servers
 - ▶ Contact information, public keys, statistics
 - ▶ Directory servers are replicated for security
- ▶ Clients choose servers randomly with bias towards high BW/uptime
- ▶ Clients build long lived Onion routes "circuits" using these servers
- ▶ Circuits are bi-directional
- ▶ Circuits are of length three

1. Circuits carry payload in units of 512 bytes.
2. Circuits live for about 10 minutes and are used for many HTTP requests and TCP streams.

Tor - How it Works - Example

▶ Example of Tor client circuit



2024-08-26

NEXT GENERATION INTERNET

└ How does onion routing work?

└ Tor - How it Works - Example

▶ Example of Tor client circuit



1. The first node is called a **guard** node. Guards learn the IP address of the actual client, and are thus particularly sensitive. Tor tries to avoid switching guards too much to minimize the chances of eventually picking a malicious guard.
2. The middle node is the least sensitive, it is primarily chosen by its performance characteristics.
3. The third node is called an **exit** node. It learns the destination and is chosen so that its exit policy allows the specific request. Exit nodes may observe the traffic in the clear, so it is important to use TLS or other types of encryption when using Tor.

Hidden Services in Tor

- ▶ Hidden services allow Tor servers to receive incoming connections anonymously
- ▶ Can provide access to services available *only* via Tor
 - ▶ Web, IRC, etc.
 - ▶ For example, host a website without your ISP knowing

2024-08-26

NEXT GENERATION INTERNET

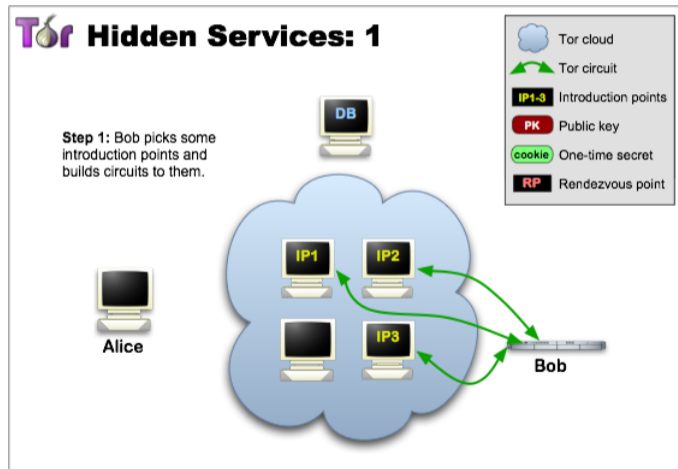
└ How does onion routing work?

└ Hidden Services in Tor

1. Hidden services are basically what is called the “Darknet” in popular media.
2. The design for exit nodes is again a bit inspired by Mixminion: There is first a need to learn how to contact the pseudonymous operator, and then both parties establish part of a path through the network.

- ▶ Hidden services allow Tor servers to receive incoming connections anonymously
- ▶ Can provide access to services available only via Tor
 - ▶ Web, IRC, etc.
 - ▶ For example, host a website without your ISP knowing

Hidden Services Example 1



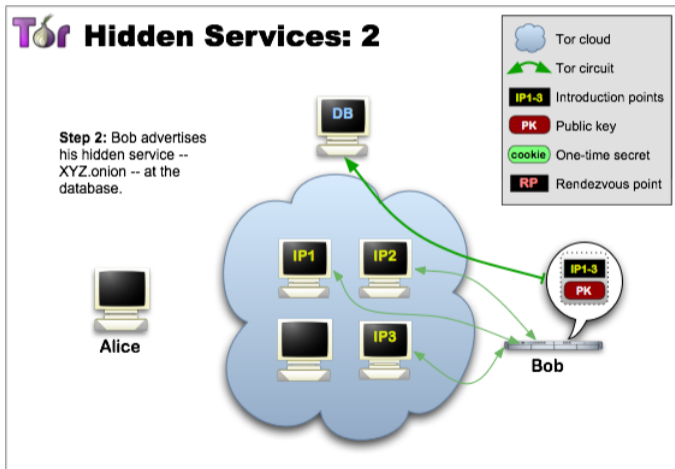
1. A hidden service establishes “Introduction points”, which allow clients to contact hidden server (while keeping its location hidden).

Hidden Services Example 2

2024-08-26

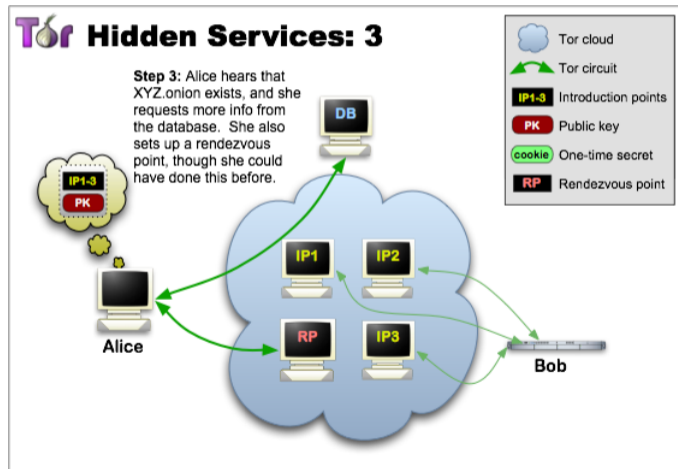
NEXT GENERATION INTERNET
└ How does onion routing work?

└ Hidden Services Example 2



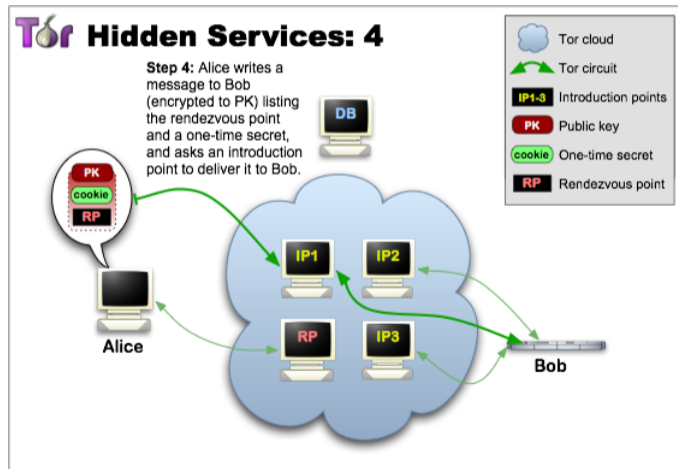
1. Information about the introduction points is published in a distributed hash table under the public key of the hidden service.

Hidden Services Example 3



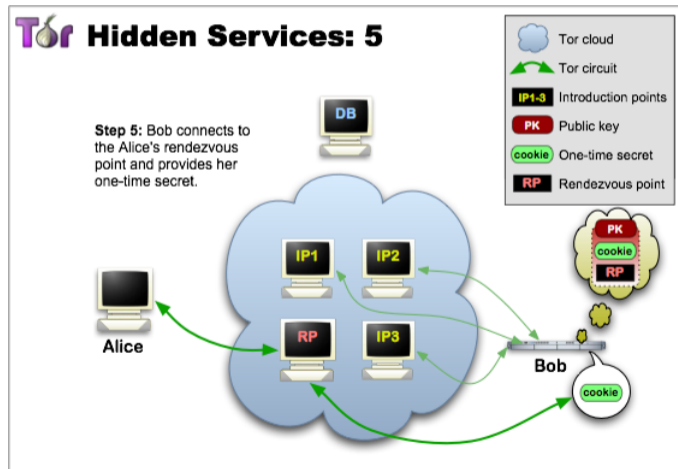
1. Clients look them up using the public keys as a label under the “.onion” pseudo-TLD.
2. Clients then set up a circuit to a “Rendezvous point” where two Tor circuits will be connected.

Hidden Services Example 4



1. The client then sends the address of their random rendezvous point to hidden server by establishing a circuit to the introduction point and sharing a "cookie" (access token) with the hidden service.
2. This keeps the load on the introduction points minimal: they only relay the cookies.

Hidden Services Example 5



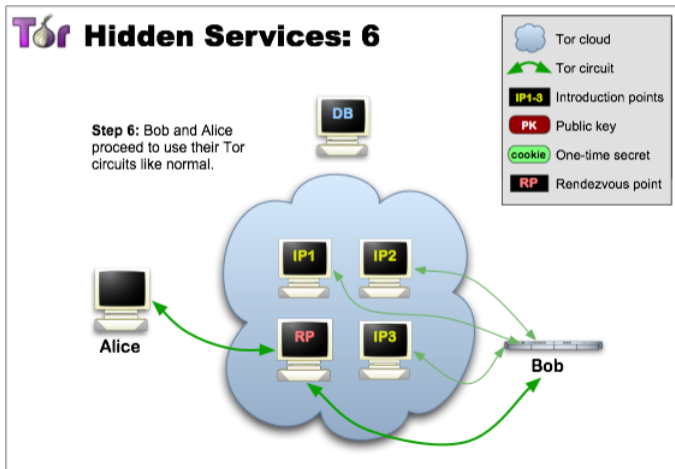
1. The hidden service then establishes its circuit to the rendezvous point.
2. The cookie assures the RP that this is really the hidden service.

Hidden Services Example 6

2024-08-26

NEXT GENERATION INTERNET
└ How does onion routing work?

└ Hidden Services Example 6



1. Data travels a total of 7 hops between the client and the hidden service.

Types of Attacks on Tor

- ▶ Exit relay snooping
- ▶ Website fingerprinting
- ▶ Traffic analysis
- ▶ Intersection attacks
- ▶ DoS [7]

An avoidable (but historically common) issue are badly configured hidden services that directly expose critical information about the operator by accident over the application protocol.

2024-08-26

NEXT GENERATION INTERNET

└ How does onion routing work?



└ Types of Attacks on Tor

- ▶ Exit relay snooping
- ▶ Website fingerprinting
- ▶ Traffic analysis
- ▶ Intersection attacks
- ▶ DoS [7]

An avoidable (but historically common) issue are badly configured hidden services that directly expose critical information about the operator by accident over the application protocol.

1. Encrypting traffic that exits the Tor network is critical. There are historic examples where exit operators spied on user traffic that was in cleartext!
2. Tor's low latency design and 512-byte cell size is not very good at hiding traffic patterns. A particular Web site may have characteristic request-response patterns. If the target site is complex and known to an attacker monitoring the traffic between client and guard, they can probably match the traffic.
3. Exit nodes and target sites could furthermore modulate their bandwidth to introduce easily detectable timing signals to determine the client.
4. If a hidden service goes down if and only if this university is offline, it is probably hosted here. An attacker may even force hosts down to confirm.
5. Denial-of-service attacks on Tor routers could be used to force clients to use specific Tor routers for circuits.

References I

-  Alessandro Acquisti, Roger Dingledine, and Paul Syverson.
On the economics of anonymity.
In Rebecca N. Wright, editor, *Financial Cryptography*, pages 84–102,
Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.
-  Daniel Arp, Fabian Yamaguchi, and Konrad Rieck.
Torben: A practical side-channel attack for deanonymizing tor
communication.
In *Proceedings of the 10th ACM Symposium on Information,
Computer and Communications Security*, pages 597–602, 2015.

2024-08-26

NEXT GENERATION INTERNET

└References



└References

 Alessandro Acquisti, Roger Dingledine, and Paul Syverson.
On the economics of anonymity.
In Rebecca N. Wright, editor, *Financial Cryptography*, pages 84–102,
Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.



 Daniel Arp, Fabian Yamaguchi, and Konrad Rieck.
Torben: A practical side-channel attack for deanonymizing tor
communication.
In *Proceedings of the 10th ACM Symposium on Information,
Computer and Communications Security*, pages 597–602, 2015.

1. Before using Tor, you might want to skim over some of the references presenting a wide range of attacks on the system.
2. History shows that Mix networks are *hard* to implement securely, and Tor uses a particularly *weak* adversary model.



References II

-  George Danezis, Roger Dingledine, and Nick Mathewson.
Mixminion: Design of a type iii anonymous remailer protocol.
In Proceedings of the 2003 IEEE Symposium on Security and Privacy, SP '03, 2003.
-  George Danezis and Ian Goldberg.
Sphinx: A compact and provably secure mix format.
In 2009 30th IEEE Symposium on Security and Privacy, pages 269–282. IEEE, 2009.

References III

-  Claudia Diaz, Harry Halpin, and Aggelos Kiayias.
The nym network.
2021.
-  Roger Dingledine, Nick Mathewson, and Paul Syverson.
Tor: the second-generation onion router.
In Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13, SSYM'04, page 21, USA, 2004. USENIX Association.

References IV

-  Nathan S Evans, Roger Dingledine, and Christian Grothoff.
A practical congestion attack on tor using long paths.
In USENIX Security Symposium, pages 33–50, 2009.
-  Alfonso Iacovazzi, Daniel Frassinelli, and Yuval Elovici.
The {DUSTER} attack: Tor onion service attribution based on flow watermarking with track hiding.
In 22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019), pages 213–225, 2019.

2024-08-26

NEXT GENERATION INTERNET


└References

└References



 Nathan S Evans, Roger Dingledine, and Christian Grothoff.
A practical congestion attack on tor using long paths.
In USENIX Security Symposium, pages 33–50, 2009.

 Alfonso Iacovazzi, Daniel Frassinelli, and Yuval Elovici.
The {DUSTER} attack: Tor onion service attribution based on flow watermarking with track hiding.
In 22nd International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2019), pages 213–225, 2019.

References V

-  Rob Jansen, Florian Tschorsch, Aaron Johnson, and Björn Scheuermann.
The sniper attack: Anonymously deanonymizing and disabling the tor network.
In NDSS, 2014.
-  Zhen Ling, Junzhou Luo, Wei Yu, Xinwen Fu, Dong Xuan, and Weijia Jia.
A new cell-counting-based attack against tor.
IEEE/ACM Transactions On Networking, 20(4):1245–1261, 2012.

References VI

-  Brad Miller, Ling Huang, A.D. Joseph, and J.D. Tygar.
I know why you went to the clinic: Risks and realization of https traffic analysis.
<http://arxiv.org/abs/1403.0297>, 2014.
-  David Anthony Stainton.
Post quantum sphinx.
Cryptology ePrint Archive, 2023.

Acknowledgements

Co-funded by the European Union (Project 101135475).



Co-funded by
the European Union

Co-funded by SERI (HEU-Projekt 101135475-TALER).

Project funded by



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs,
Education and Research EAER
**State Secretariat for Education,
Research and Innovation SERI**

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.



Co-funded by
the European Union

Project funded by

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation
Federal Department of Economic Affairs,
Education and Research EAER
**State Secretariat for Education,
Research and Innovation SERI**

Anonymity

Christian Grothoff

Using Tor

1. Use Tor

- Install Tor
- Configure Tor relay
- Setup hidden service

2. Risk analysis

Perform a risk analysis for your hidden service being deanonymized. How could an attacker proceed?

See also: <https://www.eff.org/document/20141228-speigel-analytics-security-tor-hidden-services>

NEXT GENERATION INTERNET

Blind Signatures

Christian Grothoff

31.05.2024

Learning Objectives

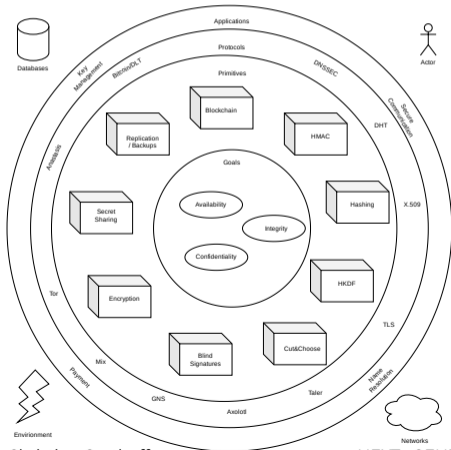
How do standard RSA signatures work?

What are blind signatures?

How do RSA blind signatures work?

What are the main applications for blind signatures?

Blind signatures on our map



How do standard RSA signatures work?

Reminder: RSA

Generate random p, q primes and e such that

$$\text{GCD}((p-1)(q-1), e) = 1 \quad (1)$$

- ▶ Define $n = pq$,
- ▶ compute d such that $ed \equiv 1 \pmod{(p-1)(q-1)}$.
- ▶ Let $s := m^d \pmod n$.
- ▶ Then $m \equiv s^e \pmod n$.

RSA Summary

- ▶ Public key: n, e
- ▶ Private key: $d \equiv e^{-1} \pmod{\phi(n)}$ where $\phi(n) = (p - 1) \cdot (q - 1)$
- ▶ Encryption: $c \equiv m^e \pmod{n}$
- ▶ Decryption: $m \equiv c^d \pmod{n}$
- ▶ Signing: $s \equiv m^d \pmod{n}$
- ▶ Verifying: $m \equiv s^e \pmod{n}$?

These equations are heavily simplified and should not be used like this in production!

Low Encryption Exponent Attack

- ▶ e is known
 - ▶ m maybe small
 - ▶ $C = m^e < n$?
 - ▶ If so, can compute $m = \sqrt[e]{C}$
- ⇒ Small e can be bad!

Padding and RSA Symmetry

- ▶ Padding can be used to avoid low exponent issues (and issues with $m = 0$ or $m = 1$)
- ▶ Randomized padding defeats chosen plaintext attacks
- ▶ Padding breaks RSA symmetry:

$$D_{A_{priv}}(D_{B_{priv}}(E_{A_{pub}}(E_{B_{pub}}(m)))) \neq m \quad (2)$$

- ▶ PKCS#1 / RFC 3447 define a padding standard

What are blind signatures?

How do RSA blind signatures work?

Blind signatures with RSA [3]

1. Obtain public key (e, n)
2. Compute $f := FDH_n(m)$,
 $f < n$.
3. Generate random blinding factor $b \in \mathbb{Z}_n$
4. Transmit $f' := fb^e \pmod n$

Blind signatures with RSA [3]

1. Obtain public key (e, n)
 2. Compute $f := \text{FDH}_n(m)$,
 $f < n$.
 3. Generate random blinding factor $b \in \mathbb{Z}_n$
 4. Transmit $f' := fb^e \pmod n$
1. Receive f' .
 2. Compute $s' := f'^d \pmod n$.
 3. Send s' .

Blind signatures with RSA [3]

1. Obtain public key (e, n)
2. Compute $f := \text{FDH}_n(m)$,
 $f < n$.
3. Generate random blinding factor $b \in \mathbb{Z}_n$
4. Transmit $f' := fb^e \pmod n$

1. Receive f' .
2. Compute $s' := f'^d \pmod n$.
3. Send s' .

1. Receive s' .
2. Compute $s := s'b^{-1} \pmod n$

Other blind signature systems

- ▶ Using Schnorr: [6, 5, 2, 4]
- ▶ Lattice-based: [1]

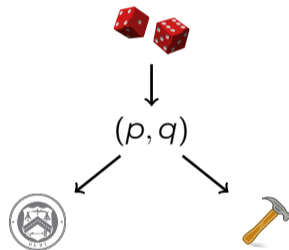
What are the main applications for blind signatures?

Applications for blind signatures

- ▶ Untraceable payments
- ▶ Unlinkable access tokens (PrivacyPass)

Provider setup: Create a denomination key (RSA)

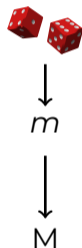
1. Generates random primes p, q .
2. Computes $n := pq$,
 $\phi(n) = (p - 1)(q - 1)$
3. Picks small $e < \phi(n)$ such that $d := e^{-1} \pmod{\phi(n)}$ exists.
4. Publishes public key (e, n) .



Merchant setup: Create a signing key (EdDSA)

- ▶ Generates random $m \pmod{o}$ as private key
- ▶ Computes public key $M := mG$

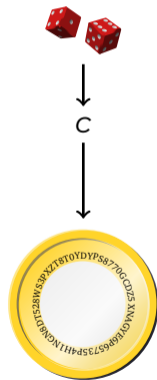
Capability: $m \Rightarrow$ 



Customer: Create a planchet (EdDSA)

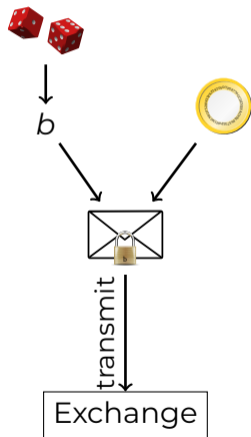
- ▶ Generates random $c \pmod{o}$ as private key
- ▶ Computes public key $C := cG$

Capability: $c \Rightarrow$ 



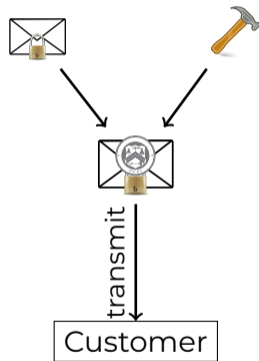
Customer: Blind planchet (RSA)

1. Obtains public key (e, n)
2. Computes $f := FDH_n(C), f < n$.
3. Generates random blinding factor $b \in \mathbb{Z}_n$
4. Transmits $f' := fb^e \pmod n$



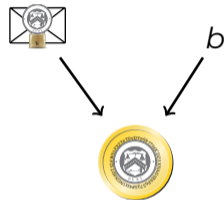
Provider: Blind sign (RSA)

1. Receives f' .
2. Computes $s' := f'^d \pmod n$.
3. Sends signature s' .

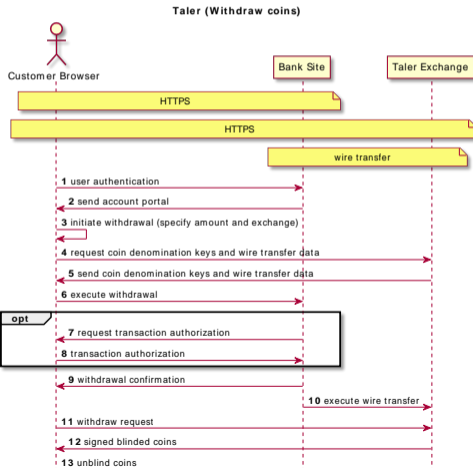


Customer: Unblind signature (RSA)

1. Receives s' .
2. Computes $s := s'b^{-1} \pmod n$



Withdrawing coins on the Web



Customer: Build shopping cart

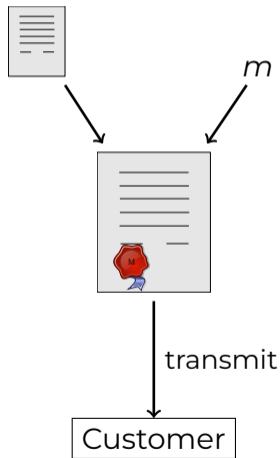


↓ transmit



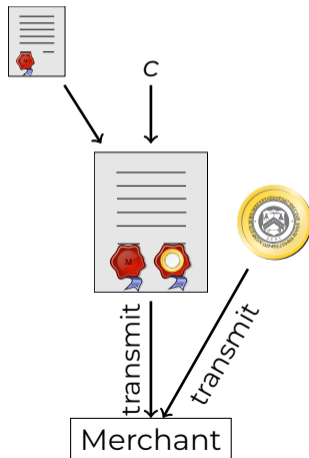
Merchant: Propose contract (EdDSA)

1. Complete proposal D .
2. Send $D, EdDSA_m(D)$



Customer: Spend coin (EdDSA)

1. Receive proposal D , $EdDSA_m(D)$.
2. Send s , C , $EdDSA_c(D)$



Merchant and provider: Verify coin (RSA)

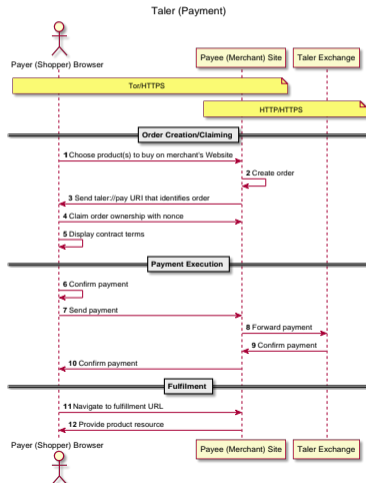
$$s^e \pmod n \stackrel{?}{\equiv} FDH_n(C)$$





The provider (Taler: exchange) does not only verify the signature, but also checks that the coin was not double-spent.

GNU Taler is an online payment system.



Payment processing with blind signatures



References I

-  Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav. Practical, round-optimal lattice-based blind signatures. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 39–53, 2022.
-  Fabrice Benhamouda, Tancrede Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ros. Cryptology ePrint Archive, Report 2020/945, 2020. <https://ia.cr/2020/945>.

References II

-  David Chaum.
Blind Signature System, pages 153–153.
Springer US, Boston, MA, 1984.
-  Gian Demarmels and Lucien Hezeveldt.
Adding schnorr's blind signature in taler.
Master's thesis, Bern University of Applied Sciences, 2022.
-  Claus Peter Schnorr.
Security of blind discrete log signatures against interactive attacks.
In *ICICS 2001, LNCS 2229*, pages 1–12. Springer-Verlag, 2001.

References III

-  Claus Peter Schnorr.
Enhancing the security of perfect blind dl-signatures.
Universität Frankfurt, 2004.
<https://www.math.uni-frankfurt.de/~dmst/teaching/SS2012/Vorlesung/EBS5.pdf>.

Acknowledgements

Funded by the European Union (Project 101135475).



**Co-funded by
the European Union**

Funded by SERI (HEU-Projekt 101135475-TALER).

Project funded by



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs,
Education and Research EAER
**State Secretariat for Education,
Research and Innovation SERI**

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Blind signatures

Christian Grothoff

Implementing RSA blind signatures

1. Review RSA code

1. Find `rsa.py` in the course resources.
2. Run the script, making sure the `python3-pycryptodome` dependency is installed on your system.
3. Review the code.

2. Add blind signatures

1. Add a function to compute a blinding factor given a public key.
2. Add a function to apply the FDH to a message and blind the result with the blinding factor, returning a blinded message.
3. Make a copy of the `rsa_sign` function and modify it to sign over a blinded message and to return a blind signature.
4. Add a function to unblind a blinded message given a blinding factor.
5. Modify the main function to blind a message, use blind signing (instead `rsa_sign`), and then unblind the blind signature.
6. Check that the final signature is still valid.

3. Measure performance

1. How long does each step take? Run each step in a loop 1000 times and measure the time.
2. What is the most expensive operation?

NEXT GENERATION INTERNET

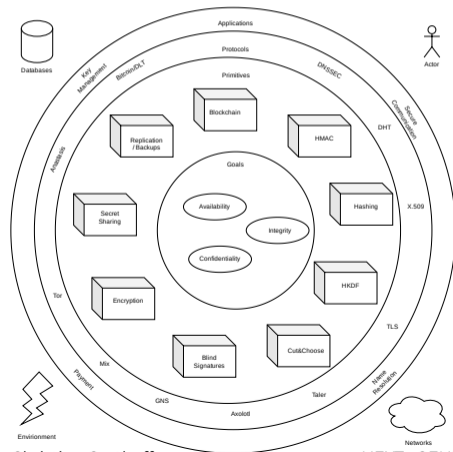
Key management

Christian Grothoff

07.06.2024

1. This lecture is about key management, covering different approaches for managing secret keys.

Key management on our map



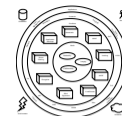
Christian Grothoff

NEXT , GENERATION , INTERNET

2024-08-26

NEXT GENERATION INTERNET

Key management on our map



1. Keys are central for encryption and signatures and they themselves primarily require confidentiality and availability. Thus, they play a role in virtually everything we do.
2. The main primitives are secret sharing and HKDFs. The main application we will look at is GNU Anastasis.
3. Related topics are replication/backups, but also X.509 (certification), DNSSEC (online/offline keys), and GNS (revocation).

Learning Objectives

How can we protect confidentiality of private keys?

How does Shamir Secret Sharing work?

Key escrow and recovery: From Shamir to Anastasis

What are threshold signatures?

What does key management look like in practice?

2024-08-26

NEXT GENERATION INTERNET

Learning Objectives

1. First, we will take a high level look at technologies to protect confidentiality.
2. Shamir Secret Sharing (SSS) is a technique to protect both confidentiality and availability.
3. But SSS has issues in practice, so we will introduce GNU Anastasis as a more practical solution.
4. Then, we will look at threshold signatures as a way to protect highly valuable signing keys.
5. Finally, we'll look at the different types of keys in GNU Taler as a practical example for how complex applications require various types of keys with different protection strategies.

How can we protect confidentiality of private keys?
How does Shamir Secret Sharing work?
Key escrow and recovery: From Shamir to Anastasis
What are threshold signatures?
What does key management look like in practice?

How can we protect confidentiality of private keys?

Software based Personal Security Environments (PSE): PKCS#12

PKCS#12 is the most common format for software PSEs:

- ▶ PKCS#12 is a file container format used for storage and transport of private keys (and possibly certificates).
- ▶ Information is protected with a password-based symmetric key (e.g. a password).
- ▶ The security of a software PKCS#12 is based on the strength of the password protecting it.

Problem: A PKCS#12 soft-token may be copied unnoticed.

2024-08-26

NEXT GENERATION INTERNET

└─ How can we protect confidentiality of private keys?

└─ Software based Personal Security Environments (PSE): PKCS#12

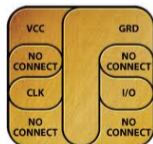
PKCS#12 is the most common format for software PSEs:

- ▶ PKCS#12 is a file container format used for storage and transport of private keys (and possibly certificates).
- ▶ Information is protected with a password-based symmetric key (e.g. a password).
- ▶ The security of a software PKCS#12 is based on the strength of the password protecting it.

Problem: A PKCS#12 soft-token may be copied unnoticed.

1. Naturally, there are other formats for storing private keys.
2. Password protection is of course already a good idea, but limited as sometimes the password will need to be entered to decrypt the key, and then a compromised host could result in the private key being leaked.
3. Furthermore, while the password *should* be strong, an adversary getting hold of the PKCS#12 data might still be able to brute-force it in practice. Not everybody uses 58-character passwords.

Smartcards and Cryptotokens



2024-08-26

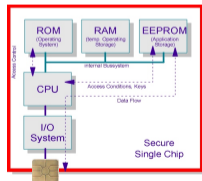
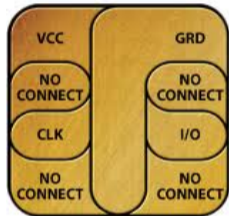
NEXT GENERATION INTERNET

└ How can we protect confidentiality of private keys?

└ Smartcards and Cryptotokens

1. A general purpose PC or smart phone can run a huge range of applications and has a complex operating system and thus a huge attack surface. The use of dedicated hardware that only runs the cryptographic functions plus some access control logic is thus safer.

Properties of Crypto-tokens/cards



- ▶ Crypto-cards have the ability of a secure container for secret data and have an executive platform for cryptographic algorithms.
- ▶ A Crypto-card looks like a “Black Box” from the outside, where some operations can only be used over a very restrictive hard- and software interface which is able to enforce specific security policies.
- ▶ Access to sensitive data areas (i.e. private keys) is physically “impossible” from the outside.

2024-08-26

NEXT GENERATION INTERNET

└ How can we protect confidentiality of private keys?

└ Properties of Crypto-tokens/cards



- ▶ Crypto-cards have the ability of a secure container for secret data and have an executive platform for cryptographic algorithms.
- ▶ A Crypto-card looks like a “Black Box” from the outside, where some operations can only be used over a very restrictive hard- and software interface which is able to enforce specific security policies.
- ▶ Access to sensitive data areas (i.e. private keys) is physically “impossible” from the outside.

1. Depending on the hardware, it is also possible to make it hard to extract the private key. This is of course a double-edged sword, as if the key is generated on the hardware, this would also prevent key backups.
2. Finally, cheaper hardware may also lack good entropy generators, requiring the keys to be provisioned, for example by the manufacturer, creating new risks.

Example: Yubikey and Personal Identity Verification (PIV)

- ▶ Yubikey provides Smart Card functionality based on the Personal Identity Verification (PIV) interface specified in NIST SP 800-73.
- ▶ Yubikeys perform RSA or ECC sign/decrypt operations using a private key stored on the token, through common interfaces such as PKCS#11.
- ▶ Supported key sizes: RSA 2048 or ECC 256/384.
- ▶ The “universal smartcard minidriver” provides “standard smart” functionality and additional certificate and PIN management features.
- ▶ Special Yubikeys obtained FIPS 140-2 security level certification.

2024-08-26

NEXT GENERATION INTERNET

└ How can we protect confidentiality of private keys?

└ Example: Yubikey and Personal Identity Verification (PIV)

- ▶ Yubikey provides Smart Card functionality based on the Personal Identity Verification (PIV) interface specified in NIST SP 800-73.
- ▶ Yubikeys perform RSA or ECC sign/decrypt operations using a private key stored on the token, through common interfaces such as PKCS#11.
- ▶ Supported key sizes: RSA 2048 or ECC 256/384.
- ▶ The “universal smartcard minidriver” provides “standard smart” functionality and additional certificate and PIN management features.
- ▶ Special Yubikeys obtained FIPS 140-2 security level certification.

1. A key issue with such devices is that they usually only support a limited set of cryptographic primitives.
2. The speed of operations is also generally only useful for personal use, like when signing an e-mail or authorizing a login, and not for APIs that need to automatically sign thousands of messages per second.
3. Finally, over 20 CVEs against Yubikey, including some with high or even critical severity (CVE-2015-3298, CVE-2021-43399, CVE-2011-4120) show that certification does not warrant the absence of critical security flaws, and merely shows that the manufacturer followed standard procedures.

Hardware Security Modules (HSM)

Common functionality:

- ▶ Secure storing and use of keys
- ▶ Random number generator
- ▶ Key pair generation
- ▶ Digital signing
- ▶ Key archiving
- ▶ Acceleration for crypto schemes

Should protect keys against:

- ▶ Mechanical & chemical attacks
- ▶ Temperature attacks
- ▶ Manipulation of voltage



2024-08-26

NEXT GENERATION INTERNET

└ How can we protect confidentiality of private keys?

└ Hardware Security Modules (HSM)

- Common functionality:
- ▶ Secure storing and use of keys
 - ▶ Random number generator
 - ▶ Key pair generation
 - ▶ Digital signing
 - ▶ Key archiving
 - ▶ Acceleration for crypto schemes
- Should protect keys against:
- ▶ Mechanical & chemical attacks
 - ▶ Temperature attacks
 - ▶ Manipulation of voltage



1. For server-side use where hundreds or thousands of public key operations are required per second, you can buy expensive high-end HSMs.
2. These HSMs not only offer higher performance, but also improved lifecycle management (entropy generation, key generation, key rotation and archival) and security.
3. Specifically, such high-end HSMs for servers are expected to have more solid protections against attacks. While consumer-grade devices can often leak private keys when exposed to physical attacks, these HSMs are expected to detect physical attacks and destroy private keys before they get leaked.
4. Plus, they are more unwieldy and thus harder to steal from your data center with cameras watching the racks.

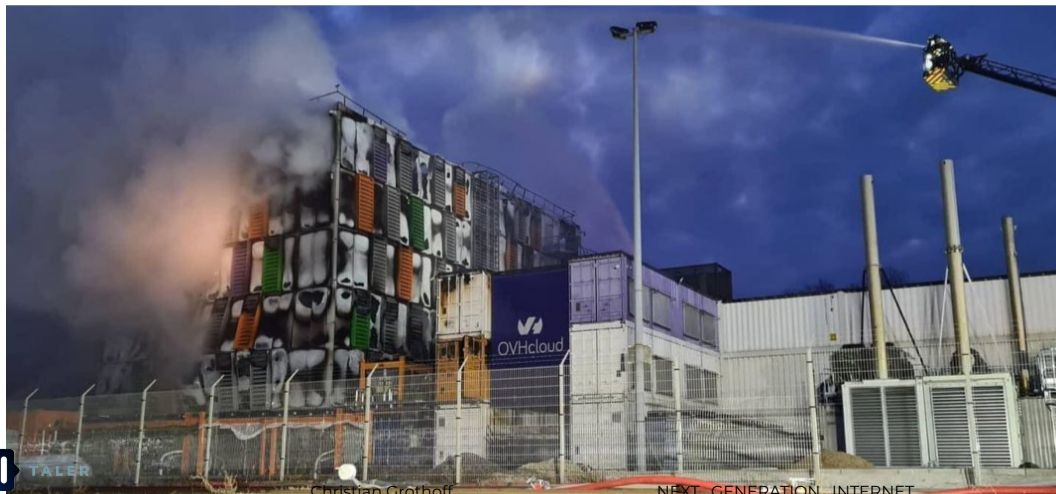
Availability

2024-08-26

NEXT GENERATION INTERNET

└ How can we protect confidentiality of private keys?

└ Availability



1. HSMs may be helpful to protect the confidentiality of keys. However, most are proprietary, which means your control over the keys is worse than with FLOSS on COTS hardware.
2. Confidentiality is most important for signing keys; signing keys are easily replaced when the HSM breaks.
3. But for decryption keys, availability is at least equally critical.
4. Even a high-end HSM doesn't cannot ensure availability, as the OVH data center fire showed.

How does Shamir Secret Sharing work?

Problem 1: Availability

If you give one person (or data center) a secret, it may get lost.

2024-08-26

NEXT GENERATION INTERNET

└ How does Shamir Secret Sharing work?

└ Problem 1: Availability

If you give one person (or data center) a secret, it may get lost.

1. The usual answer to this is to have multiple backups.
2. This way, if one copy is lost, another is still available.

Problem 1: Availability

If you give one person (or data center) a secret, it may get lost.

⇒ So give it to more than one person (or data center)!

2024-08-26

NEXT GENERATION INTERNET

└ How does Shamir Secret Sharing work?

└ Problem 1: Availability

If you give one person (or data center) a secret, it may get lost.

⇒ So give it to more than one person (or data center)!

1. The usual answer to this is to have multiple backups.
2. This way, if one copy is lost, another is still available.

Problem 2: Confidentiality

If you give many entities a secret, it may get disclosed.

2024-08-26

NEXT GENERATION INTERNET

└ How does Shamir Secret Sharing work?

└ Problem 2: Confidentiality

If you give many entities a secret, it may get disclosed.

1. Now, we still want to keep the key a secret, and giving copies out to many entities to improve availability makes confidentiality harder.
2. A simple solution here is to not give everyone a working key, but a share of the key.
3. The idea being that then some subset of the shares, often k out of n , are enough to reconstruct the secret.

Problem 2: Confidentiality

If you give many entities a secret, it may get disclosed.

⇒ So give them only a key share!

2024-08-26

NEXT GENERATION INTERNET

└ How does Shamir Secret Sharing work?

└ Problem 2: Confidentiality

If you give many entities a secret, it may get disclosed.

→ So give them only a key share!

1. Now, we still want to keep the key a secret, and giving copies out to many entities to improve availability makes confidentiality harder.
2. A simple solution here is to not give everyone a working key, but a share of the key.
3. The idea being that then some subset of the shares, often k out of n , are enough to reconstruct the secret.

Problem 3: Scalability

If you want k out of n entities to coordinate to recover a secret, there are

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad (1)$$

combinations to consider.

2024-08-26

NEXT GENERATION INTERNET

└ How does Shamir Secret Sharing work?

└ Problem 3: Scalability

If you want k out of n entities to coordinate to recover a secret, there are

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad (1)$$

combinations to consider.

1. The number of subsets of size k of a set of size n can be large.
2. Polynomial interpolation gives us a way to do this efficiently.

Problem 3: Scalability

If you want k out of n entities to coordinate to recover a secret, there are

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad (1)$$

combinations to consider.

⇒ Use polynomials!

2024-08-26

NEXT GENERATION INTERNET

└ How does Shamir Secret Sharing work?

└ Problem 3: Scalability

If you want k out of n entities to coordinate to recover a secret, there are

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \quad (1)$$

combinations to consider.

⇒ Use polynomials!

1. The number of subsets of size k of a set of size n can be large.
2. Polynomial interpolation gives us a way to do this efficiently.

Polynomials

A polynomial of degree $k - 1$ is fully determined by k data points

$$(x_0, y_0), \dots, (x_j, y_j), \dots, (x_{k-1}, y_{k-1}),$$

where no two x_j may be identical.



2024-08-26

NEXT GENERATION INTERNET

└ How does Shamir Secret Sharing work?

└ Polynomials

A polynomial of degree $k - 1$ is fully determined by k data points
 $(x_0, y_0), \dots, (x_j, y_j), \dots, (x_{k-1}, y_{k-1})$,
where no two x_j may be identical.



1. The idea is that $L(0)$ is the secret. The other coefficients of the polynomial of degree $k - 1$ are chosen at random.
2. We then give the n entities $y_i = L(i)$ for a polynomial of degree $k - 1$.
3. This way, everyone has to store only (i, y_i) and k entities can reconstruct the polynomial and then compute $L(0)$.

Lagrange interpolation

The interpolation polynomial in the Lagrange form is:

$$L(x) := \sum_{j=0}^k y_j l_j(x)$$

where

$$l_j(x) := \prod_{\substack{0 \leq m \leq k \\ m \neq j}} \frac{x - x_m}{x_j - x_m} = \frac{(x - x_0) \dots (x - x_{j-1}) (x - x_{j+1}) \dots (x - x_k)}{(x_j - x_0) \dots (x_j - x_{j-1}) (x_j - x_{j+1}) \dots (x_j - x_k)} \quad (2)$$

for $0 \leq j \leq k$.

2024-08-26

NEXT GENERATION INTERNET

└ How does Shamir Secret Sharing work?

└ Lagrange interpolation

The interpolation polynomial in the Lagrange form is:

$$L(x) := \sum_{j=0}^k y_j l_j(x)$$

where

$$l_j(x) := \prod_{\substack{0 \leq m \leq k \\ m \neq j}} \frac{x - x_m}{x_j - x_m} = \frac{(x - x_0) \dots (x - x_{j-1}) (x - x_{j+1}) \dots (x - x_k)}{(x_j - x_0) \dots (x_j - x_{j-1}) (x_j - x_{j+1}) \dots (x_j - x_k)} \quad (2)$$

for $0 \leq j \leq k$.

1. This fun equation tells us how to recompute the coefficients of L given k points.
2. l_j can be pre-computed from just knowing the subset k and without knowing the secret shares y_i .

Practical considerations

- ▶ Our secrets will typically be integers. Calculations with floating points are *messy*.
- ⇒ Use finite field arithmetic, not \mathbb{R} .

2024-08-26

NEXT GENERATION INTERNET

└ How does Shamir Secret Sharing work?

└ Practical considerations

▶ Our secrets will typically be integers. Calculations with floating points are messy.

⇒ Use finite field arithmetic, not \mathbb{R} .

1. If we did the calculation in \mathbb{R} , rounding errors will in most cases make the result non-sensical.
2. So instead, we need to do basically the same math over some finite field $GF(q)$.
3. Surprisingly, this is not actually useful in practice for key management.

Real world scalability

n/k	1	2	3	4	5	6
1	1	2	3	4	5	6
2		1	3	6	10	15
3			1	4	10	20
4				1	5	15
5					1	6
6						1

Other values:

- ▶ $\binom{10}{5} = 252$
- ▶ $\binom{20}{10} = 184756$
- ▶ $\binom{30}{15} = 155117520$

└ How does Shamir Secret Sharing work?

└ Real world scalability

n/k	1	2	3	4	5	6
1	1	2	3	4	5	6
2		1	3	6	10	15
3			1	4	10	20
4				1	5	15
5					1	6
6						1

- Other values:
- ▶ $\binom{10}{5} = 252$
 - ▶ $\binom{20}{10} = 184756$
 - ▶ $\binom{30}{15} = 155117520$

1. Looking at practical values for k and n , we first of all find that $\binom{n}{k}$ is usually not very big.
2. $n = 20$ and $k = 10$ is already pretty big for distributing key shares.
3. $n = 30$ and $k = 15$ is where we may start to see a need for scalability, but even that is still doable on modern computers.

Do we have a scalability problem?

How many people do you have to share your secrets with?

How many people realistically participate in recovery?

2024-08-26

NEXT GENERATION INTERNET

└ How does Shamir Secret Sharing work?

└ Do we have a scalability problem?

How many people do you have to share your secrets with?

How many people realistically participate in recovery?

1. The real problem we have is usability, not scalability.
2. Here, k out of n is not flexible enough in practice. If Alice is the boss, we may want Alice or Bob and Alice or Dave, but not Bob and Dave without Alice to be able to reconstruct the key!
3. Usability includes how we authenticate during recovery. Here also, it may make sense to have a policy like password plus SMS-TAN or password plus Email-TAN, but not SMS-TAN plus E-mail TAN as both of those factors might just need control over the same smartphone!
4. Finally, we want to apply data minimization also to the key escrow services that hold the shares to minimize risk.

Key escrow and recovery: From Shamir to Anastasis

What is GNU Anastasis? [2]

- ▶ Distributed key escrow and recovery service
- ▶ Users split their secret keys and distribute shares across multiple service providers
- ▶ Only the authorized user can recover the key by following standard authentication procedures
- ▶ Service providers learn nothing about the user, except possibly some details about how to authenticate the user

2024-08-26

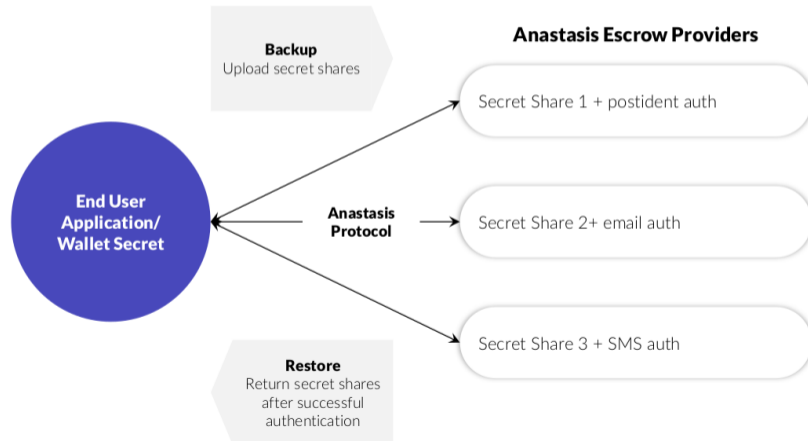
NEXT GENERATION INTERNET

└─ Key escrow and recovery: From Shamir to Anastasis

└─ What is GNU Anastasis? [2]

- ▶ Distributed key escrow and recovery service
- ▶ Users split their secret keys and distribute shares across multiple service providers
- ▶ Only the authorized user can recover the key by following standard authentication procedures
- ▶ Service providers learn nothing about the user, except possibly some details about how to authenticate the user

1. GNU Anastasis addresses these challenges, allowing flexible recovery policies (not just k -out-of- n) to be freely configured.
2. Moreover, the system focuses on allowing different methods to authorize key recovery, and considers the economic aspect of operating such a service by integrating *payment*.
3. Finally, it minimizes data exposed to the providers *and* addresses the issue of users forgetting their credentials.



1. At a high level, the user shares their secret shares with the various providers together with information on the authentication method chosen for each share.
2. During recovery, the user authenticates to a subset of the providers to recover the shares.
3. Given enough shares, they can then reconstruct the secret.
4. Next, we'll look at a much simplified version of the protocol in a bit more detail.

Step 1: Enter secret information



2024-08-26

NEXT GENERATION INTERNET

└ Key escrow and recovery: From Shamir to Anastasis

└ Step 1: Enter secret information



1. The cryptographic process starts with the master secret being provided to the application.
2. In practice, this is actually almost the last step in the user-story, but it is the first step for the cryptography.

Step 2: Split information

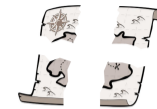


2024-08-26

NEXT GENERATION INTERNET

└ Key escrow and recovery: From Shamir to Anastasis

└ Step 2: Split information



1. First, the master secret is “split” into various shares. The master secret is encrypted with multiple random symmetric keys, each of which is derived from the respective set of nonces used as key shares.
2. Regardless, we can view this as splitting up the original secret into various pieces that allow recovery.

Step 3: Hash user identification



2024-08-26

NEXT GENERATION INTERNET

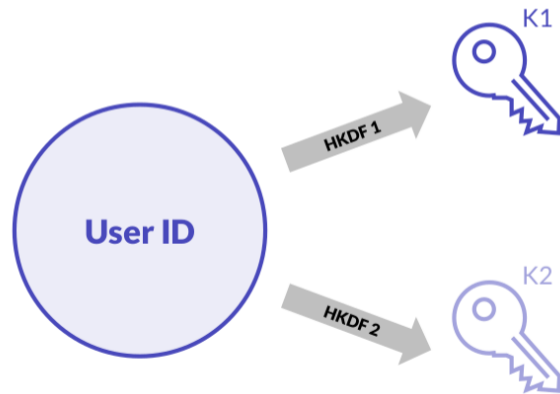
└ Key escrow and recovery: From Shamir to Anastasis

└ Step 3: Hash user identification



1. This is a highly unusual step. The user is first asked to provide various highly personal details about themselves. The design calls for attributes to be provided that ideally never change, cannot be forgotten, and at the same time are ideally not well-known.
2. The user can actually provide false data here, but what is critical is that they remember the data that they provide! So in practice, it is recommended to be perfectly honest, and to provide as much data as possible.
3. All provided data is then run through an expensive cryptographic hash function to create the **user ID**, a unique identifier for the user.
4. Here, we thus have a privacy paradox: the more private data is provided, the less predictable or invertible the user ID will be, and thus the more privacy and confidentiality the system will achieve.

Step 4: Key derivation



2024-08-26

NEXT GENERATION INTERNET

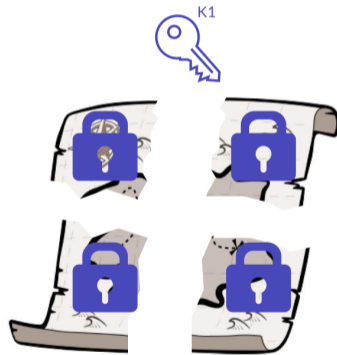
└ Key escrow and recovery: From Shamir to Anastasis

└ Step 4: Key derivation



1. From the user ID, two keys are derived using different HKDF functions.

Step 5: Encrypt parts



2024-08-26

NEXT GENERATION INTERNET

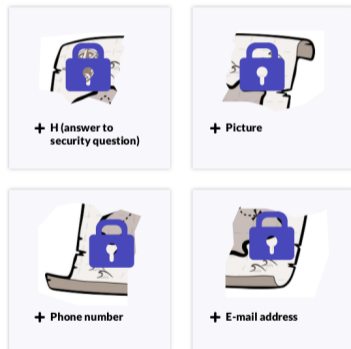
└─ Key escrow and recovery: From Shamir to Anastasis

└─ Step 5: Encrypt parts



1. The first key is then used to encrypt the key shares.
2. This ensures that the providers do not even learn the key shares that they are storing, except if they somehow **guess** or **know** their user's identity with all of the attributes and are thus able to mount a confirmation attack to invert the hash function.

Step 6: Add truth



2024-08-26

NEXT GENERATION INTERNET

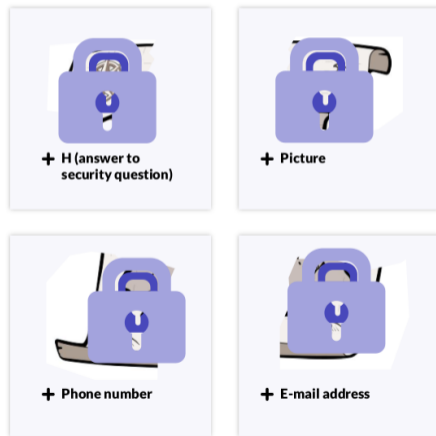
└─ Key escrow and recovery: From Shamir to Anastasis

└─ Step 6: Add truth



1. Anastasis calls the authentication data of the user the **truth**. This can be the (salted hash of the) answer to a security question, an image of the user for video identification, their phone number or e-mail address to send a TAN, or basically any other data that would be required to perform some authorization check.
2. Each key share is combined with the respective truth that defines the authorization check that the user will need to pass to recover the encrypted key share.

Step 7: Encrypt truth



2024-08-26

NEXT GENERATION INTERNET

└─ Key escrow and recovery: From Shamir to Anastasis

└─ Step 7: Encrypt truth



1. The truth (and the associated encrypted key share) are then encrypted with the second key that was derived from the user's identity.
2. This ensures that the providers do not learn the truth during backup.

Step 8: Store data



2024-08-26

NEXT GENERATION INTERNET

└ Key escrow and recovery: From Shamir to Anastasis

└ Step 8: Store data



1. The last step of the backup process is to then upload these encrypted truth and double-encrypted key shares to the Anastasis providers.
2. The providers store this information under provider-specific unlinkable identifiers. Thus, they only learn the time and the size of the data (and possibly the IP address of the client, which they should immediately discard).
3. Note that this is actually only part of the story, as independently more encrypted information is stored to enable recovery, but for the **simplified** protocol description we will leave it at this.

Step 9: User identification



2024-08-26

NEXT GENERATION INTERNET

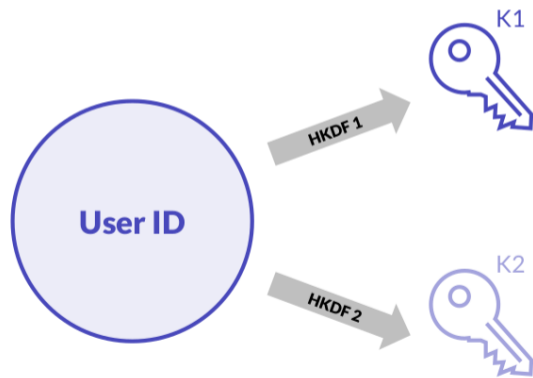
└─ Key escrow and recovery: From Shamir to Anastasis

└─ Step 9: User identification



1. To recover their secret, a user first must again enter all of their personal information. It is again hashed to compute the unique user ID.

Step 10: Key derivation

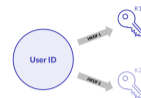


2024-08-26

NEXT GENERATION INTERNET

└ Key escrow and recovery: From Shamir to Anastasis

└ Step 10: Key derivation



1. The client can then again derive the same two symmetric encryption keys.

Step 11: Provide key



2024-08-26

NEXT GENERATION INTERNET

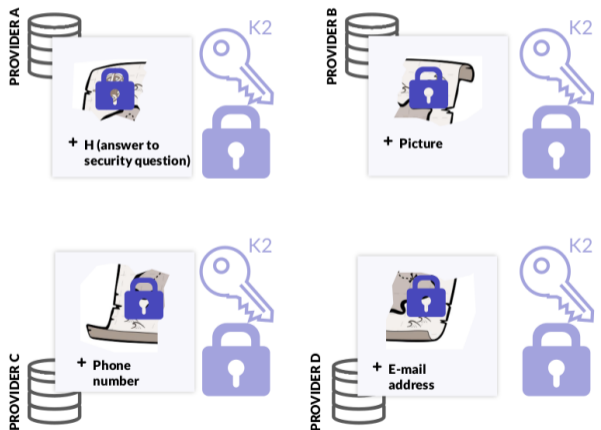
└ Key escrow and recovery: From Shamir to Anastasis

└ Step 11: Provide key



1. The encryption keys are sent to the respective providers.
2. Note that in practice, each provider will be given a different key, but this is again a detail.

Step 12: Decrypt truth



2024-08-26

NEXT GENERATION INTERNET

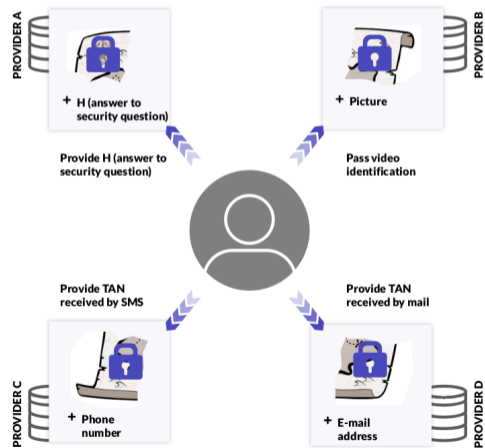
└ Key escrow and recovery: From Shamir to Anastasis

└ Step 12: Decrypt truth



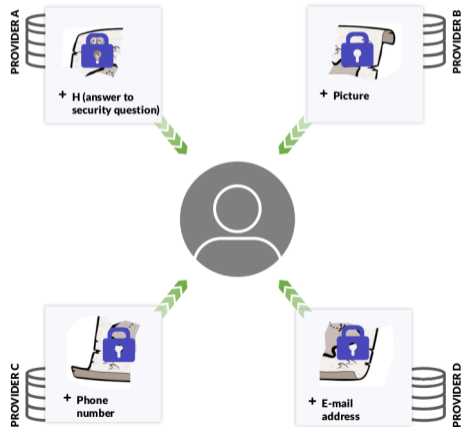
1. The providers can then decrypt the truth and challenge the user to pass the authorization check.
2. This can be quite different depending on the check, but we can imagine providers sending TAN codes to the e-mail address or mobile phone number they decrypted.

Step 13: Authenticate



1. Next, the user must pass the authorization challenge, for example by sending back the TAN.

Step 14: Receive parts



2024-08-26

NEXT GENERATION INTERNET

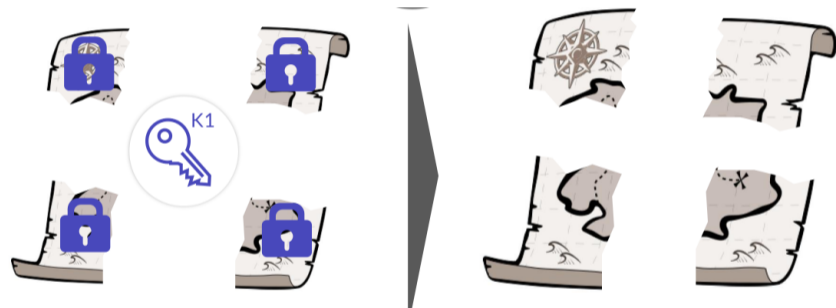
└ Key escrow and recovery: From Shamir to Anastasis

└ Step 14: Receive parts



1. In response, the client is given the encrypted key shares for the challenges that they have passed.

Step 15: Decrypt parts



2024-08-26

NEXT GENERATION INTERNET

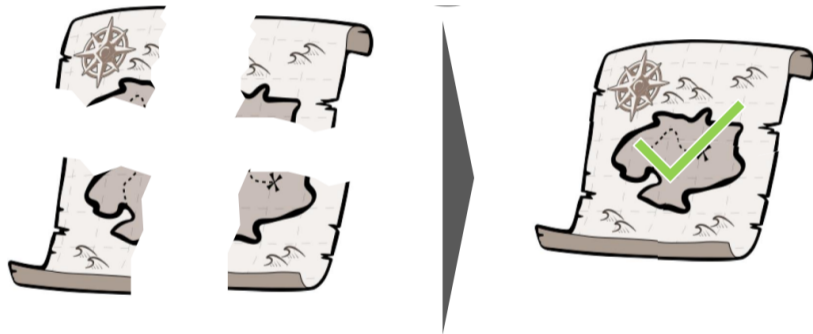
└ Key escrow and recovery: From Shamir to Anastasis

└ Step 15: Decrypt parts



1. The client can then decrypt the key shares using the first key derived from the user ID.

Step 16: Reassembly



2024-08-26

NEXT GENERATION INTERNET

└ Key escrow and recovery: From Shamir to Anastasis

└ Step 16: Reassembly



1. Given enough shares, the original master secret can be recovered.

Reality is more complex



Policies to allow more flexible splitting than 4/4



Recovery document to remember policies and providers



Distinction between core secret and master secret



Payment processing



Provider salts



Anti-DoS provisions in protocol / request limits



Versioning



Liability limitations

2024-08-26

NEXT GENERATION INTERNET

└ Key escrow and recovery: From Shamir to Anastasis

└ Reality is more complex



1. Policy documents are created and stored by providers under another hash derived from the user ID to remember which combinations of secret shares at which providers would allow recovery of the master secret.
2. Per-provider salts are used to ensure that key material differs between providers.
3. Request limits and payments are used to protect Anastasis providers against attacks, and key shares against brute-force attacks.
4. A **core secret** is used for the actual “larger” data the user wants to back up, while the **master secret** is a symmetric key used to encrypt the core secret. Versioning support ensures that the same user can upload and recover more than one core secret.
5. Providers can also set liability limits, so that users that pay for the service know the amount of legal liability the providers assume for keeping the secret shares.

What are threshold signatures?

Everything is Broken

Alice wants to create a cryptographic signature, but:

- ▶ No single piece of hardware is trusted
- ▶ No single service provider is trusted

But: Using t independent signature service providers might be ok!

If we need t providers, we probably should initially sign up with n providers so that we can still create signatures if only t/n are available...

2024-08-26

NEXT GENERATION INTERNET

└─What are threshold signatures?

└─Everything is Broken

Alice wants to create a cryptographic signature, but:

- ▶ No single piece of hardware is trusted
- ▶ No single service provider is trusted

But: Using t independent signature service providers might be ok!

If we need t providers, we probably should initially sign up with n providers so that we can still create signatures if only t/n are available...

1. GNU Anastasis still requires the user to have a secure client as the core secret originates from the user's computer and is fully restored on a computer presumed to be under the user's control. For applications involving encryption, this is probably the best practical solution today.
2. However, in practice, end-user devices may be compromised. For encryption keys, there is little we can do as the user will need to access their data on their system in the end.
3. But, for **signatures**, we can actually achieve security even if we cannot trust any individual piece of client hardware and also no single provider!
4. The solution here is to use **threshold signatures**, where t/n providers are required to contribute to create a signature.

Flexible Round-Optimized Schnorr Threshold (FROST) is a t -out-of- n threshold signature scheme:

- ▶ Distributed key generation protocol can be used to ensure private key is never stored on a single device
- ▶ t providers required to collaborate to create digital signature

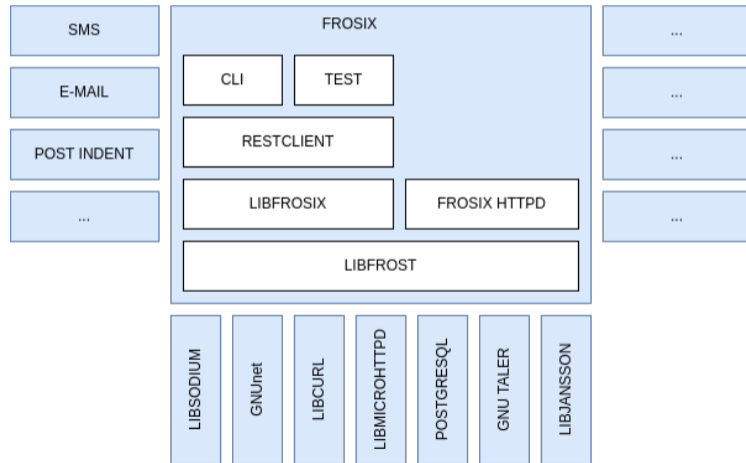
1. FROST is a modern threshold signature protocol that generates Schnorr signatures.
2. If we do not want to trust the client or any individual provider, distributed key generation must be used to collaboratively create private key shares and a master public key without ever having the private key available on any system involved.
3. We can create n private key shares, and then use $t \leq n$ key shares to create a signature that verifies against the master public key.
4. FROST only defines the cryptographic primitives, but is not a usable system by itself.

Free Software implementation for threshold signatures using FROST with:

- ▶ RESTful API to interact between signer and signing services
- ▶ Configurable authentication methods to authorize creation of signature
- ▶ Client should still use multiple devices (for authorization and to check distributed key generation) to avoid single point of failure
- ▶ Command-line tool to interact with FROSIX service providers

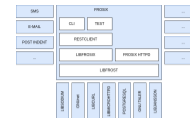
1. FROSIX is an effort to add proper authorization-procedures (in the same style as Anastasis) and a REST protocol to orchestrate the process from a client.
2. Given that we do not trust the client, a user would need to use multiple client devices, after all a compromised client might display that they have set up a distributed key signing system but only use a single compromised provider (or even only local computation!) behind the scenes/screens.
3. Thus, FROSIX has provisions to check which providers were involved in the key generation on other devices.

System components overview



└ What are threshold signatures?

└ System components overview



1. This is an overview of the FROSIX architecture, consisting of various supporting libraries to implement the REST service and payments below, the core cryptographic FROST implementation, and then the FROSIX protocol implementation.
2. Various authentication methods are implemented, but the system is extensible.
3. What is missing today is any kind of nice user interface, as only a command-line tool exists.

FROSIX: Future Work

Open issues:

- ▶ Support additional signature schemes beyond EdDSA
- ▶ Pay signature service providers for their service
- ▶ Graphical user interfaces (Gtk+, WebUI, ...)

2024-08-26

NEXT GENERATION INTERNET

└─What are threshold signatures?

└─FROSIX: Future Work

Open issues:
▶ Support additional signature schemes beyond EdDSA
▶ Pay signature service providers for their service
▶ Graphical user interfaces (Gtk+, WebUI, ...)

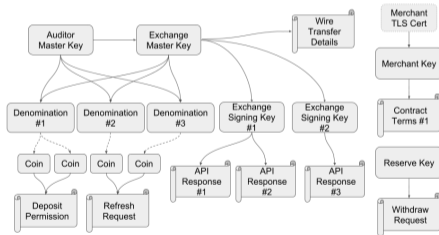
1. Public administrations are spending Billions today on digital signature solutions with highly questionable assurance levels.
2. The availability of a **usable** FLOSS alternative for digital signatures thus could provide massive savings. Currently (2024), a single digital signature with Swisscom costs **CHF 2.40**. Politically, such excessive costs are buried in large IT budgets and thus rarely discussed in budget negotiations.
3. Who is up to the challenge?

What does key management look like in practice?

Key management in GNU Taler

GNU Taler has many types of keys:

- ▶ Coin keys (EdDSA + ECDHE)
- ▶ Denomination keys (blind)
- ▶ Online message signing keys
- ▶ Offline key signing keys
- ▶ Merchant keys
- ▶ Auditor key
- ▶ Security module keys
- ▶ Transfer keys (ECDHE)
- ▶ Wallet keys
- ▶ *TLS keys, DNSSEC keys*



2024-08-26

NEXT GENERATION INTERNET

└─What does key management look like in practice?

└─Key management in GNU Taler

- GNU Taler has many types of keys:
- ▶ Coin keys (EdDSA + ECDHE)
 - ▶ Denomination keys (blind)
 - ▶ Online message signing keys
 - ▶ Offline key signing keys
 - ▶ Merchant keys
 - ▶ Auditor key
 - ▶ Security module keys
 - ▶ Transfer keys (ECDHE)
 - ▶ Wallet keys
 - ▶ TLS keys, DNSSEC keys

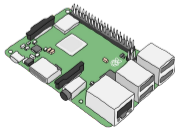


1. This is an overview of the different types of keys used in GNU Taler.
2. The diagram on the right gives an overview of which keys are used to sign what, and quite often keys are used to basically sign over other keys.
3. Most of the keys are signing keys using EdDSA.
4. But some keys are different also because the cryptography they used with is different, especially the transfer keys (ECDHE), the denomination keys (blind signatures) and the coin keys (EdDSA + ECDHE).
5. An actual Taler deployment will also have some key material outside of the core scope of GNU Taler, such as those for securing TLS and DNS.
6. The keys also differ by the party that owns them, and in some cases the protection level applied (at a high level: online vs. offline)

Offline keys

Both exchange and auditor use offline keys.

- ▶ Those keys must be backed up and remain highly confidential!
- ▶ We recommend that computers that have ever had access to those keys to NEVER again go online.
- ▶ We recommend using a Raspberry Pi for offline key operations. Store it in a safe under multiple locks and keys.
- ▶ Apply full-disk encryption on offline-key signing systems.
- ▶ Have 3–5 full-disk backups of offline-key signing systems.



└─What does key management look like in practice?

└─Offline keys

1. Offline keys are the most protected keys in the system.
2. They are used to sign or revoke online keys, and the public keys are distributed with the software to other users in the system.
3. Because they are offline, signing with them is extremely expensive, and may require waking up people in the middle of the night to get the signature. So they should be used rarely in operation, as otherwise their deliberately low availability might threaten the availability of the overall system.
4. HSMs, physical security (a physical vault), passwords and possibly threshold signatures are techniques to provide the highest possible level of protection for such offline keys.

Both exchange and auditor use offline keys.

- ▶ Those keys must be backed up and remain highly confidential!
- ▶ We recommend that computers that have ever had access to those keys to NEVER again go online.
- ▶ We recommend using a Raspberry Pi for offline key operations. Store it in a safe under multiple locks and keys.
- ▶ Apply full-disk encryption on offline-key signing systems.
- ▶ Have 3-5 full-disk backups of offline-key signing systems.



Online keys

The exchange needs RSA and EdDSA keys to be available for online signing.

- ▶ Knowledge of these private keys will allow an adversary to mint digital cash, possibly resulting in huge financial losses.
- ▶ The corresponding public keys are certified using Taler's public key infrastructure (which uses offline-only keys).

`taler-exchange-offline` can be used to **revoke** the online signing keys, if we find they have been compromised.

2024-08-26

NEXT GENERATION INTERNET

└─What does key management look like in practice?

└─Online keys

1. Some keys inherently need to be online, say because users frequently interact with them. For these online signing keys, we actually expect that a Taler operator at scale may use them tens of thousands of times per second.
2. Thus, the primary protection has to be that they are still used on a machine with tight logical and physical access control.
3. An HSM is theoretically an option, but today HSMs on the market rarely support blind signatures and are very expensive if you need them to do tens of thousands of signatures per seconds.

The exchange needs RSA and EdDSA keys to be available for online signing.

- ▶ Knowledge of these private keys will allow an adversary to mint digital cash, possibly resulting in huge financial losses.
- ▶ The corresponding public keys are certified using Taler's public key infrastructure (which uses offline-only keys).

`taler-exchange-offline` can be used to **revoke** the online signing keys, if we find they have been compromised.

Protecting online keys

The exchange needs RSA and EdDSA keys to be available for online signing.

- ▶ `taler-exchange-secmo`-* are the only processes that must have access to the private keys. These `secmod` processes should run under a different UID, but share the same GID with the exchange.
- ▶ The `secmods` generate the keys, allow `taler-exchange-httpd` to sign with them, and eventually delete the private keys.
- ▶ Communication between `secmods` and `taler-exchange-httpd` is via a UNIX domain socket.
- ▶ Online private keys are stored on disk (not in database!) and should NOT be backed up (RAID should suffice). If disk is lost, we can always create fresh replacement keys!

2024-08-26

NEXT GENERATION INTERNET

└─What does key management look like in practice?

└─Protecting online keys

1. GNU Taler today thus implements a different protection strategy, that is minimizing the amount of code that has the right to interact with the key, minimizing the surface to interact with that code, and running that code under a specific UID.
2. The entire key lifecycle management is implemented in these well-isolated and easy to audit helper processes: key creation, key use and key deletion.
3. Furthermore, while the private keys are stored on disk, it is conceivable to make that disk a RAM disk — assuming sufficient availability can be assured by someone signing replacement keys with the offline tool in a timely fashion after a loss of power!

The exchange needs RSA and EdDSA keys to be available for online signing.

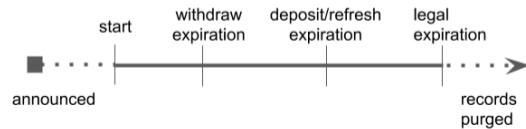
- ▶ `taler-exchange-secmo`-* are the only processes that must have access to the private keys. These `secmod` processes should run under a different UID, but share the same GID with the exchange.
- ▶ The `secmods` generate the keys, allow `taler-exchange-httpd` to sign with them, and eventually delete the private keys.
- ▶ Communication between `secmods` and `taler-exchange-httpd` is via a UNIX domain socket.
- ▶ Online private keys are stored on disk (not in database!) and should NOT be backed up (RAID should suffice). If disk is lost, we can always create fresh replacement keys!

What happens if private keys are disclosed or lost?

1. Especially for such online signing keys, we always have to do a threat model as to what could happen if control over these keys is lost.
2. For signing keys, the case where they are lost can imply downtime or (in the case of keys representing financial assets) a (hopefully small) financial loss to the owner.
3. What if signing keys used to create digital cash are disclosed to an adversary?

Denomination key (d, n) disclosure

- ▶ Auditor and exchange can detect this once the total number of deposits exceeds the number of legitimate coins.
 - ▶ At this point, (e, n) is *revoked*. Users of *unspent* legitimate coins reveal b from their withdrawal operation and obtain a *refund*.
 - ▶ The financial loss of the exchange is *bounded* by the number of legitimate coins signed with d .
- ⇒ Taler frequently rotates denomination signing keys and deletes d after the signing period of the respective key expires.



2024-08-26

NEXT GENERATION INTERNET

└─ What does key management look like in practice?

└─ Denomination key (d, n) disclosure

- ▶ Auditor and exchange can detect this once the total number of deposits exceeds the number of legitimate coins.
- ▶ At this point, (e, n) is *revoked*. Users of *unspent* legitimate coins reveal b from their withdrawal operation and obtain a *refund*.
- ▶ The financial loss of the exchange is *bounded* by the number of legitimate coins signed with d .
- ⇒ Taler frequently rotates denomination signing keys and deletes d after the signing period of the respective key expires.



1. An attacker who learns d can sign an arbitrary number of illicit coins into existence and deposit them.
2. Detecting that signatures were created outside of the control of the legitimate system is key. Consider also Certificate Transparency for X.509: we cannot prevent bad certificates from being signed, but at least we learn about it!
3. The next step is to assess the damage, and to stop further losses. Like in X.509, this involves revoking the compromised keys and possibly issuing new signatures to legitimate users.
4. Finally, we can try to limit the extent of the damage: the private key is deleted after the “withdraw” period expires, and signatures with it are only accepted up to a certain “deposit” time. This way, an adversary has limited windows of opportunity.

References I

 Deirdre Connolly, Chelsea Komlo, Ian Goldberg, and Christopher A. Wood.

Two-round threshold schnorr signatures with frost.
Technical report, IRTF, 2023.

<https://datatracker.ietf.org/doc/draft-irtf-cfrg-frost/>.

 Dominik Samuel Meister and Dennis Neufeld.

Anastasis: Password-less key recovery via multi-factor multi-party authentication.

Master's thesis, Bern University of Applied Sciences, June 2020.

2024-08-26

NEXT GENERATION INTERNET

└References

└References

 Deirdre Connolly, Chelsea Komlo, Ian Goldberg, and Christopher A. Wood.
Two-round threshold schnorr signatures with frost.
Technical report, IRTF, 2023.
<https://datatracker.ietf.org/doc/draft-irtf-cfrg-frost/>.

 Dominik Samuel Meister and Dennis Neufeld.
Anastasis: Password-less key recovery via multi-factor multi-party authentication.
Master's thesis, Bern University of Applied Sciences, June 2020.

Acknowledgements

Co-funded by the European Union (Project 101135475).



Co-funded by
the European Union

Co-funded by SERI (HEU-Projekt 101135475-TALER).

Project funded by



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs,
Education and Research EAER
**State Secretariat for Education,
Research and Innovation SERI**

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Key management

Christian Grothoff

Using GNU Anastasis

1. Installing anastasis-gtk

To install binaries on Debian or Ubuntu, follow the first steps in section 3.3.1 or 3.3.3 of <https://docs.taler.net/taler-merchant-manual.html> to add the `deb.taler.net` repository to your system. Then, you can just:

```
# apt install anastasis-gtk
```

You could of course also try to build the system from source, starting with its dependencies.

Alternatively, but **not** as securely, you can use <https://webui.anastasis.lu/> in your browser.

2. Backup and recovery

Use GNU Anastasis to back up a secret (does not have to be a real one) and to recover it. The GUI will check some of your identity data client-side for validity.

NEXT GENERATION INTERNET

Availability

Christian Grothoff

DD.MM.YYYY

2024-08-26

NEXT GENERATION INTERNET

NEXT
GENERATION
INTERNET

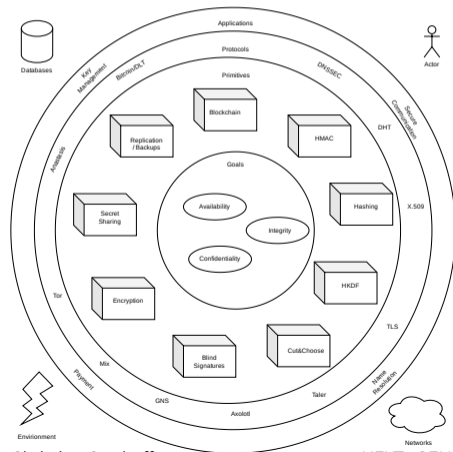
Availability

Christian Grothoff

DD.MM.YYYY

1. Availability is probably the most frequently neglected key security objective in security courses.
2. At the same time, it is the one we really cannot ignore, and that is hard to address.
3. DDoS attacks sometimes make the news, but even rapid customer growth can be a problem.
4. For example, some medical testing businesses saw rapid growth during the COVID pandemic, but their IT systems were not prepared for even just a 10x spike in traffic! Dealing with such problems only when they happen is stressful and could be costly.

Availability on our map



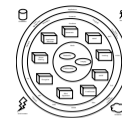
Christian Grothoff

NEXT , GENERATION , INTERNET

2024-08-26

NEXT GENERATION INTERNET

Availability on our map



1. Availability is one of the core security goals, but unlike confidentiality and integrity it is one that cryptography does not directly help with.
2. To ensure availability, we need to look at the entire system: hardware, network, databases, staff
3. Related topics on the map are secret sharing, replication, backups, and blockchains.

Learning objectives

How to architect systems for high-availability?

What are specific considerations for the network setup?

What are specific considerations for HTTP servers?

What are specific considerations for the application logic?

What are specific considerations for databases?

What are specific considerations for monitoring?

How to know your limits?

2024-08-26

NEXT GENERATION INTERNET

Learning objectives

1. In this lecture we will look at how to prepare our systems for high-availability, which includes high scalability and defense-in-depth, because a system that is down due to a security incident is of course also unavailable.
2. Security is holistic, so we will look at what we should consider across the entire stack.
3. As always, we will use the GNU Taler architecture to serve as a running example.

How to architect systems for high-availability?
What are specific considerations for the network setup?
What are specific considerations for HTTP servers?
What are specific considerations for the application logic?
What are specific considerations for databases?
What are specific considerations for monitoring?
How to know your limits?

How to architect systems for high-availability?

High-availability principles

Use:

- ▶ redundancy at all levels:
 - ▶ RAID (5, 1+1, 1+1+1), ECC RAM, redundant power supplies, ...
 - ▶ redundant servers
 - ▶ multiple power sources (grid, generator, battery, UPS)
 - ▶ multiple data centers
- ▶ a layered architecture

2024-08-26

NEXT GENERATION INTERNET

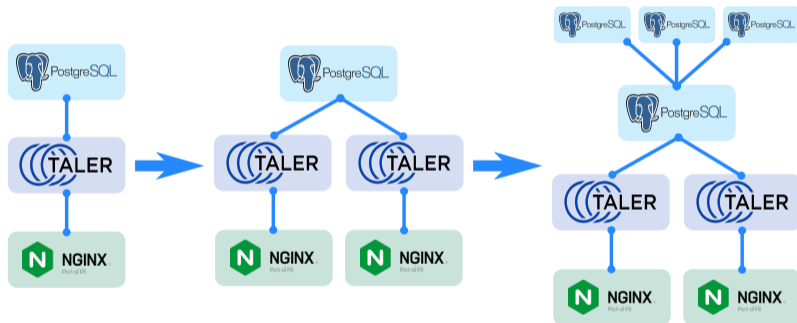
└─ How to architect systems for high-availability?

└─ High-availability principles

- Use:
- ▶ redundancy at all levels:
 - ▶ RAID (5, 1+1, 1+1+1), ECC RAM, redundant power supplies, ...
 - ▶ redundant servers
 - ▶ multiple power sources (grid, generator, battery, UPS)
 - ▶ multiple data centers
 - ▶ a layered architecture

1. A modern IBM mainframe has CPUs with redundant cores, so that if one CPU core is detected to fail, another CPU core can take over on-the-spot.
2. Modern mainframes also have redundant power cables running into the system from two sides, so if someone stumbles over a cable on one side, it does not unplug the cable on the other side.
3. A layered architecture enables both defense-in-depth as an attacker has to get through the layers, and also prepares for horizontal scaling as the different layers can serve to multiplex the load onto different systems.

Horizontal distribution



2024-08-26

NEXT GENERATION INTERNET

└─ How to architect systems for high-availability?

└─ Horizontal distribution



1. Taler here simply stands symbolically for basically any typical REST-based application.
2. The goal is to be **able** to scale by moving towards the right, adding more resources.
3. Early on, you may put all three components even onto the same host (but use different system users). The goal of modern architecture is to scale as necessary!
4. Moving towards the right increases latency between the components, which can **hurt** performance. Measure how your system reacts to such latency increases **early**.

High-availability principles

- ▶ Know your (performance) targets and limits
- ▶ Know (monitor) your load (and availability)
- ▶ Know your interactions and dependencies

2024-08-26

NEXT GENERATION INTERNET

└─ How to architect systems for high-availability?

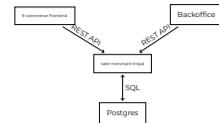
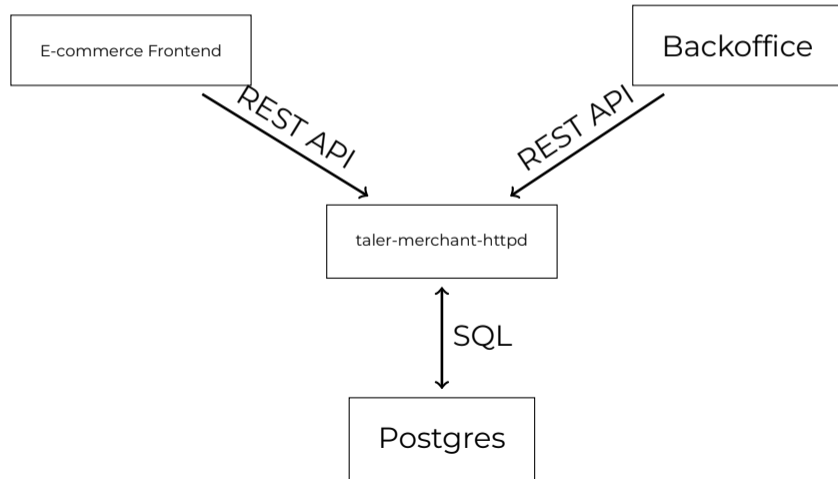
└─ High-availability principles

- ▶ Know your (performance) targets and limits
- ▶ Know (monitor) your load (and availability)
- ▶ Know your interactions and dependencies

1. Management may give you high-level performance targets, but this is also about knowing the performance of each part, for example cryptographic signature routines or database queries. What are the limits, and when will you be in trouble?
2. You also need to know what your current actual load is, so that you know how much headroom you still have. Make sure you have enough so you can plan, implement and evaluate counter-measures before problems materialize for users!
3. High-availability also means that you know your interactions. If external systems you depend upon are down, having redundancy for your own systems is moot. Knowing the interactions between components (internal or external) also helps you understand the access control logic of your system.

Taler example

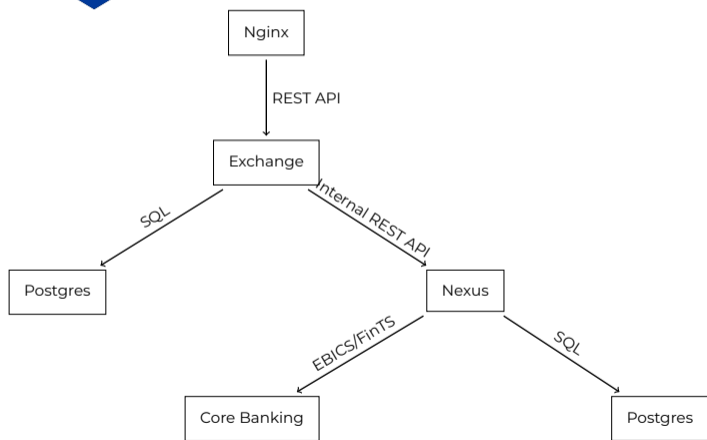
Merchant architecture



1. This is a very canonical example of a simple REST architecture.
2. We have the core application logic, which is backed by an SQL database.
3. The core application logic has two groups of users, the general public and the service administrators.
4. Both access the core system via a REST API. The access control may vary between the endpoints, from being completely public to being restricted via various access control mechanisms.
5. Often a **reverse proxy** is added, for example to host many different services (and domain names) behind the same global IP address.

Taler example

Bank architecture



2024-08-26

NEXT GENERATION INTERNET

└─ How to architect systems for high-availability?

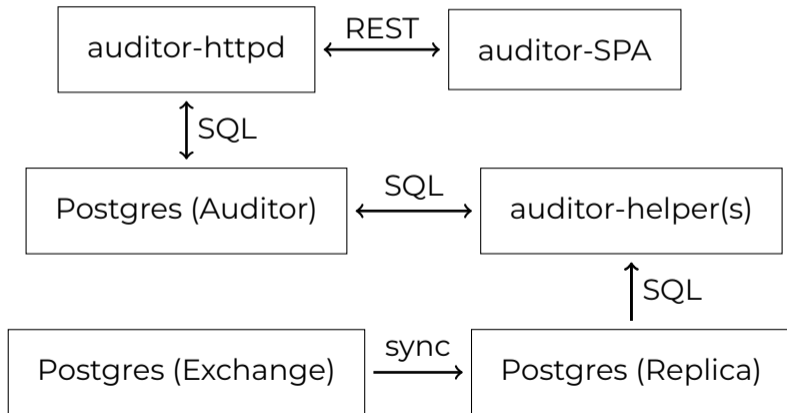
└─ Taler example



1. This is a realistic example for a more complex architecture that shows a security engineer the key interactions between components. It also achieves a bit of defense-in-depth.
2. The Nginx is where the external traffic comes in. The application logic for the Taler payment system has its own database.
3. The interaction with the core banking system is facilitated by Nexus, which has its own (much smaller) REST API.
4. The core banking system uses EBICS, a traditional core banking API.
5. An attacker would have to pass through several layers to even get near the core banking system. Each layer runs as a different user, possibly on a different computer, and also blocks malformed requests and ensures only a fraction of requests are even passed down.

Taler example

Auditor architecture

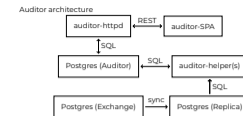


2024-08-26

NEXT GENERATION INTERNET

└ How to architect systems for high-availability?

└ Taler example



1. The GNU Taler auditor uses a slightly more interesting architecture where we have some Web interface above the application logic (auditor-httpd) that primarily shows the status of the audit. The auditor-httpd also has a few public endpoints that can be used by other Taler components to provide input for the auditor.
2. To do the audit, the auditor requires a copy of the Exchange database. This is typically provided via asynchronous database replication over a network. This way, the auditor has its own full copy locally and has some control over the schema.
3. Helper processes are used for the actual auditing logic. They feed from both databases and store the result of their audit into the auditor database to be exported via the auditor-httpd.

What are specific considerations for the network setup?

Network setup for availability

- ▶ IPv4 + IPv6 dual stack: some users today are only on IPv4, others only on IPv6
- ▶ Good data centers have multiple, redundant up-links via different providers
- ▶ Data centers should have backups for everything: cooling, battery power, fuel for on-site generators, etc.
- ▶ Even better are multiple data centers; data centers do burn...
- ▶ High-availability hosters monitor for disasters and migrate operations out of dangerous areas

2024-08-26

NEXT GENERATION INTERNET

└─What are specific considerations for the network setup?

└─Network setup for availability

- ▶ IPv4 + IPv6 dual stack: some users today are only on IPv4, others only on IPv6
- ▶ Good data centers have multiple, redundant up-links via different providers
- ▶ Data centers should have backups for everything: cooling, battery power, fuel for on-site generators, etc.
- ▶ Even better are multiple data centers; data centers do burn...
- ▶ High-availability hosters monitor for disasters and migrate operations out of dangerous areas

1. Especially with IPv6 there are still commonly routing outages. It thus can be helpful to have multiple IPv6 addresses reachable via different routes!
2. Naturally, not all of these apply to small-scale operations, but if you are operating critical infrastructure, this matters.

DNS is critical

Microsoft learned the hard way, twice

2024-08-26

NEXT GENERATION INTERNET

└ What are specific considerations for the network setup?

└ DNS is critical



Network Connectivity

Engineers are currently investigating DNS resolution issues affecting network connectivity to Azure services. More information will be provided as it becomes available.

Refresh every

2 minutes

Good Warning Error Information

PRODUCTS AND SERVICES	Americas												
	NON-REGIONAL*	EAST US	EAST US 2	CENTRAL US	NORTH CENTRAL US	SOUTH CENTRAL US	WEST CENTRAL US	WEST US	WEST US 2	CANADA EAST	CANADA CENTRAL	BRAZIL SOUTH	
IMPACTED SERVICES													
Network Infrastructure	✖	✖	✖	✖	✖	✖	✖	✖	✖	✖	✖	✖	
COMPUTE													
Azure VMware Solution by CloudSimple		✔						✔					
Virtual Machines		✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	✔	

1. In 2016 and again 2019 Microsoft experienced global service outages due to DNS failures.
2. Ensuring DNS is correctly configured and **equally redundant** as your main operational services is critical.

Firewalls?

- ▶ Firewalls add complexity, misconfiguration, hardware- and software-failures can all harm availability
 - ▶ Deep packet inspection firewalls parse all types of untrusted input, and they must do so quickly and often with high privileges; this is one of the most problematic types of software one could ever use
 - ▶ An attacker who successfully takes over your firewall does not merely defeat this layer of security, but is now in a great position to monitor your network.
- ⇒ Firewalls are dangerous.

2024-08-26

NEXT GENERATION INTERNET

└─What are specific considerations for the network setup?

└─Firewalls?

1. That firewalls create all of these issues is well-known. Why are they then used?
2. Possible answers: compliance, cover-my-ass, lack of knowledge about alternatives, insecure client systems being papered over

- ▶ Firewalls add complexity, misconfiguration, hardware- and software-failures can all harm availability
- ▶ Deep packet inspection firewalls parse all types of untrusted input, and they must do so quickly and often with high privileges; this is one of the most problematic types of software one could ever use
- ▶ An attacker who successfully takes over your firewall does not merely defeat this layer of security, but is now in a great position to monitor your network.
- ⇒ Firewalls are dangerous.

Good network access control

- ▶ First, only use IP if needed. UNIX domain sockets are faster and more secure for inter-process communication on the same host! You can also run HTTP over a UNIX domain socket.
- ▶ Second, if software does not support UNIX domain sockets, at least binding to loopback (::1) should be widely supported. Use `netstat -np1` to check which addresses local services are bound to.
- ▶ If network access is required but only from a particular host, *maybe* a simple host-based firewall can be useful as an additional security layer.
- ▶ Alternatively, use TLS and/or a WireGuard VPN to build a secure tunnel. TLS client-certificates could be used to authenticate the client.

2024-08-26

NEXT GENERATION INTERNET

└─What are specific considerations for the network setup?

└─Good network access control

1. Regardless of what network access control mechanism is implemented, security should start with the application itself being secure. Even restricted to a UNIX domain socket or otherwise locked down, the application should treat inputs as untrusted, parse them carefully and if possible minimize the rights granted to the code that interacts with the network.
2. Even network services is written in a “safe” language can still have plenty of security problems (see: log4j).

- ▶ First, only use IP if needed. UNIX domain sockets are faster and more secure for inter-process communication on the same host! You can also run HTTP over a UNIX domain socket.
- ▶ Second, if software does not support UNIX domain sockets, at least binding to loopback (::1) should be widely supported. Use `netstat -np1` to check which addresses local services are bound to.
- ▶ If network access is required but only from a particular host, *maybe* a simple host-based firewall can be useful as an additional security layer.
- ▶ Alternatively, use TLS and/or a WireGuard VPN to build a secure tunnel. TLS client-certificates could be used to authenticate the client.

- ▶ DNS with high TTL values improves caching and performance, but may increase down-time for unplanned migrations to new IP addresses
- ▶ Including multiple A/AAAA records and randomizing their order in DNS responses can be used for load-balancing among front-end servers
- ▶ DNSSEC is the only viable defense against DNS cache poisoning attacks. Use it to ensure your users are not sent elsewhere by DNS!

└─What are specific considerations for the network setup?

└─DNS and DNSSEC

1. Advanced DNS setups can even adapt the responses to the current load experienced on front-end servers.
2. DNSSEC does not provide end-to-end security, but DNS cache poisoning is a bigger threat. Users may use DoH or DoT to secure the connection to their recursive resolver, but this is outside of your control anyway.
3. Due to lack of end-to-end security and implementation support, DANE is pretty much useless, so we still need X.509 CAs.

- ▶ DNS with high TTL values improves caching and performance, but may increase down-time for unplanned migrations to new IP addresses
- ▶ Including multiple A/AAAA records and randomizing their order in DNS responses can be used for load-balancing among front-end servers
- ▶ DNSSEC is the only viable defense against DNS cache poisoning attacks. Use it to ensure your users are not sent elsewhere by DNS!

X.509 considerations

- ▶ Extended validation is largely useless, as users are very unlikely to take note.
- ▶ Use HTTP Strict Transport Security (HSTS) to force clients to always use HTTPS.
- ▶ Use tools like **certspotter** to detect TLS certificates issued for your domain.

2024-08-26

NEXT GENERATION INTERNET

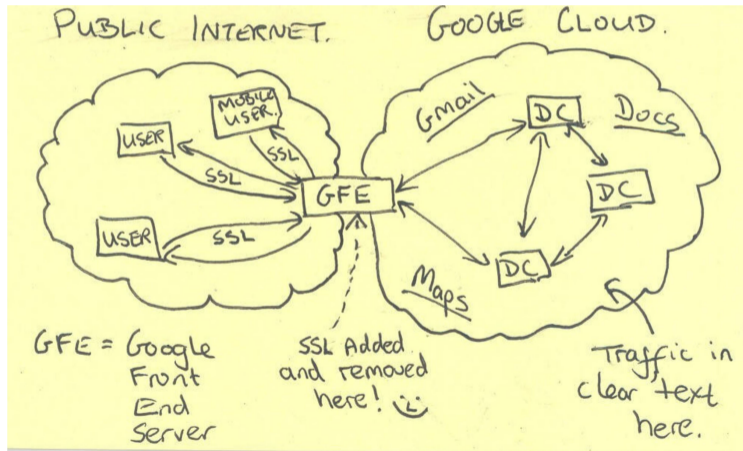
└─What are specific considerations for the network setup?

└─X.509 considerations

1. Ideally, get your site into the HSTS preload list to ensure browsers use HTTPS from the very first connection.
2. Certspotter builds on certificate transparency, where CAs must post certificates they issue for a domain.

- ▶ Extended validation is largely useless, as users are very unlikely to take note.
- ▶ Use HTTP Strict Transport Security (HSTS) to force clients to always use HTTPS.
- ▶ Use tools like **certspotter** to detect TLS certificates issued for your domain.

Don't be one of these guys...

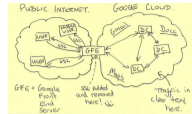


2024-08-26

NEXT GENERATION INTERNET

└ What are specific considerations for the network setup?

└ Don't be one of these guys...



1. This is a graphic of the NSA from the Snowden archive. It shows the Google front-end servers performing TLS termination and then traffic flowing in cleartext between Google's data centers.
2. Despite Google also giving the NSA front-door access to their data via the PRISM program, the NSA **additionally** spied on Google's unencrypted internal traffic.
3. Thus, while it is normal to do TLS-termination on front-end servers, you should encrypt all non-loopback traffic even within internal networks.
4. As internal TLS sessions can be long-lived and symmetric cryptography is extremely cheap, this is primarily a key management issue and **not** a performance issue.

What are specific considerations for HTTP servers?

HTTP service considerations

- ▶ Cache-control is your friend, both for performance and to make CDNs effective against DDoS.
- ▶ Ensure your applications enable caching with long durations and good ETags whenever possible.
- ▶ Defending against a DDoS requires enough bandwidth and computational power to handle requests. You can get bandwidth from a CDN for resources that can be cached. Make sure you have enough computational resources to handle non-cachable requests!
- ▶ Scalable system architecture is key: ideally you can add resources only for the duration of the DDoS.
- ▶ In addition or as an alternative to DNS-based load balancing, HTTP(S) reverse proxies can also be used for load balancing.

2024-08-26

NEXT GENERATION INTERNET

└─What are specific considerations for HTTP servers?

└─HTTP service considerations

1. Good Etags use values that the server can check quickly without having to build the actual response. Thus, a cryptographic hash of the response is often not the best answer.
2. You can simplify your cache control configuration by serving static resources from particular paths.
3. Sometimes it can be useful to not return complex answers in response to a POST; instead re-direct the POSTer to a GET request where the answer can be cached!
4. Rate-limiting IP addresses that send too many (expensive) requests is only a last resort: advanced attackers could have many IP addresses, and CG-NAT may route many legitimate users via the same IP address!

- ▶ Cache-control is your friend, both for performance and to make CDNs effective against DDoS.
- ▶ Ensure your applications enable caching with long durations and good ETags whenever possible.
- ▶ Defending against a DDoS requires enough bandwidth and computational power to handle requests. You can get bandwidth from a CDN for resources that can be cached. Make sure you have enough computational resources to handle non-cachable requests!
- ▶ Scalable system architecture is key: ideally you can add resources only for the duration of the DDoS.
- ▶ In addition or as an alternative to DNS-based load balancing, HTTP(S) reverse proxies can also be used for load balancing.

- ▶ Simplistic benchmarks like `ab` or `h2load` mostly measure your HTTP server, which is rarely the bottleneck.
- ▶ Good benchmarking requires generating **realistic** load across your API. So use your actual client applications, or at least derive traffic patterns from real-world load on your system.
- ▶ It is **also** useful to do a worst-case analysis, trying to find out what the most expensive requests are, especially in case an attacker tries to take you down based on those.

1. Modern HTTP servers can easily do 100,000 requests per second if the application logic is simple enough (like serving a static file). Often tools struggle to generate such high request loads. Make sure you know if you are already measuring your server and not still your load generator.
2. Requests that are often slow are those executing complex search queries or filters on large amounts of data in the database. Responses to such requests can generally be cached.

- ▶ Simplistic benchmarks like `ab` or `h2load` mostly measure your HTTP server, which is rarely the bottleneck.
- ▶ Good benchmarking requires generating **realistic** load across your API. So use your actual client applications, or at least derive traffic patterns from real-world load on your system.
- ▶ It is **also** useful to do a worst-case analysis, trying to find out what the most expensive requests are, especially in case an attacker tries to take you down based on those.

What are specific considerations for the application logic?

Scaling application logic

- ▶ Vertical scaling (using multiple CPU cores) is the obvious solution for improving application scalability.
- ▶ Multi-threading significantly increases application complexity, and also does not work for horizontal scaling. Thus, it should be used rarely (such as for expensive cryptographic operations) if at all!
- ▶ Using multiple **processes** (one per core) is marginally more expensive, but much simpler, works for vertical and horizontal scaling, and generally more secure!

2024-08-26

NEXT GENERATION INTERNET

└─ What are specific considerations for the application logic?

└─ Scaling application logic

1. Avoid threads. Avoid threads. Use processes. Processes are cheap.
2. We will talk about IPC to coordinate between processes a bit later.

- ▶ Vertical scaling (using multiple CPU cores) is the obvious solution for improving application scalability.
- ▶ Multi-threading significantly increases application complexity, and also does not work for horizontal scaling. Thus, it should be used rarely (such as for expensive cryptographic operations) if at all!
- ▶ Using multiple **processes** (one per core) is marginally more expensive, but much simpler, works for vertical and horizontal scaling, and generally more secure!

Architect for least privilege

- ▶ Software engineers often debate between micro-services and monoliths. Both are wrong!
 - ▶ Monoliths are bad because they tightly couple too many business concerns.
 - ▶ Micro-services are bad because they are too simple and thus require complex orchestration.
- ⇒ Instead, architect for least privilege: if a business concern requires specific rights distinct from other business concerns, that is a good reason to move it into a separate component!

2024-08-26

NEXT GENERATION INTERNET

└─What are specific considerations for the application logic?

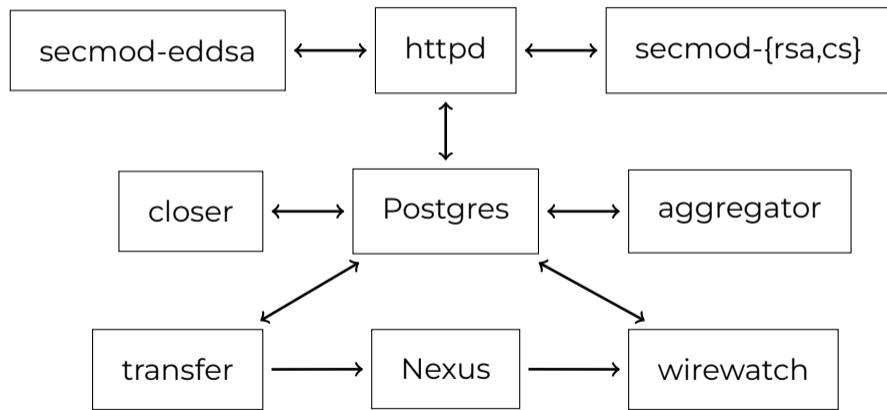
└─Architect for least privilege

- ▶ Software engineers often debate between micro-services and monoliths. Both are wrong!
- ▶ Monoliths are bad because they tightly couple too many business concerns.
- ▶ Micro-services are bad because they are too simple and thus require complex orchestration.
- ⇒ Instead, architect for least privilege: if a business concern requires specific rights distinct from other business concerns, that is a good reason to move it into a separate component!

1. Another dividing line is eliminating dead functionality. If some logic is only needed some of the time or for some deployments, it is again better to keep it separate. That way, attacks against that logic cannot impact users that do not even need it!
2. `curl` is a great example of getting it wrong: the monolith library bundles dozens of protocols (and as a result has historically had countless vulnerabilities) instead of using a modular architecture where applications can specify which protocols to support and then only integrating those.
3. GNU `libextractor` does the opposite: each of dozens of file formats supported by the library is provided by a separate plugin (and run in a separate process); applications can specify which plugins they want to use. Those that are not desired are not even loaded into memory.

Taler example

Exchange architecture

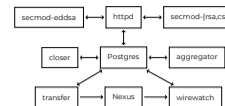


2024-08-26

NEXT GENERATION INTERNET

└─ What are specific considerations for the application logic?

└─ Taler example



1. A GNU Taler exchange deployment consists of many processes where most interact via a shared Postgres database.
2. Additionally, the “secmod” processes are minimal code bases that are the only ones with access to private online signing keys. They run under a different UNIX user and the main `httpd` communicates with them via a UNIX domain socket that is only available to a particular UNIX group.
3. The `wirewatch` and `transfer` tools also run as separate UNIX users: they are the only ones with access to the credentials to interact with the core banking system (here: Nexus).
4. The `aggregator` and `closer` processes only need database access. `aggregator`, `closer`, `transfer` and `wirewatch` can be stopped with limited impact on the core payment logic.

Minor havoc should be mandatory...

- ▶ Havoc is a technique where faults are regularly injected into a system to ensure programmers develop fault-tolerant software
- ▶ `systemd` can and should be used to limit service process lifetimes (to say 1h). This prevents minor memory leaks and memory fragmentation issues from becoming relevant in production. Postgres also does **not** like long-lived client connections.
- ▶ Ensure to configure `systemd` to **auto-restart** services (also good after crashes).

2024-08-26

NEXT GENERATION INTERNET

└─ What are specific considerations for the application logic?

└─ Minor havoc should be mandatory...

1. Limiting the lifetime of each service process by design is generally great, but raises the question of how to handle ongoing requests during the restart.
2. Ideally, we want zero down-time, even while the service restarted!

- ▶ Havoc is a technique where faults are regularly injected into a system to ensure programmers develop fault-tolerant software
- ▶ `systemd` can and should be used to limit service process lifetimes (to say 1h). This prevents minor memory leaks and memory fragmentation issues from becoming relevant in production. Postgres also does **not** like long-lived client connections.
- ▶ Ensure to configure `systemd` to **auto-restart** services (also good after crashes).

Taler's fork-close-systemd-listen trick

- ▶ Systemd can `listen` on a socket and pass the already open listen socket to an application process. This has the advantage that the listen socket remains open (and the kernel answers SYN packets) even if the application process is briefly down.
- ▶ When a Taler service is asked (`SIGTERM`) to terminate, it `closes` its listen socket(s), forks and the child continues to handle active clients and exits once those requests are finished. The original process exits immediately.

2024-08-26

NEXT GENERATION INTERNET

└─What are specific considerations for the application logic?

└─Taler's fork-close-systemd-listen trick

- ▶ Systemd can `listen` on a socket and pass the already open listen socket to an application process. This has the advantage that the listen socket remains open (and the kernel answers SYN packets) even if the application process is briefly down.
- ▶ When a Taler service is asked (`SIGTERM`) to terminate, it `closes` its listen socket(s), forks and the child continues to handle active clients and exits once those requests are finished. The original process exits immediately.

1. systemd listening ensures that a socket is always in listen mode and no new clients are ever rejected.
2. By forking and exiting, systemd sees the service process terminating and can launch a new service on the same listen socket, allowing the new process to pick up new clients.
3. The child process ensures that clients that are already interacting with the service also do not experience any failures from the hand-over to the new process.
4. The overall logic is very simple to implement (in native code).

What are specific considerations for databases?

Database performance

- ▶ Database performance is largely determined by writing good queries that make good use of indices.
- ▶ Indices can also be expensive. Learn about **partial indices**.
- ▶ When using a separate database host, and especially when sharding, the latency to the database can matter. Here, using **stored procedures** instead of multiple SQL statements can minimize performance issues arising from latency.

2024-08-26

NEXT GENERATION INTERNET

└─What are specific considerations for databases?

└─Database performance

- ▶ Database performance is largely determined by writing good queries that make good use of indices.
- ▶ Indices can also be expensive. Learn about **partial indices**.
- ▶ When using a separate database host, and especially when sharding, the latency to the database can matter. Here, using **stored procedures** instead of multiple SQL statements can minimize performance issues arising from latency.

1. We will not talk about **prepared statements** today. Everybody knows that one must **exclusively** use prepared statements, right?
2. A partial index is one with a `WHERE` clause so that only those items that satisfy the `WHERE` clause are indexed. Useful if the queries using the index have a similar clause.

Database versioning

- ▶ Database schema **will** evolve over time as application requirements change
- ▶ Database versioning should be used to track the current state of the database schema.
- ▶ The database version should be stored **within** the database itself and updated with the transaction that does the schema migration.
- ▶ <https://www.depesz.com/2010/08/22/versioning/> has code for a good simple approach.

2024-08-26

NEXT GENERATION INTERNET

└─What are specific considerations for databases?

└─Database versioning

1. Keeping the version within the same database has the key advantage that it can be bumped if and only if the transaction that changes the schema commits.
2. It is recommended to primarily version the schema changes (ALTER TABLE, etc.).
3. For stored procedures, a simple approach can be to just re-load the latest version of all stored procedures after migrating to a new schema.

- ▶ Database schema **will** evolve over time as application requirements change
- ▶ Database versioning should be used to track the current state of the database schema.
- ▶ The database version should be stored **within** the database itself and updated with the transaction that does the schema migration.
- ▶ <https://www.depesz.com/2010/08/22/versioning/> has code for a good simple approach.

Least privilege ...

... also applies to databases

- ▶ GRANT only the required rights to each DB client
- ▶ In particular, schema updates should probably only be done by the DB owner

2024-08-26

NEXT GENERATION INTERNET

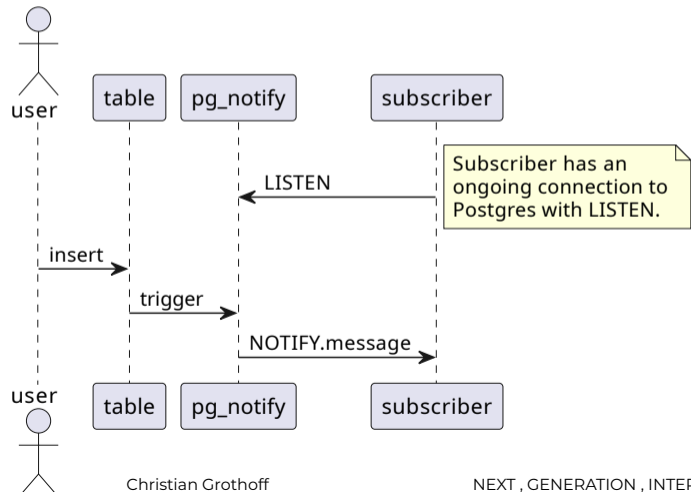
└─What are specific considerations for databases?

└─Least privilege ...

- ▶ GRANT only the required rights to each DB client
- ▶ In particular, schema updates should probably only be done by the DB owner

1. Determining the right grants for non-trivial systems is rather tedious.
2. I would love to learn about tools to automatically derive the right GRANTS given a list of prepared statements.

Postgres as a pub-sub service

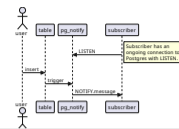


2024-08-26

NEXT GENERATION INTERNET

What are specific considerations for databases?

Postgres as a pub-sub service



1. The key pattern for the subscriber is to first LISTEN, then SELECT for the data as notifications might have been missed. Only INSERTs in transactions after the SELECT are assured to NOTIFY.
2. NOTIFY can be done via a trigger, but also explicitly as say a prepared statement.
3. NOTIFY can carry a message which is then provided to the subscriber.

Postgres as a pub-sub service

Implementation concerns

- ▶ In some programming languages, LISTEN must be done in a separate database connection.
- ▶ NOTIFY and LISTEN are very high performance operations.
- ▶ NOTIFY is transactional, so notification is guaranteed.
- ▶ LISTEN and NOTIFY are great mechanisms for inter-process communication between multiple processes that use the same database!

2024-08-26

NEXT GENERATION INTERNET

└ What are specific considerations for databases?

└ Postgres as a pub-sub service

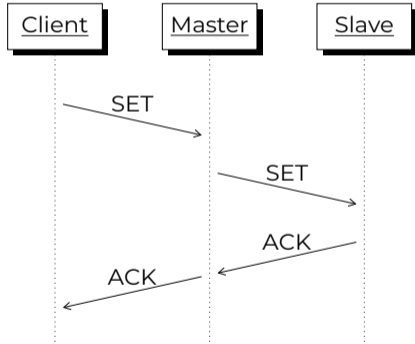
- ▶ In some programming languages, LISTEN must be done in a separate database connection.
- ▶ NOTIFY and LISTEN are very high performance operations.
- ▶ NOTIFY is transactional, so notification is guaranteed.
- ▶ LISTEN and NOTIFY are great mechanisms for inter-process communication between multiple processes that use the same database!

1. Especially JDBC-based languages seem to have trouble with nicely supporting LISTEN.
2. Using LISTEN and NOTIFY to coordinate many processes that use the same database has the advantage that a NOTIFY can trivially be sent to many LISTENers, and that LISTENers can decide which events they actually care about.
3. This even works when the processes are not on the same physical machine and share the same database over the network.
4. Processes + NOTIFY + LISTEN = great horizontal and vertical scaling!

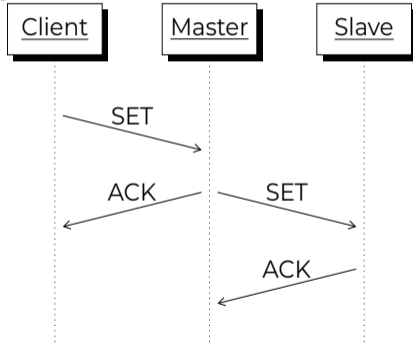
Database backups

Synchronous vs. asynchronous replication

Synchronous:



Asynchronous:



2024-08-26

NEXT GENERATION INTERNET

└─What are specific considerations for databases?

└─Database backups



1. Use synchronous replication **within** the same data center.
2. Usually, asynchronous replication to a **remote** data centers is best: you can probably tolerate a few transactions being lost if an entire data center is nuked, and you do not want to see high latency spikes or hard unavailability on network troubles between data centers.

What are specific considerations for monitoring?

Network monitoring

Monitor at least:

- ▶ IP connectivity
- ▶ DNS resolution
- ▶ Service availability

and if possible from **external** infrastructure.

Automatically **notify** staff of incidents.

2024-08-26

NEXT GENERATION INTERNET

└─What are specific considerations for monitoring?

└─Network monitoring

Monitor at least:

- ▶ IP connectivity
- ▶ DNS resolution
- ▶ Service availability

and if possible from **external** infrastructure.
Automatically **notify** staff of incidents.

1. The notification should not rely on any of your core infrastructure working.
2. Have ways to communicate with key staff independent of your infrastructure (say via SMS).

Monitoring latency

Application latency is often best monitored at the HTTP reverse proxy:

- ▶ reverse proxies widely support logging request latency
- ▶ the most interesting information – which endpoint – is available
- ▶ network latency to the client is excluded, which is good as it is usually outside of your control
- ▶ no need to send telemetry data from the client (less bandwidth, more reliable, simpler architecture, better for privacy)

2024-08-26

NEXT GENERATION INTERNET

└─What are specific considerations for monitoring?

└─Monitoring latency

Application latency is often best monitored at the HTTP reverse proxy:

- ▶ reverse proxies widely support logging request latency
- ▶ the most interesting information – which endpoint – is available
- ▶ network latency to the client is excluded, which is good as it is usually outside of your control
- ▶ no need to send telemetry data from the client (less bandwidth, more reliable, simpler architecture, better for privacy)

1. Unusually latency is a good indicator that something is wrong. Look no further than the xz malware patch detected by 500 ms of additional latency.
2. Bad configurations, high load, targeted attacks and DDoS all **could** show up via latency measurements.
3. Plus, low latency is usually a key business objective.

Application-specific monitoring

- ▶ Design your applications to enable monitoring.
- ▶ Log ERRORS if interventions are urgent, WARNINGS if investigations are in order.
- ▶ Have counters for key events.
- ▶ Export application-specific performance metrics.

2024-08-26

NEXT GENERATION INTERNET

└─What are specific considerations for monitoring?

└─Application-specific monitoring

1. Logs are only useful if somebody actually reads them.

- ▶ Design your applications to enable monitoring.
- ▶ Log ERRORS if interventions are urgent, WARNINGS if investigations are in order.
- ▶ Have counters for key events.
- ▶ Export application-specific performance metrics.

Monitor for slow queries

- ▶ Use “slow query” logging of your database to detect problematic queries and then ANALYZE them.
- ▶ Query optimizers can be fickle: a query may perform well at first, but then due to changes in database-internal metrics a query optimizer may switch to a much **worse** execution plan.

2024-08-26

NEXT GENERATION INTERNET

└─ What are specific considerations for monitoring?

└─ Monitor for slow queries

- ▶ Use “slow query” logging of your database to detect problematic queries and then ANALYZE them.
- ▶ Query optimizers can be fickle: a query may perform well at first, but then due to changes in database-internal metrics a query optimizer may switch to a much **worse** execution plan.

1. When monitoring slow queries, start “high” at like 1 second, but once you have addressed the slow queries, lower the threshold. Depending on the application, thresholds in the range of a few milliseconds could even be appropriate!

System-level monitoring

On each host, you want to monitor:

- ▶ System load (CPU load, memory utilization, disk utilization, disk IO load, bandwidth)
- ▶ Running processes (total, application specific services, state)
- ▶ Core dumps / crashes
- ▶ Other metrics (depending on your application)

2024-08-26

NEXT GENERATION INTERNET

└─What are specific considerations for monitoring?

└─System-level monitoring

On each host, you want to monitor:

- ▶ System load (CPU load, memory utilization, disk utilization, disk IO load, bandwidth)
- ▶ Running processes (total, application specific services, state)
- ▶ Core dumps / crashes
- ▶ Other metrics (depending on your application)

1. Disk full or out-of-memory are a very bad reasons for down time.
2. Unusual changes in the number of processes might be indicative of a Problem. This is especially true if key services your application requires go down. Postgres likes to terminate in out-of-memory situations.
3. Other metrics might be GPU load, load on different network interfaces, CPU utilization of specific processes, context switching, kernel network buffers, open sockets (`ss`), ...
4. For hardcore performance analysis, look at the Linux `perf` tools.

Visualize

A picture is more than 1000 data points

- ▶ Plot your data. This is the only way to **quickly** spot problems.
 - ▶ Create dashboards. You will need to not just plot individual data points, but also spot correlations.
 - ▶ Combine data sources: host, network, application monitoring; logging, performance metrics
- ⇒ Use tools like Grafana or <https://nagios.org> or Zabbix

2024-08-26

NEXT GENERATION INTERNET

└─What are specific considerations for monitoring?

└─Visualize

- ▶ Plot your data. This is the only way to **quickly** spot problems.
 - ▶ Create dashboards. You will need to not just plot individual data points, but also spot correlations.
 - ▶ Combine data sources: host, network, application monitoring; logging, performance metrics
- ⇒ Use tools like Grafana or <https://nagios.org> or Zabbix

1. Grafana should probably be combined with Loki for log analysis.

2024-08-26

NEXT GENERATION INTERNET

└ What are specific considerations for monitoring?

Example dashboard

1. The video shows a Grafana dashboard created for Taler performance analysis.

Example dashboard

How to know your limits?

Micro-benchmarks

- ▶ Micro-benchmarks give you a critical **upper bound** on system performance.
- ▶ If your database takes 300ms per query and does at most 20 queries in parallel, what is your maximum transaction rate?
- ▶ If a transaction requires 50 kilobytes of bandwidth and you have 1 GB/s, what is your maximum transaction rate?
- ▶ <https://bench.cr.yp.to/> provides extensive benchmarks for cryptographic primitives. Study it to know how fast your cryptographic routines will be!

2024-08-26

NEXT GENERATION INTERNET

└─ How to know your limits?

└─ Micro-benchmarks

- ▶ Micro-benchmarks give you a critical **upper bound** on system performance.
- ▶ If your database takes 300ms per query and does at most 20 queries in parallel, what is your maximum transaction rate?
- ▶ If a transaction requires 50 kilobytes of bandwidth and you have 1 GB/s, what is your maximum transaction rate?
- ▶ <https://bench.cr.yp.to/> provides extensive benchmarks for cryptographic primitives. Study it to know how fast your cryptographic routines will be!

1. Database example is based on real-world experience with a company that ran into this problem but did not realize this was their bottleneck.
2. Consider all key performance dimensions (RAM, CPU, IO-load, latency) for critical operations; TCP streams at high bandwidth require large buffers, databases with limited number of concurrent operations may be limited by (disk or network) latency, etc.
3. Micro-benchmarks only give you a theoretical upper bound, but that number becomes very useful once your system-level benchmarks get close to it, and you need to figure out how to scale-up.

Burn your systems

Before going into production, burn your systems:

- ▶ memtest86+ (RAM)
- ▶ stress (CPU, RAM)
- ▶ stress-ng (CPU, RAM)
- ▶ lookbusy (CPU, RAM, disk)
- ▶ pgbench (CPU, RAM, disk)
- ▶ Blender-benchmark (GPU)

2024-08-26

NEXT GENERATION INTERNET

└─ How to know your limits?

└─ Burn your systems

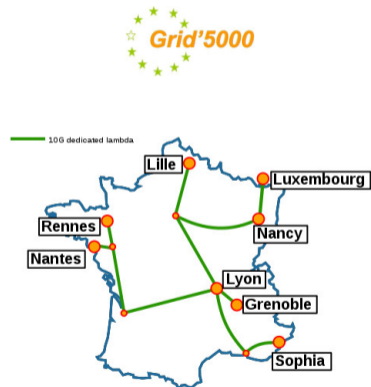
1. This can be a good way to detect faulty hardware before launching.
2. Might be a good idea to do several of these at the same time to check power supply and cooling issues.

Before going into production, burn your systems:

- ▶ memtest86+ (RAM)
- ▶ stress (CPU, RAM)
- ▶ stress-ng (CPU, RAM)
- ▶ lookbusy (CPU, RAM, disk)
- ▶ pgbench (CPU, RAM, disk)
- ▶ Blender-benchmark (GPU)

Grid'5000 [1]

- ▶ Large-scale flexible testbed
- ▶ 800 nodes with total 15'000 cores
- ▶ Bare metal deployments
- ▶ Fully customizable software stack



2024-08-26

NEXT GENERATION INTERNET

└ How to know your limits?

└ Grid'5000 [1]

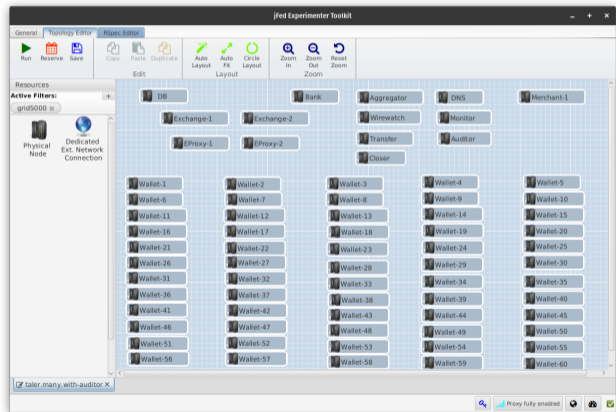
- ▶ Large-scale flexible testbed
- ▶ 800 nodes with total 15'000 cores
- ▶ Bare metal deployments
- ▶ Fully customizable software stack



1. Flexible system for large-scale distributed experiments operated by INRIA.
2. Probably the ultimate way to do end-to-end evaluations: real network with real latencies and with full physical access to the hardware!

Platform Access

jFed - Java-based GUI and CLI

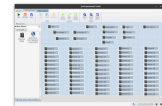


2024-08-26

NEXT GENERATION INTERNET

└ How to know your limits?

└ Platform Access



1. jFed can be a bit confusing as it is designed to work with a variety of testbeds, and so many features do not apply to Grid5k.
2. In particular, you basically only allocate hosts, and all hosts can talk to all hosts. Do not try to draw network links in jFed for Grid5k.
3. The placement of the nodes on the screen is also not significant, but may help you organize different types of nodes a bit.

Running an experiment

2024-08-26

NEXT GENERATION INTERNET

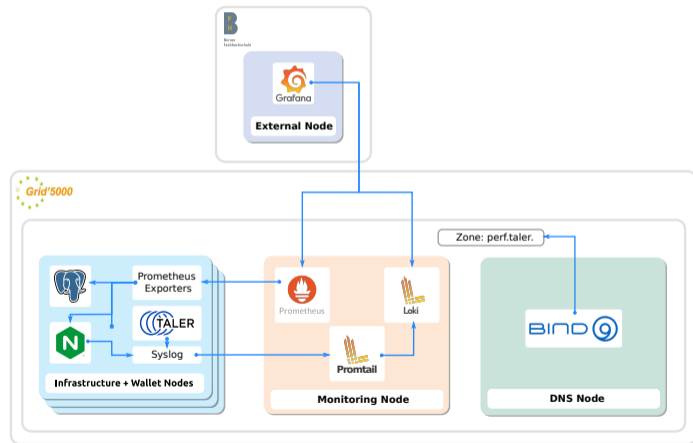
└ How to know your limits?

└ Running an experiment



1. Building an image with Kameleon [6] is best done under Docker.
2. Allocating experiment resources is the hardest part, as it frequently fails due to the Grid being overloaded.

Experiment architecture



2024-08-26

NEXT GENERATION INTERNET

└ How to know your limits?

└ Experiment architecture



1. Within the Grid you need to allocate resources for both the system to evaluate and the load generation logic (here: wallet nodes). It is not uncommon to need most resources for load generation, as load generation tools are rarely the primary optimization target!
2. Monitoring itself can be a bottleneck. Thus, it is a good idea to deploy monitoring within the Grid5k itself.
3. DNS can be critical, especially for latency. Thus, again, deploying DNS within Grid5k can be a good idea.
4. All Grid resources are released upon termination of the experiment. Thus, export all critical data to an external node for post-experiment analysis.

Performance

Various payment systems

Bitcoin	PayPal	Visa
4 TPS	193 TPS	1'667 TPS
e-Krona [5] (Sweden)	e-CNY [4] (China)	Project Hamilton [2] (MIT)
100 TPS	10'000 TPS	1'700'000 TPS

Know your requirements when benchmarking!

2024-08-26

NEXT GENERATION INTERNET

└ How to know your limits?


└ Performance

Bitcoin	PayPal	Visa
4 TPS	193 TPS	1'667 TPS
e-Krona [5] (Sweden)	e-CNY [4] (China)	Project Hamilton [2] (MIT)
100 TPS	10'000 TPS	1'700'000 TPS

Know your requirements when benchmarking!

1. <https://www.riksbank.se/globalassets/media/rapporter/e-krona/2022/e-krona-pilot-phase-2.pdf>
2. <https://www.atlanticcouncil.org/blogs/econographics/a-report-card-on-chinas-central-bank-digital-currency-the-e-cny/>
3. <https://www.bostonfed.org/-/media/Documents/Project-Hamilton/Project-Hamilton-Phase-1-Whitepaper.pdf>
4. Note that Hamilton mostly did an embarassingly parallel database benchmark and did not consider that transaction load varies between accounts of supermarket chains and individual citizens. Thus, their simulation is of somewhat questionable value due to lack of realism.
5. Business requirements need to be clear about performance requirements. Bitcoin clearly missed the required scale by several orders of magnitude, while Hamilton overshot in the other direction.

References I

-  Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, Emmanuel Jeannot, Emmanuel Jeanvoine, Adrien Lèbre, David Margery, Nicolas Niclausse, Lucas Nussbaum, Olivier Richard, Christian Pérez, Flavien Quesnel, Cyril Rohr, and Luc Sarzyniec. Adding virtualization capabilities to the Grid'5000 testbed. In *Cloud Computing and Services Science*, volume 367, pages 3–20. Springer International Publishing, 2013.

2024-08-26

NEXT GENERATION INTERNET

└References

└References

Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, Emmanuel Jeannot, Emmanuel Jeanvoine, Adrien Lèbre, David Margery, Nicolas Niclausse, Lucas Nussbaum, Olivier Richard, Christian Pérez, Flavien Quesnel, Cyril Rohr, and Luc Sarzyniec. Adding virtualization capabilities to the Grid'5000 testbed. In *Cloud Computing and Services Science*, volume 367, pages 3–20. Springer International Publishing, 2013.

References II

-  Jim Cunha, Robert Bench, James Lovejoy, Cory Fields, Madars Virza, Tyler Frederick, David Urness, Kevin Karwaski, Anders Brownworth, Neha Narula.
Project hamilton phase 1 a high performance payment processing system designed for central bank digital currencies.
Technical report, Federal Reserve Bank of Boston and Massachusetts Institute of Technology Digital Currency Initiative, Feb 2022.
Available at <https://www.bostonfed.org/-/media/Documents/Project-Hamilton/Project-Hamilton-Phase-1-Whitepaper.pdf> [05.05.2022].

2024-08-26


NEXT GENERATION INTERNET

└References

└References

Jim Cunha, Robert Bench, James Lovejoy, Cory Fields, Madars Virza, Tyler Frederick, David Urness, Kevin Karwaski, Anders Brownworth, Neha Narula.
Project hamilton phase 1 a high performance payment processing system designed for central bank digital currencies.
Technical report, Federal Reserve Bank of Boston and Massachusetts Institute of Technology Digital Currency Initiative, Feb 2022.
Available at <https://www.bostonfed.org/-/media/Documents/Project-Hamilton/Project-Hamilton-Phase-1-Whitepaper.pdf> [05.05.2022].

References III

-  [Ananya Kunar.](#)
A report card on china's central bank digital currency: the e-cny, Jan 2022.
Available at [https://www.atlanticcouncil.org/blogs/econographics/a-report-card-on-chinas-central-bank-digital-currency-the-e-cny/\[05.05.2022\]](https://www.atlanticcouncil.org/blogs/econographics/a-report-card-on-chinas-central-bank-digital-currency-the-e-cny/[05.05.2022]).

2024-08-26

NEXT GENERATION INTERNET

└References

└References

 [Ananya Kunar.](#)
A report card on china's central bank digital currency: the e-cny, Jan 2022.
Available at [https://www.atlanticcouncil.org/blogs/econographics/a-report-card-on-chinas-central-bank-digital-currency-the-e-cny/\[05.05.2022\]](https://www.atlanticcouncil.org/blogs/econographics/a-report-card-on-chinas-central-bank-digital-currency-the-e-cny/[05.05.2022]).

References IV

-  People's Bank of China.
Progress of research & development of e-cny in china.
Technical report, People's Bank of China, Jul 2021.
Available at <http://www.pbc.gov.cn/en/3688110/3688172/4157443/4293696/2021071614584691871.pdf> [05.05.2022].
-  Sveriges Riskbank.
e-krona pilot phase 2.
Technical report, Sveriges Riskbank, Apr 2022.
Available at <https://www.riksbank.se/globalassets/media/rapporter/e-krona/2022/e-krona-pilot-phase-2.pdf> [05.05.2022].

References V

-  Cristian Ruiz, Salem Harrache, Michael Mercier, and Olivier Richard. Reconstructable Software Appliances with Kameleon. *Operating Systems Review*, 49(1):80–89, 2015.

2024-08-26

NEXT GENERATION INTERNET

└References

└References

 Cristian Ruiz, Salem Harrache, Michael Mercier, and Olivier Richard. Reconstructable Software Appliances with Kameleon. *Operating Systems Review*, 49(1):80–89, 2015.

Acknowledgements

Co-funded by the European Union (Project 101135475).



Co-funded by
the European Union

Co-funded by SERI (HEU-Projekt 101135475-TALER).

Project funded by



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs,
Education and Research EAER
**State Secretariat for Education,
Research and Innovation SERI**

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Availability

Christian Grothoff

Availability

1. Monitor HTTP request latency

Configure Apache or nginx to log request latency, HTTP status code and URL for every request.

2. Listen on a UNIX domain socket

Implement a minimal HTTP server that listens on a UNIX domain socket. It should return 32 bytes from `/dev/random` to serve as a public entropy source. Make it available via your reverse proxy.

3. Listen and notify

Implement two programs, one that prints all of the rows in a table by monotonically increasing row ID, and a second that inserts a new row. Use `LISTEN` and `NOTIFY` on a `TRIGGER` to have the first program print new rows immediately when they are inserted, without busy-waiting. Check that the listener works reliably.

4. Database access control

Limit the process `LISTENing` to only allow `SELECT` on the respective table. Validate by trying to `UPDATE`, `DELETE` or `DROP`.

5. Version your database

Add versioning to your database. Create the schema using an SQL file. Upgrade the schema (and version) using another SQL file.

6. Replicate your database

Setup two databases using your schema. Then apply log-based replication between the two databases. Run your listener on the replica database, and insert into the master database. Check that the replica listener immediately prints the freshly inserted record.

NEXT GENERATION INTERNET

GNU Taler

Christian Grothoff

07.06.2024

2024-08-26

NEXT GENERATION INTERNET

NEXT
GENERATION
INTERNET
GNU Taler

Christian Grothoff

07.06.2024

1. In this lecture, we will do a deep dive into the cryptography behind GNU Taler.

Learning objectives

How should we pay?

Introduction to GNU Taler

How does cut-and-choose work?

How to prove protocols secure with cryptographic games?

What are the future plans for GNU Taler?

2024-08-26

NEXT GENERATION INTERNET

└ Learning objectives

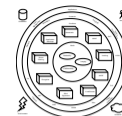
1. First, following Prof. Rogaway's call [4] for a community-wide effort to develop more effective means to resist mass surveillance, we will start with a moral analysis of the problem: How should we pay?
2. Then, you will get an overview of GNU Taler and its architecture. You should already be familiar with blind signatures, one key cryptographic building block.
3. We will then use GNU Taler's cryptography to introduce two more advanced cryptographic concepts, namely cut-and-choose protocols and an advanced example for provable security using cryptographic games.
4. Finally, we'll take a brief peek at the GNU Taler roadmap to see what probably lies ahead.

How should we pay?
Introduction to GNU Taler
How does cut-and-choose work?
How to prove protocols secure with cryptographic games?
What are the future plans for GNU Taler?

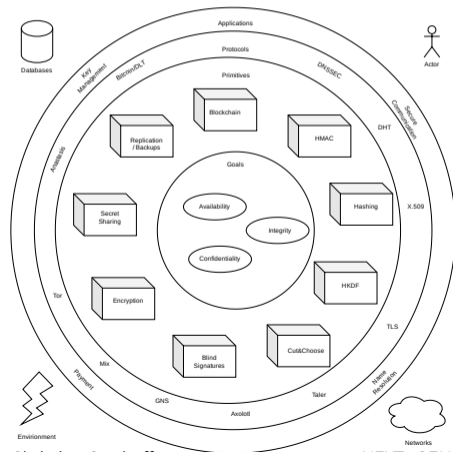
Taler on our map

2024-08-26

NEXT GENERATION INTERNET



Taler on our map



1. Taler is a payment protocol offering anonymity.
2. Key building blocks are blind signatures and cut&choose constructions.
3. The main related topics are Tor (anonymity), Blockchain (payment) and Integration (payments usually need to be integrated with other business processes).



2024-08-26

How should we pay?

Surveillance

2024-08-26

NEXT GENERATION INTERNET

└ How should we pay?

└ Surveillance



1. What domain of digital communication should we be most concerned about?

Surveillance concerns

- ▶ Everybody knows about Internet surveillance.
- ▶ But is it **that** bad?
 - ▶ You can choose when and where to use the Internet
 - ▶ You can anonymously access the Web using Tor
 - ▶ You can find open access points that do not require authentication
 - ▶ IP packets do not include your precise location or name
 - ▶ ISPs typically store this meta data for days, weeks or months

2024-08-26

NEXT GENERATION INTERNET

└ How should we pay?

└ Surveillance concerns

- ▶ Everybody knows about Internet surveillance.
- ▶ But is it **that** bad?
 - ▶ You can choose when and where to use the Internet
 - ▶ You can anonymously access the Web using Tor
 - ▶ You can find open access points that do not require authentication
 - ▶ IP packets do not include your precise location or name
 - ▶ ISPs typically store this meta data for days, weeks or months

1. Internet mass-surveillance may be bad, but it is to some degree avoidable or escapable.

Where is it worse?

This was a question posed to RAND researchers in 1971:

“Suppose you were an advisor to the head of the KGB, the Soviet Secret Police. Suppose you are given the assignment of designing a system for the surveillance of all citizens and visitors within the boundaries of the USSR. The system is not to be too obtrusive or obvious. What would be your decision?”

2024-08-26

NEXT GENERATION INTERNET

└ How should we pay?

└ Where is it worse?

This was a question posed to RAND researchers in 1971:
“Suppose you were an advisor to the head of the KGB, the Soviet Secret Police. Suppose you are given the assignment of designing a system for the surveillance of all citizens and visitors within the boundaries of the USSR. The system is not to be too obtrusive or obvious. What would be your decision?”

1. The result: an electronic funds transfer system that looks strikingly similar today’s debit card system.
2. What is surprising is that Snowden says this is **one of the worst things**, as he obviously had a bunch of rather large concerns.

Where is it worse?

This was a question posed to RAND researchers in 1971:

“Suppose you were an advisor to the head of the KGB, the Soviet Secret Police. Suppose you are given the assignment of designing a system for the surveillance of all citizens and visitors within the boundaries of the USSR. The system is not to be too obtrusive or obvious. What would be your decision?”

“I think one of the big things that we need to do, is we need to get a way from true-name payments on the Internet. The credit card payment system is one of the worst things that happened for the user, in terms of being able to divorce their access from their identity.” –Edward Snowden, IETF 93 (2015)

2024-08-26

NEXT GENERATION INTERNET

└─ How should we pay?

└─ Where is it worse?

This was a question posed to RAND researchers in 1971:
“Suppose you were an advisor to the head of the KGB, the Soviet Secret Police. Suppose you are given the assignment of designing a system for the surveillance of all citizens and visitors within the boundaries of the USSR. The system is not to be too obtrusive or obvious. What would be your decision?”

“I think one of the big things that we need to do, is we need to get a way from true-name payments on the Internet. The credit card payment system is one of the worst things that happened for the user, in terms of being able to divorce their access from their identity.” –Edward Snowden, IETF 93 (2015)

1. The result: an electronic funds transfer system that looks strikingly similar today’s debit card system.
2. What is surprising is that Snowden says this is **one of the worst things**, as he obviously had a bunch of rather large concerns.

Why is it worse?

- ▶ When you pay by CC, the information includes your name
- ▶ When you pay in person with CC, your location is also known
- ▶ You often have no alternative payment methods available
- ▶ You hardly ever can use someone else's CC
- ▶ Anonymous prepaid cards are difficult to get and expensive
- ▶ Payment information is typically stored for 6-10 years!

2024-08-26

NEXT GENERATION INTERNET

└ How should we pay?

└ Why is it worse?

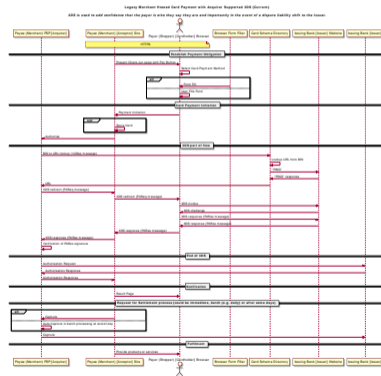
1. For digital payments, surveillance has become completely normalized.
2. It is also basically inescapable, except by using cash. and cash sometimes cannot be used!

- ▶ When you pay by CC, the information includes your name
- ▶ When you pay in person with CC, your location is also known
- ▶ You often have no alternative payment methods available
- ▶ You hardly ever can use someone else's CC
- ▶ Anonymous prepaid cards are difficult to get and expensive
- ▶ Payment information is typically stored for 6-10 years!

Credit cards have problems, too!

3D secure ("verified by visa") is a nightmare:

- ▶ Complicated process
- ▶ Shifts liability to consumer
- ▶ Significant latency
- ▶ Can refuse valid requests
- ▶ Legal vendors excluded
- ▶ No privacy for buyers



2024-08-26

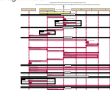
NEXT GENERATION INTERNET

└ How should we pay?

└ Credit cards have problems, too!

3D secure ("verified by visa") is a nightmare:

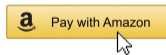
- ▶ Complicated process
- ▶ Shifts liability to consumer
- ▶ Significant latency
- ▶ Can refuse valid requests
- ▶ Legal vendors excluded
- ▶ No privacy for buyers



1. Now, the modern online CC process is also a nightmare, from privacy, security, usability and cost perspectives.
2. Our claim: online credit card payments will be replaced. The question is, with what?

The bank's Problem

- ▶ Global tech companies push oligopolies
- ▶ Privacy and federated finance are at risk
- ▶ Economic sovereignty is in danger



2024-08-26

NEXT GENERATION INTERNET

└ How should we pay?

└ The bank's Problem

1. And Apple would like us to pay 30% fees on everything for their walled surveillance garden.

Predicting the future

- ▶ Google and Apple will be your bank and run your payment system
- ▶ They can target advertising based on your purchase history, location and your ability to pay
- ▶ They will provide more usable, faster and broadly available payment solutions; our federated banking system will be history
- ▶ After they dominate the payment sector, they will start to charge fees befitting their oligopoly size
- ▶ Competitors and vendors not aligning with their corporate “values” will be excluded by policy and go bankrupt
- ▶ The imperium will have another major tool for its financial warfare

2024-08-26

NEXT GENERATION INTERNET

└ How should we pay?

└ Predicting the future

1. “Do you want to live under total surveillance?” Sure, this may sound unlikely, but let’s listen to some experts on this.

- ▶ Google and Apple will be your bank and run your payment system
- ▶ They can target advertising based on your purchase history, location and your ability to pay
- ▶ They will provide more usable, faster and broadly available payment solutions; our federated banking system will be history
- ▶ After they dominate the payment sector, they will start to charge fees befitting their oligopoly size
- ▶ Competitors and vendors not aligning with their corporate “values” will be excluded by policy and go bankrupt
- ▶ The imperium will have another major tool for its financial warfare

1. The key sentence here is that they **will have absolute control** over how we use digital cash.
2. The director of the BIS points to this as a fact. Note that the BIS is basically the United Nations of the central banks of the world. Their headquarters is in Basel.

2024-08-26

NEXT GENERATION INTERNET
└ How should we pay?

The Emergency Act of Canada, February 2022, <https://www.youtube.com/watch?v=Nel>

The Emergency Act of Canada, February 2022, <https://www.youtube.com/watch?v=Nel>

Introduction to GNU Taler

Digital cash, made **socially responsible**.



Privacy-Preserving, Practical, Taxable, Free Software, Efficient



1. Bold claims.

What is Taler?

<https://taler.net/en/features.html>

Taler is

- ▶ a Free/Libre software *payment system* infrastructure project
- ▶ ... with a surrounding software ecosystem
- ▶ ... and a company (Taler Systems S.A.) and community that wants to deploy it as widely as possible.

However, Taler is

- ▶ *not* a currency
- ▶ *not* a long-term store of value
- ▶ *not* a network or instance of a system
- ▶ *not* decentralized
- ▶ *not* based on proof-of-work or proof-of-stake
- ▶ *not* a speculative asset / “get-rich-quick scheme”

2024-08-26

NEXT GENERATION INTERNET

└ Introduction to GNU Taler

└ What is Taler?

1. Not a currency, not a crypto-currency, no blockchain, just a payment system.
2. For your day-to-day expenses, not your retirement fund or buying a house.
3. Not like PayPal where you have one operator, primarily a protocol, like HTTP!
4. Not a P2P network, there are still easily identifiable (and accountable) payment service providers.
5. Efficient, no burning down the planet for 3 transactions per second.
6. Again, not a currency, you pay with GNU Taler, not *in* Taler. You pay in EUR/CHF/USD.

Taler is

- ▶ a Free/Libre software payment system infrastructure project
- ▶ ... with a surrounding software ecosystem
- ▶ ... and a company (Taler Systems S.A.) and community that wants to deploy it as widely as possible.

However, Taler is

- ▶ *not* a currency
- ▶ *not* a long-term store of value
- ▶ *not* a network or instance of a system
- ▶ *not* decentralized
- ▶ *not* based on proof-of-work or proof-of-stake
- ▶ *not* a speculative asset / “get-rich-quick scheme”

Design goals

... for the GNU Taler payment system

GNU Taler must ...

1. ... be implemented as **free software**.
2. ... protect the **privacy of buyers**.
3. ... must enable the state to **tax income** and crack down on illegal business activities.
4. ... prevent payment fraud.
5. ... only **disclose the minimal amount of information necessary**.
6. ... be usable.
7. ... be efficient.
8. ... avoid single points of failure.
9. ... foster **competition**.

2024-08-26

NEXT GENERATION INTERNET

└ Introduction to GNU Taler

└ Design goals

GNU Taler must ...

1. ... be implemented as **free software**.
2. ... protect the **privacy of buyers**.
3. ... must enable the state to **tax income** and crack down on illegal business activities.
4. ... prevent payment fraud.
5. ... only **disclose the minimal amount of information necessary**.
6. ... be usable.
7. ... be efficient.
8. ... avoid single points of failure.
9. ... foster **competition**.

1. The design goals are presented in order of priority. Those higher up are more important.
2. If you have other priorities, you will end up with a different design. When designing a system, try to come up with priorities first.
3. Of course the order is not absolute: we would not sacrifice an insane amount of efficiency for a tiny gain in usability. But it is important to have priorities when the trade-offs are plausible.
4. Objective 5 is relevant as objective 2 is only about privacy of buyers, but there is other data to minimize in a complex system.
5. How do you foster competition? By making it possible for various components to be commercially offered by different parties; using proper protocols ensures there can be different implementations, operators and integrators.

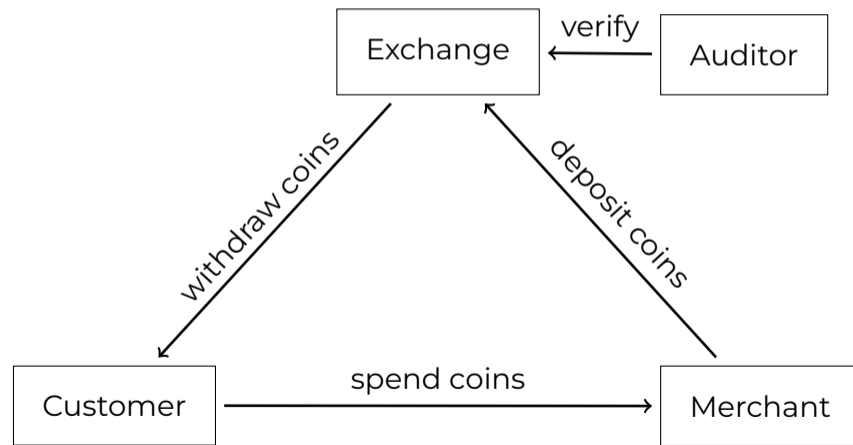
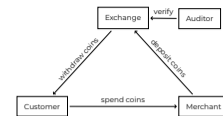
Taler overview

2024-08-26

NEXT GENERATION INTERNET

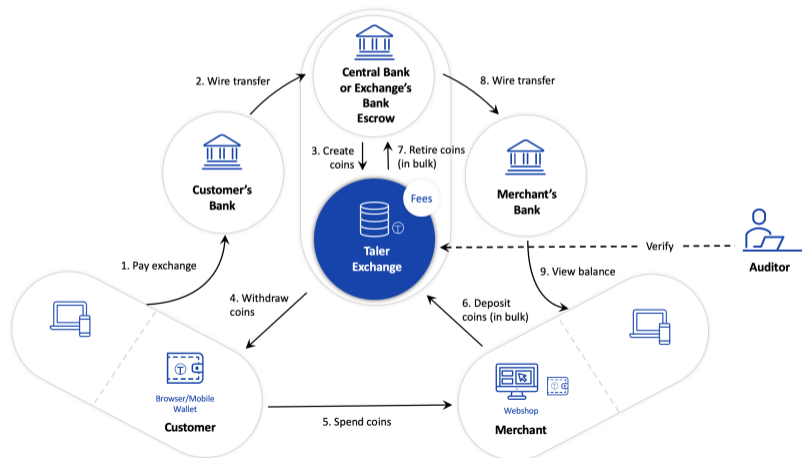
└ Introduction to GNU Taler

└ Taler overview



1. This figure shows the key parties in the GNU Taler system.
2. The exchange operates the payment system: it issues digital coins and allows them to be redeemed.
3. Customers obtain digital cash can can spend it.
4. Merchants accept digital cash.
5. The auditor checks that the exchange is operating correctly.

Architecture of Taler



1. Illustration of the payment process of GNU Taler and its integration with an existing core banking system. Macro-payments between bank accounts (steps 2 and 8) are for large sums. Step 2 represents buyers withdrawing money from their bank accounts, and step 8 merchants receiving their aggregated (daily, weekly, monthly, etc.) revenues. In contrast, cryptographic payments within GNU Taler (steps 4, 5 and 6) are much cheaper.
2. The purchase in step 5 is unlinkable to the withdrawal in step 4 due to the use of blind signatures, which protect the anonymity for the buyer spending coins in step 5.

Usability of Taler

`https://demo.taler.net/`

1. Install Web extension.
2. Visit the `bank.demo.taler.net` to withdraw coins.
3. Visit the `shop.demo.taler.net` to spend coins.

2024-08-26

NEXT GENERATION INTERNET

└ Introduction to GNU Taler

└ Usability of Taler

`https://demo.taler.net/`

1. Install Web extension.
2. Visit the `bank.demo.taler.net` to withdraw coins.
3. Visit the `shop.demo.taler.net` to spend coins.

1. KUDOS is a “fake” currency used for the demonstration, EUR or CHF would be used in practice.
2. The demo can be done using a WebExtension or a Taler wallet running on a mobile phone, or both.
3. You can also demonstrate P2P payments between the Taler wallet browser extension and the mobile phone.

How does cut-and-choose work?

We say Taler is taxable because:

- ▶ Merchant's income is visible from deposits.
- ▶ Hash of contract is part of deposit data.
- ▶ State can trace income and enforce taxation.

Limitations:

- ▶ withdraw loophole
- ▶ *sharing* coins among family and friends

Other contemporary payment systems have similar limitations on identification, and thus these limitations should not be a legal issue.

We say Taler is taxable because:

- ▶ Merchant's income is visible from deposits.
- ▶ Hash of contract is part of deposit data.
- ▶ State can trace income and enforce taxation.

Limitations:

- ▶ withdraw loophole
- ▶ *sharing* coins among family and friends

Other contemporary payment systems have similar limitations on identification, and thus these limitations should not be a legal issue.

1. When withdrawing, Taler does not exactly determine that the owner of the account is also the owner of the wallet. The withdraw loophole is basically equivalent to somebody putting their bank card into an ATM and someone else taking the cash. While [3] explains a way to address the loophole, doing so would put the privacy of payer's at risk, so we decided against it and will not cover it here.
2. The sharing loophole is when the owner of a coin decides to simply give the private key and signature of a digital coin to another user. As the user cannot be sure that the owner really deleted their copy of the private key material (without any backup), both users then *share* access to the value of the coin. The first to spend it, will succeed. We call this *sharing* as opposed to a transaction: in a transaction, ownership is transferred between parties that do not trust each other.
3. Sharing is thus like giving your spouse the password to your bank account.

Giving change

It would be inefficient to pay EUR 100 with 1 cent coins!

- ▶ Denomination key represents value of a coin.
- ▶ Exchange may offer various denominations for coins.
- ▶ Wallet may not have exact change!
- ▶ Usability requires ability to pay given sufficient total funds.

2024-08-26

NEXT GENERATION INTERNET

└ How does cut-and-choose work?

└ Giving change

It would be inefficient to pay EUR 100 with 1 cent coins!

- ▶ Denomination key represents value of a coin.
- ▶ Exchange may offer various denominations for coins.
- ▶ Wallet may not have exact change!
- ▶ Usability requires ability to pay given sufficient total funds.

1. Taler issues digital cash using blind signatures, where each signature conveys the respective coin a particular value.
2. We want to avoid cryptographic expenses linear in the amount being paid!
3. Thus we need a way to get change, but doing so must not void our security assurances, specifically unlinkability (and anonymity) for the payer, and income transparency for the payee.
4. The high-level approach for getting change is pretty simple: when paying with a coin, the (EdDSA) coin signature can specify that not the full value of the coin is to be spent, but only a fraction. The exchange then allows a wallet to request change by creating a second signature using the partially spent coin's private (EdDSA) key over a change request with fresh (blinded) digital coins that total up to the amount of change that is due.

Giving change

It would be inefficient to pay EUR 100 with 1 cent coins!

- ▶ Denomination key represents value of a coin.
- ▶ Exchange may offer various denominations for coins.
- ▶ Wallet may not have exact change!
- ▶ Usability requires ability to pay given sufficient total funds.

Key goals:

- ▶ maintain unlinkability
- ▶ maintain taxability of transactions

2024-08-26

NEXT GENERATION INTERNET

└ How does cut-and-choose work?

└ Giving change

1. Taler issues digital cash using blind signatures, where each signature conveys the respective coin a particular value.
2. We want to avoid cryptographic expenses linear in the amount being paid!
3. Thus we need a way to get change, but doing so must not void our security assurances, specifically unlinkability (and anonymity) for the payer, and income transparency for the payee.
4. The high-level approach for getting change is pretty simple: when paying with a coin, the (EdDSA) coin signature can specify that not the full value of the coin is to be spent, but only a fraction. The exchange then allows a wallet to request change by creating a second signature using the partially spent coin's private (EdDSA) key over a change request with fresh (blinded) digital coins that total up to the amount of change that is due.

It would be inefficient to pay EUR 100 with 1 cent coins!

- ▶ Denomination key represents value of a coin.
- ▶ Exchange may offer various denominations for coins.
- ▶ Wallet may not have exact change!
- ▶ Usability requires ability to pay given sufficient total funds.

Key goals:

- ▶ maintain unlinkability
- ▶ maintain taxability of transactions

Giving change

It would be inefficient to pay EUR 100 with 1 cent coins!

- ▶ Denomination key represents value of a coin.
- ▶ Exchange may offer various denominations for coins.
- ▶ Wallet may not have exact change!
- ▶ Usability requires ability to pay given sufficient total funds.

Key goals:

- ▶ maintain unlinkability
- ▶ maintain taxability of transactions

Method:

- ▶ Contract can specify to only pay *partial value* of a coin.
- ▶ Exchange allows wallet to obtain *unlinkable change* for remaining coin value.

2024-08-26

NEXT GENERATION INTERNET

└ How does cut-and-choose work?

└ Giving change

1. Taler issues digital cash using blind signatures, where each signature conveys the respective coin a particular value.
2. We want to avoid cryptographic expenses linear in the amount being paid!
3. Thus we need a way to get change, but doing so must not void our security assurances, specifically unlinkability (and anonymity) for the payer, and income transparency for the payee.
4. The high-level approach for getting change is pretty simple: when paying with a coin, the (EdDSA) coin signature can specify that not the full value of the coin is to be spent, but only a fraction. The exchange then allows a wallet to request change by creating a second signature using the partially spent coin's private (EdDSA) key over a change request with fresh (blinded) digital coins that total up to the amount of change that is due.

It would be inefficient to pay EUR 100 with 1 cent coins!

- ▶ Denomination key represents value of a coin.
- ▶ Exchange may offer various denominations for coins.
- ▶ Wallet may not have exact change!
- ▶ Usability requires ability to pay given sufficient total funds.

Key goals:

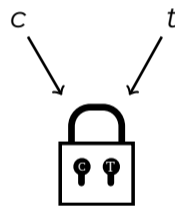
- ▶ maintain unlinkability
- ▶ maintain taxability of transactions

Method:

- ▶ Contract can specify to only pay *partial value* of a coin.
- ▶ Exchange allows wallet to obtain *unlinkable change* for remaining coin value.

Diffie-Hellman (ECDH)

1. Create private keys $c, t \pmod{o}$
2. Define $C := cG$
3. Define $T := tG$
4. Compute DH:
 $cT = c(tG) = t(cG) = tC$



2024-08-26

NEXT GENERATION INTERNET

└ How does cut-and-choose work?

└ Diffie-Hellman (ECDH)

1. Create private keys $c, t \pmod{o}$
2. Define $C := cG$
3. Define $T := tG$
4. Compute DH:
 $cT = c(tG) = t(cG) = tC$



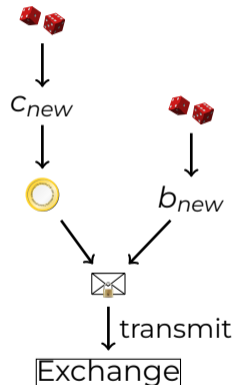
1. Before we can introduce the change protocol, we need another pretty picture for a well-known cryptographic primitive, Diffie-Hellman (DH). Taler uses ECDH, but that does not matter except for performance.
2. A good way to think of DH is a lock with two keys where either key opens the lock.
3. Note that we will use DH in a rather unusual way in the following protocol.

Straw-man solution

Given partially spent private coin key c_{old} :

1. Pick random $c_{new} \bmod o$ private key
2. Compute $C_{new} := c_{new}G$ public key
3. Pick random b_{new}
4. Compute $f_{new} := FDH(C_{new}), m < n$.
5. Transmit $f'_{new} := f_{new}b_{new}^e \bmod n$

... and sign request for change with c_{old} .



2024-08-26

NEXT GENERATION INTERNET

└ How does cut-and-choose work?

└ Straw-man solution

- Given partially spent private coin key c_{old} :
1. Pick random $c_{new} \bmod o$ private key
 2. Compute $C_{new} := c_{new}G$ public key
 3. Pick random b_{new}
 4. Compute $f_{new} := FDH(C_{new}), m < n$.
 5. Transmit $f'_{new} := f_{new}b_{new}^e \bmod n$
- ... and sign request for change with c_{old} .

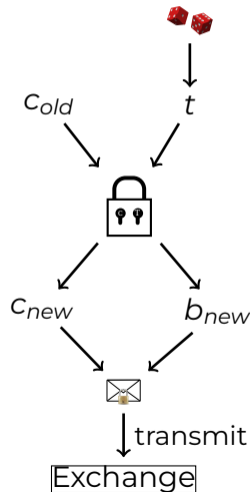


1. A straw-man solution is one that does not work, but still could be useful to illuminate the issue.
2. Here, the protocol allows users to obtain change (c_{new}) by signing the request for change (the envelope) with an old coin c_{old} that has some residual value from a previous purchase (that signature is not shown).
3. **Problem:** Owner of c_{new} may differ from owner of c_{old} breaks income-transparency / enables tax evasion!

Customer: Transfer key setup (ECDH)

Given partially spent private coin key c_{old} :

1. Let $C_{old} := c_{old}G$ (as before)
2. Create random private transfer key $t \pmod{o}$
3. Compute public transfer key $T := tG$
4. Compute $X := c_{old}(tG) = t(c_{old}G) = tC_{old}$
5. Derive c_{new} and b_{new} from X using HKDF
6. Compute $C_{new} := c_{new}G$
7. Compute $f_{new} := FDH(C_{new})$
8. Transmit $f'_{new} := f_{new}b_{new}^e$



2024-08-26

NEXT GENERATION INTERNET

└ How does cut-and-choose work?

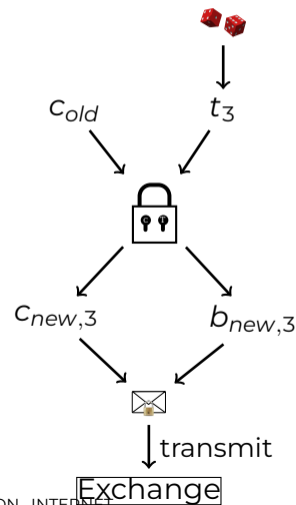
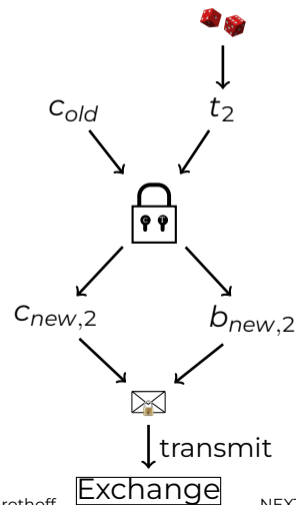
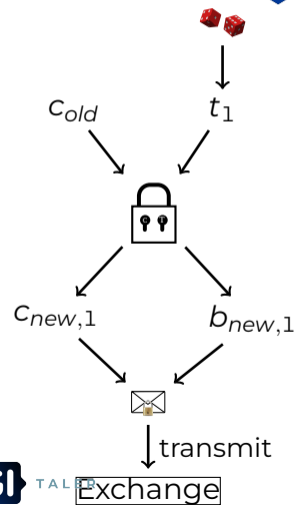
└ Customer: Transfer key setup (ECDH)

- Given partially spent private coin key c_{old} :
1. Let $C_{old} := c_{old}G$ (as before)
 2. Create random private transfer key $t \pmod{o}$
 3. Compute public transfer key $T := tG$
 4. Compute $X := c_{old}(tG) = t(c_{old}G) = tC_{old}$
 5. Derive c_{new} and b_{new} from X using HKDF
 6. Compute $C_{new} := c_{new}G$
 7. Compute $f_{new} := FDH(C_{new})$
 8. Transmit $f'_{new} := f_{new}b_{new}^e$



1. In this construction, we *derive* the blinding factor b_{new} and the private key of the new coin c_{new} from the DH of the c_{old} and a newly created transfer key t . Note that it is a bit unusual but perfectly fine that we here have **both** private keys to compute the DH.
2. The resulting blinded public key of the new coin (public key derivation and blinding are elided to keep the diagram concise) is then signed with c_{old} to request change.
3. This approach has an obvious problem: from the perspective of the Exchange, we cannot even tell that the user followed this procedure as the resulting request with the blinded coin is indistinguishable from the previous construction.

Cut-and-Choose

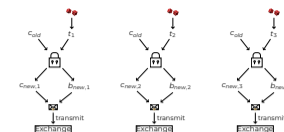


2024-08-26

NEXT GENERATION INTERNET

How does cut-and-choose work?

Cut-and-Choose



1. This DH-construction thus obviously does not work, so in the usual approach of an insane person, we don't just do it once, but three times using three different transfer keys t_1 , t_2 , and t_3 instead of just t .
2. Now, before you decide that we have just gone mad, this is actually a well-known technique called **cut-and-choose**. Here, we do a protocol step multiple times to basically be able to **burn** some of these iterations to **prove** our honesty.
3. There are also **non-interactive** cut-and-choose protocols, but this one is a simple interactive one.

Exchange: Choose!

Exchange sends back random $\gamma \in \{1, 2, 3\}$ to the customer.

2024-08-26

NEXT GENERATION INTERNET

└ How does cut-and-choose work?

└ Exchange: Choose!

Exchange sends back random $\gamma \in \{1, 2, 3\}$ to the customer.

1. This is the typical interaction: the Exchange picks one of the three at random, basically deciding on which iterations to challenge the wallet's honesty.
2. γ primarily needs to be **unpredictable** for the wallet.
3. Note that the protocol has a security parameter $\kappa = 3$, and so the wallet could guess correctly in $\frac{1}{3}$ of the cases. Usually in security we would think of this to be way too low, and you will see much higher values in other cut-and-choose protocols. But, we will see why $\kappa = 3$ is actually enough for GNU Taler!

Customer: Reveal

1. If $\gamma = 1$, send t_2, t_3 to exchange
2. If $\gamma = 2$, send t_1, t_3 to exchange
3. If $\gamma = 3$, send t_1, t_2 to exchange

2024-08-26

NEXT GENERATION INTERNET

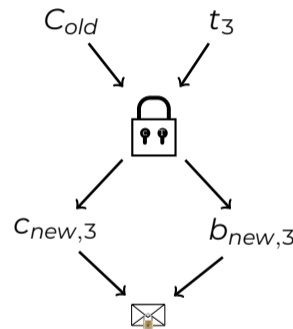
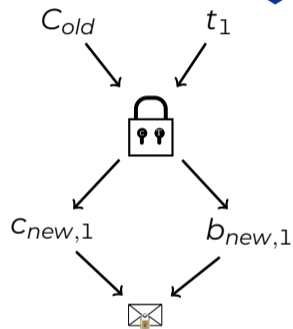
└ How does cut-and-choose work?

└ Customer: Reveal

1. If $\gamma = 1$, send t_2, t_3 to exchange
2. If $\gamma = 2$, send t_1, t_3 to exchange
3. If $\gamma = 3$, send t_1, t_2 to exchange

1. So given the γ challenge value, the wallet has to send back the t_i values for $i \neq \gamma$.

Exchange: Verify ($\gamma = 2$)



2024-08-26

NEXT GENERATION INTERNET

└ How does cut-and-choose work?

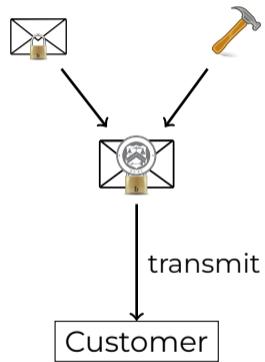
└ Exchange: Verify ($\gamma = 2$)



1. Given those two values the exchange can **validate** the construction as it can compute the DH from the **transfer private keys** t_i and the **coin public key** C_{old} .
2. If the result matches with the original request from the wallet, the exchange has established that with $\frac{2}{3}$ probability the wallet made an honest request for change following the prescribed construction.
3. If the wallet is unable (or unwilling) to produce the required t_i values, or if the resulting blinded values do not match, the entire change is forfeit, and the customer loses their money.
4. Thus, trying to cheat on income-transparency is punished with what amounts to a **66.67% tax**. Thus, a security level of κ is sufficient as long as the *effective* income tax (after deductions, on the full income) is below $\frac{\kappa-1}{\kappa}$. Taler always uses $\kappa = 3$.

Exchange: Blind sign change (RSA)

1. Take $f'_{new,\gamma}$.
2. Compute $s' := f'^d_{new,\gamma} \pmod n$.
3. Return signature s' .



2024-08-26

NEXT GENERATION INTERNET

└ How does cut-and-choose work?

└ Exchange: Blind sign change (RSA)

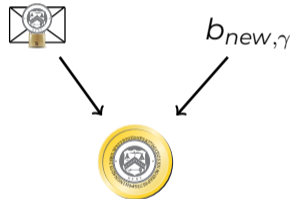


1. Take $f'_{new,\gamma}$.
2. Compute $s' := f'^d_{new,\gamma} \pmod n$.
3. Return signature s' .

1. If the customer's request did follow the DH-construction, the exchange takes the third envelope, the one where t_γ was not disclosed, and signs this one to issue the change.

Customer: Unblind change (RSA)

1. Receive s' .
2. Compute $s := s' b_{new,\gamma}^{-1} \pmod n$.



2024-08-26

NEXT GENERATION INTERNET

└ How does cut-and-choose work?

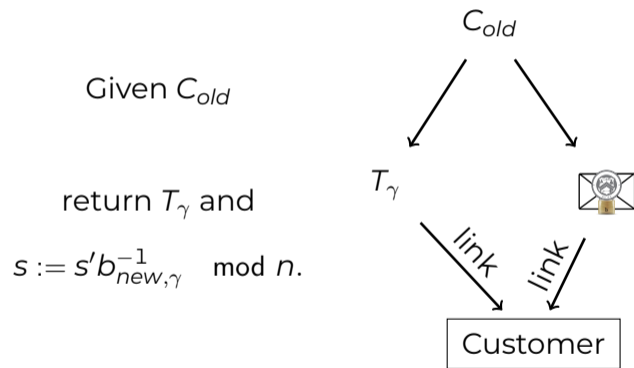
└ Customer: Unblind change (RSA)

1. Receive s' .
2. Compute $s := s' b_{new,\gamma}^{-1} \pmod n$.



1. As with the ordinary blind-signature based withdraw, the customer can then unblind the signature and has a valid coin.
2. Without knowledge of c_{old} or t_γ , the coins derived from this process are indistinguishable from coins that were withdrawn directly from an account.
3. Most importantly, without knowledge of t_γ or c_{old} , the c_{new} is unlinkable to c_{old} .

Exchange: Allow linking change



2024-08-26

NEXT GENERATION INTERNET

└ How does cut-and-choose work?

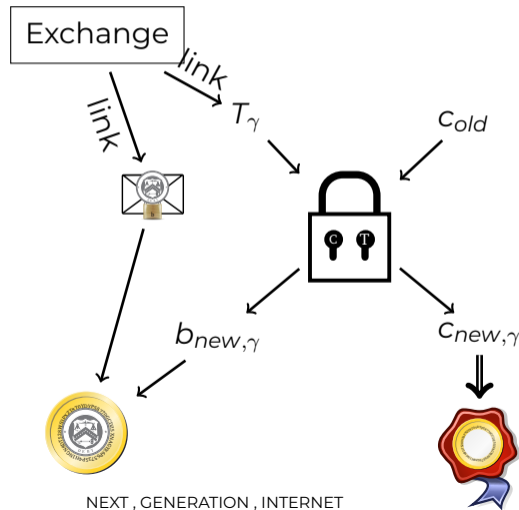
└ Exchange: Allow linking change



1. But, how does this address the issue that c_{old} may have a different owner from $c_{new,\gamma}$? Well, so far it does not! In principle, the envelope can easily be constructed by someone who was not the original owner of c_{old} .
2. So how does this help? Well, the exchange has one more sub-protocol, which is the **link** protocol. Given the old coin's public key, C_{old} , it returns T_γ , the **public transfer key**, and the blind signature over the new coin that was rendered as change.
3. Note that this is a request that the owner of c_{old} can always trivially make, as they know C_{old} .
4. So how does that help?

Customer: Link (threat!)

1. Have c_{old} .
2. Obtain T_{γ} s from exchange
3. Compute $X_{\gamma} = c_{old}T_{\gamma}$
4. Derive $c_{new,\gamma}$ and $b_{new,\gamma}$ from X_{γ}
5. Unblind $s := s'b_{new,\gamma}^{-1} \pmod n$



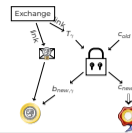
2024-08-26

NEXT GENERATION INTERNET

└ How does cut-and-choose work?

└ Customer: Link (threat!)

1. Have c_{old}
2. Obtain T_{γ} s from exchange
3. Compute $X_{\gamma} = c_{old}T_{\gamma}$
4. Derive $c_{new,\gamma}$ and $b_{new,\gamma}$ from X_{γ}
5. Unblind $s := s'b_{new,\gamma}^{-1} \pmod n$



1. Well, given these two values, the owner of the original c_{old} can **also** compute the DH (this time from c_{old} and T_{γ}), and then also derive $c_{new,\gamma}$ and also unblind the exchange's signature using $b_{new,\gamma}$.
2. As a result, the owner of the old coin can always compute the change, and thus is effectively **also** always an owner of the change rendered!
3. Thus, we have **reduced** the possibility of abusing the change protocol for a transaction that would result in a **mutually exclusive transfer of ownership** to the case where the ownership of the change is **shared**.
4. But, we previously explained that **sharing** is not something we can or would care to prevent, so the change protocol does not weaken income transparency.

Refresh protocol summary

- ▶ Customer asks exchange to convert old coin to new coin
- ▶ Protocol ensures new coins can be recovered from old coin
- ⇒ New coins are owned by the same entity!

Thus, the refresh protocol allows:

- ▶ To give unlinkable change.
- ▶ To give refunds to an anonymous customer.
- ▶ To expire old keys and migrate coins to new ones.
- ▶ To handle protocol aborts.

Transactions via refresh are equivalent to *sharing* a wallet.

- ▶ Customer asks exchange to convert old coin to new coin
 - ▶ Protocol ensures new coins can be recovered from old coin
 - ⇒ New coins are owned by the same entity!
- Thus, the refresh protocol allows:
- ▶ To give unlinkable change.
 - ▶ To give refunds to an anonymous customer.
 - ▶ To expire old keys and migrate coins to new ones.
 - ▶ To handle protocol aborts.

Transactions via refresh are equivalent to *sharing* a wallet.

1. In Taler, the overall protocol is called the **refresh** protocol, not the **change** protocol, as it has uses beyond getting unlinkable change.
2. A merchant can grant a refund to an anonymous customer by telling the exchange to nullify the original deposit. Then the anonymous owner of the original coin can obtain the refund via the refresh protocol.
3. If a coin is about to expire (because the exchange only accepts deposits for a certain denomination key for a limited amount of time), the refresh protocol can be used to obtain fresh coins, signed with the current denomination key. This is like rolling over to a fresh series of bank notes.
4. Finally, we can handle situations where the customer did try to spend digital cash, but then the message was lost, say due to a power outage, before the transaction was actually completed. But, the customer might not be sure that nobody else saw the public key of the coin! So, to ensure that transactions remain unlinkable (and that the merchant cannot deposit the coin later), the wallet can again use the refresh protocol.

How to prove protocols secure with cryptographic games?

Reminder: Cryptographic games

An *oracle* is a party in a game that the adversary can call upon to indirectly access information that is otherwise hidden from it.

For example, **IND-CPA** can be formalized like this:

Setup Generate random key k , select $b \in \{0, 1\}$ for $i \in \{1, \dots, q\}$.

Oracle Given M_0 and M_1 (of same length), return $C := \text{enc}(k, M_b)$.

The adversary wins, if it can guess b with probability greater than $\frac{1}{2} + \epsilon(\kappa)$ where $\epsilon(\kappa)$ is a negligible function in the security parameter κ .

2024-08-26

NEXT GENERATION INTERNET

└ How to prove protocols secure with cryptographic games?

└ Reminder: Cryptographic games

An oracle is a party in a game that the adversary can call upon to indirectly access information that is otherwise hidden from it.
For example, **IND-CPA** can be formalized like this:
Setup Generate random key k , select $b \in \{0, 1\}$ for $i \in \{1, \dots, q\}$.
Oracle Given M_0 and M_1 (of same length), return $C := \text{enc}(k, M_b)$.
The adversary wins, if it can guess b with probability greater than $\frac{1}{2} + \epsilon(\kappa)$ where $\epsilon(\kappa)$ is a negligible function in the security parameter κ .

Age restriction in E-commerce

Problem:

Verification of minimum age requirements in e-commerce.

Common solutions:

1. ID Verification
2. Restricted Accounts
3. Attribute-based

2024-08-26

NEXT GENERATION INTERNET

- └ How to prove protocols secure with cryptographic games?
 - └ Age restriction in E-commerce

Problem:

Verification of minimum age requirements in e-commerce.

Common solutions:

1. ID Verification
2. Restricted Accounts
3. Attribute-based

1. *Restricted Accounts* are for example Credit or Debit cards specific for children. Such cards exist e.g. in the US and only allow to purchase in certain stores. More generally, Debit cards are sometimes also used as proxy to proof that an adult is doing a purchase, e. g. when buying cigarettes at a vending machine with cash money.
2. *Attribute-based* systems that are operational are e.g. the IRMA app in the Netherlands. Another (not widespread) example is re:claimid, work done by the GNUnet e.V.,

Age restriction in E-commerce

Problem:

Verification of minimum age requirements in e-commerce.

Common solutions:

	Privacy
1. ID Verification	bad
2. Restricted Accounts	bad
3. Attribute-based	good

2024-08-26

NEXT GENERATION INTERNET

└ How to prove protocols secure with cryptographic games?

└ Age restriction in E-commerce

Problem:

Verification of minimum age requirements in e-commerce.

Common solutions:

	Privacy
1. ID Verification	bad
2. Restricted Accounts	bad
3. Attribute-based	good

1. *Restricted Accounts* are for example Credit or Debit cards specific for children. Such cards exist e.g. in the US and only allow to purchase in certain stores. More generally, Debit cards are sometimes also used as proxy to proof that an adult is doing a purchase, e. g. when buying cigarettes at a vending machine with cash money.
2. *Attribute-based* systems that are operational are e.g. the IRMA app in the Netherlands. Another (not widespread) example is re:claimid, work done by the GNUnet e.V.,

Age restriction in E-commerce

Problem:

Verification of minimum age requirements in e-commerce.

Common solutions:

	Privacy	Ext. authority
1. ID Verification	bad	required
2. Restricted Accounts	bad	required
3. Attribute-based	good	required

2024-08-26

NEXT GENERATION INTERNET

└ How to prove protocols secure with cryptographic games?

└ Age restriction in E-commerce

Problem:

Verification of minimum age requirements in e-commerce.

Common solutions:

	Privacy	Ext. authority
1. ID Verification	bad	required
2. Restricted Accounts	bad	required
3. Attribute-based	good	required

1. *Restricted Accounts* are for example Credit or Debit cards specific for children. Such cards exist e.g. in the US and only allow to purchase in certain stores. More generally, Debit cards are sometimes also used as proxy to proof that an adult is doing a purchase, e. g. when buying cigarettes at a vending machine with cash money.
2. *Attribute-based* systems that are operational are e.g. the IRMA app in the Netherlands. Another (not widespread) example is re:claimid, work done by the GNUnet e.V.,

Age restriction in E-commerce

Problem:

Verification of minimum age requirements in e-commerce.

Common solutions:

	Privacy	Ext. authority
1. ID Verification	bad	required
2. Restricted Accounts	bad	required
3. Attribute-based	good	required

Principle of Subsidiarity is violated

└ How to prove protocols secure with cryptographic games?

└ Age restriction in E-commerce

Problem:
Verification of minimum age requirements in e-commerce.

Common solutions:

	Privacy	Ext. authority
1. ID Verification	bad	required
2. Restricted Accounts	bad	required
3. Attribute-based	good	required

Principle of Subsidiarity is violated

1. *Restricted Accounts* are for example Credit or Debit cards specific for children. Such cards exist e.g. in the US and only allow to purchase in certain stores. More generally, Debit cards are sometimes also used as proxy to proof that an adult is doing a purchase, e. g. when buying cigarettes at a vending machine with cash money.
2. *Attribute-based* systems that are operational are e.g. the IRMA app in the Netherlands. Another (not widespread) example is re:claimid, work done by the GNUnet e.V.,

Principle of Subsidiarity

Functions of government—such as granting and restricting rights—should be performed *at the lowest level of authority possible*, as long as they can be performed *adequately*.

2024-08-26

NEXT GENERATION INTERNET

└ How to prove protocols secure with cryptographic games?

└ Principle of Subsidiarity

Functions of government—such as granting and restricting rights—should be performed *at the lowest level of authority possible*, as long as they can be performed *adequately*.

Historically, the concept of *subsidiarity* was developed in the law philosophy and social theory of the Catholic church and dates back to mid 1900. It is now also a core principle under which the European Union and its constituting member states operate.

Principle of Subsidiarity

Functions of government—such as granting and restricting rights—should be performed *at the lowest level of authority possible*, as long as they can be performed *adequately*.

For age-restriction, the lowest level of authority is:

Parents, guardians and caretakers

2024-08-26

NEXT GENERATION INTERNET

└ How to prove protocols secure with cryptographic games?

└ Principle of Subsidiarity

Functions of government—such as granting and restricting rights—should be performed *at the lowest level of authority possible*, as long as they can be performed *adequately*.

For age-restriction, the lowest level of authority is:

Parents, guardians and caretakers

Historically, the concept of *subsidiarity* was developed in the law philosophy and social theory of the Catholic church and dates back to mid 1900. It is now also a core principle under which the European Union and its constituting member states operate.

Age restriction

Design goals

1. Tie age restriction to the **ability to pay** (not to ID's)
2. maintain **anonymity of buyers**
3. maintain **unlinkability of transactions**
4. align with **principle of subsidiarity**
5. be **practical and efficient**

2024-08-26

NEXT GENERATION INTERNET

- └ How to prove protocols secure with cryptographic games?
 - └ Age restriction

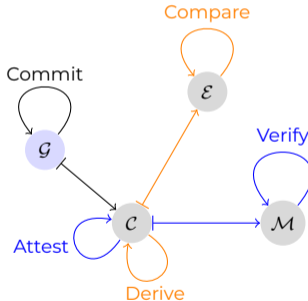
1. Tie age restriction to the **ability to pay** (not to ID's)
2. maintain **anonymity of buyers**
3. maintain **unlinkability of transactions**
4. align with **principle of subsidiarity**
5. be **practical and efficient**

1. Binding the age-restriction to the *ability* to pay also allows to make exceptions: For instance, a guardian might allow its 14 year old child a limited amount of money with a higher maximum age, or no age-restriction attached to it. This is also an example for the application of the principle of subsidiarity.
2. In times of Blockchains it seems to be necessary to insist on the *practicality and efficiency* of such schemes.

Age restriction

Assumptions and scenario

- ▶ Assumption: Checking accounts are under control of eligible adults/guardians.
- ▶ *Guardians* **commit** to an maximum age
- ▶ *Minors* **attest** their adequate (minimum) age
- ▶ *Merchants* **verify** the attestations
- ▶ Minors **derive** age commitments from existing ones
- ▶ *Exchanges* **compare** the derived age commitments



2024-08-26

NEXT GENERATION INTERNET

- └ How to prove protocols secure with cryptographic games?
 - └ Age restriction

- ▶ Assumption: Checking accounts are under control of eligible adults/guardians.
- ▶ Guardians **commit** to an maximum age
- ▶ Minors **attest** their adequate (minimum) age
- ▶ Merchants **verify** the attestations
- ▶ Minors **derive** age commitments from existing ones
- ▶ Exchanges **compare** the derived age commitments



1. This scheme is designed independent of any particular payment service protocol.
2. Note the difference between the *maximum* age, that a guardian commits to, and the required *minimum* (or *adequate*) age that the merchant requires. The maximum age is what the child can prove to be *at the maximum*. In a particular purchase, the required minimum age can also be lower than what the child can prove (but not higher)
3. Of course, the scheme also needs to work for *adults*! In that case, the two participants in the scheme *guardian* and *child* are the same person. Adults simply commit themselves to the maximum age (or age group) permitted in the scheme.
4. The derive and compare operations exist to avoid that a child has to ask the guardian *again* for a new commitment after each purchase. The guardian might not be available, while the payment service usually is permanently online. Therefore, the compare step to ensure the equivalence of two

Formal function signatures

Searching for functions with the following signatures

Commit :	$(a, \omega) \mapsto (Q, P)$	$\mathbb{N}_M \times \Omega \rightarrow \mathbb{O} \times \mathbb{P}$,
Attest :	$(m, Q, P) \mapsto T$	$\mathbb{N}_M \times \mathbb{O} \times \mathbb{P} \rightarrow \mathbb{T} \cup \{\perp\}$,
Verify :	$(m, Q, T) \mapsto b$	$\mathbb{N}_M \times \mathbb{O} \times \mathbb{T} \rightarrow \mathbb{Z}_2$,
Derive :	$(Q, P, \omega) \mapsto (Q', P', \beta)$	$\mathbb{O} \times \mathbb{P} \times \Omega \rightarrow \mathbb{O} \times \mathbb{P} \times \mathbb{B}$,
Compare :	$(Q, Q', \beta) \mapsto b$	$\mathbb{O} \times \mathbb{O} \times \mathbb{B} \rightarrow \mathbb{Z}_2$,

with $\Omega, \mathbb{P}, \mathbb{O}, \mathbb{T}, \mathbb{B}$ sufficiently large sets.

Basic and security requirements are defined later.

Mnemonics:

$\mathbb{O} = c\mathbb{O}mmitments$, $Q = Q\text{-}mitment$ (commitment), $\mathbb{P} = \mathbb{P}roofs$, $P = P\text{ro}of$,

$\mathbb{T} = a\mathbb{T}testations$, $T = a\mathbb{T}testation$, $\mathbb{B} = \mathbb{B}lindings$, $\beta = \beta\text{l}inding$.

└ How to prove protocols secure with cryptographic games?

└ Formal function signatures

Searching for functions with the following signatures

Commit :	$(a, \omega) \mapsto (Q, P)$	$\mathbb{N}_M \times \Omega \rightarrow \mathbb{O} \times \mathbb{P}$,
Attest :	$(m, Q, P) \mapsto T$	$\mathbb{N}_M \times \mathbb{O} \times \mathbb{P} \rightarrow \mathbb{T} \cup \{\perp\}$,
Verify :	$(m, Q, T) \mapsto b$	$\mathbb{N}_M \times \mathbb{O} \times \mathbb{T} \rightarrow \mathbb{Z}_2$,
Derive :	$(Q, P, \omega) \mapsto (Q', P', \beta)$	$\mathbb{O} \times \mathbb{P} \times \Omega \rightarrow \mathbb{O} \times \mathbb{P} \times \mathbb{B}$,
Compare :	$(Q, Q', \beta) \mapsto b$	$\mathbb{O} \times \mathbb{O} \times \mathbb{B} \rightarrow \mathbb{Z}_2$,

with $\Omega, \mathbb{P}, \mathbb{O}, \mathbb{T}, \mathbb{B}$ sufficiently large sets.

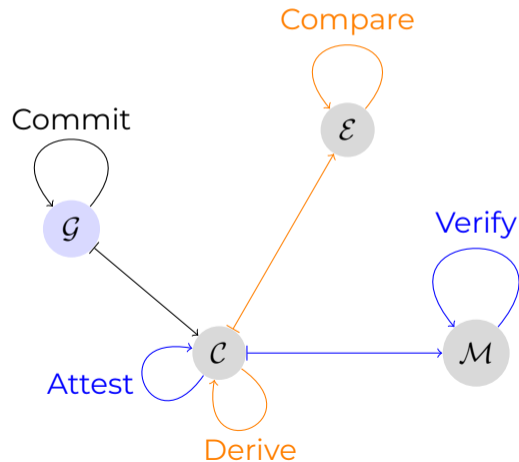
Basic and security requirements are defined later.

Mnemonics:
 $\mathbb{O} = c\mathbb{O}mmitments$, $Q = Q\text{-}mitment$ (commitment), $\mathbb{P} = \mathbb{P}roofs$, $P = P\text{ro}of$,
 $\mathbb{T} = a\mathbb{T}testations$, $T = a\mathbb{T}testation$, $\mathbb{B} = \mathbb{B}lindings$, $\beta = \beta\text{l}inding$.

1. The sets can be all considered to be sufficiently large finite subsets of $\{0, 1\}^*$.

Age restriction

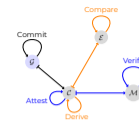
Naïve scheme



2024-08-26

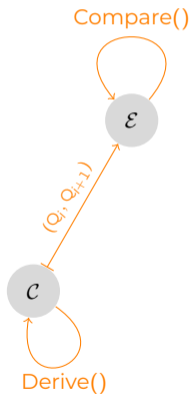
NEXT GENERATION INTERNET

- How to prove protocols secure with cryptographic games?
 - Age restriction



- The diagram provides a sketch of the participants in the scheme, the callers and calls to the five functions and the data flow between the participants.
- It is important to hint at the iterative nature of Derive-Compare in particular, which—in this naive version of the scheme—leads to the linkability of sequences of commitments (see next slide).

Achieving unlinkability



Simple use of `Derive()` and `Compare()` is problematic.

- ▶ Calling `Derive()` iteratively generates sequence (Q_0, Q_1, \dots) of commitments.
- ▶ Exchange calls `Compare(Q_i, Q_{i+1}, \dots)`

⇒ **Exchange identifies sequence**

⇒ **Unlinkability broken**

2024-08-26

NEXT GENERATION INTERNET

└ How to prove protocols secure with cryptographic games?

└ Achieving unlinkability

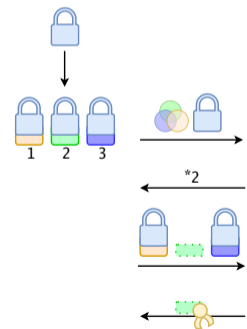


Simple use of `Derive()` and `Compare()` is problematic.

- ▶ Calling `Derive()` iteratively generates sequence (Q_0, Q_1, \dots) of commitments.
- ▶ Exchange calls `Compare(Q_i, Q_{i+1}, \dots)`
- **Exchange identifies sequence**
- **Unlinkability broken**

Achieving unlinkability

Cut&choose protocol **DeriveCompare $_{\kappa}$** using **Derive()** and **Compare()**:



1. \mathcal{C} derives commitments (Q_1, \dots, Q_{κ}) from Q_0 by calling **Derive()** with blindings $(\beta_1, \dots, \beta_{\kappa})$
2. \mathcal{C} calculates $h_0 := H(H(Q_1, \beta_1) || \dots || H(Q_{\kappa}, \beta_{\kappa}))$
3. \mathcal{C} sends Q_0 and h_0 to \mathcal{E}
4. \mathcal{E} chooses $\gamma \in \{1, \dots, \kappa\}$ randomly
5. \mathcal{C} reveals $h_{\gamma} := H(Q_{\gamma}, \beta_{\gamma})$ and all (Q_i, β_i) , except $(Q_{\gamma}, \beta_{\gamma})$
6. \mathcal{E} compares h_0 and $H(H(Q_1, \beta_1) || \dots || h_{\gamma} || \dots || H(Q_{\kappa}, \beta_{\kappa}))$ and evaluates **Compare** (Q_0, Q_i, β_i) .
7. On success, Q_{γ} will be the result

2024-08-26

NEXT GENERATION INTERNET

How to prove protocols secure with cryptographic games?

Achieving unlinkability

Cut&choose protocol **DeriveCompare**, using **Derive()** and **Compare()**:

1. \mathcal{C} derives commitments (Q_1, \dots, Q_{κ}) from Q_0 by calling **Derive()** with blindings $(\beta_1, \dots, \beta_{\kappa})$
2. \mathcal{C} calculates $h_0 := H(H(Q_1, \beta_1) || \dots || H(Q_{\kappa}, \beta_{\kappa}))$
3. \mathcal{C} sends Q_0 and h_0 to \mathcal{E}
4. \mathcal{E} chooses $\gamma \in \{1, \dots, \kappa\}$ randomly
5. \mathcal{C} reveals $h_{\gamma} := H(Q_{\gamma}, \beta_{\gamma})$ and all (Q_i, β_i) , except $(Q_{\gamma}, \beta_{\gamma})$
6. \mathcal{E} compares h_0 and $H(H(Q_1, \beta_1) || \dots || h_{\gamma} || \dots || H(Q_{\kappa}, \beta_{\kappa}))$ and evaluates **Compare** (Q_0, Q_i, β_i) .
7. On success, Q_{γ} will be the result

1. This is another example where cut&choose is used as a generic, knowledge-hiding "overlay"-protocol.
2. It leads to the same *combined* results of the calls to **Derive** and **Compare** in succession. In particular, the protocol **DeriveCompare $_{\kappa}$** , considered as function, has the signature

$$\mathbb{O} \times \mathbb{P} \times \Omega \rightarrow \mathbb{O} \times \mathbb{P} \times \mathbb{B} \times \{0, 1\}$$

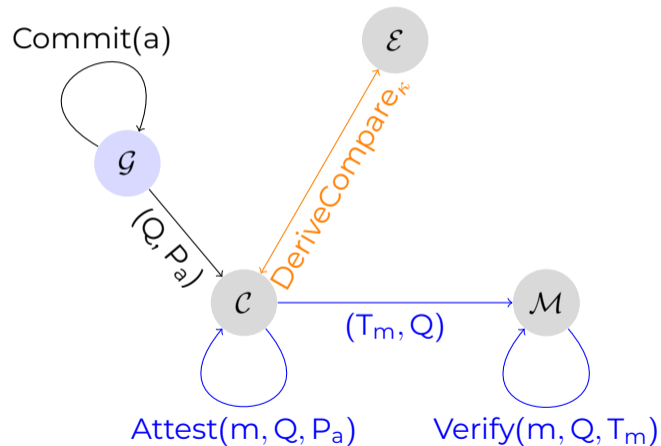
3. Scheme is similar to the *refresh* protocol in GNU Taler.

Achieving unlinkability

With **DeriveCompare $_{\kappa}$**

- ▶ \mathcal{E} learns nothing about Q_{γ} ,
- ▶ trusts outcome with $\frac{\kappa-1}{\kappa}$ certainty,
- ▶ i.e. \mathcal{C} has $\frac{1}{\kappa}$ chance to cheat.

Note: Still need Derive and Compare to be defined.



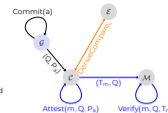
2024-08-26

NEXT GENERATION INTERNET

How to prove protocols secure with cryptographic games?

Achieving unlinkability

With **DeriveCompare $_{\kappa}$** ,
▶ \mathcal{E} learns nothing about Q_{γ} ,
▶ trusts outcome with $\frac{\kappa-1}{\kappa}$ certainty,
▶ i.e. \mathcal{C} has $\frac{1}{\kappa}$ chance to cheat.
Note: Still need Derive and Compare to be defined.



1. The image on the right shows the refined scheme using **DeriveCompare $_{\kappa}$** for unlinkability.
2. Consider also a variant of the scheme, where instead of sending *one* γ , \mathcal{E} sends $\vec{\gamma} \in \{1, \dots, \kappa\}^M$ (that is: one value per age-group). Then the chance for \mathcal{C} to cheat would *fall* (depending on how many age groups are above the committed one and are therefore worth cheating).

Basic requirements

Candidate functions

(Commit, Attest, Verify, Derive, Compare)

must first meet *basic* requirements:

- ▶ Existence of attestations
- ▶ Efficacy of attestations
- ▶ Derivability of commitments and attestations

2024-08-26

NEXT GENERATION INTERNET

└ How to prove protocols secure with cryptographic games?

└ Basic requirements

Candidate functions
(Commit, Attest, Verify, Derive, Compare)
must first meet basic requirements:
▶ Existence of attestations
▶ Efficacy of attestations
▶ Derivability of commitments and attestations

Basic requirements

Formal details

Existence of attestations

$$\forall_{\substack{a \in \mathbb{N}_M \\ \omega \in \Omega}} : \text{Commit}(a, \omega) =: (Q, P) \implies \text{Attest}(m, Q, P) = \begin{cases} T \in \mathbb{T}, & \text{if } m \leq a \\ \perp & \text{otherwise} \end{cases}$$

Efficacy of attestations

$$\text{Verify}(m, Q, T) = \begin{cases} 1, & \text{if } \exists_{P \in \mathbb{P}} : \text{Attest}(m, Q, P) = T \\ 0 & \text{otherwise} \end{cases}$$

$$\forall_{n \leq a} : \text{Verify}(n, Q, \text{Attest}(n, Q, P)) = 1.$$

etc.

2024-08-26

NEXT GENERATION INTERNET

└ How to prove protocols secure with cryptographic games?

└ Basic requirements

Existence of attestations

$$\forall_{\substack{a \in \mathbb{N}_M \\ \omega \in \Omega}} : \text{Commit}(a, \omega) =: (Q, P) \implies \text{Attest}(m, Q, P) = \begin{cases} T \in \mathbb{T}, & \text{if } m \leq a \\ \perp & \text{otherwise} \end{cases}$$

Efficacy of attestations

$$\text{Verify}(m, Q, T) = \begin{cases} 1, & \text{if } \exists_{P \in \mathbb{P}} : \text{Attest}(m, Q, P) = T \\ 0 & \text{otherwise} \end{cases}$$

etc.

$$\forall_{n \leq a} : \text{Verify}(n, Q, \text{Attest}(n, Q, P)) = 1.$$

1. The first one is basically that the Attest function will always be successful if asked to provide an age restriction proof for an age smaller than the commitment, and fail if the requested age is larger.
2. The second equation simply states that Verify must pass age restriction proofs generated from Attest for the correct age, and fail otherwise.
3. Equivalent intuitive requirements on the preservation of the age restriction permissions apply to Derive and Compare.

Security requirements

Candidate functions must also meet *security* requirements. Those are defined via security games:

- ▶ Game: Age disclosure by commitment or attestation
- ↔ Requirement: Non-disclosure of age
- ▶ Game: Forging attestation
- ↔ Requirement: Unforgeability of minimum age
- ▶ Game: Distinguishing derived commitments and attestations
- ↔ Requirement: Unlinkability of commitments and attestations

Meeting the security requirements means that adversaries can win those games only with negligible advantage.

Adversaries are arbitrary polynomial-time algorithms, acting on all relevant input.

2024-08-26

NEXT GENERATION INTERNET

└─ How to prove protocols secure with cryptographic games?

└─ Security requirements

Candidate functions must also meet *security* requirements. Those are defined via security games:

- ▶ Game: Age disclosure by commitment or attestation
- ↔ Requirement: Non-disclosure of age
- ▶ Game: Forging attestation
- ↔ Requirement: Unforgeability of minimum age
- ▶ Game: Distinguishing derived commitments and attestations
- ↔ Requirement: Unlinkability of commitments and attestations

Meeting the security requirements means that adversaries can win those games only with negligible advantage.
Adversaries are arbitrary polynomial-time algorithms, acting on all relevant input.

Security requirements

Simplified example

Game $G_{\mathcal{A}}^{\text{FA}}(\lambda)$ —Forging an attest:

1. $(a, \omega) \xleftarrow{\$} \mathbb{N}_{M-1} \times \Omega$
2. $(Q, P) \leftarrow \text{Commit}(a, \omega)$
3. $(m, T) \leftarrow \mathcal{A}(a, Q, P)$
4. Return 0 if $m \leq a$
5. Return $\text{Verify}(m, Q, T)$

Requirement: Unforgeability of minimum age

$$\forall \mathcal{A} \in \mathfrak{A}(\mathbb{N}_M \times \mathbb{O} \times \mathbb{P} \rightarrow \mathbb{N}_M \times \mathbb{T}) : \Pr \left[G_{\mathcal{A}}^{\text{FA}}(\lambda) = 1 \right] \leq \epsilon(\lambda)$$

└ How to prove protocols secure with cryptographic games?

└ Security requirements

Game $G_{\mathcal{A}}^{\text{FA}}(\lambda)$ —Forging an attest:

1. $(a, \omega) \xleftarrow{\$} \mathbb{N}_{M-1} \times \Omega$
2. $(Q, P) \leftarrow \text{Commit}(a, \omega)$
3. $(m, T) \leftarrow \mathcal{A}(a, Q, P)$
4. Return 0 if $m \leq a$
5. Return $\text{Verify}(m, Q, T)$

Requirement: Unforgeability of minimum age

$$\forall \mathcal{A} \in \mathfrak{A}(\mathbb{N}_M \times \mathbb{O} \times \mathbb{P} \rightarrow \mathbb{N}_M \times \mathbb{T}) : \Pr \left[G_{\mathcal{A}}^{\text{FA}}(\lambda) = 1 \right] \leq \epsilon(\lambda)$$

1. This is a simplified version of the actual game from the paper. In this simplified version we don't consider the Derive operation, i. e. we don't provide the adversary with additional commitments and proofs coming from a calls to Derive (instead of Commit).

Solution: Instantiation with ECDSA



age groups	key ID's
1:	b5bb9d
2:	801fa0
3:	19d8de
4:	52f23c

To **Commit** to age (group) $a \in \{1, \dots, M\}$

- Guardian generates ECDSA-keypairs, one per age (group):

$$\langle (q_1, p_1), \dots, (q_M, p_M) \rangle$$

- Guardian then **drops** all private keys p_i for $i > a$:

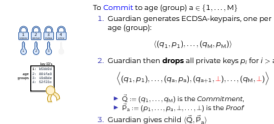
$$\langle (q_1, p_1), \dots, (q_a, p_a), (q_{a+1}, \perp), \dots, (q_M, \perp) \rangle$$

- ▶ $\vec{Q} := (q_1, \dots, q_M)$ is the *Commitment*,
- ▶ $\vec{P}_a := (p_1, \dots, p_a, \perp, \dots, \perp)$ is the *Proof*

- Guardian gives child $\langle \vec{Q}, \vec{P}_a \rangle$

How to prove protocols secure with cryptographic games?

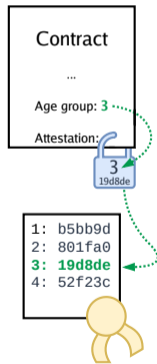
Solution: Instantiation with ECDSA



- The scheme requires M keys for M age groups.
- Thus, some key costs are linear in the number of age groups.
- Fortunately, below 6 and above 21 usually nobody cares, so M is small.
- In principle, the keys could represent arbitrary capabilities of programmable money; using them for anything but age-restrictions may raise ethical concerns. GNU Taler only supports the use-case for age-restrictions.

Instantiation with ECDSA

Definitions of Attest and Verify



Child has

- ▶ ordered public-keys $\vec{Q} = (q_1, \dots, q_M)$,
- ▶ (some) private-keys $\vec{P} = (p_1, \dots, p_a, \perp, \dots, \perp)$.

To Attest a minimum age $m \leq a$:

Sign a message with ECDSA using private key

p_m

Merchant gets

- ▶ ordered public-keys $\vec{Q} = (q_1, \dots, q_M)$
- ▶ Signature σ

To Verify a minimum age m :

Verify the ECDSA-Signature σ with public key

q_m .

2024-08-26

NEXT GENERATION INTERNET

└ How to prove protocols secure with cryptographic games?

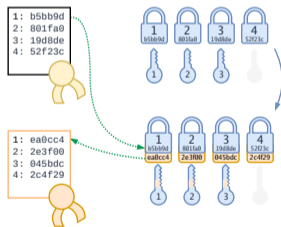
└ Instantiation with ECDSA



1. Performance primarily matters for the user during payment.
2. All other operations usually happen in the background.
3. Thus, it is good that attestation and verification are cheap ($O(1)$) and independent of M .

Instantiation with ECDSA

Definitions of Derive and Compare



Child has $\vec{Q} = (q_1, \dots, q_M)$ and $\vec{P} = (p_1, \dots, p_a, \perp, \dots, \perp)$.

To Derive new \vec{Q}' and \vec{P}' :

Choose random $\beta \in \mathbb{Z}_g$ and calculate

$$\vec{Q}' := (\beta * q_1, \dots, \beta * q_M),$$

$$\vec{P}' := (\beta p_1, \dots, \beta p_a, \perp, \dots, \perp)$$

Note: $(\beta p_i) * G = \beta * (p_i * G) = \beta * q_i$
 $\beta * q_i$ is scalar multiplication on the elliptic curve.

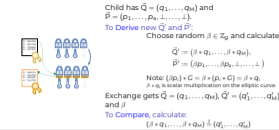
Exchange gets $\vec{Q} = (q_1, \dots, q_M)$, $\vec{Q}' = (q'_1, \dots, q'_M)$ and β

To Compare, calculate:

$$(\beta * q_1, \dots, \beta * q_M) \stackrel{?}{=} (q'_1, \dots, q'_M)$$

How to prove protocols secure with cryptographic games?

Instantiation with ECDSA



1. The cryptography used for blinding is inspired by [5].

Instantiation with ECDSA

Functions (Commit, Attest, Verify, Derive, Compare)
as defined in the instantiation with ECDSA

- ▶ meet the basic requirements,
- ▶ also meet all security requirements.
Proofs by security reduction, details are in the paper.

2024-08-26

NEXT GENERATION INTERNET

└ How to prove protocols secure with cryptographic games?

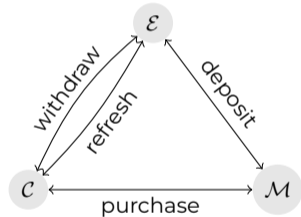
└ Instantiation with ECDSA

Functions (Commit, Attest, Verify, Derive, Compare)
as defined in the instantiation with ECDSA

- ▶ meet the basic requirements,
- ▶ also meet all security requirements.
Proofs by security reduction, details are in the paper.

Integration with GNU Taler

Key operations in the original system



- ▶ Coins are public-/private key-pairs (C_p, c_s) .
- ▶ Exchange blindly signs $\text{FDH}(C_p)$ with denomination key d_p
- ▶ Verification:

$$1 \stackrel{?}{=} \text{SigCheck}(\text{FDH}(C_p), D_p, \sigma_p)$$

(D_p = public key of denomination and σ_p = signature)

2024-08-26

NEXT GENERATION INTERNET

└ How to prove protocols secure with cryptographic games?

└ Integration with GNU Taler



- ▶ Coins are public-/private key-pairs (C_p, c_s) .
- ▶ Exchange blindly signs $\text{FDH}(C_p)$ with denomination key d_p
- ▶ Verification:
 $1 \stackrel{?}{=} \text{SigCheck}(\text{FDH}(C_p), D_p, \sigma_p)$
(D_p = public key of denomination and σ_p = signature)

The diagram on the left now is a sketch of the main protocols of GNU Taler. The participants in our scheme and in GNU Taler are almost the same: The guardian is missing in GNU Taler.

Integration with GNU Taler

Binding age restriction to coins

To bind an age commitment Q to a coin C_p , instead of signing $\text{FDH}(C_p)$, \mathcal{E} now blindly signs

$$\text{FDH}(C_p, H(Q))$$

Verification of a coin now requires $H(Q)$, too:

$$1 \stackrel{?}{=} \text{SigCheck}(\text{FDH}(C_p, H(Q)), D_p, \sigma_p)$$

2024-08-26

NEXT GENERATION INTERNET

└ How to prove protocols secure with cryptographic games?

└ Integration with GNU Taler

To bind an age commitment Q to a coin C_p , instead of signing $\text{FDH}(C_p)$, \mathcal{E} now blindly signs

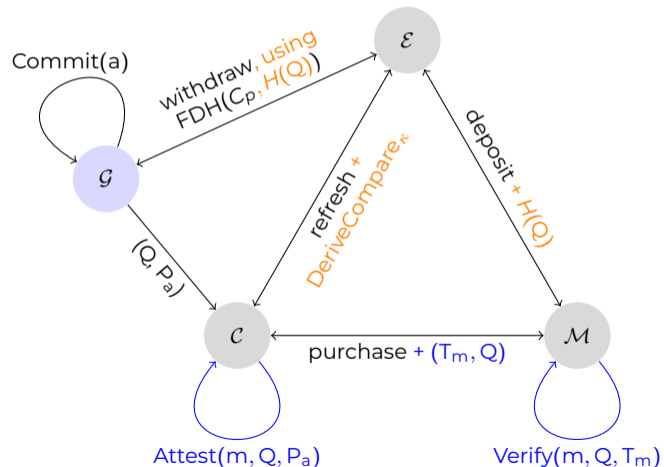
$$\text{FDH}(C_p, H(Q))$$

Verification of a coin now requires $H(Q)$, too:

$$1 \stackrel{?}{=} \text{SigCheck}(\text{FDH}(C_p, H(Q)), D_p, \sigma_p)$$

Integration with GNU Taler

Integrated schemes

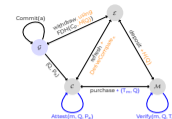


2024-08-26

NEXT GENERATION INTERNET

How to prove protocols secure with cryptographic games?

Integration with GNU Taler



1. In the diagram we do not mention the actual coin material. For example, the guardian not only gives the age-commitment and -proof to the child, but also the corresponding coin's private key to which the commitment is bound to.
2. In an actual implementation of the subsidiary flow of the age-restriction scheme, one might choose to place the Commit and withdraw step actually onto the wallet of the *child*, and instruct parents to make sure that the child has set appropriate age-restrictions prior to filling the reserve.
3. Note that any two coins of the same denomination and with age restriction set are still indistinguishable.

Instantiation with Edx25519

Paper also formally defines another signature scheme: Edx25519.

- ▶ Scheme already in use in GNUnet,
- ▶ based on EdDSA (Bernstein et al.),
- ▶ generates compatible signatures and
- ▶ allows for key derivation from both, private and public keys, independently.

Current implementation of age restriction in GNU Taler uses Edx25519.

2024-08-26

NEXT GENERATION INTERNET

└ How to prove protocols secure with cryptographic games?

└ Instantiation with Edx25519

Paper also formally defines another signature scheme: Edx25519.

- ▶ Scheme already in use in GNUnet,
- ▶ based on EdDSA (Bernstein et al.),
- ▶ generates compatible signatures and
- ▶ allows for key derivation from both, private and public keys, independently.

Current implementation of age restriction in GNU Taler uses Edx25519.

- ▶ Approach can be used with any token-based payment scheme
- ▶ Subsidiarity requires bank accounts being owned by adults
- ▶ Scheme can be adapted to case where minors have bank accounts
 - ▶ Assumption: banks provide minimum age information during bank transactions.
 - ▶ Child and Exchange execute a variant of the cut&choose protocol.

└ How to prove protocols secure with cryptographic games?

└ Discussion

- ▶ Approach can be used with any token-based payment scheme
- ▶ Subsidiarity requires bank accounts being owned by adults
- ▶ Scheme can be adapted to case where minors have bank accounts
 - ▶ Assumption: banks provide minimum age information during bank transactions.
 - ▶ Child and Exchange execute a variant of the cut&choose protocol.

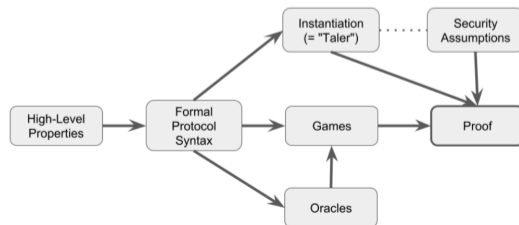
1. It is worth noting that the scheme would not work as a pure age-verification scheme, *independent* of any payment: Without the financial punishment for cheating, the child can try $\text{DeriveCompare}_\kappa$ often enough until it succeeds in gets the maximal possible age commitment.

What are the future plans for GNU Taler?

Summary

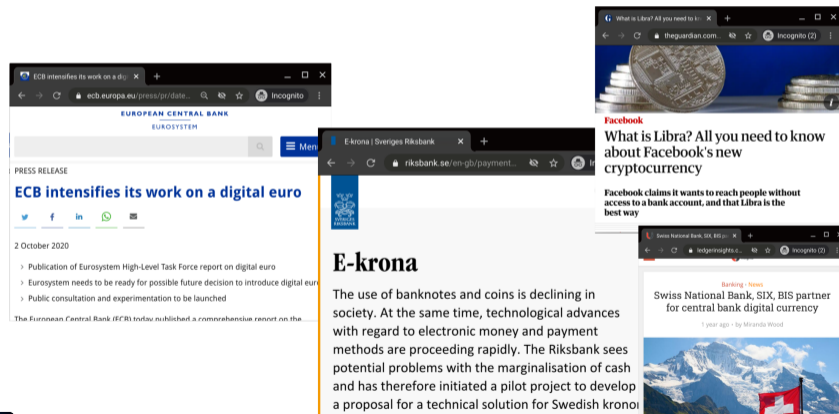
GNU Taler:

- ▶ Gives change, can provide refunds
- ▶ Integrates nicely with HTTP, handles network failures
- ▶ Has high performance
- ▶ Is Free Software
- ▶ Includes formal security proofs



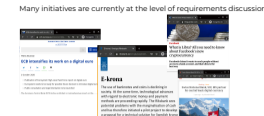
CBDC initiatives and GNU Taler

Many initiatives are currently at the level of requirements discussion:



2024-08-26

NEXT GENERATION INTERNET
└─ What are the future plans for GNU Taler?
└─ CBDC initiatives and GNU Taler



1. CBDC = Central Bank Digital Currency, so digital payment systems operated by the central bank and where the money is a liability of the central bank.
2. Taler can serve as the foundation for a *bearer-based retail* CBDC.
3. Taler replicates physical cash rather than bank deposits

Unique regulatory features for CBs

1. Central bank issues digital coins equivalent to issuing cash
2. Architecture with consumer accounts at commercial banks
3. Withdrawal limits and denomination expiration
4. Income transparency and possibility to set fees
5. Revocation protocols and loss limitations
6. Privacy by cryptographic design not organizational compliance

Political support needed, talk to your representatives!

2024-08-26

NEXT GENERATION INTERNET

└─What are the future plans for GNU Taler?

└─Unique regulatory features for CBs

1. ⇒ monetary policy remains under CB control
2. ⇒ no competition for commercial banking (S&L) and
⇒ CB does not have to manage KYC, customer support
3. ⇒ protects against bank runs and hoarding
4. ⇒ additional insights into economy and new policy options
5. ⇒ exit strategy and handles catastrophic security incidents
6. ⇒ **CB cannot be forced to facilitate mass-surveillance**

1. Central bank issues digital coins equivalent to issuing cash
2. Architecture with consumer accounts at commercial banks
3. Withdrawal limits and denomination expiration
4. Income transparency and possibility to set fees
5. Revocation protocols and loss limitations
6. Privacy by cryptographic design not organizational compliance

Political support needed, talk to your representatives!

Ongoing work

- ▶ Post-quantum blind signatures
- ▶ Integration into more physical machines
- ▶ Integration with KYC/AML providers
- ▶ Deployment for regional currency in Basel
- ▶ Integration with Swiss Postfinance EBICS API
- ▶ Wallet backup and recovery with Anastasis
- ▶ Internationalization ⇒ <https://weblate.taler.net/>

<https://bugs.taler.net/> tracks open issues.

2024-08-26

NEXT GENERATION INTERNET

└─ What are the future plans for GNU Taler?

└─ Ongoing work

- ▶ Post-quantum blind signatures
- ▶ Integration into more physical machines
- ▶ Integration with KYC/AML providers
- ▶ Deployment for regional currency in Basel
- ▶ Integration with Swiss Postfinance EBICS API
- ▶ Wallet backup and recovery with Anastasis
- ▶ Internationalization ⇒ <https://weblate.taler.net/>

<https://bugs.taler.net/> tracks open issues.

1. This list naturally changes frequently.
2. The key message at this time is that Taler is currently operated at a small scale, but we expect to grow it much bigger soon.
3. Help with translations is always welcome.

Open issues

- ▶ Address remaining scalability challenges (multiple topics)
- ▶ Porting to more platforms (Web shops, iOS, embedded, ...)
- ▶ Integration with e-commerce frameworks (Prestashop, OpenCart, ECWID, ...)
- ▶ Currency conversion
- ▶ SAP integration
- ▶ Design and usability for illiterate and innumerate users
- ▶ Federated exchange (wads)
- ▶ ...

Help needed, talk to us (e.g. at <https://ich.taler.net/>)

- ▶ Address remaining scalability challenges (multiple topics)
- ▶ Porting to more platforms (Web shops, iOS, embedded, ...)
- ▶ Integration with e-commerce frameworks (Prestashop, OpenCart, ECWID, ...)
- ▶ Currency conversion
- ▶ SAP integration
- ▶ Design and usability for illiterate and innumerate users
- ▶ Federated exchange (wads)
- ▶ ...

Help needed, talk to us (e.g. at <https://ich.taler.net/>)


1. GNU Taler already performs well, but there are many, many scenarios to test and further optimize. So plenty of work to go around!
2. Similarly, payments have many applications, so integrating GNU Taler into any business process that involves a customer payment is literally work that will never end. But, it should be simple if you actually understand the business process.
3. Usability is another critical topic, and we hope to address the needs of more than just the 99% of commercially viable users.

- ▶ Be paid to read advertising, starting with spam
- ▶ Give welfare without intermediaries taking huge cuts
- ▶ Forster regional trade via regional currencies
- ▶ Eliminate corruption by making all income visible
- ▶ Stop the mining by making crypto-currencies useless for anything but crime

- ▶ Be paid to read advertising, starting with spam
- ▶ Give welfare without intermediaries taking huge cuts
- ▶ Forster regional trade via regional currencies
- ▶ Eliminate corruption by making all income visible
- ▶ Stop the mining by making crypto-currencies useless for anything but crime

1. These are some of the more speculative results that we hope to eventually achieve using GNU Taler.
2. Do you have any interesting ideas of what one could also accomplish with this system?

References I

-  Jeffrey Burdges, Florian Dold, Christian Grothoff, and Marcello Stanisci.
Enabling secure web payments with GNU Taler.
In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors, *6th International Conference on Security, Privacy and Applied Cryptographic Engineering*, number 10076 in LNCS, pages 251–270. Springer, Dec 2016.

2024-08-26


NEXT GENERATION INTERNET

└References



└References

Jeffrey Burdges, Florian Dold, Christian Grothoff, and Marcello Stanisci.
Enabling secure web payments with GNU Taler.
In Claude Carlet, M. Anwar Hasan, and Vishal Saraswat, editors, *6th International Conference on Security, Privacy and Applied Cryptographic Engineering*, number 10076 in LNCS, pages 251–270. Springer, Dec 2016.

References II

-  David Chaum, Christian Grothoff, and Thomas Moser.
How to issue a central bank digital currency.
In *SNB Working Papers*, number 2021-3. Swiss National Bank,
February 2021.
-  Florian Dold.
*The GNU Taler system: practical and provably secure electronic
payments. (Le système GNU Taler: Paiements électroniques
pratiques et sécurisés).*
PhD thesis, University of Rennes 1, France, 2019.

References III

-  Phillip Rogaway.
The moral character of cryptographic work.
Austin, TX, August 2016. USENIX Association.
-  Martin Schanzenbach, Christian Grothoff, and Bernd Fix.
The GNU Name System.
RFC 9498, November 2023.

2024-08-26

NEXT GENERATION INTERNET

└References

└References

 Phillip Rogaway.
The moral character of cryptographic work.
Austin, TX, August 2016. USENIX Association.
 Martin Schanzenbach, Christian Grothoff, and Bernd Fix.
The GNU Name System.
RFC 9498, November 2023.

Acknowledgements


Co-funded by the European Union (Project 101135475).



Co-funded by
the European Union

Co-funded by SERI (HEU-Projekt 101135475-TALER).

Project funded by

 Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs,
Education and Research EAER
**State Secretariat for Education,
Research and Innovation SERI**

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

GNU Taler

Christian Grothoff

Security and Privacy in Digital Payments

1. Ethical case study: Private payments

“A company is developing new software for private payments. This will enable its customers to transact with “complete” privacy (like cash). The solution does not include backdoors, and thus the company cannot block payments to support trade embargos or anti money laundering efforts.”

- Discuss virtues and vices affected.
- Does it make a difference if the software is Free Software developed by an individual or a community instead of proprietary software from a company?
- Research the case of Alexey Pertsev!

2. Use GNU Taler

- Install the GNU Taler wallet Web extension in Firefox from <https://wallet.taler.net/>
- Go to “Help”, “Add-ons and themes”, click the settings icon and select “Debug Add-ons”, then select “Inspect” for the “GNU Taler wallet” to see the developer console for the Taler wallet.
- Register for an account and withdraw funds from <https://bank.demo.taler.net/> and then buy an article at <https://shop.demo.taler.net/>. Observe the HTTPS requests made by the Web extension. During withdraw, you should see “config”, “keys”, “reserves” and “batch-withdraw”. During payment, you should see a “claim”, “pay”, and likely a “melt” and “reveal”. If you did not see a “melt” and “reveal”, purchase more articles until you do.
- Use <https://docs.taler.net/> to map these requests to the protocol explained in the course and use this to answer the following questions:
 - How much bandwidth does a payment take? What does this suggest about the amount of storage an exchange or a wallet need per transaction?
 - Given a 1 Gigabit/second link, what is the maximum TPS per second Taler could do?

- Given a 1 Terrabyte disk, how many transactions should **roughly** fit?
- Look at the “Timings” tab. Which operation took the longest? What is the reason?
- Where is the γ value for the cut-and-choose step in the protocol? Find the respective request and the JSON field name in the response.
- Keep the inspector console open and surf the Internet. What else could you do to establish that the wallet is respecting your privacy as advertised?

NEXT GENERATION INTERNET

Distributed systems

Christian Grothoff

07.06.2024

2024-08-26

NEXT GENERATION INTERNET

NEXT
GENERATION
INTERNET
Distributed systems

Christian Grothoff

07.06.2024

1. Distributed systems are defined as “computer systems whose inter-communicating components are located on different networked computers”.
2. Information security is mostly interesting in the context of distribute systems.

Learning objectives

What are typical attacks on decentralized systems?

What should we think about when building distributed systems?

What are impossibly hard problems in distributed systems security?

How can we work around such hard issues?

Bonus: Depolymerization [7]

2024-08-26

NEXT GENERATION INTERNET

Learning objectives

1. We will begin by looking at attacks on decentralized systems, which are distributed systems where any party is easily replaced by any other party.
2. Next, we will look at common mistakes system architects make, to inform what we need to think of when designing distributed systems.
3. Then, we will study various common hard problems, some of which have formal impossibility proofs and others that have “merely” been identified as extremely hard and likely impossible. Knowing that you deal with a hard problem is informative: it implies that you need to find ways to work around it, instead of attacking it directly.
4. Finally, we will look at common ways to work around some of these hard problems.

What are typical attacks on decentralized systems?

What should we think about when building distributed systems?

What are impossibly hard problems in distributed systems security?

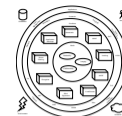
How can we work around such hard issues?

Bonus: Depolymerization [7]

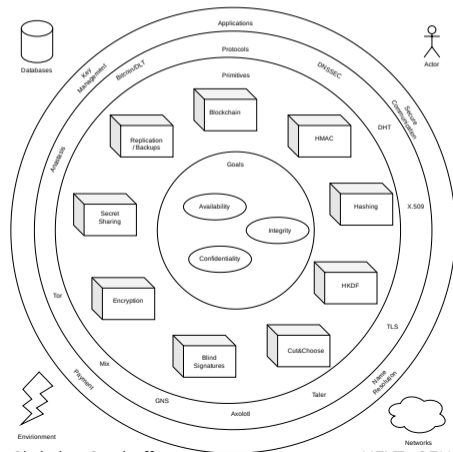
Distributed systems on our map

2024-08-26

NEXT GENERATION INTERNET



└ Distributed systems on our map



1. Secure name resolution, payment, key management and communication are all applications in the domain of distributed systems. Without a network, computer security would be rather boring.
2. Key considerations are availability (replication, backups, environmental conditions, networks) and integrity (database theory, consistency, atomicity).

What are typical attacks on decentralized systems?

Background:

- ▶ Ancient Greece: Sybils were prophetesses that prophesied under the divine influence of a deity. Note: At the time of prophecy not the person but a god was speaking through the lips of the Sybil.
- ▶ 1973: Flora Rheta Schreiber published a book "Sybil" about a woman with 16 separate personalities.

1. Just some background on the origin of the name "Sybil".

The Sybil attack



The Sybil attack [6]:

- ▶ Insert a node multiple times into a network, each time with a different identity
- ▶ Position a node for next step on attack:
 - ▶ Attack connectivity of the network
 - ▶ Attack replica set
 - ▶ In case of majority votes, be the majority!

2024-08-26

NEXT GENERATION INTERNET

└ What are typical attacks on decentralized systems?

└ The Sybil attack



The Sybil attack [6]:

- ▶ Insert a node multiple times into a network, each time with a different identity
- ▶ Position a node for next step on attack:
 - ▶ Attack connectivity of the network
 - ▶ Attack replica set
 - ▶ In case of majority votes, be the majority!

1. Sybil attacks also happen on online social networks; there they are usually associated with “sock puppet accounts” (pseudonymous accounts or false identities).
2. Open network protocols that want a “diversity” of operators (common in file-sharing, mix networks, onion routing) are often vulnerable.

Defenses against Sybil Attacks

- ▶ Use authentication with trusted party that limits identity creation
- ▶ Use “external” identities (IP address, MAC, e-mail)
- ▶ Use “expensive” identities (solve computational puzzles, require payment)

Douceur [6]: Without trusted authority to certify identities, no realistic approach exists to completely stop the Sybil attack.

2024-08-26

NEXT GENERATION INTERNET

└ What are typical attacks on decentralized systems?

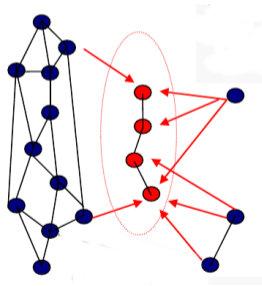
└ Defenses against Sybil Attacks

- ▶ Use authentication with trusted party that limits identity creation
 - ▶ Use “external” identities (IP address, MAC, e-mail)
 - ▶ Use “expensive” identities (solve computational puzzles, require payment)
- Douceur [6]: Without trusted authority to certify identities, no realistic approach exists to completely stop the Sybil attack.

1. Trusted third parties that limit the ability to join the network are an issue for open networks, especially file-sharing and anonymizing networks where also the operators may like some privacy for themselves.
2. External proxy identities are often surprisingly cheap, some universities have tens of thousands of unused IPv4 addresses, and anyone with an SMTP server has basically an unlimited number of e-mail addresses.
3. An adversary might be willing to spend many times the resources on computational puzzles or payments than normal honest users.

Eclipse attack: Goal

- ▶ Separate a node or group of nodes from the rest of the network
- ▶ isolate peers (DoS, surveillance) or isolate data (censorship)



2024-08-26

NEXT GENERATION INTERNET

└ What are typical attacks on decentralized systems?

└ Eclipse attack: Goal

- ▶ Separate a node or group of nodes from the rest of the network
- ▶ isolate peers (DoS, surveillance) or isolate data (censorship)



1. Isolation of individuals defeats anonymization networks, and isolation of data enables censorship.

Eclipse Attack: Techniques

- ▶ Use Sybil attack to increase number of malicious nodes
- ▶ Take over routing tables, peer discovery
- ⇒ Details depend on overlay structure

2024-08-26

NEXT GENERATION INTERNET

└─ What are typical attacks on decentralized systems?

└─ Eclipse Attack: Techniques

- ▶ Use Sybil attack to increase number of malicious nodes
- ▶ Take over routing tables, peer discovery
- ⇒ Details depend on overlay structure

1. Most Eclipse attacks use some form of Sybil attack to increase the number of malicious points of presence in the network.
2. Eclipse attacks primarily depend on the routing table structure and maintenance procedures. Routing protocol vulnerabilities (see: BGP hijacking) can also help the attacker.

Eclipse Attack: Defenses

- ▶ Large number of connections
- ▶ Aggressive discovery (“continuous” bootstrap)
- ▶ Prefer long-lived connections / old peers
- ▶ Replication
- ▶ Diverse neighbor selection (different IP subnets, geographic locations)
- ▶ Audit neighbor behavior (if possible)

2024-08-26

NEXT GENERATION INTERNET

└ What are typical attacks on decentralized systems?

└ Eclipse Attack: Defenses

- ▶ Large number of connections
- ▶ Aggressive discovery (“continuous” bootstrap)
- ▶ Prefer long-lived connections / old peers
- ▶ Replication
- ▶ Diverse neighbor selection (different IP subnets, geographic locations)
- ▶ Audit neighbor behavior (if possible)

1. The more sparse the routing tables, the easier they usually are to attack.
2. Constantly searching for new peers to connect to can help stay connected despite churn.
3. By preferring long-lived connections, attackers need to be present in the network for a long time, which costs them resources and may help detect them before they can strike.
4. Creating many replicas of data at different points in the network makes it harder to block all paths.
5. Some attackers may have many IP addresses, but possibly from few subnets or geographic locations.
6. If the protocol allows honest nodes to detect malicious activity and attribute it precisely, it may prevent dishonest nodes from continuing to participate.

Poisoning Attacks

Nodes provide false information:

- ▶ wrong routing tables [8]
- ▶ wrong meta data
- ▶ wrong performance measurements

2024-08-26

NEXT GENERATION INTERNET

└ What are typical attacks on decentralized systems?

└ Poisoning Attacks

Nodes provide false information:

- ▶ wrong routing tables [8]
- ▶ wrong meta data
- ▶ wrong performance measurements

1. These attacks apply to protocols that are adaptive. For example, Tor and other mix networks may bias node selection towards nodes that perform (or claim to perform) well. Similarly, file-sharing networks often want to incentivize nodes that contribute resources.
2. The PitchBlack [8] attack on Freenet poisoned the routing tables, causing routing performance to collapse.

Timing Attacks [11]

Nodes can:

- ▶ measure latency to determine origin of data
- ▶ delay messages
- ▶ send messages using particular timing patterns to aid correlation
- ▶ include wrong timestamps (or just have the wrong time set...)

2024-08-26

NEXT GENERATION INTERNET

└─What are typical attacks on decentralized systems?

└─Timing Attacks [11]

Nodes can:
▶ measure latency to determine origin of data
▶ delay messages
▶ send messages using particular timing patterns to aid correlation
▶ include wrong timestamps (or just have the wrong time set...)

1. Timing attacks are a major threat for anonymizing networks, especially those that try to anonymize users while offering low latency.
2. While we mostly think of timing attacks as just measuring time, attackers can also delay messages or modulate transmissions to have particular characteristics.
3. Various protocols (starting with the timestamp option in TCP!) include timestamps that attackers could manipulate.
4. I am not aware of any general solution to these types of attacks; take timing attacks into due consideration when threat modelling and develop system-specific solutions if necessary.

What should we think about when building distributed systems?

The 8 Fallacies of Distributed Computing¹

1. The network is reliable
2. Latency is zero
3. Bandwidth is infinite
4. The network is secure
5. Topology does not change
6. There is one administrator
7. Transport cost is zero
8. The network is homogeneous

¹According to Peter Deutsch and James Gosling

└─What should we think about when building distributed systems?

└─The 8 Fallacies of Distributed Computing^a

1. The network is reliable
 2. Latency is zero
 3. Bandwidth is infinite
 4. The network is secure
 5. Topology does not change
 6. There is one administrator
 7. Transport cost is zero
 8. The network is homogeneous
- ^aAccording to Peter Deutsch and James Gosling

1. These are **fallacies**, which means they are both obviously false assumptions that are still frequently made.
2. Often these assumptions are not made **explicitly**, the more common issue is that a system design does not adequately handle situations where the assumption does not hold.

Self stabilization (Dijkstra 1974) [4]

- ▶ A system is self-stabilizing, if starting from any state, it is guaranteed that the system will eventually reach a correct state (convergence).
- ▶ Given that the system is in a correct state, it is guaranteed to stay in a correct state, provided that no fault happens (closure).
- ▶ Self-stabilization enables a distributed algorithm to recover from a transient fault **regardless of its nature**.

Example: Spanning-tree Protocol from Networking!

2024-08-26

NEXT GENERATION INTERNET

- └ What should we think about when building distributed systems?
 - └ Self stabilization (Dijkstra 1974) [4]

- ▶ A system is self-stabilizing, if starting from any state, it is guaranteed that the system will eventually reach a correct state (convergence).
- ▶ Given that the system is in a correct state, it is guaranteed to stay in a correct state, provided that no fault happens (closure).
- ▶ Self-stabilization enables a distributed algorithm to recover from a transient fault **regardless of its nature**.

Example: Spanning-tree Protocol from Networking!

1. Self stabilization is something to aspire to. This is how truly robust systems are built.
2. This kind of thinking went into Voyager 1&2 (the spacecraft), but also applies to any other highly reliable system.
3. Heartbeat mechanisms are a common technique to recover from faults.

What are impossibly hard problems in distributed systems security?

Ryge's Triangle

Ryge's Triangle postulates three key management goals for a system associating cryptographic keys with addresses or names:

- ▶ Non-interactive: the system should require no user interface
- ▶ Flexible: addresses/names can be re-used by other participants
- ▶ Secure: the system is secure against active attackers

Ryge's triangle says that one can only have two of the three.

2024-08-26

NEXT GENERATION INTERNET

- └ What are impossibly hard problems in distributed systems security?
 - └ Ryge's Triangle

Ryge's Triangle postulates three key management goals for a system associating cryptographic keys with addresses or names:

- ▶ Non-interactive: the system should require no user interface
- ▶ Flexible: addresses/names can be re-used by other participants
- ▶ Secure: the system is secure against active attackers

Ryge's triangle says that one can only have two of the three.

1. Ryge's Triangle was postulated in the context of Vula, an mDNS-based key management protocol for WireGuard VPNs. Vula has different modes, effectively allowing the user to pick their edge of the triangle.
2. TOFU can be used to allow non-interactive and secure communication, but it is not flexible.
3. Manual verification can be used to be secure and flexible, but requires user interaction.
4. Anything goes is non-interactive and flexible, but not at all secure against active attacks.

Anonymity Trilemma

The Anonymity Trilemma [3] states that given the objectives of:

- ▶ Strong anonymity
- ▶ Low bandwidth overhead
- ▶ Low latency

... one can only have two of the three.

2024-08-26

NEXT GENERATION INTERNET

└─What are impossibly hard problems in distributed systems security?

└─Anonymity Trilemma

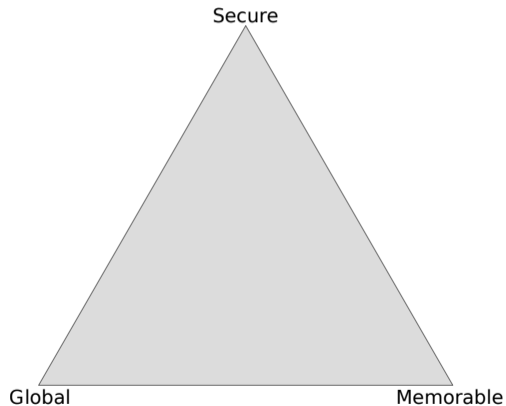
The Anonymity Trilemma [3] states that given the objectives of:

- ▶ Strong anonymity
- ▶ Low bandwidth overhead
- ▶ Low latency

... one can only have two of the three.

1. This trilemma is really bad news for privacy, as strong anonymity **also** requires a large user-base, and without low latency and low resource usage, it is difficult for any anonymity system to gain a large user-base!
2. Design examples include Piplinet [2] (low latency, strong anonymity, high bandwidth overhead), Tor [5] (low latency, low bandwidth overhead, weak anonymity), and Mix networks (high latency, low bandwidth overhead, strong anonymity)

Zooko's Triangle



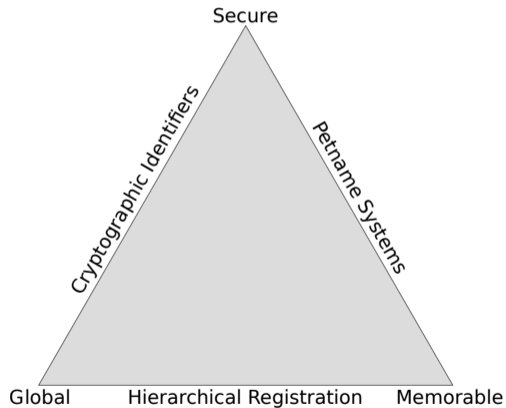
A name system can only fulfill **two!**

- └ What are impossibly hard problems in distributed systems security?
 - └ Zooko's Triangle



1. This triangle was proposed in the context of name resolution, that is securely mapping human-readable names to some authoritative value.
2. Secure here considers a very strong adversary, one that could take over **any role** of the system (including the registry/CA) and has more (computational) resources than all other participants combined.
3. Memorable means that names can be **enumerated by brute-force**, that is the attacker can just try all possible values.
4. Global means that all users get the **same result** for the same name.

Zooko's Triangle



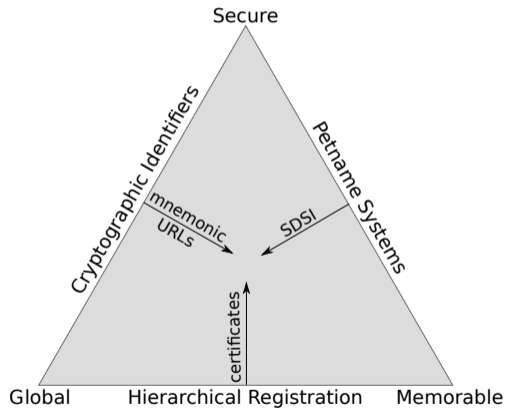
DNS, ".onion" IDs and /etc/hosts/ are representative designs.

- └ What are impossibly hard problems in distributed systems security?
 - └ Zooko's Triangle



1. In DNS, names are global and human-readable.
2. In ".onion", the cryptographic identifier ensures that only the authority can publish introduction points, but the names are not memorable.
3. In /etc/hosts names are secure and memorable, but not global.

Zooko's Triangle



2024-08-26

NEXT GENERATION INTERNET

- What are impossibly hard problems in distributed systems security?
 - Zooko's Triangle



- In DNSSEC, security is still limited (adversary model: what if the registry is the attacker)
- Mnemonics try to give humans nicer representations that make it easier to remember cryptographic material.
- Petname systems like SDSI [15] make local certifications usable for other users, but do not achieve global names.

Limits on authentication

Theorem (Boyd's Theorem I)

"Suppose that a user has either a confidentiality channel to her, or an authentication channel from her, at some state of the system. Then in the previous state of the system such a channel must also exist. By an inductive argument, such a channel exists at all previous states."

Theorem (Boyd's Theorem II)

"Secure communication between any two users may be established by a sequence of secure key transfers if there is a trusted chain from each one to the other."

2024-08-26

NEXT GENERATION INTERNET

- └ What are impossibly hard problems in distributed systems security?
 - └ Limits on authentication

Theorem (Boyd's Theorem I)

"Suppose that a user has either a confidentiality channel to her, or an authentication channel from her, at some state of the system. Then in the previous state of the system such a channel must also exist. By an inductive argument, such a channel exists at all previous states."

Theorem (Boyd's Theorem II)

"Secure communication between any two users may be established by a sequence of secure key transfers if there is a trusted chain from each one to the other."

1. In practice you *hope* that you had a secure channel when you obtained your operating system, and that the operating system had a secure channel to the CAB forum, and the CAB forum had a secure channel to the CAs.

Solution space: Zfone Authentication (ZRTP) [12]

Idea: combine human interaction proof and baby duck approach:

- ▶ A and B perform Diffie-Hellman exchange
- ▶ Keying material from previous sessions is used (duckling)
- ▶ Short Authentication String (SAS) is generated (hash of DH numbers)
- ▶ Both users read the SAS to each other, recognize voice

⇒ ZRTP foils standard man-in-the-middle attack.

2024-08-26

NEXT GENERATION INTERNET

└─What are impossibly hard problems in distributed systems security?

└─Solution space: Zfone Authentication (ZRTP) [12]

Idea: combine human interaction proof and baby duck approach:
▶ A and B perform Diffie-Hellman exchange
▶ Keying material from previous sessions is used (duckling)
▶ Short Authentication String (SAS) is generated (hash of DH numbers)
▶ Both users read the SAS to each other, recognize voice
⇒ ZRTP foils standard man-in-the-middle attack.

1. The duckling basically allows us to establish the security of *past* communications in the future.
2. Zfone does not technically violate Boyd's theorem, as users must have learned each other's voice-print over a secure channel previously.
3. This is a very good example for hacking around an impossibility theorem!

Blockchain Trilemma

Blockchains try to achieve three properties:

- ▶ Decentralization: there are many participants, and each participant only needs to have a small amount of resources, say $O(c)$
- ▶ Scalability: the system scales to $O(n) > O(c)$ transactions
- ▶ Security: the system is secure against attackers with $O(n)$ resources

The Blockchain trilemma [13] is that one can only have two of the three.

2024-08-26

NEXT GENERATION INTERNET

└─What are impossibly hard problems in distributed systems security?

└─Blockchain Trilemma

Blockchains try to achieve three properties:

- ▶ Decentralization: there are many participants, and each participant only needs to have a small amount of resources, say $O(c)$
- ▶ Scalability: the system scales to $O(n) > O(c)$ transactions
- ▶ Security: the system is secure against attackers with $O(n)$ resources

The Blockchain trilemma [13] is that one can only have two of the three.

1. Naturally, anyone trying to sell you some shitcoin is going to claim that their shitcoin satisfies all three.
2. In practice, they will have made some trade-off which could be more-or-less interesting.
3. Polkadot [1] has an interesting approach where they achieve scalability by offloading transactions onto side-chains (sharding) and security when the side-chains are merged back into the main chain.

The CAP theorem [9]

No distributed system can be *consistent, available and partition tolerant* at the same time.

- ▶ Consistency: A *read* sees the changes made by all previous *writes*
- ▶ Availability: *Reads* and *writes* always succeed
- ▶ Partition tolerance: The system operates even when network connectivity between components is broken

2024-08-26

NEXT GENERATION INTERNET

- └ What are impossibly hard problems in distributed systems security?
 - └ The CAP theorem [9]

No distributed system can be consistent, available and partition tolerant at the same time.

- ▶ Consistency: A read sees the changes made by all previous writes
- ▶ Availability: Reads and writes always succeed
- ▶ Partition tolerance: The system operates even when network connectivity between components is broken

1. The CAP theorem is one of the most important results in distributed systems. Postulated in 1998, a formal proof was published in 2002.
2. Applied to payment systems, we can read consistency as agreement on who owns how much money, availability as the ability to perform payments, and partition tolerance being about some groups of participants being isolated (offline) from the rest of the system. Naturally, if participants that cannot talk to the rest of the network do make payments, we cannot all agree on who owns how much as some transactions are not visible to some members.
3. Eventual consistency is an approach to ensure that the system will converge to a coherent transaction set after the partition heals (that is, users are again online). Note that eventual consistency does not require that there is a time where everyone is again online, the connected subsets can be completely fluent! Conflict-free replicated data types [14] are a popular principled approach for ensuring eventual consistency.

“EZB: Digitaler Euro benötigt Secure Element des iPhones” –Heise.

“Ohne Lösung des Single-Spending-Problems wird das nichts”:

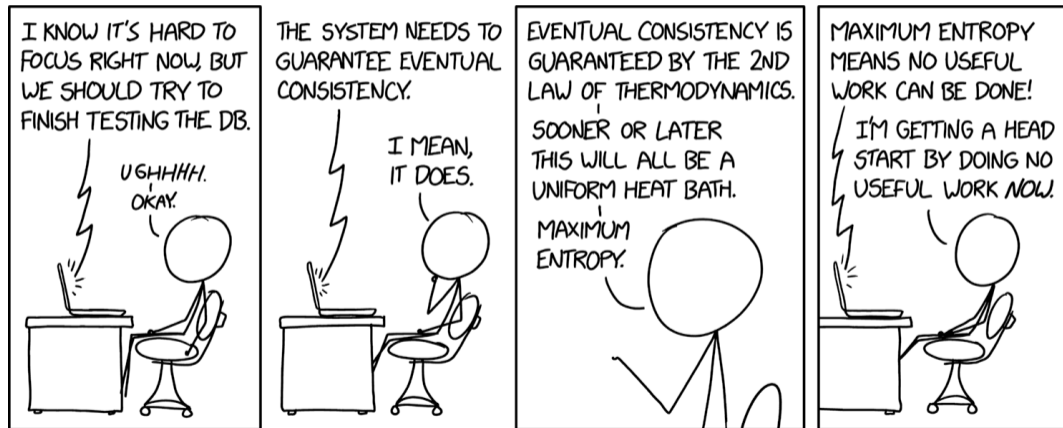
“Im klassischen Geldverkehr gibt es ja das Problem, dass man jeden Euro nur einmal ausgeben kann. Das ist einer der großen Nachteile von Bargeld. Eine digitale Wahrung hatte den groen Vorteil, dass man jeden Euro mehrfach ausgeben konnte, und somit mehr Geld erschaffen konnte. Das ware der groe Vorteil gegenuber Bargeld und Kartenzahlungen. So lange das nicht geht, ist es im Prinzip wie DRM, ein Versuch, mit viel Aufwand, irgendwie die Zeit zuruck zu drehen um alte Geschaftsmodelle 1:1 am Leben zu erhalten.” –Casandro

└ What are impossibly hard problems in distributed systems security?

└ Heise Forum: Posting 43921013

“EZB: Digitaler Euro benotigt Secure Element des iPhones” –Heise.
“Ohne Losung des Single-Spending-Problems wird das nichts”:
“Im klassischen Geldverkehr gibt es ja das Problem, dass man jeden Euro nur einmal ausgeben kann. Das ist einer der groen Nachteile von Bargeld. Eine digitale Wahrung hatte den groen Vorteil, dass man jeden Euro mehrfach ausgeben konnte, und somit mehr Geld erschaffen konnte. Das ware der groe Vorteil gegenuber Bargeld und Kartenzahlungen. So lange das nicht geht, ist es im Prinzip wie DRM, ein Versuch, mit viel Aufwand, irgendwie die Zeit zuruck zu drehen um alte Geschaftsmodelle 1:1 am Leben zu erhalten.” –Casandro

1. The poster is NOT insane, the posting is sarcastic.
2. Translation: “In traditional monetary transactions there is the problem that you can only spend each euro once. This is one of the big disadvantages of cash. A digital currency would have the great advantage that you could spend each euro multiple times, thus creating more money. That would be the big advantage over cash and card payments. As long as that’s not possible, it’s basically like DRM, an attempt, with a lot of effort, to somehow turn back time in order to keep old business models alive 1:1.”



What are impossibly hard problems in distributed systems security?

<https://xkcd.com/2315/>



How can we work around such hard issues?

Digital offline payments ...

... are incompatible with the CAP theorem

- ▶ Offline capabilities are often cited as a requirement for digital payments by central banks
- ▶ All implementations must either use restrictive hardware elements and/or introduce counter-party risk.
- ⇒ Permanent offline features weaken a digital payment solution (privacy, security)
- ⇒ Introduces unwarranted competition for physical cash (endangers emergency-preparedness).

2024-08-26

NEXT GENERATION INTERNET

└ How can we work around such hard issues?

└ Digital offline payments ...

1. Without privacy, you can go after double-spenders and **try** to get the money back.
2. We recommend pure online-only, bearer-based digital payments.
3. For offline payments, central banks should stick to physical cash. Physical cash is also best for emergencies (power outage, catastrophic cyber incidents).

- ▶ Offline capabilities are often cited as a requirement for digital payments by central banks
- ▶ All implementations must either use restrictive hardware elements and/or introduce counter-party risk.
- ⇒ Permanent offline features weaken a digital payment solution (privacy, security)
- ⇒ Introduces unwarranted competition for physical cash (endangers emergency-preparedness).

The scenario

God is offline, but customer pays online



2024-08-26

NEXT GENERATION INTERNET

└ How can we work around such hard issues?

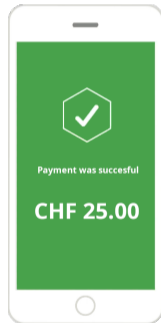
└ The scenario



1. We will not consider a common situation for **offline** payments that we actually can do securely.
2. This is a classical case of changing impossible requirements to work around the CAP theorem.
3. The **modified** offline scenario is simple: the merchant is offline, but the customer **is** online.
4. Now, in the case of a worshipper donating to god, we already have no problem as clearly the worshipper will be **honest**.

1. We have the same scenario in commercial applications where the customer might not be so honest.
2. Here, many major providers (Twint, PayPal, AliPay, PayTM) all use the same basic method.
3. The customer scans some QR code that identifies the payee, enters the amount, confirms the payment, and the payment service provider checks the confirmation screen on the device of the user.
4. In the background, the payment service provider wires the funds to the merchant's bank account.

Secure payment ... Everything green?



2024-08-26

NEXT GENERATION INTERNET

└ How can we work around such hard issues?

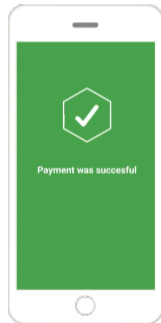
└ Secure payment ...



1. Sadly, the security of this approach is basically in this wonderful screen, which is unfortunately on a device under the full control of the customer.

Exploit “Code”

Programming optional



2024-08-26

NEXT GENERATION INTERNET

└ How can we work around such hard issues?

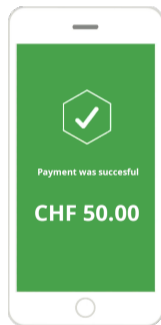
└ Exploit “Code”



1. As a result, for anyone who can take and edit a screenshot ...

“Customers” love Twint ...

Daily non-business for merchants



2024-08-26

NEXT GENERATION INTERNET

└ How can we work around such hard issues?

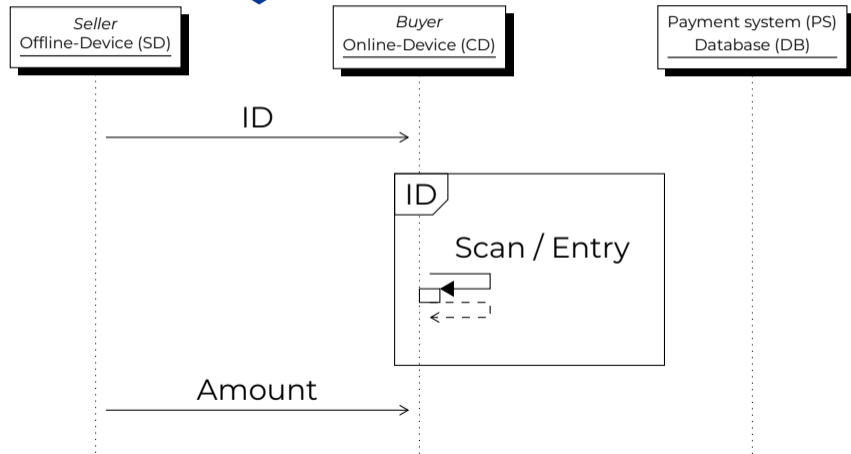
└ “Customers” love Twint ...



1. ... payment is optional.
2. Rumors are that particularly vulnerable merchants (those with many one-off customers, say at events, festivals, etc.) get paid about 80% of the time...

... more secure with GNU Taler! (1/4)

First steps are identical ...

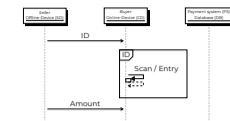


2024-08-26

NEXT GENERATION INTERNET

└ How can we work around such hard issues?

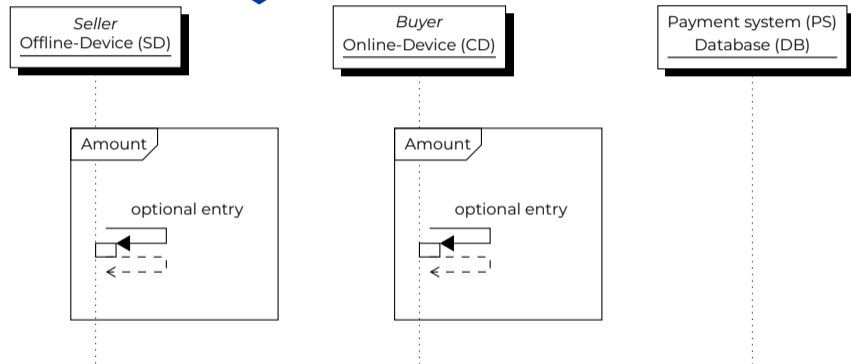
└ ... more secure with GNU Taler! (1/4)



1. We will now present a practical approach [10] to improve security in the above scenario.
2. A key constraint is that we do not want to require the seller to have Internet connectivity.
3. Furthermore, we do not require them to have an expensive and fragile smart phone — a simple € 10 off-line device will be enough! The approach is not specific to GNU Taler, but was first implemented there.

... more secure with GNU Taler! (2/4)

Optional: enter variable amount

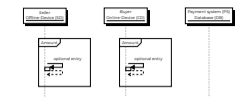


2024-08-26

NEXT GENERATION INTERNET

└ How can we work around such hard issues?

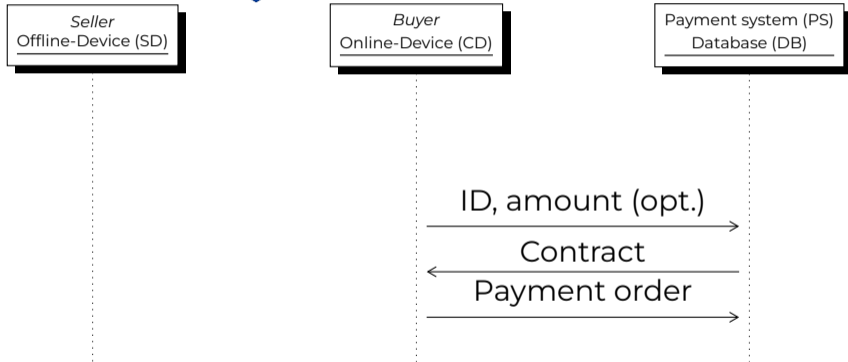
└ ... more secure with GNU Taler! (2/4)



1. As with canonical systems, the customer has to enter the amount they want to pay.
2. This is optional, as the amount might be fixed and in this case the data entry could be skipped.
3. Note that the **seller** also may need to enter the amount to be paid on their offline device if they want to be sure that the buyer paid the correct amount.

... more secure with GNU Taler! (3/4)

The customer still has to pay

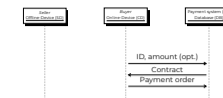


2024-08-26

NEXT GENERATION INTERNET

└ How can we work around such hard issues?

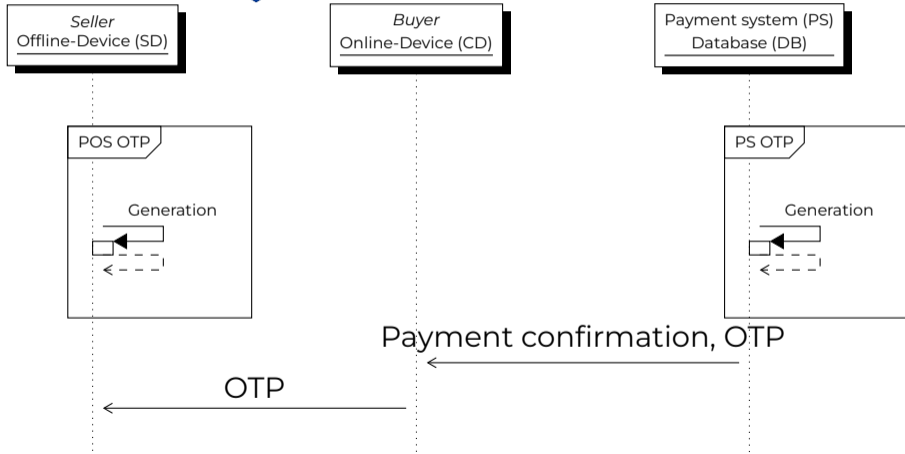
└ ... more secure with GNU Taler! (3/4)



1. This is just the canonical payment process where the customer contacts the (trusted) payment system, receives payment instructions and confirms the payment.

... more secure with GNU Taler! (4/4)

New: OTP in reverse direction!

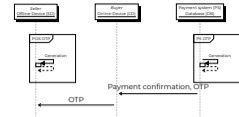


2024-08-26

NEXT GENERATION INTERNET

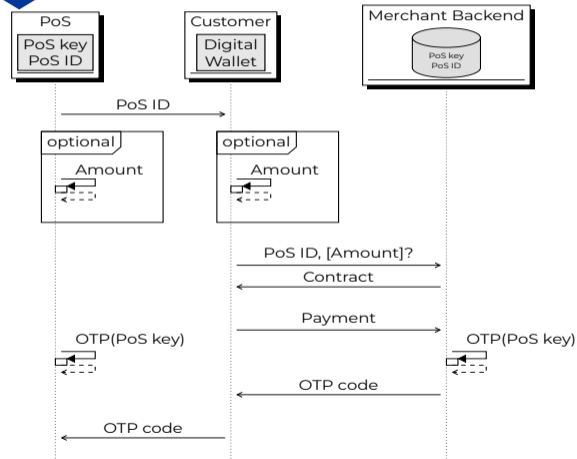
How can we work around such hard issues?

... more secure with GNU Taler! (4/4)



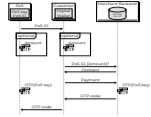
1. This is the special step. Instead of merely returning a payment confirmation, the payment system uses a **secret** that it **shares** with the offline device of the seller to compute a one-time-password.
2. If a variable amount is involved and the OTP generator of the merchant supports it, the amount itself would also be included in the hash function used to compute the OTP code.
3. The OTP code is sent to the untrusted device of the buyer. A malicious buyer cannot generate such a code themselves because they do not know the shared secret. They cannot re-use it because it is a one-time-password and thus only valid once!
4. The seller can check that the OTP code presented by the buyer matches their OTP generator. The OTP code is equivalent to a compact designated verifier signature by the payment system for the offline point-of-sale affirming that a certain amount was paid.

Summary: Partially Offline Payments



How can we work around such hard issues?

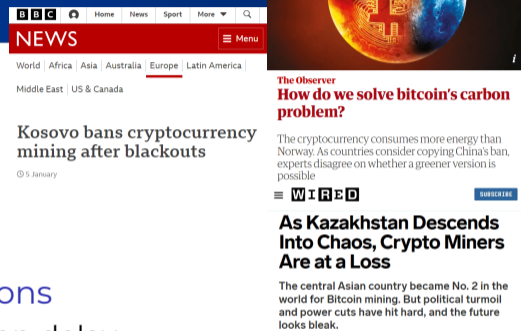
Summary: Partially Offline Payments



1. The CAP theorem is not violated, because the point-of-sale basically communicates via the customer; the trick is that the communication is secure despite the customer's device not being trusted.
2. The same effect could have been achieved with an ordinary cryptographic signature from the merchant backend and the point-of-sale knowing the respective public key.
3. However, the signature would have been much larger (basically requiring the customer to show a QR code or offer an NFC tag) than the short OTP code. Furthermore, the offline PoS device would have to be much beefier to scan a QR code and perform public key cryptography. So the OTP solution is both **cheaper** and more **usable**.

Bonus: Depolymerization [7]

Blockchain based cryptocurrencies



Biggest cryptocurrencies

- ▶ **BTC** Bitcoin
- ▶ **ETH** Ethereum

Common blockchain limitations

- ▶ **Delay** block and confirmation delay
- ▶ **Cost** transaction fees
- ▶ **Scalability** limited amount of transaction per second
- ▶ **Ecological impact** computation redundancy
- ▶ **Privacy & regulatory compliance**

2024-08-26

NEXT GENERATION INTERNET

└ Bonus: Depolymerization [7]

└ Blockchain based cryptocurrencies

1. The blockchain trilemma is not the only dilemma for cryptocurrencies.
2. We will now look at some practical approaches for addressing or lessening some of these issues.



Related work

Centralization - Coinbase off-chain sending

- + Fast and cheap: off chain transaction
- Trust in Coinbase: privacy, security & transparency

Layering - Lightning Network

- + Fast and cheap: off-chain transactions
- Requires setting up bidirectional payment channels
- Fraud attempts are mitigated via a complex penalty system

2024-08-26

NEXT GENERATION INTERNET
└ Bonus: Depolymerization [7]

└ Related work

Centralization - Coinbase off-chain sending

- + Fast and cheap: off chain transaction
- Trust in Coinbase: privacy, security & transparency

Layering - Lightning Network

- + Fast and cheap: off-chain transactions
- Requires setting up bidirectional payment channels
- Fraud attempts are mitigated via a complex penalty system

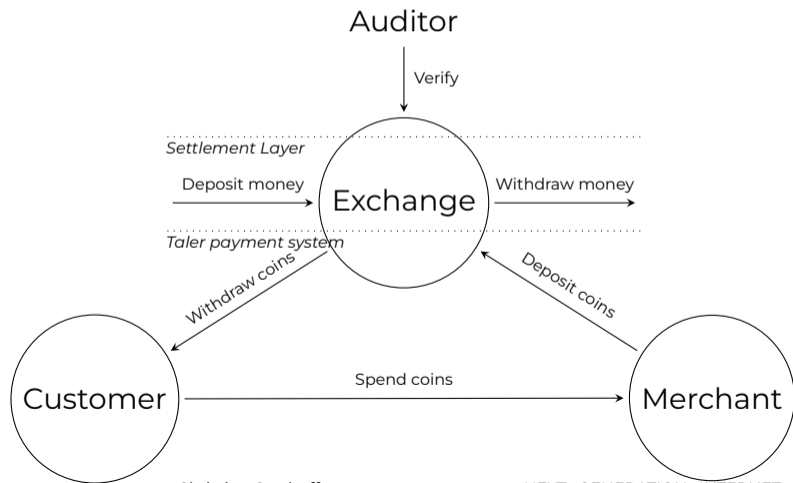
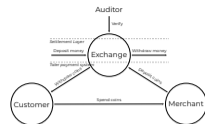
1. One solution is to trade crypto-currencies off-chain. This involves creating an account at a “trusted” intermediary such as FTX and hoping that they do not embezzle the funds. Providers pick jurisdictions favorable to them.
2. In light of the blockchain trilemma, this is the solution that basically **only** offers performance and no cryptographic security or decentralization.
3. The Lightning network creates a layer-2 network over Bitcoin where pairs of nodes can perform fast off-chain transactions over payment channels. Opening a payment channel requires locking up funds, and payments are limited to the amount locked up by both parties. When the channel is closed, the final delta between the accounts is transferred on-chain.
4. The main issue is the requirement to lock up funds (limiting the availability of effective payment channels), and as a result lightning is fast and cheap if it works, but for some payments it may simply not work at all because no route with adequate capacity exists between payer and payee!

Taler Architecture

2024-08-26

NEXT GENERATION INTERNET
└ Bonus: Depolymerization [7]

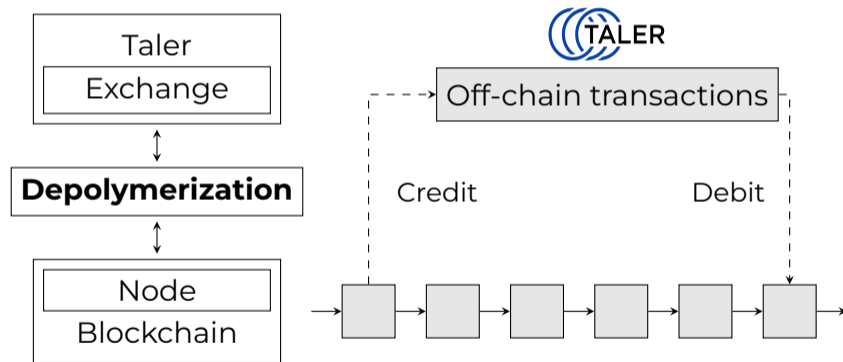
└ Taler



1. This is a review of the Taler architecture.
2. Key for Taler is the **existence** of a settlement layer. Usually, this is some existing wholesale payment system, such as SEPA, UPC or SWIFT. This is the “core banking system” of the respective fiat currency used by banks to make transactions.
3. But, Taler is not limited to traditional core banking systems and fiat currencies!

Project Depolymerization

Taler with blockchain settlement layer

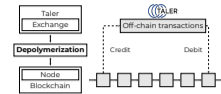


2024-08-26

NEXT GENERATION INTERNET

└ Bonus: Depolymerization [7]

└ Project Depolymerization



1. A polymer, commonly called plastic, is a long-chained molecule created by the process of polymerization. For example, Poly-Ethylen (PE) is created from many Ethylen molecules.
2. The process of **depolymerization** is when we recycle plastic trash into monomers, usually by applying heat.
3. Project depolymerization turns long chains of blocks into “unlinked” digital coins useful for high-speed transactions. The digital coins can eventually be put back onto the blockchain.
4. As always, GNU Taler assumes that the operator of the Taler exchange is trustworthy because they are easily identified and must be regulated and independently audited.

Depolymerization [7]

Architecture



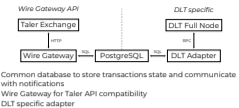
- ▶ Common database to store transactions state and communicate with notifications
- ▶ Wire Gateway for Taler API compatibility
- ▶ DLT specific adapter

2024-08-26

NEXT GENERATION INTERNET

└ Bonus: Depolymerization [7]

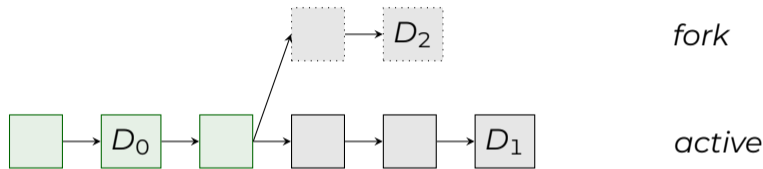
└ Depolymerization [7]



1. This is a high-level overview of the Depolymerization software architecture.
2. We use a generic wire gateway to implement the GNU Taler wire gateway REST interface, a generic database to store transaction data, and then a blockchain-specific adapter that talks to a full node over some blockchain-specific RPC API.
3. This is about 12k LOC in Rust for both Bitcoin and Ethereum, with the blockchain-specific code being around 2k LOC for each blockchain.
4. We will now take a brief look at key challenges involved in implementing this.

CAP & Bitcoin

Chain reorganization



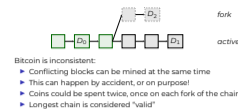
Bitcoin is inconsistent:

- ▶ Conflicting blocks can be mined at the same time
- ▶ This can happen by accident, or on purpose!
- ▶ Coins could be spent twice, once on each fork of the chain!
- ▶ Longest chain is considered "valid"

2024-08-26

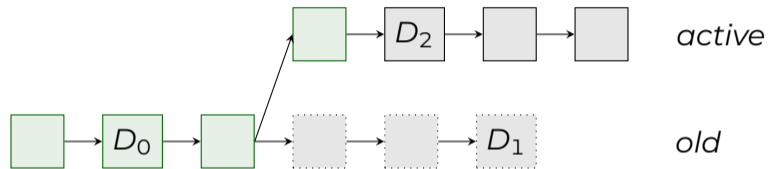
NEXT GENERATION INTERNET
└ Bonus: Depolymerization [7]

└ CAP & Bitcoin



1. A fork is when concurrent blockchain states coexist. Nodes will follow the longest chain, replacing recent blocks if necessary during a blockchain reorganization.
2. Basically, this raises the question as to when Depolymerizer can be sure that an inbound transaction is actually final. Original Bitcoin paper suggests to consider transaction confirmed only after at least 6 blocks past the transaction, but competitively long alternative chains could void durability even after 6 blocks!
3. Once Depolymerizer issues blindly signed anonymous digital coins there is no good way to "undo" this, so we need to be sure that the money did arrive in our escrow account: If a deposit transaction disappears from the blockchain, coins created from **final** Taler withdraw transactions might no longer be backed by credit!

Handling blockchain reorganization



- ▶ As small reorganizations are common, Satoshi already recommended to apply a confirmation delay to handle most disturbances and attacks.
- ▶ If a reorganization longer than the confirmation delay happens, but it did not remove credits, Depolymerizer is safe and keeps running.

2024-08-26

NEXT GENERATION INTERNET

└ Bonus: Depolymerization [7]

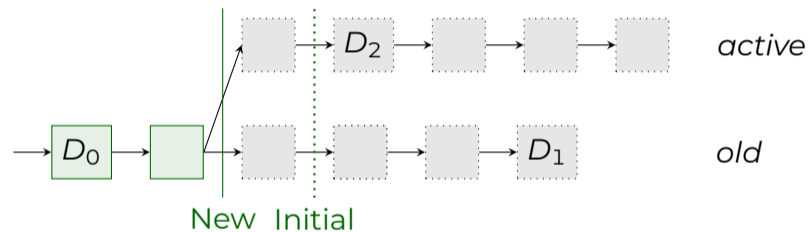
└ Handling blockchain reorganization



- ▶ As small reorganizations are common, Satoshi already recommended to apply a confirmation delay to handle most disturbances and attacks.
- ▶ If a reorganization longer than the confirmation delay happens, but it did not remove credits, Depolymerizer is safe and keeps running.

1. Thus, first of all, Depolymerizer also waits for ≥ 6 blocks before considering an inbound Bitcoin transaction to be “final”.
2. Second, if a fork is detected **despite** waiting 6 blocks, we check if this actually affected our balance. Not all forks will be **relevant**.

Adaptive confirmation

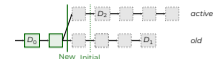


2024-08-26

NEXT GENERATION INTERNET

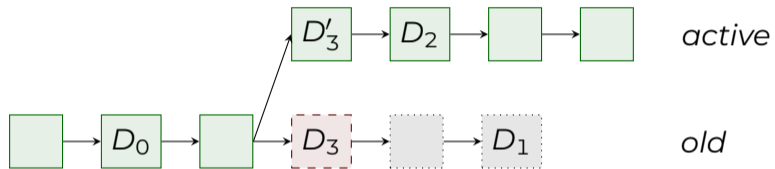
└ Bonus: Depolymerization [7]

└ Adaptive confirmation



1. Nevertheless, if we experience any such reorganization once, its dangerously likely for another one of a similar scope to happen again. Depolymerizer learns from reorganizations by increasing its confirmation delay.
2. In the previous slides, we used a confirmation delay of 3 blocks, but after experiencing a successful fork after 4 blocks, the confirmation delay would be increased to 5 blocks.
3. Of course in practice the minimum would be ≥ 6 blocks.

Handling blockchain reorganization

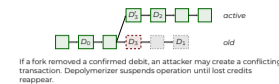


If a fork removed a confirmed debit, an attacker may create a conflicting transaction. Depolymerizer suspends operation until lost credits reappear.

2024-08-26

NEXT GENERATION INTERNET
└ Bonus: Depolymerization [7]

└ Handling blockchain reorganization



1. Now, imagine we had a confirmation delay of only two blocks, and originally accepted D_3 when the “old” chain was the longest chain. When the “fork” became the new “active” (longest) chain an incoming transfer to the Depolymerizer might disappear.
2. If Depolymerizer issued coins for a blockchain transaction that then “disappeared”, Depolymerizer stop all processing until either the administrator manually intervenes or the transaction appears again on-chain (as it *should* still be in the pool of transactions yet to be mined, unless there is now a hard conflict).

Challenges

Taler Metadata

- ▶ Metadata are required to link a wallet to credits and allow merchant to link deposits to debits
- ▶ Putting metadata in blockchain transactions can be tricky

2024-08-26

NEXT GENERATION INTERNET
└ Bonus: Depolymerization [7]

└ Challenges

Taler Metadata

- ▶ Metadata are required to link a wallet to credits and allow merchant to link deposits to debits
- ▶ Putting metadata in blockchain transactions can be tricky

1. To determine the wallet eligible to withdraw funds, GNU Taler requires a wallet-specific **reserve public key** to be encoded in the wire transfer subject.
2. However, most blockchains lack the ability to encode such meta-data nicely on-chain.

Storing metadata

Bitcoin

Bitcoin - Credit

- ▶ Transactions from code
- ▶ Only 32B + URI
- ▶ **OP_RETURN**

Bitcoin - Debit

- ▶ Transactions from common wallet software
- ▶ Only 32B
- ▶ **Fake Segwit Addresses**

2024-08-26

NEXT GENERATION INTERNET

└ Bonus: Depolymerization [7]

└ Storing metadata

Bitcoin - Credit

- ▶ Transactions from code
- ▶ Only 32B + URI
- ▶ **OP_RETURN**

Bitcoin - Debit

- ▶ Transactions from common wallet software
- ▶ Only 32B
- ▶ **Fake Segwit Addresses**

1. When doing outgoing wire transfers, Depolymerizer can encode meta-data using **OP-RETURN**.
2. However, users cannot generate such transactions with contemporary (Blockchain) wallets.
3. Thus, for **incoming** transactions, we use fake Segwit addresses. Basically, the transaction must contain three outputs, the amount to be withdrawn must be sent to the Deploymerizer's Bitcoin wallet address, and two additional outputs (with nominal amounts) must go two "fake" Bitcoin wallet addresses which actually encode the reserve public key. The (few) Satoshis involved in the transfers to the fake addresses are burned.

Storing metadata

Ethereum

Smart contract?

- ▶ Logs in smart contract is the recommend way (ethereum.org)
- ▶ Expensive (additional storage and execution fees)
- ▶ Avoidable attack surface (error prone)

Custom input format

Use input data in transactions, usually used to call smart contract, to store our metadata.

2024-08-26

NEXT GENERATION INTERNET

└ Bonus: Depolymerization [7]

└ Storing metadata

1. Ethereum recommends encoding meta-data in smart contracts. But, this is actually more expensive as the result is large and comes with execution (Gas) fees. The complexity also makes it fragile.
2. Depolymerizer instead uses a custom input format.

Smart contract?

- ▶ Logs in smart contract is the recommend way (ethereum.org)
- ▶ Expensive (additional storage and execution fees)
- ▶ Avoidable attack surface (error prone)

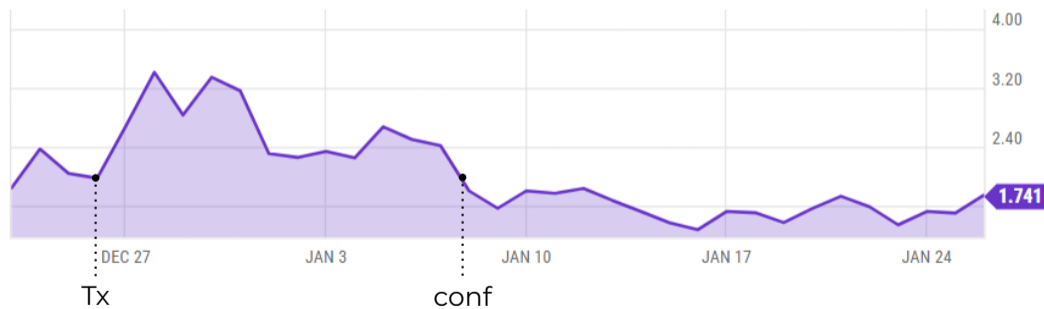
Custom input format

Use input data in transactions, usually used to call smart contract, to store our metadata.

Blockchain challenges

Transactions stuck in mempool

We want confirmed debits within a limited time frame.



2024-08-26

NEXT GENERATION INTERNET

└ Bonus: Depolymerization [7]

└ Blockchain challenges

We want confirmed debits within a limited time frame.



1. When we trigger a debit with a fee too small, it may not be confirmed in a timely fashion.
2. If we picked the current transaction cost at time Tx and the minimum cost for inclusion in a block goes up afterwards, it would take until time "conf" for the transaction to be actually included in a block and thus confirmed. And that is assuming it even stays in the mempool for this long.

Blockchain challenges

Transactions stuck in mempool

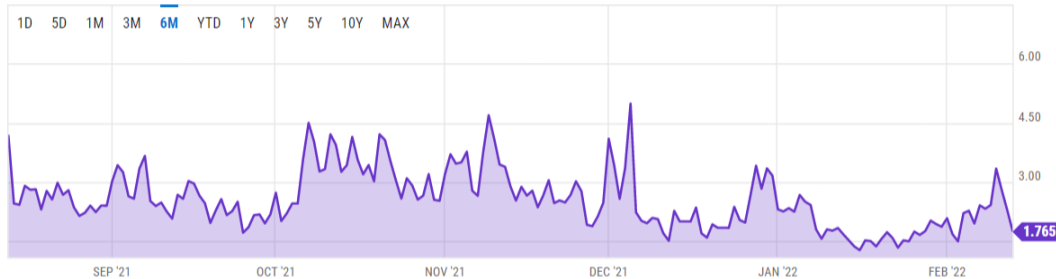


Figure: Bitcoin average transaction fee over 6 months (ychart)

2024-08-26

NEXT GENERATION INTERNET

└ Bonus: Depolymerization [7]

└ Blockchain challenges



Figure: Bitcoin average transaction fee over 6 months (ychart)

1. However, transaction fees are unpredictable as shown in the historical chart.
2. Offering high fees to miners would significantly increase transaction costs, while using low fees risks transactions not being mined in a reasonable amount of time.
3. Depolymerizer thus **monitors** stuck transactions, and if necessary increases the transaction fees paid to miners if transactions are stuck for too long.

Future work

- ▶ Support other blockchains
- ▶ Universal auditability, using sharded transactions history
- ▶ Multisig by multiple operators for transactions validation

2024-08-26

NEXT GENERATION INTERNET
└ Bonus: Depolymerization [7]

└ Future work

- ▶ Support other blockchains
- ▶ Universal auditability, using sharded transactions history
- ▶ Multisig by multiple operators for transactions validation

1. It might be interesting to see what challenges and solutions apply to other blockchains.
2. Right now, a Taler auditor always requires a full copy of an exchange database, which would not scale if “everybody” wanted to participate in an audit. Universal auditability is about allowing everyone to check that an exchange is operating correctly. With public blockchains, at least the underlying settlement layer data is already public!
3. Still, this would only detect problems **after** an incident. Requiring multiple signatures for on-chain transactions from independent operators sharing access to the escrow account would eliminate the single point of failure.

Conclusion

Blockchains can be used as a settlement layer for GNU Taler with Depolymerizer.

- Trust exchange operator or auditors
- + Fast and cheap
- + Realtime, ms latency
- + Linear scalability
- + Ecological
- + Privacy when it can, transparency when it must (avoid tax evasion and money laundering)

2024-08-26

NEXT GENERATION INTERNET

└ Bonus: Depolymerization [7]



└ Conclusion

Blockchains can be used as a settlement layer for GNU Taler with Depolymerizer.

- Trust exchange operator or auditors
- + Fast and cheap
- + Realtime, ms latency
- + Linear scalability
- + Ecological
- + Privacy when it can, transparency when it must (avoid tax evasion and money laundering)

1. Note that this approach does not address the ecological footprint of the underlying blockchain, it only addresses the problem for off-chain transactions.
2. It also does not prevent money laundering and criminal abuse on the underlying blockchain, only the off-chain part **could be made compliant**.
3. Making the off-chain part compliant is still a huge challenge, as the operator would also have to validate the legality of the **source of funds** for all incoming transactions on the blockchain, and it is not easy to do that well.

References I

-  Hanaa Abbas, Maurantonio Caprolu, and Roberto Di Pietro.
Analysis of polkadot: Architecture, internals, and contradictions.
In *2022 IEEE International Conference on Blockchain (Blockchain)*,
pages 61–70, 2022.
-  Wei Dai.
Pipenet 1.1.
<http://www.weidai.com/pipenet.txt>.

2024-08-26

NEXT GENERATION INTERNET

└References

└References

 Hanaa Abbas, Maurantonio Caprolu, and Roberto Di Pietro.
Analysis of polkadot: Architecture, internals, and contradictions.
In *2022 IEEE International Conference on Blockchain (Blockchain)*,
pages 61–70, 2022.

 Wei Dai.
Pipenet 1.1.
<http://www.weidai.com/pipenet.txt>.

References II

 Debajyoti Das, Sebastian Meiser, Esfandiar Mohammadi, and Aniket Kate.

Anonymity trilemma: Strong anonymity, low bandwidth overhead, low latency - choose two.



In 2018 IEEE Symposium on Security and Privacy (SP), pages 108–126, 2018.

 Edsger W. Dijkstra.

Self-stabilizing systems in spite of distributed control.

Commun. ACM, 17(11):643–644, nov 1974.

References III

-  Roger Dingledine, Nick Mathewson, and Paul Syverson.
Tor: the second-generation onion router.
In Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13, SSYM'04, page 21, USA, 2004. USENIX Association.
-  John Douceur.
The Sybil Attack.
In Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002), March 2002.

2024-08-26

NEXT GENERATION INTERNET

└References

└References



 Roger Dingledine, Nick Mathewson, and Paul Syverson.
Tor: the second-generation onion router.
In Proceedings of the 13th Conference on USENIX Security Symposium - Volume 13, SSYM'04, page 21, USA, 2004. USENIX Association.

 John Douceur.
The Sybil Attack.
In Proceedings of the 1st International Peer To Peer Systems Workshop (IPTPS 2002), March 2002.



References IV

-  Antoine d'Aligny, Emmanuel Benoist, and Christian Grothoff.
Project depolymerization: Tokenization of blockchains.
In 4th Conference on Blockchain Research and Applications for Innovative Networks and Services, September 2022.
-  Nathan S. Evans, Chris GauthierDickey, and Christian Grothoff.
Routing in the dark: Pitch black.
In 23rd Annual Computer Security Applications Conference (ACSAC 2007), pages 305–314. IEEE Computer Society, December 2007.


References V

-  Seth Gilbert and Nancy Lynch.
Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services.
SIGACT News, 33(2):51–59, June 2002.
-  Priscilla Huang, Emmanuel Benoist, Christian Grothoff, and Sebastian Javier Marchano.
Practical offline payments using one-time passcodes.
SUERF Policy Briefs, (622), June 2023.

References VI

-  Brian N. Levine, Michael K. Reiter, Chenxi Wang, and Matthew K. Wright.
Timing attacks in low-latency mix-based systems.
In Proceedings of Financial Cryptography (FC '04), pages 251–265, February 2004.
-  Laurianne McLaughlin.
Philip Zimmermann on what's next after pgp.
IEEE Security & Privacy, 4(1):10–13, 2006.

References VII

-  Taishi Nakai, Akira Sakurai, Shiori Hironaka, and Kazuyuki Shudo. A formulation of the trilemma in proof of work blockchain. *IEEE Access*, 12:80559–80578, 2024.
-  Nuno Preguiça, Carlos Baquero, and Marc Shapiro. Conflict-free replicated data types (CRDTs). In *Encyclopedia of Big Data Technologies*. Springer, May 2018.
-  Butler Lampson Ronald L. Rivest. Sdsi - a simple distributed security infrastructure. Technical report, April 1996.

-  Taishi Nakai, Akira Sakurai, Shiori Hironaka, and Kazuyuki Shudo. A formulation of the trilemma in proof of work blockchain. *IEEE Access*, 12:80559–80578, 2024.
-  Nuno Preguiça, Carlos Baquero, and Marc Shapiro. Conflict-free replicated data types (CRDTs). In *Encyclopedia of Big Data Technologies*. Springer, May 2018.
-  Butler Lampson Ronald L. Rivest. Sdsi - a simple distributed security infrastructure. Technical report, April 1996.

Acknowledgements

Co-funded by the European Union (Project 101135475).



Co-funded by
the European Union

Co-funded by SERI (HEU-Projekt 101135475-TALER).

Project funded by



Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Swiss Confederation

Federal Department of Economic Affairs,
Education and Research EAER
**State Secretariat for Education,
Research and Innovation SERI**

Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Security in Distributed Systems

Christian Grothoff

Security in Distributed Systems

1. Partially offline payments

- Set up a Taler merchant backend and/or use the public one at <https://backend.demo.taler.net/> (password is “sandbox”).
- Withdraw some KUDOS to your Taler wallet so you could pay at the merchant backend.
- Install an OTP app on your phone.
- Configure an OTP device in the merchant backend and share the secret with your OTP app. Do **not** use the “with amount” mode, as canonical OTP apps do not support it!
- Create a template in the merchant backend and set up the OTP device for payment configuration.
- Scan the QR code of the template to initiate a payment.
- Pay and check the confirmation code from the payment configuration with your OTP app.

2. Depolymerization

- Enter `taler://withdraw-exchange/btc.ice.bfh.ch/` into your Taler wallet (can also be done via QR code)
- Select withdraw funds, select a tiny amount of Bitcoin.
- Do an on-chain Segwit transaction following the instructions from the Taler wallet.
- Do BTC peer-to-peer transfers with your fellow students!