

DARCs and Calypso

Non-fintech things to do with a blockchain

Intro

1) 3 Chapters

- a) DARC, Calypso, Auditing

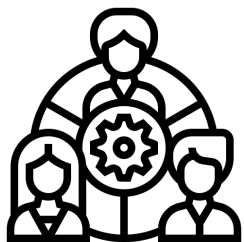
2) For every chapter

- a) **Goals** - what you should be able to know and to do afterwards
- b) **Introduction** - theory and demonstrations
- c) **Exercices** - to be done in your group - later exercises build upon earlier ones!
- d) **Bonus** - if you're fast and waiting for the others - reading material and additional exercises

Ia. DARCs

- Understand how DARCs work
 - Rules: Action / Expression
 - Delegation - trust other DARCs
- Signature verification with DARCs
 - Need latest version
 - Parse the tree
- Recovery using DARCs

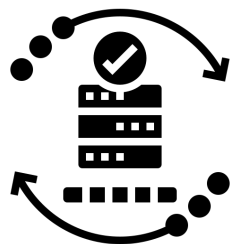
Ib. Delegation of Access



Replace the
password with
a *Delegation*



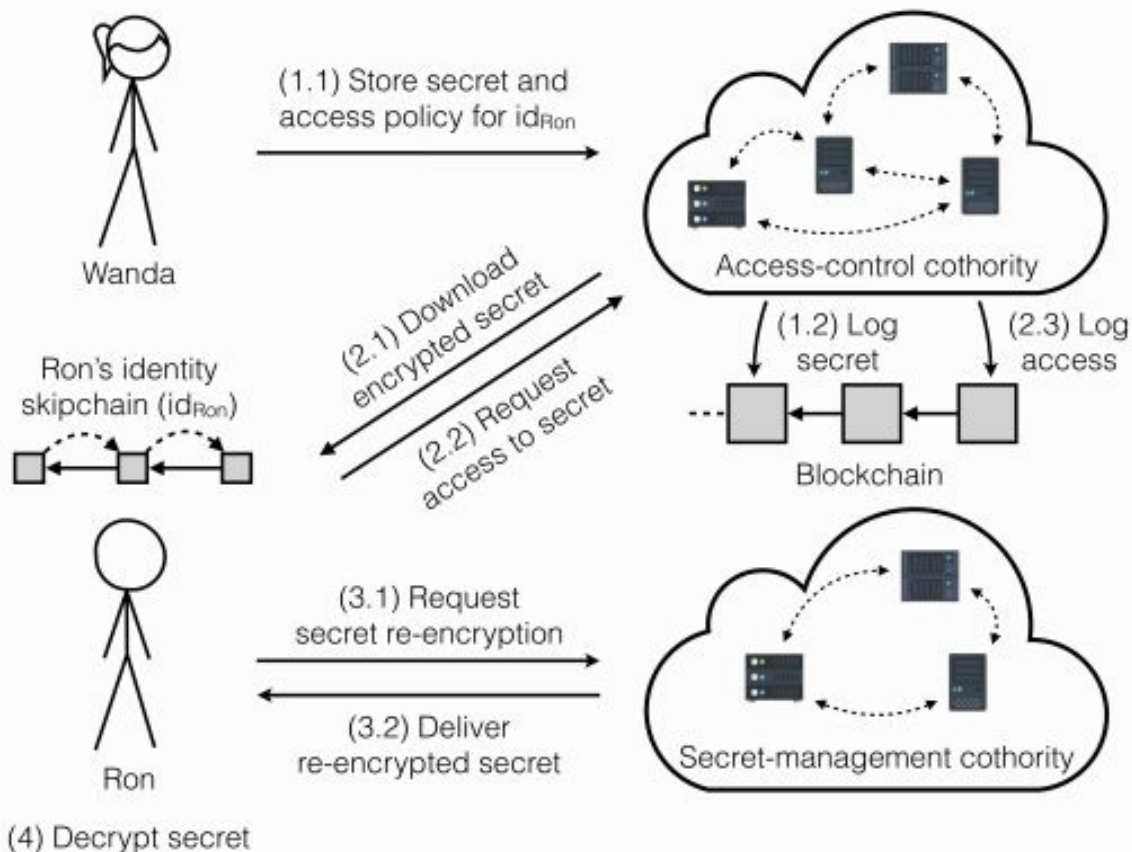
Full audit of
updates and
usage



Access can be
updated over
time

Ib. Calypso

2018/209 - Verifiable Management of Private Data under Byzantine Failures



Ib. Authentication, Authorization, and Identity

Centralized

- Authentication
 - OpenID Connect
- Authorization
 - OAuth2
 - Links Authentication to Rights
- ActiveDirectory combines both
- Little control by the user
- Insider abuse very easy
- Closed-in systems - need to verify both servers and clients

<https://www.town-crier.org/>

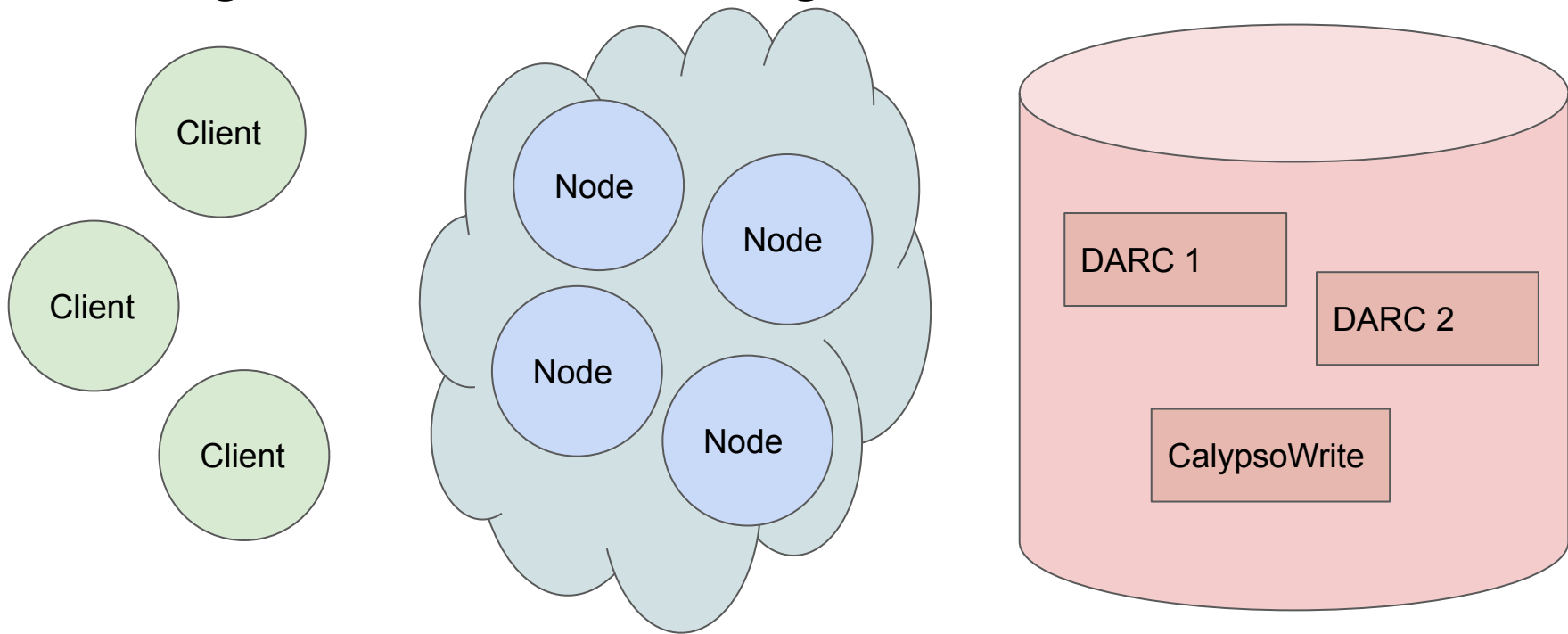
Decentralized / Distributed

- Authentication
 - W3C Decentralized IDs
- DARC -> Distributed Access Rights Control
- Is an attribute-based access control
- Current attributes supported
 - Signature authentication
 - Ed25519 / DID
 - AND / OR / () operations
 - Calls to smart contracts
- Self-sovereign identity
 - Add / remove devices
 - Key-rotation

Ib. DARC Rules

- A DARC combines the authorization, authentication, and delegation
- Every rule has
 - Action - the authorization described
 - Expression - how the action can be verified (authentication)
- Two special actions
 - **Evolve** - defines how the DARC can be changed from one version to the next
 - **Sign** - if this DARC has been delegated
- Available Expressions
 - AND / OR / ()
 - Call to view-method of a smart EVM contract
 - Signature verification (ed25519, W3C DID as WIP)
 - Delegation to another DARC

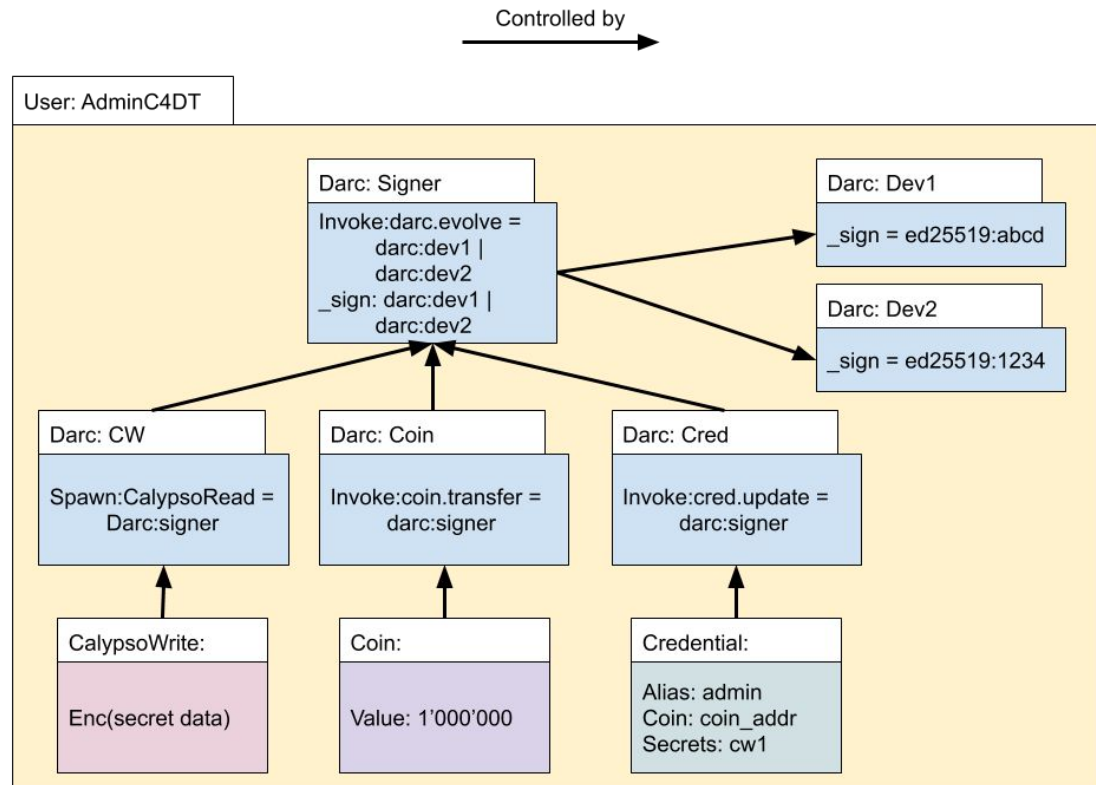
Ib. Digression - OmniLedger 101



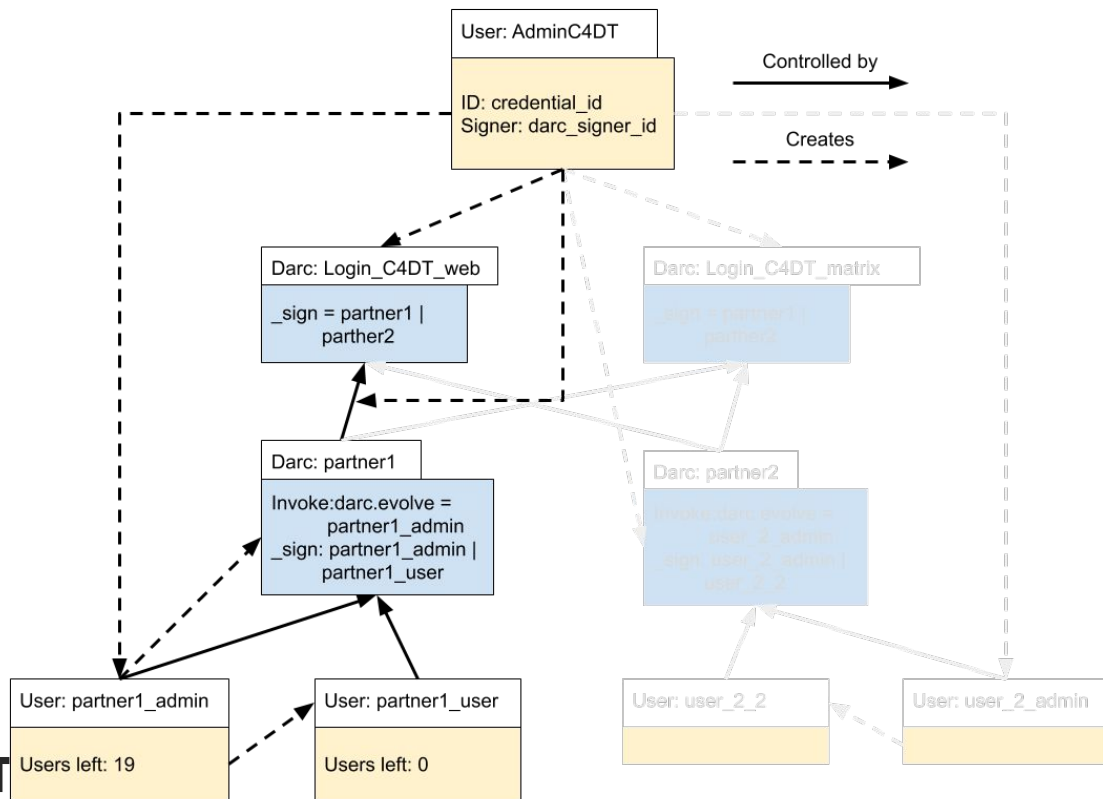
Ib. Digression - OmniLedger 101

- ByzCoin - blockchain based on [\[1602.06997\]](#) - plus *time.Sleep(166)*
- OmniLedger - [sharding](#) (not implemented) - cool name
- Every Instance
 - Has a unique ID - 32 byte hash
 - Is linked to a contract (DARC, Coin, Value, CalypsoWrite, EVM, ...)
 - Links to a DARC for access control
 - Can be updated, depending on the contract
- Blocks contain transactions and a hash of the merkle tree of the instances
- Using forward-links, OmniLedger
 - Avoids forks
 - Allows for succinct proofs
- For more information, see [GitHub](#) - [Stats](#) - [Article](#)

Ib. Delegation Example - Inside User Definition



Ib. Federated Access Delegation



Ic. Exercise 1: Create a group DARC

- Each group creates a DARC with delegation to all members of the group
 - Decide which name the DARC should have
 - One member creates a new DARC (*Contacts / Create Group*)
 - adds all member of the group to the DARC using *Edit Group*
 - and sends the BaseID of the DARC to matrix
 - All the other members of the group add a link (*Contacts / Link Group*)
- Other members: try to modify the DARC
 - What is the error message?
 - Why does it fail?
 - How to change the DARC so that all members can modify the DARC?

Ic. Exercise 2: Add Recovery

- Allow your group-darc to recover your wallet
 - *Devices / Add Recovery* - chose the group DARC from exercise 1
- Try to recover an account from a member of your group
 - *Contacts / Recover*
 - You can open the link in a 'private session' in Chrome, but not in Firefox (missing DB-support)
- Try to recover an account from a member of another group - what happens?
- We'll look in exercise 4 what happened to his/her account!
- Keep the recovery-account, so the members of your team can give you back your account, should you delete the private key!

Ic. Exercise 3: Start up StackBlitz

- For the next chapter, you'll need more access to your account
- Create a new device, but don't click on the link!: *Device / Add Device*
- Copy the URL somewhere safe
- Open [StackBlitz](#)
- On the left side, click 'Settings' (the gear-symbol), then change 'Hot reload trigger' to 'Save'
- Paste the device-URL in the open file in the `deviceURL` variable
- Save (CMD+S or CTRL+S) to reload
- Open the file *handson/handson.component.ts*

Ic. Exercise 3b - Show Around in StackBlitz

- Left: files - only need *handson.component.ts*
- Middle: editor - usually stay in *ngOnInit*
 - Extra: for one time actions, add a *buttonYourCode* method and copy/paste code in *handson.component.html*
- Right: output of *ngOnInit* and the buttons
 - Blockchain explorer - Action buttons - Output
- Buttons:
 - *Print instance* - prints out the information about an instance
 - *Create CalypsoWrite / CalypsoRead* - used for later exercises
 - *UpdateLTS* - debugging - hopefully not used ;)
 - *Send Coin* - simple example of how a transaction is created
 - *Create Value / Update Value* - used for bonus exercise

Ic. Exercise 4: What Happened while Recovering?

- Use *Print instance* to show your ‘Signer DARC’
 - Count the number of delegations
 - Compare with the number of devices and recoveries
- Use the *Print instance* to show your ‘recovered by’ device
 - In the wallet, in the *Devices* tab, click on the name of the device to get the *baseID*

Id. Bonus

- Do the Id. Bonus
- Use the Stackblitz environment to create a new value instance, protected by your group darc
 - Ask another member of your group to update the value - for this you need to copy the *Send coin* button and modify it to

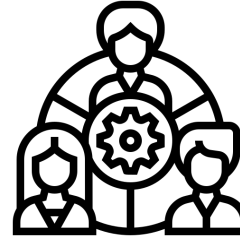
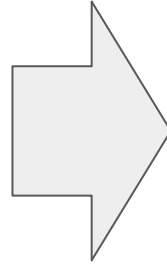
Ila. Calypso

- Explain the difficulty with classical encryption when things change
- Understand what re-encryption is
- Being able to store a secret and retrieve it

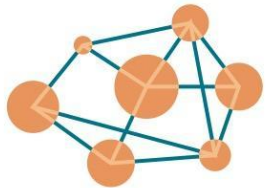
IIb. 'Password' Manager



Store key
encrypted on
blockchain



Delegate
access to the
key



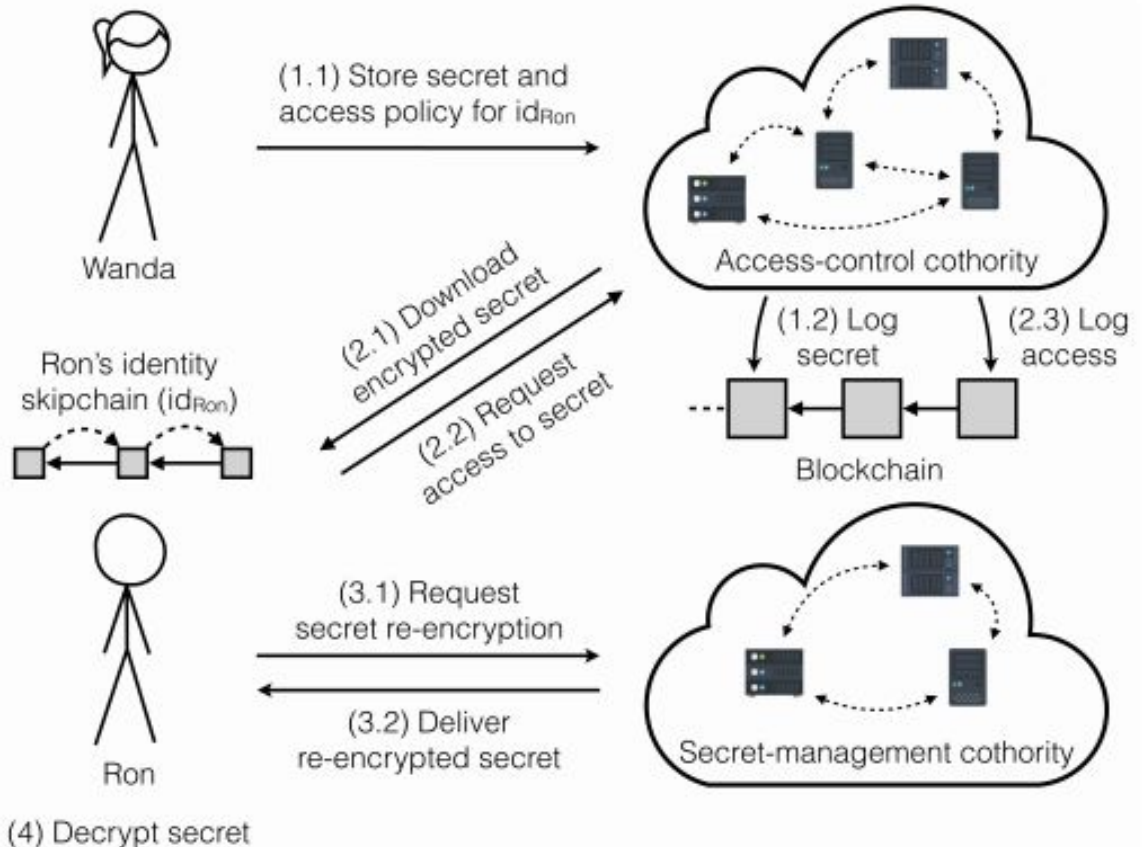
Only the block-
chain can
decrypt the key



Decryption can
be audited

IIb. Calypso

2018/209 - Verifiable Management of Private Data under Byzantine Failures



IIb.

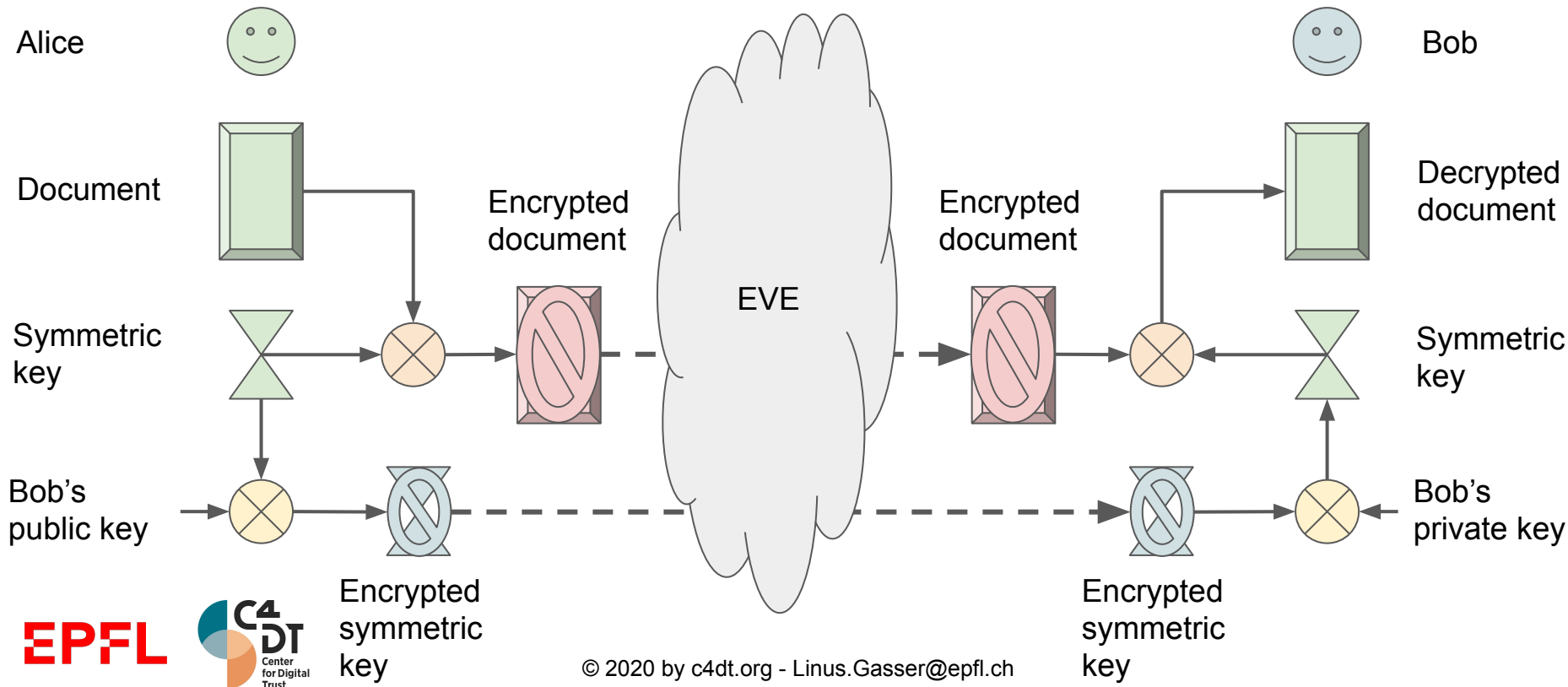
Asymmetric Encryption

- Public / Private keys
 - Private key needs to be kept safe
 - Public key is shared
- Challenge of Key Infrastructure
 - Public keys can be exchanged over unsafe medium
 - How to be sure that the keys are correct?
- Encryption
 - Everybody encrypts using the public key
 - Only the private key can decrypt
- Signature
 - Only the private key can sign
 - Everybody verifies using the public key
- Biggest downside: very slow!

Symmetric Encryption

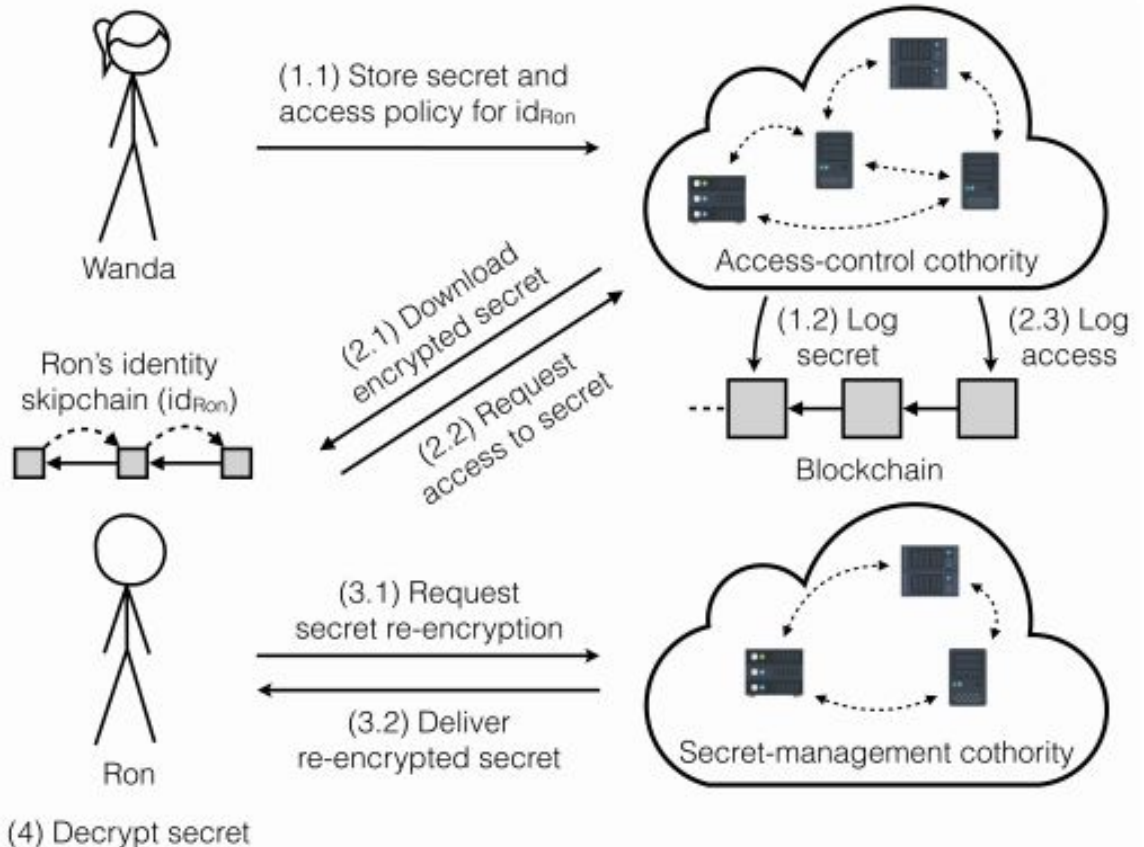
- Only one type of key
- Needs to be treated securely
- Cannot be transferred as-is over unsafe medium
- Biggest upside: very fast!

IIb. Asymmetric and Symmetric Encryption

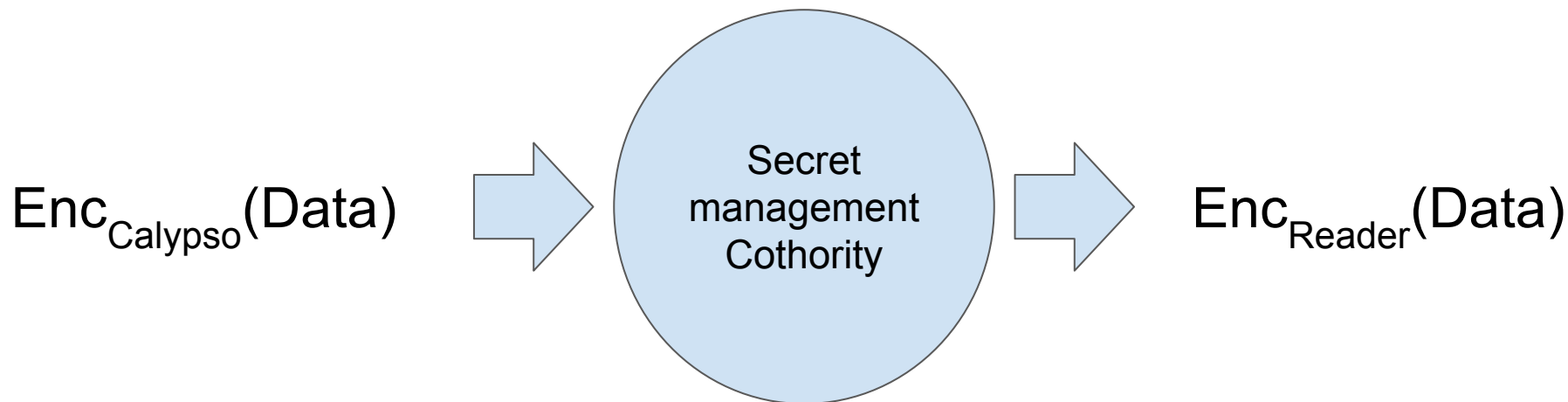


IIb. Calypso

2018/209 - Verifiable Management of Private Data under Byzantine Failures



Ilb. Calypso for Dummies



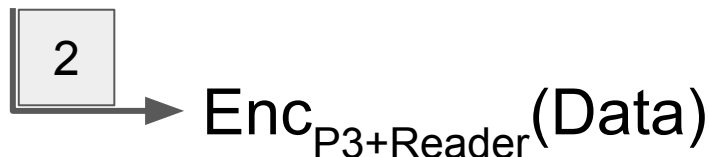
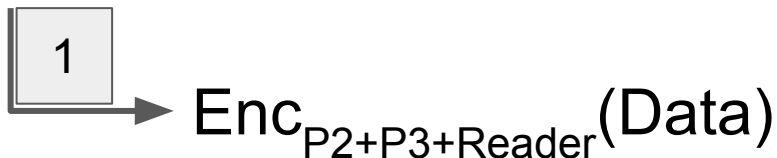
- How to
 - re-encrypt without decrypting the data?
 - proof the reader is allowed to re-encrypt?
 - avoid fraudulent re-encryption requests?
 - change access rights after encryption?

IIb. Re-encrypt Without Decrypting the Data?

$\text{Enc}_{\text{Calypso}}(\text{Data})$

$\text{Enc}_{P1+P2+P3}(\text{Data})$
)

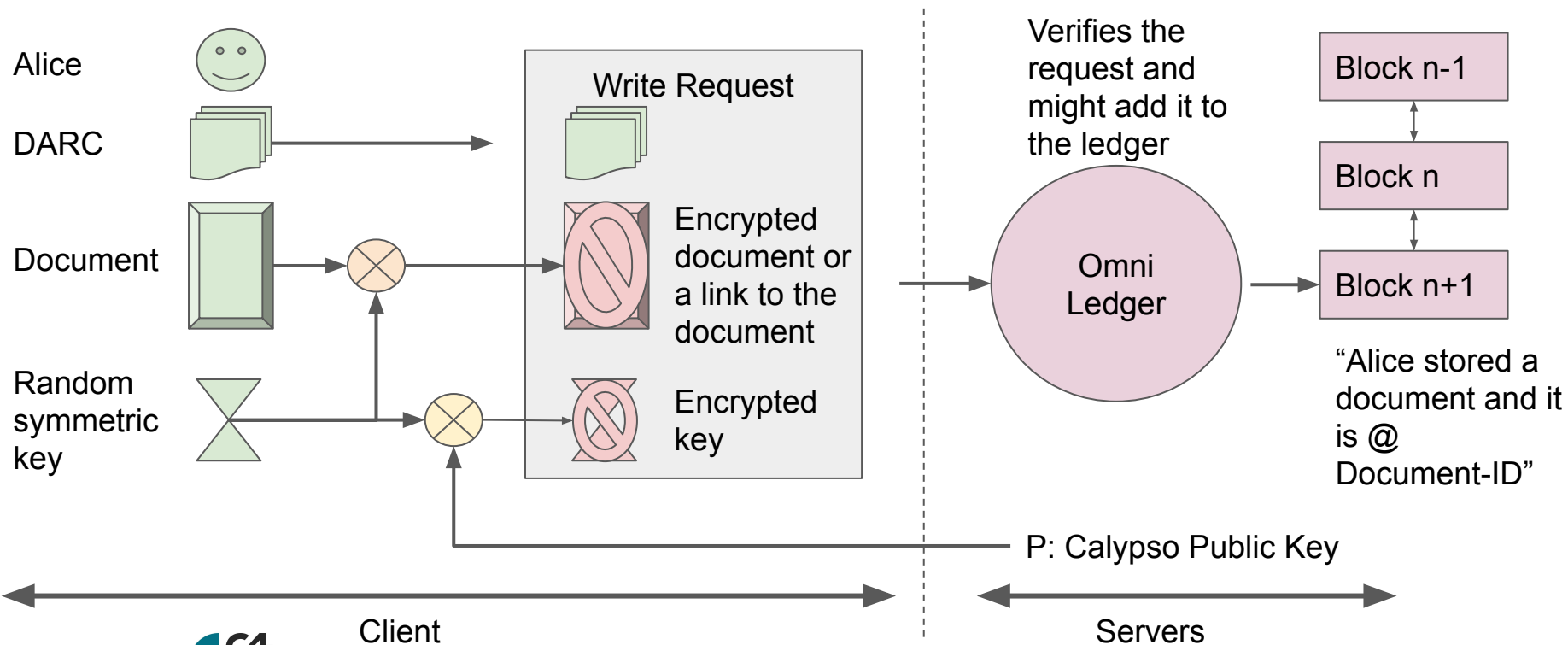
- Using homomorphic properties and zero-knowledge proofs to
 - proof the re-encryption has been done correctly
 - add the reader-encryption in every step
- Limitation using ed25519: 31 bytes of data!



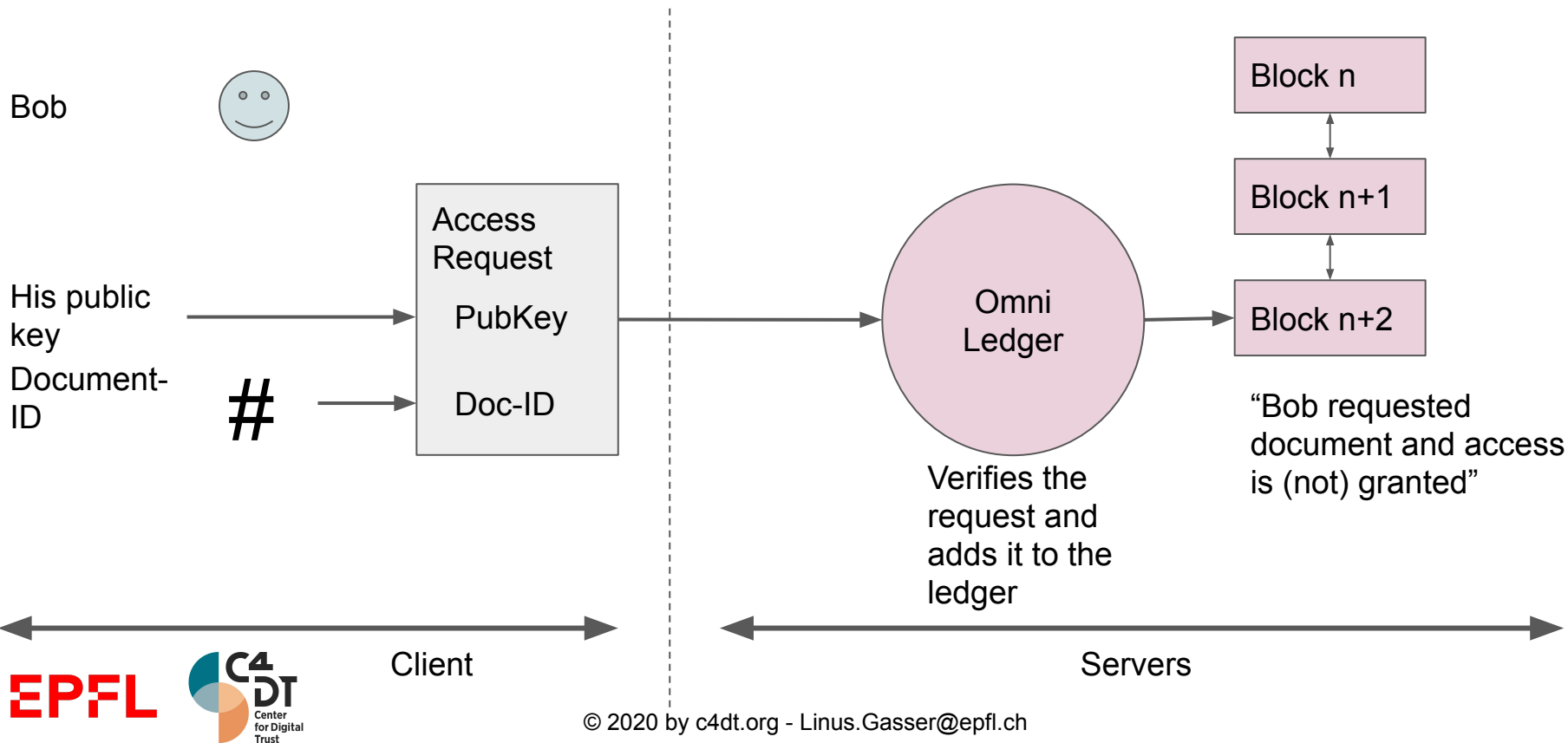
IIb. DARC usage

- How to
 - proof the reader is allowed to re-encrypt?
 - change access rights after encryption?
 - -> describe the access rights using DARCs
- How to avoid fraudulent re-encryption requests?
 - Writer needs to proof he knows the clear-text secret
 - -> more zero knowledge proofs

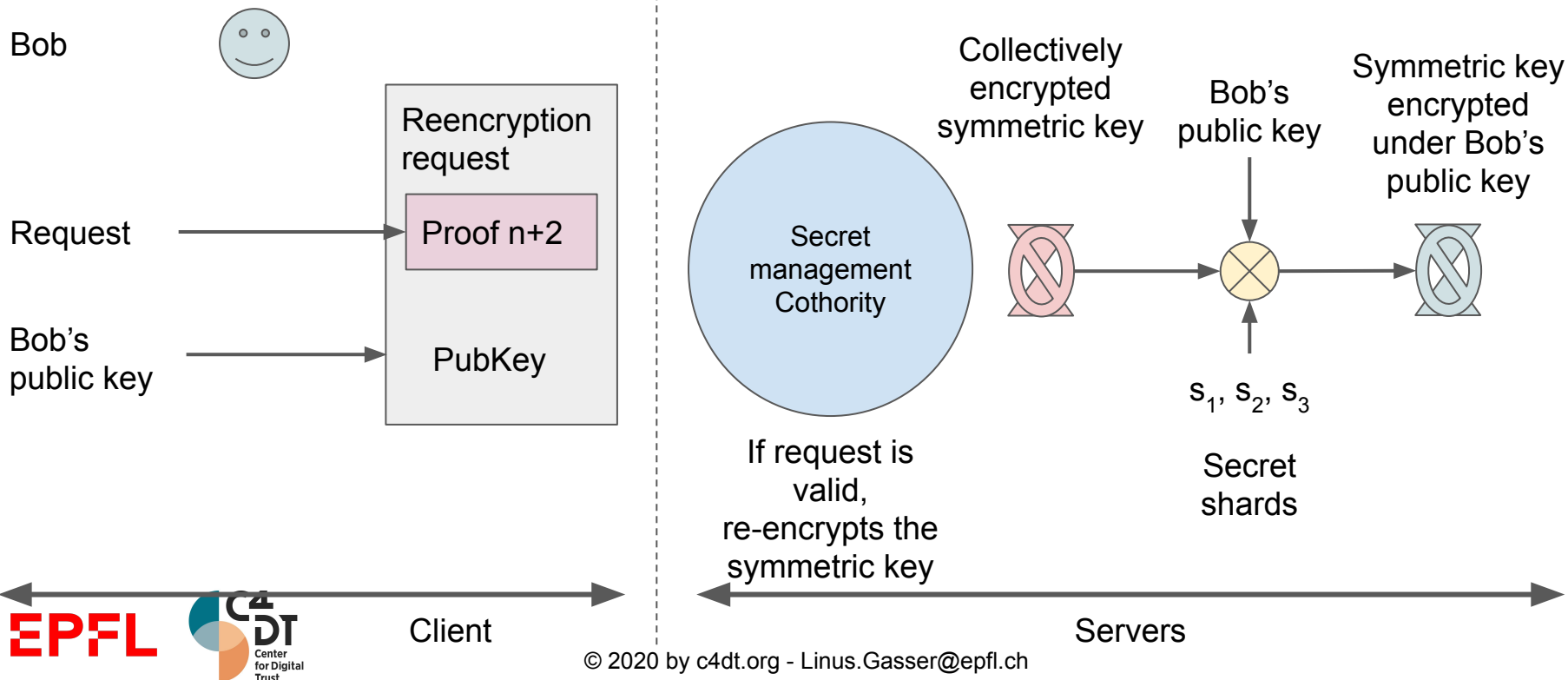
IIb. Alice Encrypts her Document



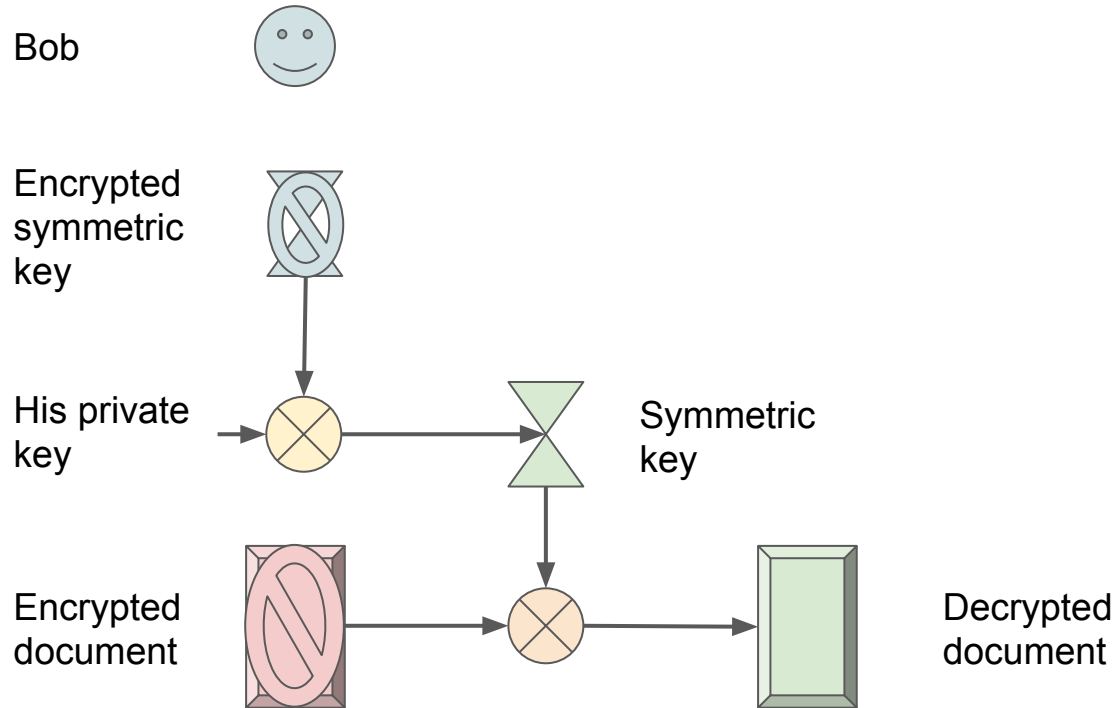
IIb. Bob Requests Reading the Document



IIb. Bob Requests the Document's Symmetric Key



IIb. Bob Decrypts the Document



IIb. Summary

- Alice
 - Doesn't need to know the reader in advance
 - As long as a majority of the nodes are honest, nobody can read the message
 - Can adjust the readers over time
- Bob
 - Can trust nobody else sees the document
- Eve
 - Cannot interfere
- Auditor
 - Sees who accessed the data

IIc. Calypso Exercise

- Encrypt a text using Calypso
 - Everybody in the group uses stackblitz to *Create Calypso-Write*
 - Use your group-DARC for the access control
 - Post the WriteID to matrix
- Decrypt the texts from your group
 - Use *Read Calypso* in stackblitz
 - Find the transaction in the blockchain
 - Can you see who read the secret?
 - How could you find out?
- Try to decrypt texts from another group
 - Take a WriteID from matrix from another group and try to decrypt it
 - Find the failed transaction in the blockchain

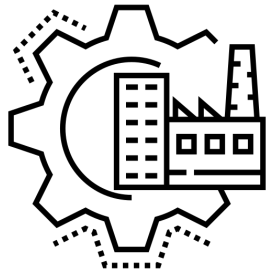
Ild. Bonus

- How could you avoid giving away who read the secret?

IIIa. Auditing using Calypso

- Understand how to support off-chain encryption
- Exercise on-chain

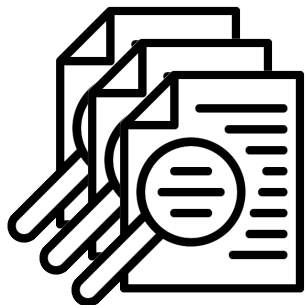
IIIb. Auditing Encrypted Certificates



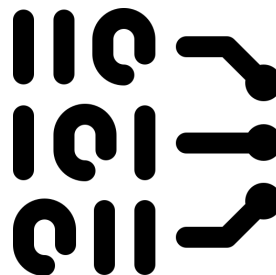
Given product
info from Pharma



Attach
delegation for
audits



Different
auditors for
different batches



Multi-signatures
Time-based
Location-based

IIIb. Auditing

- Application of Calypso
- System encrypts logs to Calypso public key
- Adds DARC for allowing re-encryption
- When auditors come, delegate re-encryption to them

IIIc. Auditing Exercise

- In each group, discuss what DARC's can be used to implement this use-case
 - 1 - every factory is independent
 - 2 - “one DARC to rule them all”
 - What are the security issues for both solutions?
 - Chose one solution and create the DARC's, share the IDs in your group
- Create the Exercise:
 - Person 1 - auditor
 - Person 2..n - factories
 - Factories create *CalypsoWrite* with the DARC chosen in the previous discussion
 - After the *CalypsoWrite* is done, every factory adds the auditor to its DARC
 - The factories send the writeIDs to the auditor
 - The auditor checks the messages

IIId. Bonus

- Switch the auditor to one of the other group, and let it decrypt all messages