Data Security in the Swiss Electronic Health Records (E-HD): Four possible architectures

Management Summary

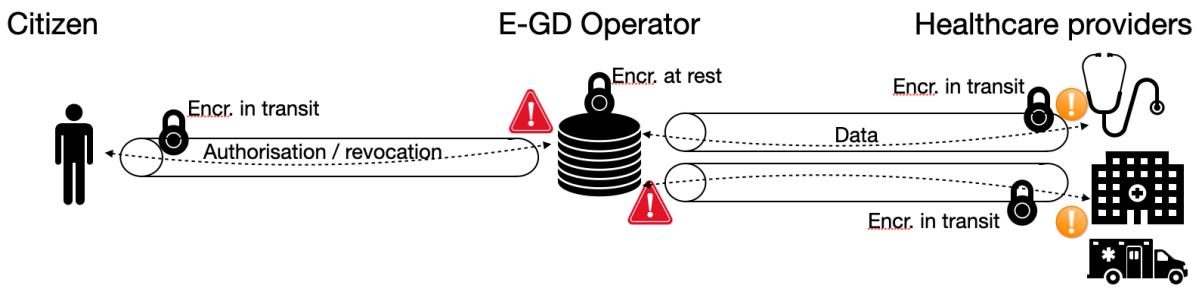
Linus Gasser, Imad Aad, Jean-Pierre Hubaux
Center for Digital Trust (C4DT), EPFL

July 17, 2026

Executive Summary

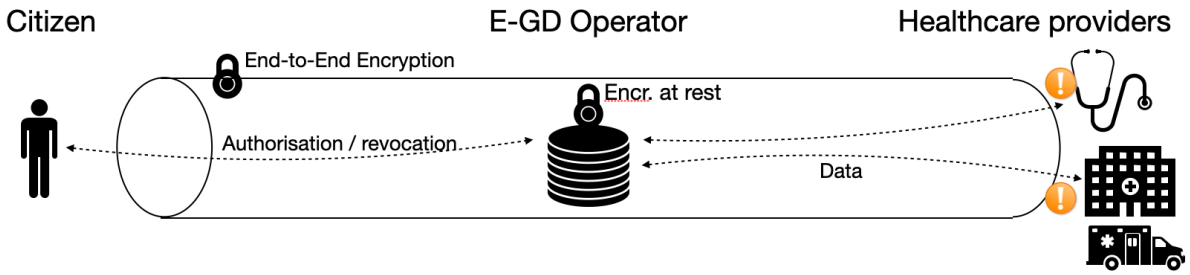
Health data must be accessible when appropriate *and* must be protected, as it is highly sensitive. For the Electronic Health Record (E-HD), a security architecture must be defined to achieve this dual goal. This document presents 4 possible architectures.

Architecture I relies on organizational measures (see figure below). It assumes that the E-HD operator (the entity selected by FOPH to run the E-HD) respects the law and that its credentials will not fall in the wrong hands. Technically, it is the simplest solution.



Data is encrypted in transit and at rest. It is decrypted and re-encrypted at the interfaces emphasized by the red exclamation marks, meaning that it is accessible to the E-HD operator.

In contrast, Architectures II, III and IV provide additional technical protections and follow the principle of end-to-end encryption (E2EE), summarized in the following picture.



End-to-end encryption: data is not accessible to the E-HD operator. The data storage by healthcare providers is outside the scope of this document

By means of Confidential Computing, Architecture II supports code verification by third parties. In Architecture III, the end user (citizen) is entrusted with key management. Architecture IV relies on the so-called t-of-n parties principle.

The main criteria to select the architecture are usability, cost, and data security.

1. Introduction / goals

This document proposes four possible architectures for the Electronic Health Dossier (E-HD), to be run under supervision of the Federal Office of Public Health (FOPH).
In line with the mandate we received from FOPH in April 2026, the focus of this work is on end-to-end encryption (E2EE), where only the relevant data is accessible to the entities dealing with the user's health (e.g., family doctor, hospital, emergency service), for a precise purpose. No other intermediaries, not even the FOPH or the service provider(s), should access the data. This fulfills the "need to know", "data minimisation / proportionality" (Art. 6, al. 2), "privacy by default and by design" (Art. 7) and "purpose" (Art. 6, al. 3) principles of the new Swiss Federal Data Protection Act (nFDPA).

Furthermore, the proposed architectures allow the users to control their own data by stating who has access to what. For each proposed architecture, we analyse the privacy properties it fulfills.

The proposed architectures try to minimise the amount of accessible metadata (e.g., user X visited hospital Y). However, bearing in mind that the E-HD is not a simple storage system but rather a complex platform for data sharing with dynamic access rights, emergency services etc., accessible metadata cannot be eliminated without disproportionately expensive measures.

In line with the mandate, and due to time constraints, we do not delve into the security measures (e.g., technical and organisational measures against cyberattacks). The proposed architectures remain compatible with these measures, and they can be deployed separately. For security against malware hidden in encrypted data, we propose several approaches.

The proposed architectures are agnostic on data formats. Data elements can be scans of paper documents, unstructured pdf files, or data structured according to specific standards. The proposed architectures, data protection mechanisms, and data exchange protocols apply equally to all data formats. Our proposition are measures that affect the intended usage and functionality of the E-HD to enable digital workflows.

In the next section (2) we describe individual architecture elements: Encryption models, key-related operations, data classification, access control etc. In Section 3 we propose four possible architectures and describe the interactions. Section 4 tackles additional considerations like scanning malware and encrypted searches.

2. Basic concepts

In this section we introduce the basic concepts used in Section 3.

2.1 Encryption Models

To hide data from unintended readers/eavesdroppers, the content must be encrypted. Only entities possessing the proper key can decrypt and read the data. Two encryption models exist and can be combined.

Symmetric encryption

Like a safe, it requires the *same code*: symmetric encryption uses the same key for encrypting and decrypting data (e.g., sending an encrypted zip file by mail). This method requires sharing the cryptographic key to the recipient via a separate channel (e.g., phone call, sms).

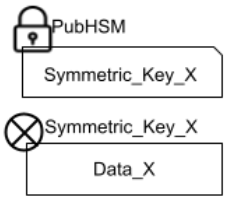
Symmetric encryption is relatively simple, but it requires communicating the secret key on a different secure channel. This limits the usage to specific scenarios, typically one-to-one communications. Consider for instance a bank offering services on its webpage. The bank cannot establish separate symmetric keys with all its clients, on separate secure channels, before each time they connect securely to the webpage.

Asymmetric encryption

To enable encryption between parties which meet for the first time, asymmetric encryption is used (at the basis of http**s**:// we use in our browsers). The bank has a private/public key pair. Anyone can encrypt data using the bank's public key (the lock), while only the bank is able to decrypt the data, using its private key (the key of the lock). The public key (locks) is accessible to anyone, while only the bank possesses the private key (a single key for all the locks that are accessible by anyone). Only the bank is capable of decrypting the data using its private key. Intermediaries can only see encrypted data.

Hybrid encryption

Combining the above 2 encryption methods, data can be first encrypted symmetrically (put in a "safe"). Then the encrypted data is re-encrypted asymmetrically using the destination's public key. Different combinations can be used to meet different conditions, as we'll see in the architectures below.



2.2 Onboarding, Key Establishment and Management

Onboarding Healthcare Providers

Authentication: We assume the E-HD operator can authenticate Healthcare Providers using AGov or other methods used in the health sector.

Access credentials: After authentication, access credentials are established such that the Healthcare Provider can communicate and upload/download user data to/from the E-HD. This can be done via a webpage, or an API allowing health service applications to access the database.

Onboarding Citizens

Onboarding a citizen means connecting them to their E-HD and giving them the means to modify the access control and manage their documents. In our four architectures, citizen authentication online can be done using AGov or the forthcoming eID (swiyu). After onboarding, the citizen possesses credentials (for future online sign-ins) and the necessary keys used for encryption (depending on the architecture)

Note that the E-HD law mandates that all health services share relevant data to the E-HD. In practice, this may happen while the users are not onboarded yet. Hence, the healthcare provider should be able to upload the citizen's data, encrypted, before the citizen gets their credentials or encryption keys.

2.3 Data Classification (emergency, general, private)

To fulfill various types of health services and their respective requirements of data accesses, 2 different data categories are defined:
General / Emergency: Data accessible to any health service with user confirmation (in the app or written) or with emergency access.
Private: The user defines which service accesses data from this category. No other Healthcare Providers can access them

All access to the data is logged. In addition, after emergency access, the client is notified with SMS and email.

2.4 Access Control and User profiles

Access control is the mechanism of granting specific users access to specific data. When both users and data operate within a single organisation, access control can be enforced at the system level, allowing user X to access data Y, or user category A to access data category B. On the other hand, when users, data, and platform operators belong to different organisations that do not necessarily trust each other (e.g. bad governance, or malicious system administrators), access control can be implemented using encryption: Citizen X gives Healthcare provider Y the key to decrypt the data stored on the servers at organisation Z. This is the case in all architectures proposed below.

Different people have different opinions about health data privacy. Accordingly, the system will propose several profiles. At one extreme, data will be easily accessible (ex.“permissive”). At the other extreme, only minimal access will be provided by the citizen (ex.”restrictive”). Several intermediate profiles can be defined. The default profile, that is the one adopted without user intervention, can be defined by the regulator. At enrollment time, citizens will be invited to select the profile they prefer. The motivated citizen will also have the option to parametrize manually each of the access rights.

2.5 Break the Glass

If the citizen does not prohibit emergency services from accessing his/her Emergency Data, upon an emergency event access to this data without the need of user intervention (the user can be unconscious) will be granted. In practice, the emergency care provider will access the emergency data of the patient through the app installed on the emergency care provider’s phone or computer, with appropriate credentials (ex. HIN identity, eID, using 2FA etc.). This assumes of course the patient to be identified, using various methods depending on whether the patient is conscious, uses eID etc. To prevent abuse, all data accesses are logged in a system that cannot be altered by the BAG (ex. administered by a separate organisation) and each citizen is informed of each access. To prevent mass data leaks by the exploitation of this feature (for example through the stolen phone of a healthcare provider), a simple solution consists in capping the number of records that a given emergency care provider can access (e.g., not more than 20 treated patients per day).

2.6 Hardware Security Module

An HSM (Hardware Security Module) is a hardened physical device used to securely generate, safeguard, and manage cryptographic keys. It executes sensitive encryption, decryption, and digital signing operations within a physically isolated, tamper-resistant environment. By keeping master keys separate from general servers, it prevents attackers or unauthorized software from extracting sensitive credentials. It establishes a "root of trust" for an organization's cryptographic infrastructure. Commonly deployed in finance, government, and cloud data centers, it ensures compliance with data security regulations.

2.7 Trusted Execution Environments

A Trusted Execution Environment (TEE) is a secure area within a processor that isolates sensitive code and data from the rest of the system, including the operating system and other applications. It provides confidentiality and integrity by ensuring that only authorized code can access protected information, even if the main system is compromised. TEEs are commonly used for tasks such as secure authentication, cryptographic key management, digital payments, and privacy-preserving computing.
From a technical point of view, a TEE produces a signature on the input data, the program code, and the output data. Furthermore, the actual running of the program code is secured through cryptographic measures to make it difficult for other parts of the system to spy on the execution.

For our use-case, this allows us to put all the work on encrypted data in a TEE and tell the HSM to allow decryption data requests only if it comes from an audited code which is run in the TEE.

2.8 Confidential Computing

Confidential computing is a security approach that protects data while it is being processed, not just when it is stored or transmitted. It uses hardware-based trusted execution environments (TEEs) to isolate computations from the rest of the system.
This prevents cloud providers, administrators, or malware from accessing sensitive data during processing. Applications can perform computations on encrypted or protected data with stronger privacy guarantees. It is commonly used for secure cloud computing, AI workloads, healthcare data, and financial services.

2.9 Attackers / Trust models

In the context of the E-HD, several types of attackers must be considered:

- An external hacker trying to steal the data or disrupt operations
- A rogue employee working for the E-HD operator illegally accessing the data.
- The Cloud provider, using the data stored at its premises for other purposes, such as to train LLMs
- An authoritarian government that would change the laws, thus granting itself (or other entities) access to data that was previously restricted. Note: Hoping to thwart this threat by a security architecture is naive.

Trust, in the context of the proposed architectures, relies on two components:

- "Compromise threshold" reflects the amount of effort to break the system. The higher the threshold, the better.
- "Public verifiability" reflects how much the public (e.g. civil society organisations, academics, researchers) can access and verify the code that is run by the organisations running the E-HD platform. The higher the public verifiability, the better. It is fundamental to be able to verify the full code running in the Trusted Execution Environments, operated by an untrusted operator (e.g. the E-HD operator), or by a trusted organisation that may turn untrusted in the future.

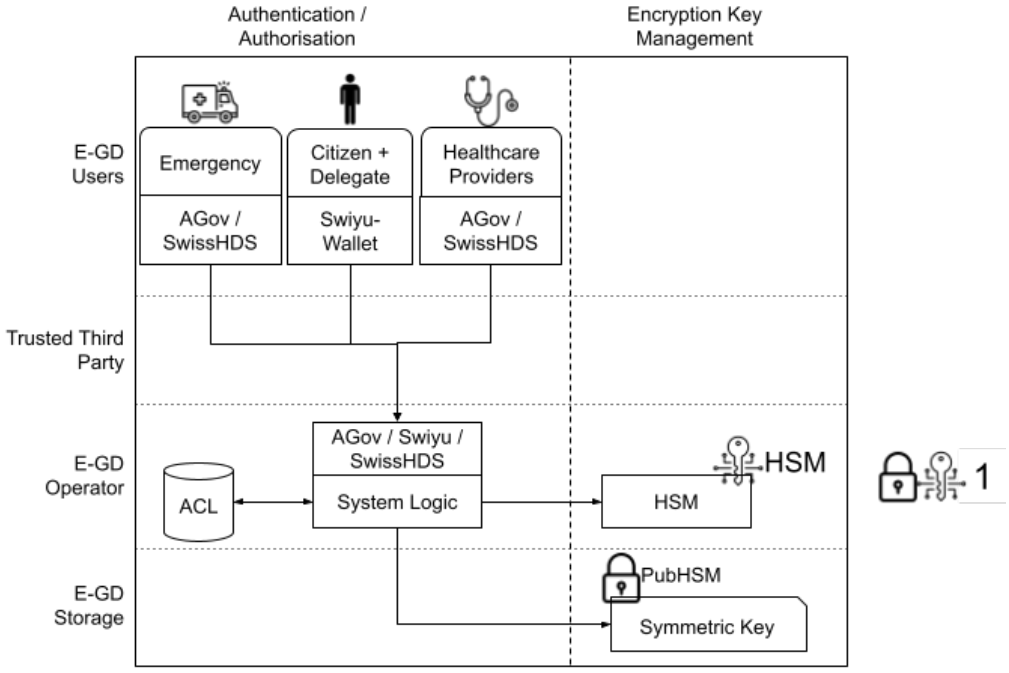
3. Four Possible Architectures

In the following sections, we present four possible architectures, ranging from fully-centralized, to fully-decentralized. We present here Architectures I, II and III. Architecture IV is more futuristic; its full description is in the appendix.

3.1 Overview of the four proposed architectures

		Technical Protection		
	Architecture I Organizational Protection	**Architecture II Confidential Computing**	**Architecture III Citizen-Managed Keys**	**Architecture IV Fully Decentralized Key Management**
Brief description	Trust the law	Protect against some abuses	Users have full control	Logged and fully open
Security robustness	Low	Medium	Medium	High
Technical Maturity	High (TRL 9)	High (TRL 8)	Medium (TRL 6-7)	Low (TRL 4-5)
Biggest downside	Access Control Management is single point of failure	Complexity; only few HSM products fulfil the requirements	Infrastructure and key recovery from users	No standards and only MVP code available
Recovery Mechanisms for credentials	Centralised, logged	Centralised, logged	Recovery decryption key, t-of-n key	Delegation, recovery key, t-of-n key, all logged
Delegation, Tutor	Centralised	Centralised	In the credentials on the wallets	Delegated as far as necessary
Break the Glass emergency	Centralised	one-time tokens accepted by HSM	t-of-n key with logging	Delegated urgency tokens, fully logged
Access Control Management	Government service	Credentials stored in Swiyu wallet	Credentials stored in Swiyu wallet	Access control in distributed ledger
Decryption	HSM		Every actor has their own decryption key	Re-encryption service
Identity Providers	Swiyu			
Encrypted data storage	Cloud service			

3.2 Architecture I: Organizational Protection



Architecture I - Organisational Security; PubHSM is the E-HD Operator's public key.
See Section 3.5 for description of the components

Architecture I uses standard components and relies on organisational security. We use it as a baseline to show how key management and security can be improved by removing responsibility away from the *E-HD Operator*. We suppose that the components are programmed according to the law, and that security best practices are followed. The *E-HD Operator* has full control over the HSM and the *E-HD Storage*, and only the implementation of correctness and log files can prevent abuse by the *E-HD Operator* or prove it did afterwards.

Encryption / Decryption: all encryption and decryption are done within the realm of the *E-HD operator*. The HSM is the only holder of the E-HD Operator private key, but it is driven by its *System Logic*, which only gives organisational security: an error or a bad implementation will allow decryption to more than the necessary participants in the system.

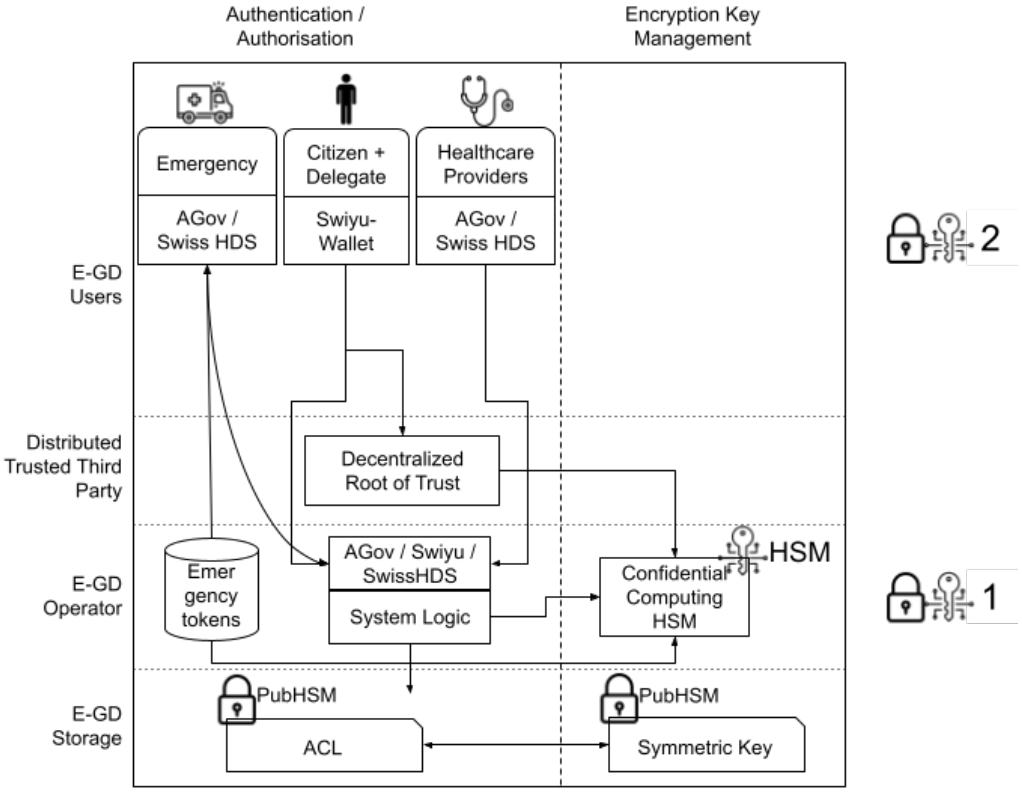
The advantages of this system are:

- Relatively simple to implement
- Known components and mature libraries

The downsides of this system are:

- Single point of failure with the *System Logic* tied so closely to the *HSM*
- No protection in case of change of ACL database

3.3 Architecture II: Technical Security - Confidential Computing



Architecture II - Confidential computing. The decryption decision is moved from the System Logic of the E-HD operator to the Confidential Computing HSM. PubHSM is the E-HD Operator's public key.

This second architecture improves the *single compromise threshold* of the first architecture by adding an HSM with confidential computing, e.g., [Thales Luna]. Instead of trusting the governance and correct implementation by the government, the clients can verify the code, but they still must trust the HSM producer to not collude with the government. This solution has a high maturity, as these HSMs are available commercially and have been extensively tested.

Encryption / Decryption: The decryption of the symmetric key is done in the Confidential Computing HSM ("1" in the figure), which means that there is a possibility of collusion between the manufacturer of the HSM and the E-HD Operator. On the other hand, the decryption of the documents is usually only done by the E-HD Users ("2" in the figure).

To make this work, a couple of changes with respect to Architecture I are necessary:

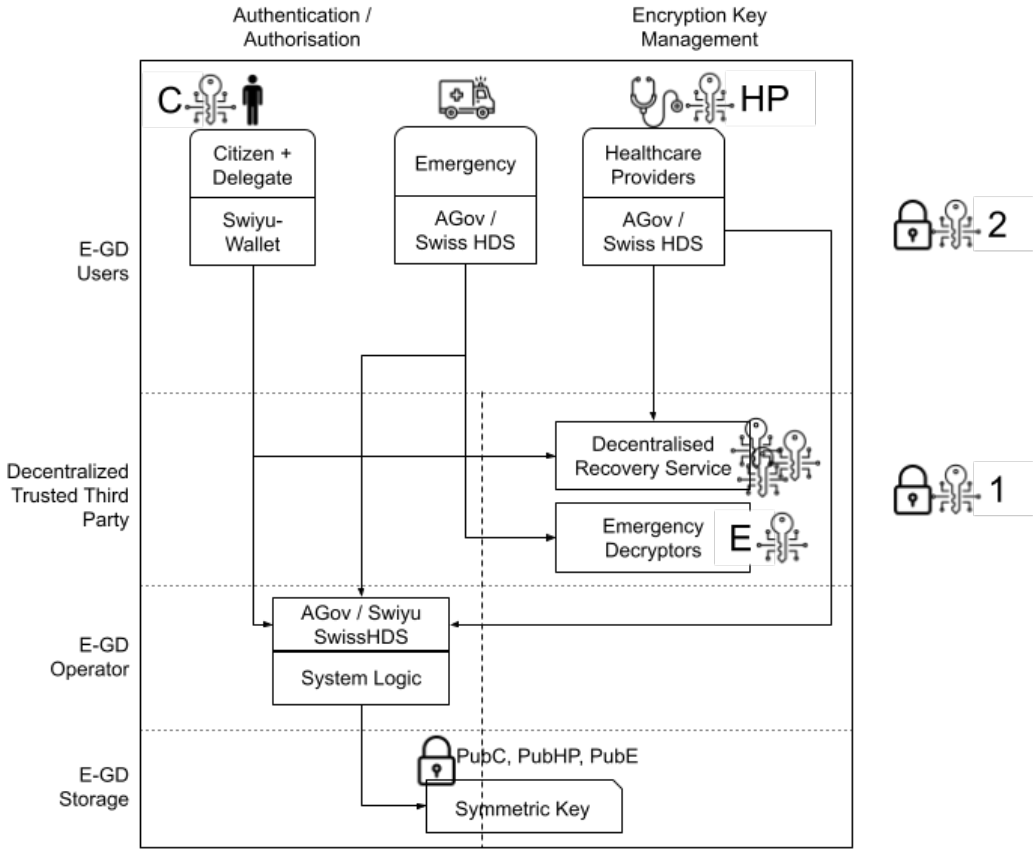
- *The medical data users get a credential in their swiyu-wallet* which allows them to authenticate towards the HSM, ask for decryption and/or to modify the ACL in the storage system.
- *There may be distrust that the E-HD Operator can abuse the system, changing the ACL to grant itself access to specific data. Therefore, a decentralized root of trust* is created with the permissions to update the ACL, but only if a threshold number of the members of the system sign off on it. This allows the members of this system (e.g.,

cantons, civil society groups) to log the requests, and flag abusive patterns. Instead of trusting the E-HD Operator, the E-HD users must only trust that at least one of the members is honest and will flag abusive behaviour, e.g., a big number of read requests from a single actor.
- *One-time emergency tokens* are created for the the emergency access: they enable a logged one-time access to the emergency data through the HSM.
- *The Access Control List* is encrypted on the storage system, and can be updated by the client, their delegate, or the decentralized root of trust. Each client has their own ACL.

The advantages of this system are:
- The *Compromise threshold* is increased, as now the government cannot unilaterally decrypt the data, because the HSM would not allow that.
- The *Client verifiability* increases: the client can verify that the HSM runs the required software, and interest groups can read the logs of the decentralized root of trust service to make sure no rogue keys are created.

The downsides of this system are:
- Complexity increases: instead of running everything on a central server, the development needs to take into account various pieces and make sure that they fit together perfectly.
- Even though the Confidential Computing HSM does exist as a commercial solution, they are not common and must be carefully chosen.

3.4 Architecture III: Technical Security - Citizen-Managed Keys



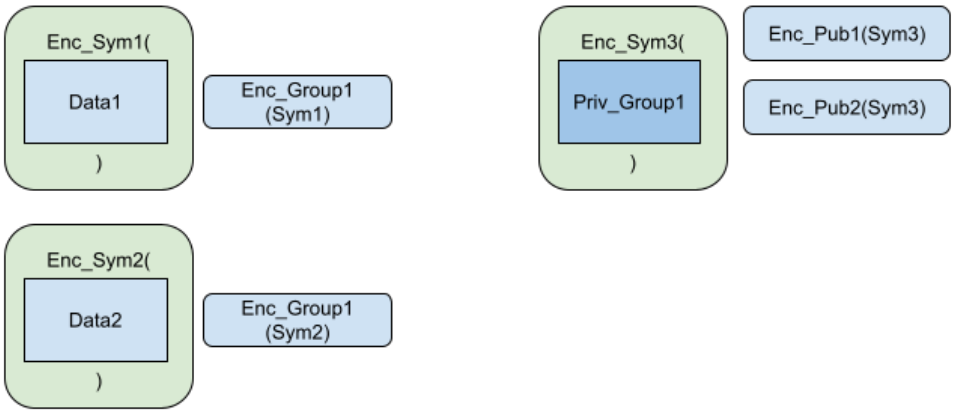
Architecture III - Citizen-managed keys. A decentralized Trusted Third Party holds partial copies of the keys for recovery and for emergency decryption. PubC, PubHP, and PubE stand for the public keys of the Citizen, the Healthcare Provider, and the Emergency Services, respectively.

While Architectures I and II keep the actual decryption keys in the HSM for easier recovery, in Architecture III the decryption keys are stored in the devices of the users (the citizens). This makes recovery of these keys particularly important, as in practice users will lose their keys - often!
But thanks to this storage of the keys in the clients' devices, the non-client compromise threshold rises again and prevents the government from accessing the medical documents.

Encryption / Decryption: all relevant keys are only stored with the E-HD Users (“2” in the figure) and the Decentralized Trusted Third Parties (“1” in the figure). The E-HD Operator does not have access to the keys nor to the cleartext data in any step of the operations.
To implement emergency decryptions and key recovery services, without delegating the task to the E-GD Operator, this task is implemented within the “Decentralized Trusted Third Party” as in Architecture 2. This allows the members of this system (e.g., cantons, civil society groups) to log the requests, and flag abusive patterns. Therefore the E-GD Operator is not able to abuse the system by recovering user keys in order to access their data.

ACL / RBAC: as a proof-of-concept, we only show how to implement a simple access control list using public key encryption: every data folder is encrypted with a random symmetric key, which is encrypted with the public key of every reader who needs access to the data. To change the ACL in this scheme, one can add a new reader by adding an encrypted symmetric key, or remove a reader by deleting the encryption:

It is also possible to create a Role Based Access Control and group various keys together by giving them access to a private key, as described in the following figure. Two data files are encrypted to the same group, and don't need to be updated every time the group definition changes. This simplifies the handling of the access to the data considerably.



The main differences of Architecture III with respect to Architecture II are:

- The *private key* stored in the HSM is now replaced with one or more keys stored only on the clients devices.
- The *decentralized recovery service* needs to be able to re-create the private key of the client in a secure way. We propose to use a threshold key sharing with actors of the government and the public services who hold partial keys.
- The *storage system*, which now has several encryptions of the symmetric key to the different asymmetric keys. There are various optimisations possible to avoid having to re-encrypt everything when the access control changes.
- The *medical data users*, with the exception of the emergency user, hold the corresponding private keys to decrypt the symmetric key and use it to decrypt the actual data.
- The *emergency decryptors* uses a third party service that shares the private key(s) between all parties, and needs to have a subset of the parties to re-create the decryption key. This can be either a unique key for access to all emergency data, or one key per client.

The advantages of this system are:

- Full control of decryption by the E-HD Users: nobody else sees the data in unencrypted form or can access it.
- If outside attackers want to decrypt the data, they need to crack devices of the E-HD Users. For a single user this might be easier, but for large-scale data theft the attack needs to be carried out once per user.
- All emergency accesses are properly logged by the community of the emergency decryptors.

The downsides of this system are:

- Higher complexity; such a project would be technically risky due to the increased number of steps needed compared to Architecture II, the lower maturity of the components, and the technical expertise needed to fit all the components together.
- Putting citizens in charge of their own encryption keys will lead to many cases of lost keys, which can induce citizen frustration and highly unwelcome overhead for the caregivers..
- Handling of key recovery with regard to authentication must be properly prepared for a good user experience.
- Lesser established libraries and tools to set up the decentralized key recovery and emergency decryptors services.

3.5 - Architecture Components

Hereunder we describe the components appearing in the figures depicting the architectures.
(II) indicates a component only used in Architecture II.
(III) indicates a component only used in Architecture III.

Columns:
Authentication / Authorisation - All services related to identify users and define access rights
Encryption Key Management - All services related to handling the private and symmetric keys used in the encryption and decryption of the medical data

Rows:
E-HD Users - citizens, delegates, healthcare providers, and emergency services
Trusted Third Party (II) - is a non-government entity with high acceptance by the E-HD Users and verifiable public logging of all operations
Decentralised Trusted Third Party (III) - a group of non-government and government entities which can act if a subset of them decides on an action and publicly log that action
E-HD Operator - Entity running the backend software for the E-HD system, e.g., FOPH
E-HD Storage - Entity storing the encrypted data of the E-HD Users, e.g., Swiss Gov. Cloud

Boxes:
ACL - Access Control List describes which E-HD Users have the right to decrypt this data - challenge: protect non-lawful changes in the ACL
AGov / Swiyu - Authentication services to identify users
Citizen + *Delegate* - An owner of medical data and their delegates (family, representatives)

Confidential Computing HSM (II) - Special kind of Hardware Security Module with programmable API - challenge: prove that the running software is valid and without backdoors
Decentralized Root of Trust (II) - A public service which provides a logged signature to the HSM to authorize changing the ACL of the medical data - challenge: logging and rogue recovery
Decentralized Recovery Service (III) - A group of actors (cantons, public organisations) which store a private key so that a subset can recover the key and assure logging - challenge: coordination, adding and removing members to the group
Emergency - healthcare providers needing urgent access to a subset of the medical data - challenge: preventing rogue access to the medical data available in case of an emergency
Emergency Decryptors (III) - A group of actors (cantons, public organisations) which have a partial private key and can decrypt data and assure logging - challenge: coordinating, adding and removing members to the group
Emergency Tokens (II) - Unique PINs which can be used to authorise emergency decryption by the HSM - challenge: renewing and distributing PINs
Healthcare Providers - Hospitals, medical doctors, pharmacies
PubHSM, PubC, PubHP, PubE - Public keys of the HSM, the Citizens, the Healthcare Providers and the Emergency Decryptors - challenge: public key infrastructure
Swiss HDS - Swiss Health Data Space with access to all other Healthcare Providers. Depending on the actual implementation of the Swiss HDS this can be used as an additional authentication and/or authorization provider, as well as an E-HD User - challenge: scope not clear yet, what are the needs and actions
Swiyu Wallet - The app on the device of the E-HD Users with credentials storing the certificates for the authorisation (II) or for decryption (III) - challenge: storing and using credential with private key with a secondary app
Symmetric Key - The key used to decrypt the medical data - in this document also used to represent the actual medical data - challenge: protect the key to be available only in the authorized context
System Logic - Defines how the requests are handled, what data is loaded, and interacts with the HSM - challenge: proving its validity, conformance with law

4. Additional Considerations

4.1 Detecting Malware in PDFs and Other Data

The following table summarizes possible solutions.

Term	Description	Positive	Negative
Client scanning	Before encryption and after decryption, the clients scan the PDFs before opening them on the system	Keeps full E2EE	Needs updated scanners on the clients
Transform PDFs	Strip attack vectors in	Removes most of	Even PDF/A might

	PDF files to create a PDF/A.	the potential attack vectors; PDF/A-1 and PDF/A-2 are more secure than PDF/A-3.	trigger a buffer overflow in the reader.
TEE scanning	Add a decryption key for a TEE like Intel SGX, which can take the file, decrypt it, run a virus scanner, and then give a "clean / not clean" output.	Most versatile approach, can also be used to scan documents on a regular basis.	Trust resides in the TEE and the software it runs. While it's possible to give some guarantees, the company providing the TEE needs to be trusted.

4.2 Searchable encryptions, and training AI on encrypted data

The two distinct objectives of “encrypted searches” are:

- Searches where the entity holding the data does not learn about the query contents (“query hidden from server”)
- The entity holding the data can search it without decrypting (“data hidden from server”). This is the main purpose in the E-HD context.

Different cryptographic techniques address one or both objectives. The applicability to the proposed architectures depends mainly on whether the techniques use symmetric or asymmetric encryption.

All four proposed architectures rely on encrypting different data elements with different symmetric keys, set by various Healthcare providers, which considerably limits the usage of the above techniques. One proposal is to include search indices in metadata, set by the Healthcare Provider producing the data, then encrypt this data using the technique from the corresponding architecture. The same logic applies to training AI using encrypted data.