Integrated Information Systems: Design-Options for Consortial Plattforms

Maximilian Dohlus
Physikalisch Technische Bundesanstalt
in Berlin
Abbestr. 2-12, 10587 Berlin, Germany
Email: maximilian.dohlus@ptb.de

Martin Nischwitz, Artem Yurchenko, Jan Wetzlich
Physikalisch Technische Bundesanstalt
Email: {artem.yurchenko, martin.nischwitz, jan.wetzlich}@ptb.de

***Abstract*—Implementing an Integrated Information System (IIS) comprises a lot of design-decisions. We present the 6 most important questions to be answered related to IIS-design and use decision-trees to provide an overview of the possible choices for each topic to help contextualize our solution. Our goal is to describe in detail our use-case, its requirements and why our approach is optimal within the described set of options.**

INTRODUCTION

Well known Data Integration (DI) expert Michael Stonebraker refers to DI as the "800-pound gorilla in the corner" [19]. This is due to the often vastly underestimated effort required to integrate even low numbers of data sources. Our use-case is based in the PTB Initiative "European Metrology Cloud" for creating an IIS within the realm of legal metrology [18]. We aim to present an overview of the main design-decisions involved in creating such an IIS, describe in detail our own use-case and show why it pertains to many real-world scenarios. We also consider that such systems are often influenced heavily by politics between participants and not solely by technical optimization.
For the main decisions to be made when designing an IIS, we use decision-trees to illustrate the range of possible options and explain how our approach matches the requirements set by the consortium: (1) monitor all data-sources for change to provide a secure legal basis for processes involving data shared among the participants, (2) ensure data privacy and integrity protection, (3) provide a basis for secure data transfer and process execution and (4) a queriable information system to access all shared data. Our project deals with metrological administrative processes defined by EU-/ or national law [18]. The consortium consists of manufacturers of measuring devices under legal control, the national metrological institutes (NMI), market surveillance (MS) and the verification authorities (VA) as well as the users of those measuring devices. Every stakeholder holds part of the required data regarding legal processes. This data needs to be shared with a select number of other stakeholders, while making sure that e.g. competing manufacturers cannot access their respective data. Overall participation in the system needs to be completely optional for all manufacturers and users. The main goal of the system is to provide streamlined services to private companies (manufacturers and users) by governmental agencies like the NMIs (PTB in our case) and the MS/VA (german: Eichbehörden) [18]. Thus, our goal is to integrate the roughly 2000 data sources currently in existence (in our national scope) by stakeholders in legal metrology and allow for the implementation of relevant legal processes like getting a new measuring device ready for market deployment or update the software of measuring instruments (MI).
In section I, we identify 6 major design-options for such an IIS, which are: data locality, data integration type, PACELC related options, hardware, log-storage and privacy and discuss them in detail in the following sections. Section II gives an overview of our data locality solutions within the context of all possible options and explains our choices framed by the PACELC-theorem. Section III talks about requirements for the hardware the system is build on (and why this is important). In section IV is a detailed explanation of our privacy and data storage solution. Finally, section V contains our conclusion and outlook for future work.

Contributions:

1) Based on current literature, a detailed description of the most important design-options for consortial IIS based on a large number of participants is given.
2) Our decision-trees are meant to help illustrate the range of possible options.
3) We explain our own solution and why it is optimal given the requirements we state in the introduction.

I. DESIGN-OPTIONS FOR INTEGRATED INFORMATION SYSTEMS

In literature, there are many options for designing an IIS. Some of these options contradict each other or lack practicality. Thus, we aim to present an overview of current design-options and how our own approach fits within the described range of possible configurations. From an abstract point of view, data integration (of data systems S_i) could be described as an IIS T not only linking data from all S_i to a combined whole (T) but also mapping processes from all S_i to T. We will discuss the main design-decisions to be made and how a given use-case can be easily classified to find the most

This work was done in context of The European Metrology Cloud.

appropriate system-design solution. Assuming we have too much data for one single machine to handle, the main degrees of freedom for T are:

- **Data Locality:** will the machines be physically close (i.e. data center) or spread over various locations in a distributed system (DS) - this decision mainly effects latency and bandwidth between nodes.
- **Data Integration type:** virtual data integration means implementing wrapper-type systems for on-the-fly schema-transformations from global to local (queries) or vice versa (result sets) [13] while data integration (which we will call manifested similar to the difference between virtual and manifested views in relational DBMS) means using bulk-insert - this decision pertains to implementation effort and political effects like the willingness of source system owners to change their systems.
- **PACELC:** for distributed systems, PACELC always forces us to decide between availability and consistency in the partitioned-network case and latency vs consistency otherwise - this decision affects things like using or not using (node-/data-) replication as well as enforced vs eventual consistency [9]. PACELC is an extension of the well known CAP-theorem.
- **Hardware:** which hardware will the system run on, who chooses the components, what is the storage capacity and other performance indicators - this not only affects overall performance but also cost and system security on the hardware side.
- **Storage:** how and where logs are stored, who can access what data and which information about other participants can be infered - this point is especially important because having immutable logging is one of the main requirements of our system in order to be accepted as proof in legal proceedings. Also affected are storage access times and network traffic.
- **Privacy:** multiple competing stakeholders that share their databases in one system is only possible, if the privacy of each stakeholder is guaranteed. Protecting that privacy, while still having enough transparency to e.g. manage access rights or provide consistent system logs and protocol compliance, raises multiple design questions.

II. Data

A. Data Locality and Data Integration type

Since we assume to have more data than can be stored on one single machine, we are left with the need for a network of computers. While in the past it has been clear that for data processing to be handled efficiently, those systems needed to be physically close and linked via high bandwidth LAN connections [17]. While this still holds true for most cutting edge applications and web services, thanks to ever increasing bandwidth available over the internet, a number of network based systems can be realized without this need for putting machines physically close. Some well-known examples for this are peer-to-peer-networks (p2p) like blockchain, which achieved high popularity in recent years, or decentralized applications (dApps). With p2p, dApps and similar approaches, a growing tendency towards decentralized systems exists and is likely to continue. Since we also look at stakeholder-politics, removing the necessity for a system-user of higher importance than others is crucial.

In our use-case, we have to deal with about 2000 heterogeneous data sources that need to be combined into an IIS. While modern data centers can easily hold that number of servers and would deliver very good network bandwidth and minimal latency, our own system has no need for that much traffic and can deal with reduced bandwidth and latency. On the other hand, allowing participants to freely choose where to put their machines and manage the data they want to share with the system is an obvious advantage, since it removes the need for third-party housing and administration of the hardware. Depending on system location, putting all machines in one central location also holds risks like power outages and other mostly localized phenomenathat could be avoided if the system-nodes can (not must) be spread over much larger distances and connect via Internet instead of LAN. Also this avoids having a central honeypot.

When talking about linking heterogeneous data sources, the first step is to define a global mediated schema and map all sources to that schema. The two main ways of doing this are bulk-wise (translate all data once and manifest the result similar to manifested views) or on-demand by deploying so-called wrappers that translate global queries into local ones and the local result sets back to the global schema [8], [13]. Both approaches are valid, depending on the situation. Using wrappers holds the risk of implementation errors in each of those translators which can be very hard to detect. Using bulk-insert, the process logic needs to be checked only once and data import errors can be found much more easily. As our use-case has to deal with a large number of data sources, employing wrappers is not practical.

The related decision tree looks like this: Figure 1. Using Wrappers might lead to a simplified integration of additional sources (or at least moves the implementation effort from a central authority to each source owner) but as mentioned

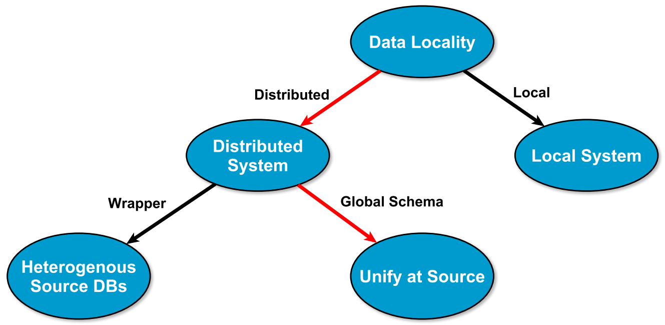

Fig. 1. Data locality and DI-type in a combined decision tree

above, may result in hard to find errors in the source-global-translations and as such reduce system stability with regards to process execution. Going the bulk-insert-route has the rather obvious advantage of being the best performing variant regarding query speeds.

Design-decision: Data Locality and Data Integration type: Part of our design-goals was to have a system with maximum flexibility and security while maintaining better-than-average performance and (mostly for political reasons) be as distributed and decentralized as possible. We chose to base our system on physically distributable nodes using a single schema on every node. By nodes, we refer to the machines located near stakeholder-systems that are linked in our systems network. We do not use node- or data-replication and thus create a distributed database in the exact same way relational tables get spread over multiple nodes: by slicing relations horizontally as well as vertically and distribute the data over all nodes. That way, there is no need for a centralized data cluster and every participant can choose what data to share via his own node. A node may be shut off at any time, although the remaining system at that point will only lose data stored on that node as we do no replication of any kind. If a use-case allows for the implementation of a centralized authority, this would in general be the simpler but less secure solution. This concludes our decision tree overview for distributed vs physically close systems (data locality) and on-demand-integration (wrapper) vs bulk-wise integration.

B. PACELC

Another degree of freedom for distributed data systems is based on the PACELC-theorem. It states the following:
In the case of having network partitioning (P), a system designer must choose between availability (A) of the system and consistency (C) (of data replicated across multiple machines). When there is no network partition (E), the designer still must choose between latency (L) and consistency (C). See: Figure 2. The first part is equivalent to the well-known CAP-theorem. The reason for this is easily explained: once a system needs to hold more data than a single node can handle, that data needs to be spread over multiple nodes. To achieve better availability, those nodes can be replicated such that when one of the data nodes A gets too many calls to handle, some of those requests can be transferred to one of its replicas A_i^*, thus increasing availability. Using replication also means that, in case a data change needs to be applied to A, only one replica gets the update, which then would need to be propagated to all other replicas A_i^* (including A itself). This is what introduces the consistency problem where systems using replication only get eventual consistency or lose out on the availability side of things while the system asserts consistency after value updates, which is only achieved once all replicas A_i^* again hold the same values as A.

Design-decision: PACELC: Since this system-design decision-tree is not of our own making, we only briefly explain our projects' approach: Since we do not implement node-replication for reasons of data-protection and privacy, we have no consistency-issues stemming from explicit replication. What we might have is a situation where two or more participants have several data cells or even full relations duplicated. In that case, our approach is to solve these issues on query execution when the node merging the result sets gets conflicting overlapping information from other nodes. Thus, we focus on consistency in the non-partitioned case. When dealing with network partitioning, we lose data from other partitions while those nodes cannot be reached. Still, we feel like our system can be said to focus on availability in the partitioned case because while not replicating data we do replicate essential services to keep the system running and all processes working as long as no data from another partition is needed to continue that process.

III. Hardware configuration

Given a decentralized system, every node makes an important security anchor for the overall network. Such trusted system requires a special focus on software security, but this is not feasible without a secure hardware foundation. The commonality of all hardware, software and firmware is directly or indirectly security-relevant and is therefore referred to as Trusted Computing Base.
Ever since the media-effective release of Spectre [4] and Meltdown [5], the issue of hardware security has caught the attention of the masses. In most cases, exploiting such vulnerabilities requires code execution on the affected system, which implies the attacker already having direct access to the system. Nevertheless, these gaps should not be underestimated. New zero-day exploits are often discovered, and additionally there is a Spectre variant which only requires the execution of a malicious JavaScript in the browser.

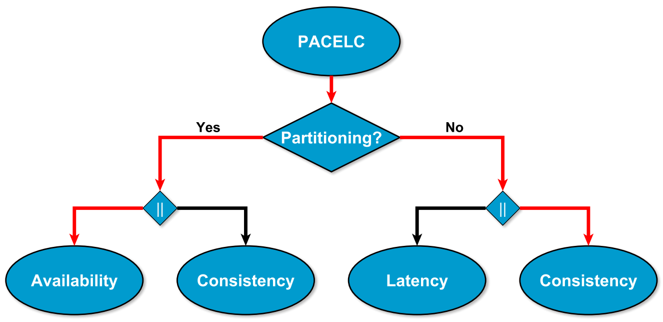

Fig. 2. Decision options stemming from the PACELC-Theorem

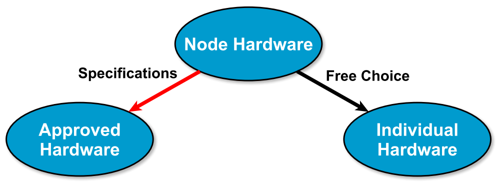

Fig. 3. Decision options for hardware configuration

Spectre and Meltdown exploit hardware bugs, but they are implemented in software and thus can potentially be detected by conventional safeguards such as antivirus software.
Another class of threats, that until recently has been assumed to be purely theoretical but has become more prevalent since the release of the StuxNet Virus, is firmware-affecting malware. This type of malware exploits vulnerabilities in the firmware of system components and thus bypasses existing protective measures. In some cases, implementation errors can weaken system security, as was the case with a bug in some TPM-modules from Infineon [6]. In other cases, the firmware vulnerabilities even allow unauthorized access to the system, bypassing system protection mechanisms, as it happened with recently discovered bugs in the Intel Active Management Technology [7]. Even permanent modifications of firmware, such as UEFI, are possible to provide stealth functionality for other malicious software [10]. Such modifications are very efficient and hard to detect. The potential possibilities extend to the modification of the CPU-microcode [11], which can be used to influence the execution of the commands in the processor on the lowest possible level. Due to strong encryption, microcode updates for current processors can be considered safe. Fortunately, such firmware-attacks today are often limited to specific hardware/firmware versions, but it is to be expected that the number of such attacks will increase over time and diversify to different hardware configurations.

These examples are intended to show that without secure hardware, integrity of the software cannot be achieved. In addition, hardware security can only be assumed for a certain time and this estimate can be subject to change. This results in a requirement to reduce hardware diversity to a manageable level, providing a better overview of the current threats and thus enabling faster reaction in terms of software or even hardware updates. Furthermore, the use of typed hard- and software also allows the potential attacker to completely focus on the given system. The choice between typed and non-typed hardware is therefore of political nature. No hardware typing is associated with tacit acceptance of some compromised network participants, while highly standardized hardware leads to an all or nothing situation. The design-options are visualized in Figure 3. Both variants are found in the market: for example, the MacOS operating system from Apple requires typed hardware, while Microsoft only sets the minimum requirements for their operating systems.

Design-decision: Hardware: For our system, the solution with typed hardware appears to be the preferred variant, as it not only keeps the development and maintenance effort within manageable limits, but also makes the security of the overall system easier to assess. Due to standardized components, the operating system no longer needs to support a variety of hardware and can therefore be stripped to the core functionalities, further reducing the attack surface while increasing the system efficiency.

IV. Storage and Privacy

Choosing distributable nodes using a single schema based on typed hardware results in no further restrictions concerning the storage technology for the data itself, although system logs need to be stored in a different, more complex way (distributed over all nodes). This again raises new privacy issues.

A. Immutable Logging

The processes in such a consortial system might imply legal consequences, thus logging of interactions is crucial. There are certain requirements for such a logging:

- immutability of logs by any stakeholder and any malware
- availability of logging service
- persistency of log files over time
- scaleability

In general there are three possibilities to achieve such kind of logging:

- **Hardware solution (HS) [16]** like write-once read many (WORM) storage as provided by optical drives (physical WORM) or flash storage (physical but mostly achieved by device controller). Each Node logs its own interactions and keeps the log files.
- **A trustworthy 3rd party entity (TWE)** keeps logbooks about processes in the platform. All interactions are reported to the TWE and only those are trusted.
- **Distributed ledgers (DL) [15]** under consortial control shared across all stake holders and proveable by cryptographic means.

Those Solutions are shown in Figure 4.

Immutability: The first requirement can be matched by HSs as well as by software solutions like logbooks kept by a TWE or DL under public control.

Availability: As processes heavily rely on logging services, a centralized logbook keeping entity would become a single point of failure. HSs require specialized hardware that is more costly than software solutions that only require more standard hardware.

Persistency: If the legal frame requires storing of log data for an extended period, the storage has to be permanent and also redundancy is needed. HSs would require physical copies as offsite backup which would become costly. Logbooks by a TWE can be backed up with standard archiving software and

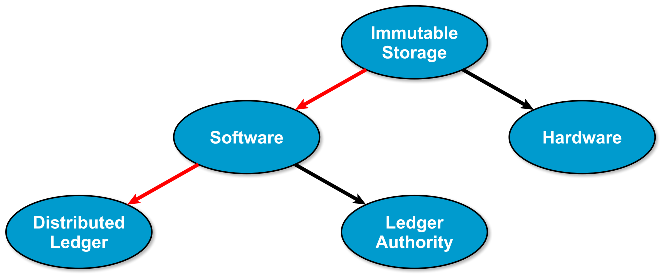

Fig. 4. Decision options for immutable logging

the archive can be stored offsite. DL are very durable by its intrinsic replication across nodes.

Acalability: Scalability is needed if the system is expected to grow over time but also to satisfy the availability requirements while the system is under heavy load. That can be achieved in both software and hardware solutions. But again, specialized hardware is more costly than software solutions that only requires more standard hardware.

Design-decision: immutable logging: We conclude that DL would fit our use-case best , as HS would be too costly and expensive , which might prevent stakeholders from joining. If a stakeholder decides to e.g. add a file to the system, he commits a file creation event to the ledger. Other stakeholders can now interact with that file, depending on their respective access rights. The logging is implemented by letting other stakeholders set a so-called *lock* on a committed file or data-object , assuming they were granted the appropriate rights from the file owner. Once a file is *locked* by another stakeholder, any changes to that file need to be communicated to and approved by every *lock*-holder. To guarantee compliance to these rules, DL Technology in the form of a blockchain is utilized, i.e. every event or process involving sensitive data of a stakeholder is written to the blockchain once consensus in the network has been established, thus garanteeing a once set lock cannot be removed without permissions from the *lock*-holders.

B. *Privacy vs verifiability*

The decision to utlize DL Technology to implement immutable logging, prompts questions on how the data is written to the ledger. The data added to the blockchain might contain sensitive information about the respective stakeholder that should only be visible to selected other stakeholders. The data needs to be encrypted and appropriate rights need to be set, on who can read or write on that data. To prevent any form of non-conform transactions and expose potentially compromised nodes, each new block (and therefore every new transaction) is verified by at least one other node in the network. To enable a node to verify a new proposed block, the node must, at least partially, be able to read the transactions in that block. This heavily compromises the privacy of each participant, especially considering the case that multiple manufacturers could trace all transactions of their respective competitors, given their node is involved in the verification process. The concept of transparency and privacy stand in direct contradiction to each other. This is even more important, given that most pariticpiants of the network are competing business entities. Any reveal of sensitive information could cause tremendous damage to the involved participant, thus rendering the utilization of such a system unappealing. This problem however, is well-known and existing blockchain technologies already implement processes to preserve varying levels of privacy.

Privacy in existing blockchains: Blockchains can be differentiated into permissioned and permissionless blockchains, depending on how new nodes are allowed to join the network. In permissioned blockchains the privacy issue can often be simplified, since the nodes are assumed to be more trustworthy and a less demanding verification process can be implemented with some form of centralized control. A fully decentralized permissionless blockchain, on the other hand, always requires the majority of the network to verify and approve each transaction. The following list gives a short overview of concepts deployed by blockchains with the goal to preserve privacy.

Design-decision: privacy vs verifiability:

- **Bitcoin [3]** utilizes multiple obfuscation schemes to hide the complete transaction history of an identity, e.g. a user can create multiple accounts (wallets) with unique private keys, thus allowing him to divide his transactions. Other approaches are “Bitcoin laundries”, that allow multiple users to mix their transactions, thus obfuscating transaction history.
- **Ethereum** is working on implementing zero-knowledge succinct non-interactive arguments of knowledge (zk-SNARKs) in their blockchain, e.g. ZoKrates [14]. Based on zero-knowledge proofs, this technology allows verification of computations, without knowing the actual content of the computation. The same technology is applied in textbfZcash, a blockchain with a focus on zkSNARKs to allow “shielded” transactions.
- **Monero** is implementing RingCT [1], a technology based on ring signatures. A ring signature proves, that the signers private key corresponds to a set of public keys, without revealing which public key specifically.
- **Hyperledger Fabric [12]** is a permissioned blockchain that solves the privacy issue by introducing channels, that encapsulate the transactions from all peers that are not involved in that transaction. Only involved participants can see and verify the content of the transactions.

Preserving privacy in public ledgers is a difficult task, since a majority of the network is required to verify all transactions to prevent attacks like double-spending. A general solution to the problem are zero-knowledge proofs, however, the current implementations are still in an early development stage and require a lot of resources to execute [2]. Ethereum is also working on implementing zkSNARKs in their blockchain, but due to the turing-complete virtual machine behind the blockchain, the process is even more resource intensive compared to using a zkSNARK for specific tasks, as is done in Zcash.

Preserving privacy on permissioned blockchains on the other

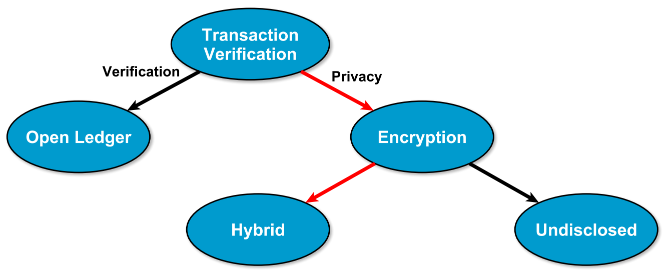

Fig. 5. Decision options for verification vs. privacy

hand can often be simplified, since the participating nodes are more trustworthy and known identities.

Preserving privacy is always based on encryption. Different technologies can be applied, resulting in different levels of transparency. Ring signatures can be implemented to obfuscate the identity in a group or simple symmetric encryption to enable only the key owners to see and verify the respective transactions. The latter could be used in a scheme where consensus is only established if every single owner of a group of symmetric key owners has signed a transaction encrypted with that key. This could be described as an undisclosed ledger. The ideal solution would be a low-cost and fast zero knowledge proof, enabling every node to verify every transaction without reading the actual content. In the case of immutable logging: if a party e.g. proposes a transaction to delete a file, the current verifier only needs to evaluate that no other party has an active *lock* on that file. The identities of those involved and the content of the file itself bear no relevance to the verification process. This variant can be described as a hybrid approach between verifiability and privacy.
The design-options are visualized in Figure 5, where our focus on the hybrid approach is highlighted. Implementing an efficient zero-knowledge proof would offer the most verifiability, given the requirements of immutable logging.

V. Conlusion

We showed that our consortial platform setting is a new problem and that there exists a sufficent solution for it. The design-decisions are cascading, as data locality implies different requirements for all following decisions.We described our solution that satisfies all given requirements, including data-locality, data-integration, a platform trust anchor and logging with an emphasis on privacy issues. Designing an IIS for a consortial platform is a relevant task for various other scenarios in the industry. Combining the varying interests of different stakeholders into a system that implements immutable logging of processes to be available as legal proof can be applied to areas in healthcare, insurance policies, and similar scenarios. The presented design-decisions only give an overview of what technologies we consider fitting for a consortial plattform. There is still a lot of room left for what specific implementations are the most efficient. An evaluation of already existing and comparable solutions is required, to derive what implementations can be applied to our scenario.

References

[1] Shen Noether, “Ring Signature Confidential Transactions for Monero,” *Cryptology ePrint Archive, Report 2015/1098*, 2015

[2] Cultivating Sapling: Faster zk-SNARKs, https://blog.z.cash/cultivating-sapling-faster-zksnarks, 2017

[3] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system,âĂİ http://bitcoin.org/bitcoin.pdf.

[4] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz and Y. Yarom, “Spectre Attacks: Exploiting Speculative Execution,” *ArXiv e-prints*, 2018

[5] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, S. Mangard, P. Kocher, D. Genkin, Y. Yarom and M. Hamburg, “Meltdown,” *ArXiv e-prints*, 2018

[6] CVE-2017-15361. Available from cvedetails, CVE-ID CVE-2017-15361., October 16 2017. URL: https://www.cvedetails.com/cve/CVE-2017-15361/.

[7] CVE-2017-5689. Available from MITRE, CVE-ID CVE-2017-5689., February 1 2017. URL: https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2017-5689.

[8] Ziawasch Abedjan, Lukasz Golab, and Felix Naumann. Profiling relational data: a survey. *The VLDB Journal*, 24(4):557–581, Aug 2015. URL: https://doi.org/10.1007/s00778-015-0389-y, doi:10.1007/s00778-015-0389-y.

[9] Daniel Abadi. Consistency tradeoffs in modern distributed database system design: Cap is only part of the story. *Computer*, 45(2):37–42, February 2012. URL: https://doi.org/10.1109/MC.2012.33, doi:10.1109/MC.2012.33.

[10] Corey Kallenberg, Xeno Kovah, John Butterworth, and Sam Cornwell. Extreme privilege escalation on windows 8/uefi systems.

[11] P. Koppe, B. Kollenda, M. Fyrbiak, C. Kison, R. Gawlik, C. Paar, and T. Holz. Reverse engineering x86 processor microcode. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1163–1180, Vancouver, BC, 2017. USENIX Association. URL: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/koppe.

[12] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, Chrysoula S., M. Vukolić, S. W. Cocco, and J. Yellick. Hyperledger fabric: A distributed operating system for permissioned blockchains. In *Proceedings of the Thirteenth EuroSys Conference*, EuroSys ’18, pages 30:1–30:15, New York, NY, USA, 2018. ACM. URL: http://doi.acm.org/10.1145/3190508.3190538, doi:10.1145/3190508.3190538.

[13] Patrick Ziegler and Klaus R. Dittrich. *Data Integration — Problems, Approaches, and Perspectives*, pages 39–58. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. URL: https://doi.org/10.1007/978-3-540-72677-7_3, doi:10.1007/978-3-540-72677-7_3.

[14] Jacob Eberhardt and Stefan Tai. Zokrates - scalable privacy-preserving off-chain computations. In *IEEE International Conference on Blockchain*. 2018. URL: http://www.ise.tu-berlin.de/fileadmin/fg308/publications/2018/2018_eberhardt_ZoKrates.pdf.

[15] L. Aniello, R. Baldoni, E. Gaetani, F. Lombardi, A. Margheri, and V. Sassone. A prototype evaluation of a tamper-resistant high performance blockchain-based transaction log for a distributed database. In *2017 13th European Dependable Computing Conference (EDCC)*, pages 151–154, Sept 2017. doi:10.1109/EDCC.2017.31.

[16] Ben Laurie. Network forensics. *Queue*, 2(4):50, 2004.

[17] Michael Stonebraker, Samuel Madden, Daniel J. Abadi, Stavros Harizopoulos, Nabil Hachem, and Pat Helland. The end of an architectural era: (it’s time for a complete rewrite). In *Proceedings of the 33rd International Conference on Very Large Data Bases*, VLDB ’07, pages 1150–1160. VLDB Endowment, 2007. URL: http://dl.acm.org/citation.cfm?id=1325851.1325981.

[18] F Thiel, M Esche, F.G. Toro, Daniel Peters, Alexander Oppermann, J Wetzlich, and M Dohlus. A digital quality infrastructure for europe: The european metrology cloud. 127:83–97, 12 2017.

[19] Big Data, Disruption and the 800-pound Gorilla in the Corner, Michael Stonebraker, http://web.stanford.edu/class/ee380/Abstracts/160601-slides.pdf