Decoupling Permission Management from Cryptography for Privacy-Preserving Systems

Ruben De Smet
Department of Engineering Technology (INDI)
Department of Electronics and Informatics (ETRO)
Vrije Universiteit Brussel
rubedesm@vub.be

Tom Godden
Department of Engineering Technology (INDI)
Vrije Universiteit Brussel
tom.godden@vub.be

Kris Steenhaut
Department of Engineering Technology (INDI),
Department of Electronics and Informatics (ETRO)
Vrije Universiteit Brussel
kris.steenhaut@vub.be

An Braeken
Department of Engineering Technology (INDI)
Vrije Universiteit Brussel
an.braeken@vub.be

***Abstract*—The principle of privacy-by-design more often than not requires the implementation of privacy-enhancing technologies (PETs). In turn, the implementation of PETs requires in-depth knowledge of cryptography engineering, which hinders a.o. rapid prototyping, modularity, and readability. This article proposes research on *loose coupling* of cryptographic primitives to business logic. As a running example, we look at the permission management of Signal's private groups, and propose an alternative design, keeping in mind extensibility, modularity, and improved transparency and auditability.**

I. Introduction

The integration of privacy-enhancing technologies (PETs) in new software systems requires a meticulous engineering effort, and often involves *cryptography engineering*. This is a paradoxical observation: software engineers are told not to "roll their own crypto", but at the same time, have to provide the most privacy-friendly version of their software. Commonly, no prior art exists regarding the specific cryptographic requirements for new applications, and new protocols need to be devised. These new cryptographic protocols get tailored to the specific use-case of the application. This introduces a tight coupling of business logic to cryptographic implementation details.

This article demonstrates how declarative programming languages may enable the decoupling of permission systems from their underlying cryptographic implementation. Additionally, we present some work in progress, and ways forward, to improve the expressiveness and potential of declarative programming frameworks for PET development.

Workshop on Innovation in Metadata Privacy: Analysis and Construction Techniques (IMPACT) 2025
28 February 2025, San Diego, CA, USA
ISBN 979-8-9919276-2-8
https://dx.doi.org/10.14722/impact.2025.23050
www.ndss-symposium.org

As a motivating example, this article studies the Signal private group management system [1]–[3]. First, we motivate the need for modularity in cryptographic design in section II-A, and argue our point with prior art in section II-B. A brief overview of the Signal private group system is then given in section II-C, to serve as the motivating example of this article. Our approach is then explained in section III. In the discussion, section IV, we emphasize current limitations, ongoing work, and necessary future work, after which follows a brief conclusion in section V.

II. Background

A. Challenges of privacy-preserving systems

Privacy-by-design and privacy-by-default mandate that information processing systems safeguard user data against exposure. However, implementing these principles in practice is fraught with challenges.

A key tension lies between the cryptographic skills required to guarantee security and the usability needed for widespread adoption. Cryptographic systems must meet stringent requirements for correctness, confidentiality, and integrity, as even minor flaws can have catastrophic consequences. At the same time, developers integrating these systems into real-world applications often lack expertise in cryptography, leading to a high risk of implementation errors or insecure shortcuts.

The trade-off between general-purpose and application-specific solutions further complicates the design of privacy-preserving systems. On the one hand, generic primitives like Transport Layer Security (TLS) provide broadly applicable cryptographic guarantees but fail to address nuanced application requirements, such as managing fine-grained access control, metadata privacy, or privacy-preserving analytics. On the other hand, tailored cryptographic solutions can precisely meet application needs but often entangle cryptographic implementation details with application logic, creating systems that are difficult to debug, extend, standardize, or audit.

This tight coupling of business logic and cryptographic implementation introduces significant barriers to modularity and maintainability. For example, when system requirements evolve, such as adapting a permission model to support new roles or relationships, developers often face the task of modifying cryptographic constructs alongside application logic. Moreover, debugging such systems is inherently challenging, as privacy-respecting data processing and storage limits the tools and techniques available for tracing issues.

Privacy-preserving system design demands a balance between correctness, usability, and modularity. Bridging this gap requires approaches that can decouple cryptographic implementation from application logic, enabling developers to focus on high-level design without sacrificing privacy guarantees.

B. Related work

The design of PETs has often encountered the challenge of balancing cryptographic correctness with usability [4]. Certain domains of PETs have made significant progress in developing high-level frameworks and abstractions that facilitate the adoption of complex cryptographic primitives while minimizing the risk of misuse. Below, we survey notable examples of such advancements and discuss their relevance to our goal of decoupling permission systems from cryptographic implementation.

High-level frameworks for private information retrieval (PIR) provide a compelling example of how cryptographic primitives can be abstracted for easier integration. For example, MuchPIR is a plugin-style PIR solution for the mainstream PostgreSQL relational database management system (RDBMS), demonstrating how modern databases can adopt privacy-preserving queries with minimal disruption to existing workflows [5]. These frameworks emphasize usability by abstracting away cryptographic details, a principle that aligns closely with the need for modularity in privacy-preserving systems.

Declarative frameworks have been proposed to simplify policy specification and enforcement in privacy and security systems. Recently, for instance, Farkas, Toldi, Péter, *et al.* developed a Prolog-based framework for declarative, zero-knowledge verifiable policies [6]. Building on the ideas behind Circuitree [7], their framework extends the capabilities of declarative systems by enabling the expression of policies that are both enforceable and provable using zero-knowledge proofs. Circuitree's foundational contribution lies in its ability to encode relations and permissions in a high-level declarative syntax while leveraging underlying cryptographic primitives. Notably, the work of Farkas, Toldi, Péter, *et al.* compares against the original version of Circuitree, which has meanwhile seen significant improvement and extensions, and a new comparison might be of interest.

In the domain of access control, several notable models emphasize modularity and interpretability. Masoumzadeh and Joshi introduced OSNAC, an ontology-based access control model tailored for social networking systems [8]. By leveraging ontologies, OSNAC enables dynamic and context-aware access control decisions, demonstrating the potential of declarative paradigms in simplifying complex policy definitions. Similarly, the work of Fong and Siahaan on relationship-based access control (ReBAC) proposes a declarative policy language that captures complex (data) relationship-based constraints while maintaining human readability [9]. Neither of these frameworks, however, are cryptographically enforced.

Despite their theoretical potential, advanced cryptographic techniques such as oblivious transfer (OT), attribute-based encryption (ABE), and fully homomorphic encryption (FHE) often remain relatively inaccessible to non-experts. These primitives are typically implemented in bespoke ways for specific applications, resulting in tightly coupled systems that are difficult to adapt or extend. While frameworks exist for some of these primitives [10]–[12], their integration into mainstream application development remains an ongoing challenge.

These prior works collectively highlight the potential benefits of decoupling business logic from cryptographic implementation. Frameworks like those for PIRs and declarative policy specification provide valuable insights and inspiration into how abstraction can enhance modularity, extensibility, and usability in PETs. Access control systems, in particular, demonstrate how declarative paradigms can be used to define and enforce complex policies.

This article extends these ideas by advocating for a declarative approach to structuring permissions management. By abstracting away cryptographic complexity, such an approach facilitates transparency, auditability, and ease of adoption, while maintaining the cryptographic correctness required for privacy-preserving applications.

C. Signal Private Groups

Modern instant messaging systems are expected to include *group chat* functionality, which requires managing the distribution of messages among group members. Two primary models exist for this purpose: *server fan-out* and *client fan-out*.

In the *server fan-out* model, the server receives a message from the sender and distributes copies of the message to each group member. While efficient, this model requires the server to maintain knowledge of the group composition, creating potential privacy risks.

Privacy-preserving instant messaging systems, such as Signal, adopt the *client fan-out* model to mitigate these concerns. In this approach, the sender distributes a copy of the message directly to each recipient, ensuring that the server does not know the group composition. However, this distributed model complicates the management of group membership, necessitating mechanisms to ensure consistency and integrity without violating privacy.

To address these challenges, Signal introduced its "private group system" [1] in 2019. This system relies on storing an encrypted group state on the server and using an anonymous credential system to enforce well-formed and consistent updates to the group state. For example, while only administrators should have the authority to add group members [2], any member should be allowed to leave the group independently.

Signal's anonymous credential system [3] is tailored specifically to its use case, enabling it to achieve strong guarantees of privacy and security. In contrast, this work proposes leveraging a more general zero-knowledge proof (ZKP) system to achieve similar guarantees while introducing greater expressivity and flexibility. A generalized ZKP system can enable more complex assertions of group state consistency and extend the capabilities of the private group system to support additional features.

Careful attention, however, must be given to the performance trade-offs of adopting a more general approach, ensuring that the enhanced expressivity does not come at the cost of usability, efficiency, or security.

III. Declaratively modeling permission systems

Permission systems define and enforce the rules governing who can access or modify specific resources. Declarative approaches to modeling such systems provide a high-level, expressive, and human-readable way to specify these rules, separating the business logic from implementation details. This separation stimulates modularity, makes the rules easier to verify, and enables reuse across different applications or domains.

Declarative frameworks use formal languages to define the relationships and constraints in the permission system. These languages are particularly well-suited to capture the dynamic and hierarchical nature of permission systems. For example, they can specify rules such as:

a) Membership Inheritance: If a user is an administrator of a parent group, they automatically gain permissions in subgroups.

b) Context-Specific Constraints: Certain permissions may only be valid under specific conditions, such as time of access or resource state.

c) Conflict Resolution: Declarative systems may explicitly encode policies for resolving conflicts, such as when two rules grant overlapping but contradictory permissions.

Declarative systems also enable auditability and interpretability. Since the rules are expressed in a high-level, human-readable form, they can be analyzed and debugged independently of the underlying cryptographic implementation.

A. Circuitree

Circuitree [7] is a framework for declaratively modeling witnesses, public inputs, and proof statements using Datalog. Its core strength lies in leveraging the simplicity and expressiveness of Datalog to encode the relationships and constraints that define the permission system. As such, Circuitree is a ZKP system for Datalog statements. While Circuitree initially lacked support for certain features required for more expressive permission systems, arithmetic constraints have since been implemented in the framework. This addition significantly expands Circuitree's applicability, particularly for modeling permission systems that rely on numerical constraints. Moreover, the system has undergone substantial performance improvements, although both these results have yet to be formally evaluated and published. These advancements improve Circuitree's efficiency while extending its flexibility, enabling a wider range of applications.

While Circuitree provides a powerful starting point, it lacks certain features needed for more expressive permission systems. Specifically, one more missing feature in Circuitree would be some form of support for *negation*. Declarative permission systems often need to encode rules such as "no unauthorized user can access a resource." Negation in Datalog is a complex topic, and care has to be taken to assure the prover always terminates. Extending Circuitree to support stratified negation [13, Section 15.2], without negation-as-failure, would enable such constraints to be modeled directly.

In contrast to the approach proposed by Farkas, Toldi, Péter, *et al.* [6], which introduces a Prolog-based extension for policy evaluation, our proposed extensions would maintain Circuitree's efficient architecture. This should allow for efficient proof generation and verification while extending the system's flexibility to capture a broader range of permission models.

B. Modeling Signal private groups in Circuitree

Signal's private group system addresses the challenges of managing group membership in privacy-preserving messaging by storing the encrypted group state on a server. Modifications to the group (e.g. adding or removing members) must be performed in a way that ensures the integrity of the group structure and the correctness of updates, without revealing sensitive information, if any information at all, to the server.

To achieve this, Signal relies on an anonymous credential system [3] tailored to its specific use-case. Circuitree, as a more general-purpose ZKP system, can be used to achieve similar goals. Circuitree enables the encoding of rules and constraints declaratively, making it easier to extend the permission system's feature set.

Consider the scenario where a user wishes to add a new member to the group. The group state stored on the server includes the current members and their roles, and a set of rules governs how modifications can be made. The following example demonstrates how to model this process in Circuitree. The server stores the following encrypted group state:

```
% Old group structure
member(alice, prevGroup, member).
member(bob, prevGroup, admin).

% We can encode higher-order rights!
allow(add, alice, prevGroup).
```

The challenge is to prove the updated group structure. For example, we may need to demonstrate that Alice and Bob remain administrators in the new group, and that Carol is now a member. Additionally, we need to ensure that no access rights have been violated and that there is always at least one administrator in the group. The new group structure could look like this:

```
% New group structure
?- member(alice, newGroup, admin).
```

```
?- member(bob, newGroup, admin).
?- member(carol, newGroup, member).

% Access rights in the new group
?- allow(add, alice, prevGroup).

% Prove that none of the rights were
↪  violated
?- not violation.
% At least one admin remains
?- member(_, newGroup, admin).
```

The new group structure is formatted as a Datalog *query*, since it is the statement that we want to prove with Circuitree.

The system can validate these relationships by specifying a changeset, represented as a witness by the prover. For example, we could specify the change of Alice adding Carol as a member with the following:

```
% Declare the change
action(add, carol, member, alice).
```

To compute the new group structure based on the changeset, we would write Datalog rules. These rules are fixed on both the server, and the prover. The program below demonstrates this approach:

```
% Additions imply membership in the new
↪  group
member(X, newGroup, Role)
  :- action(add, X, Role, Member).
% A member in the old group also exists in
↪  the new group
member(X, newGroup, Role)
  :- not action(delete, X, Role, Member),
     member(X, prevGroup, Role).

violation
  :- action(Action, _, Role, Member),
     not allow(Action, Member, prevGroup).

% Define default rights based
allow(add, Member, Group)
  :- member(Member, Group, admin).
allow(delete, Member, Group)
  :- member(Member, Group, admin).
```

At first glance, this Datalog program seems like an elegant way to capture the desired group dynamics. However, this program relies on negation, which is not currently supported by Circuitree.

In theory, a semipositive Datalog$^\neg$ formulation should be sufficient for such a program. In practice, implementing *stratified negation* [13, Section 15.2] would likely be necessary. Currently, Circuitree's implementation does not support stratified negation, which limits its ability to directly represent this permission system.

C. Transparency and usability

Declarative permission models are inherently human-readable and interpretable, which improves transparency in privacy-preserving systems. Stakeholders, including developers, administrators, and auditors, can easily review the rules and ensure they align with organizational policies or regulatory requirements.

For example, the following rule makes it clear that only admins can add members to a group:

```
allow(add, Member, Group) :-
    member(Member, Group, admin).
```

In contrast to imperative implementations that obscure logic within code, or within the cryptography itself, declarative rules explicitly state the relationships and constraints, enhancing trust and accountability.

A key advantage of using declarative frameworks is the lower barrier to entry for non-experts in cryptography or privacy-preserving system design. Circuitree's high-level abstraction reduces the need for deep technical expertise, enabling broader participation in developing secure systems.

Consider the following simple rule for validating actions:

```
violation :-
    action(Action, _, _, Initiator),
    not allow(Action, Initiator, Group).
```

This rule encodes misuse resistance by automatically flagging any unauthorized action, reducing the likelihood of human error. Non-expert developers can confidently build systems by relying on well-defined, declarative primitives, stimulating safer and more widespread adoption of privacy-preserving technologies.

D. Integration into Signal

Up until now, this article considers Circuitree as an isolated building block for modeling permissions. In theory, it is possible to use Circuitree to prove statements about the pre-existing encrypted group state. By intelligently picking the elliptic curve used for the Circuitree ZKP, a backwards-compatible design, reusing Signal's current ElGamal-encrypted group state could be rather efficient.

Alternatively, the server could keep a cryptographic *commitment* to the group state on the Signal servers, instead of the full (encrypted) state. This approach keeps the group state information-theoretically secure. The actual new group state would need to be gossiped around the group members, but this infrastructure is already in place.

Either approach requires some bespoke ZKP design to tie the group state to Circuitree, but neither design fundamentally changes the current protocol. In both cases, a so-called "gadget" needs to be designed to transform the hidden group state into Circuitree datalog facts.

IV. Discussion

As demonstrated by the Signal example, the decoupling of underlying cryptographic systems renders the system as a whole more modular, transparent, and usable to non-cryptographers. However, it is also important to quantify the cost.

Circuitree has seen substantial performance improvements, even with new features such as arithmetic constraints included.

Whereas these optimizations remain unpublished, initial results suggest that the framework remains highly efficient, even improving its suitability for real-world applications. A detailed performance comparison with frameworks such as [6] and imperative ZKP systems like Circom [14] would further clarify these trade-offs.

The approach presented in this article allows for permission models to be developed and analyzed independently of the cryptographic machinery, significantly lowering barriers to entry for developers of privacy-preserving systems. However, this modularity does not come without trade-offs, and it is crucial to carefully quantify the associated costs and limitations, and to further study the potential security impact of the approach.

A. Performance trade-offs

One of the primary challenges lies in the computational cost of extending declarative frameworks such as Circuitree. While Datalog is inherently efficient for a variety of declarative queries, the addition of features like negation may introduce performance bottlenecks. Furthermore, practical applications often require scalability to accommodate large group sizes, such as those in messaging systems like Signal. The impact of these extensions on scalability and proof efficiency remains an open question, requiring evaluation in future work.

B. Interfacing with encrypted states

Another key challenge is the integration of declarative systems with encrypted stored states. In the case of Signal, for example, group membership and permissions are stored on the server in an encrypted format. To leverage Circuitree, this encrypted state must be transformed into a format that can be processed by the ZKP system, which involves implementing a proof-of-decryption. These gadgets introduce an additional layer of complexity, in terms of development effort, coupling and runtime performance. Developing a set of generic "input gadgets" tailored to Circuitree would be an interesting research direction.

C. Applicability to broader systems

Although this article focuses on modeling Signal's private group system, the proposed approach is general enough to be applied to a wide range of privacy-preserving systems. Future work should explore how declarative ZKP frameworks can be adapted to other domains, such as access control in online social networks, blockchain smart contracts, and privacy-preserving machine learning. Comparisons with existing systems like the Prolog-based approach of Farkas, Toldi, Péter, *et al.* [6] or Signal's native implementation will also provide valuable insights into the practical trade-offs of declarative versus specialized solutions.

D. Comparison to imperative frameworks

In addition to declarative frameworks, imperative ZKP frameworks such as Circom [15] or Cairo [16] offer powerful tools for implementing privacy-preserving systems. These frameworks provide a Turing-complete approach to designing ZKPs, allowing developers to transform imperative, C-like code to a ZKP. This imperative paradigm may provide greater familiarity to the developer, but may come at a significant performance cost.

Future work should include a direct comparison of declarative and imperative ZKP frameworks, evaluating their respective strengths in terms of performance, developer usability, and adaptability to different domains.

E. Future directions for Circuitree

Several extensions to Circuitree would enhance its utility as a privacy-preserving permission framework:

- Adding support for stratified negation, with a focus on maintaining proof efficiency.
- Investigating the feasibility of incorporating systems beyond ZKPs, such as oblivious transfer or secure multi-party computation, into the declarative framework.
- Developing practical performance benchmarks against real-world systems, including those with bespoke cryptographic implementations.
- Improving the interoperability of declarative frameworks with encrypted states through reusable input gadgets, streamlining integration with existing cryptographic systems.
- Support for open queries (e.g. `?- member(_, newGroup, _)`)

F. Limitations

The modularity and abstraction offered by declarative permission modeling come with limitations, primarily centered on performance and integration challenges. Additionally, while declarative frameworks excel at human-readable transparency, they may still require a significant learning curve for developers unfamiliar with logical programming paradigms such as Datalog. Further research is needed to ensure these systems are not only powerful but also intuitive and easy to adopt.

V. Conclusion

We present a case for the use of declarative frameworks in the modeling of privacy-preserving permission systems, with a specific focus on extending the capabilities of Circuitree. By decoupling permission management from cryptographic implementation details, declarative systems provide a pathway toward modular, transparent, and usable privacy-preserving systems.

Using the Signal private group management system as a motivating example, we demonstrated how declarative approaches can encode and enforce complex permission models. We identified key enhancements required to extend Circuitree's applicability, including support for negation, and efficient mechanisms for bridging encrypted states with declarative proofs. While these enhancements promise greater expressivity and usability, the performance trade-offs remain an open question.

Compared to imperative ZKP frameworks like Circom, declarative systems offer significant advantages in terms of

usability, modularity, and interpretability, making them a compelling choice for rapidly evolving systems. Future work should explore the practical implications of these trade-offs, through both theoretical analysis and empirical benchmarking, and investigate the potential for declarative frameworks to generalize beyond ZKPs into broader domains of privacy-preserving computation.

We believe declarative systems represent a promising direction for the development of next-generation privacy-preserving technologies. Their ability to reduce implementation complexity, facilitate iterative development, and uphold cryptographic guarantees makes them valuable tools for advancing the practical adoption of privacy-enhancing technologies.

Acknowledgment

This work is being performed within the framework of Innoviris 2024-RPF Sufficiency and Data Minimization (SDM) and Cybersecurity Research Program Flanders - second cycle (VOEWICS02).

References

[1] J. O'Leary. "Technology Preview: Signal Private Group System," Signal Messenger. (Dec. 9, 2019), [Online]. Available: https://signal.org/blog/signal-private-group-system/ (visited on 04/27/2021).

[2] R. Sarafa. "New Features Coming to Signal Groups," Signal Messenger. (Oct. 14, 2020), [Online]. Available: https : / / signal . org / blog / new - groups/ (visited on 04/27/2021).

[3] M. Chase, T. Perrin, and G. Zaverucha, "The Signal Private Group System and Anonymous Credentials Supporting Efficient Verifiable Encryption," in *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*, Virtual Event USA: ACM, Oct. 30, 2020, pp. 1445–1459, ISBN: 978-1-4503-7089-9. DOI: 10.1145/3372297.3417887. [Online]. Available: https://dl.acm.org/doi/10.1145/3372297.3417887.

[4] A. Whitten and J. D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0.," in *USENIX Security Symposium*, vol. 348, 1999, pp. 169–184. [Online]. Available: https : / / www . usenix . org / legacy / events / sec99 / full_papers / whitten / whitten . ps (visited on 06/26/2024).

[5] E. Liones, *ReverseControl/MuchPIR*, Nov. 8, 2024. [Online]. Available: https://github.com/ReverseControl/MuchPIR (visited on 01/15/2025).

[6] M. Farkas, B. Á. Toldi, B. Z. Péter, and I. Kocsis, "A Prolog-based Approach to Self-Evaluated, Declarative and Zero-Knowledge Verifiable Policies," in *2024 32nd International Conference on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Krakow, Poland: IEEE, Oct. 21, 2024, pp. 1–6, ISBN: 9798331531300. DOI: 10.1109/MASCOTS64422.2024.10786564. [Online]. Available: https://ieeexplore.ieee.org/document/10786564/ (visited on 01/13/2025).

[7] T. Godden, R. De Smet, C. Debruyne, T. Vandervelden, K. Steenhaut, and A. Braeken, "Circuitree: A Datalog Reasoner in Zero-Knowledge," *IEEE Access*, vol. 10, pp. 21 384–21 396, 2022, ISSN: 2169-3536. DOI: 10.1109/ACCESS.2022.3153366.

[8] A. Masoumzadeh and J. Joshi, "OSNAC: An Ontology-based Access Control Model for Social Networking Systems," in *2010 IEEE Second International Conference on Social Computing*, Minneapolis, MN, USA: IEEE, Aug. 2010, pp. 751–759, ISBN: 978-1-4244-8439-3. DOI: 10.1109/SocialCom.2010.116. [Online]. Available: http : / / ieeexplore . ieee . org / document / 5591484/ (visited on 04/16/2020).

[9] P. W. Fong and I. Siahaan, "Relationship-based access control policies and their policy languages," in *Proceedings of the 16th ACM Symposium on Access Control Models and Technologies*, ser. SACMAT '11, New York, NY, USA: Association for Computing Machinery, Jun. 15, 2011, pp. 51–60, ISBN: 978-1-4503-0688-1. DOI: 10.1145/1998441.1998450. [Online]. Available: https://doi.org/10.1145/1998441.1998450 (visited on 09/03/2020).

[10] FENTEC, "FENTEC Project: Increasing Trustworthiness of ICT solutions developing Functional Encryption," Madrid, Spain, Press release, Mar. 31, 2018, p. 1. [Online]. Available: https://fentec.eu/sites/default/files/fentec/public/content-files/article/FENTEC_PR_1.pdf (visited on 01/17/2025).

[11] M. Abdalla, D. Catalano, R. Gay, and B. Ursu, "Inner-Product Functional Encryption with Fine-Grained Access Control," in *Advances in Cryptology – ASIACRYPT 2020*, S. Moriai and H. Wang, Eds., Cham: Springer International Publishing, 2020, pp. 467–497, ISBN: 978-3-030-64840-4. DOI: 10.1007/978-3-030-64840-4_16.

[12] M. Tilen, J. Hartman, and G. Thibs, *Fentec-project/CiFEr*, fentec-project, 2021. [Online]. Available: https://github.com/fentec-project/CiFEr (visited on 01/17/2025).

[13] S. Abiteboul, R. Hull, and V. Vianu, *Foundations of Databases*. Reading, Mass: Addison-Wesley, 1995, 685 pp., ISBN: 978-0-201-53771-0.

[14] H. García Navarro, "Design and implementation of the Circom 1.0 compiler," M.S. thesis, Universidad Complutense de Madrid, Madrid, Spain, 2020, 57 pp.

[15] M. Bellés-Muñoz, M. Isabel, J. L. Muñoz-Tapia, A. Rubio, and J. Baylina, "Circom: A Circuit Description Language for Building Zero-Knowledge Applications," *IEEE Transactions on Dependable and Secure Computing*, vol. 20, no. 6, pp. 4733–4751, Nov. 2023, ISSN: 1941-0018. DOI: 10.1109/TDSC.2022.3232813. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10002421 (visited on 01/17/2025).

[16] L. Goldberg, S. Papini, and M. Riabzev, "Cairo – a Turing-complete STARK-friendly CPU architecture," 1063, 2021. [Online]. Available: https://eprint.iacr.org/2021/1063 (visited on 09/02/2021).