A-SIT

Secure Information Technology Center – Austria

Privacy enhancements for flexible access control policies

Privacy enhancements for flexible access control policies

Autor:
Stefan More
smore@iaik.tugraz.at

14. April 2023

Abstract:

Policies for access control systems enable the decoupling of the control of data and resource access from the business logic. In addition to the resulting flexibility, another advantage is that these policies can be created and maintained by domain experts without specific IT expertise. These policies typically define who has what permission to which resources. In addition to authorization, such policies can also define how users are authenticated. This can involve using existing systems such as eIDAS or enterprise systems, as well as newer models like self-sovereign identity (SSI).

One challenge in designing access control systems is preserving the privacy of users. To evaluate an access request against a policy, users often have to provide various (often personally identifiable) data. When these data are stored in digital documents, more data is often transmitted than necessary. To address this challenge, privacy-preserving technologies can be used. However, integrating these technologies into access control systems, especially existing ones, is often not straightforward.

Therefore, the goal of this project is to research the integration of privacy-preserving technologies into a policy system to further enhance user privacy.

Contents

1. Introduction

Access control policies are an important aspect of modern computing systems. They help to ensure that only authorized users can access certain resources or perform specific actions. Systems then enforce the policies by granting or denying access requests based on user identity, attributes, or other factors.
Typically, a user presents a credential to the verifier. An example credential is shown in Figure 2.1. This credential was previously issued by a issuer by calculating the commitment on the attributes (commonly done using a hash function) and signing the resulting digest. A conceptual overview of this process is shown in Figure 2.2. The verifier then verifies this signature. Doing so provides authenticity and integrity protection of the attributes.
However, traditional access control mechanisms are not sufficient for protecting sensitive and personal information. For example, in a healthcare setting, a patient's medical record may contain highly sensitive information that should only be accessed selectively on a need-to-know basis. Another example is the classical case of a student who wants to visit a bar which requires an age proof. While the bar's bouncer is only interested in the student's age,[1] presenting a government-issued ID document would reveal other attributes as well. In such scenarios, a traditional access control policy system may not be sufficient, as it may reveal data to individuals who do not have a legitimate need for the information.
To address this challenge, privacy-enhanced access control policies incorporate privacy considerations into the access control decision-making process. These policies aim to protect the privacy of sensitive information by limiting the amount of information that is disclosed during the access control process.
Privacy-enhanced access control policies can be implemented using various privacy-preserving authentication protocols such as zero-knowledge proofs. But, the construct of a specific protocol has enormous consequences on the achieved privacy properties.

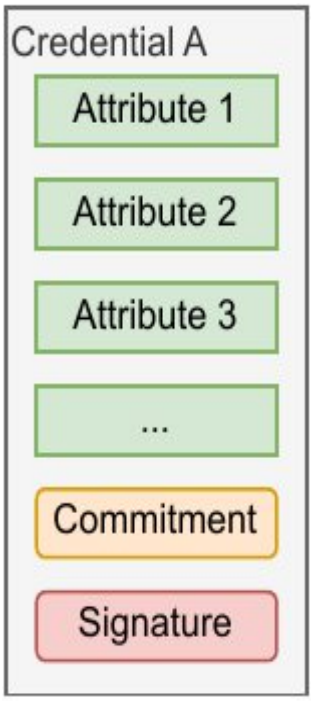

Figure 2.1: Example credential

Overall, privacy-enhanced access control policies offer a more granular and fine-grained approach to access control that can provide stronger privacy guarantees for sensitive information. By incorporating privacy considerations into the access control decision-making process, these policies can help ensure that only authorized individuals can access sensitive information while minimizing the risk of privacy violations or unauthorized disclosures.

[1] To be more concrete, the bar's bouncer is only interested whether the student's age is above a certain threshold – a so called predicate on an attribute.

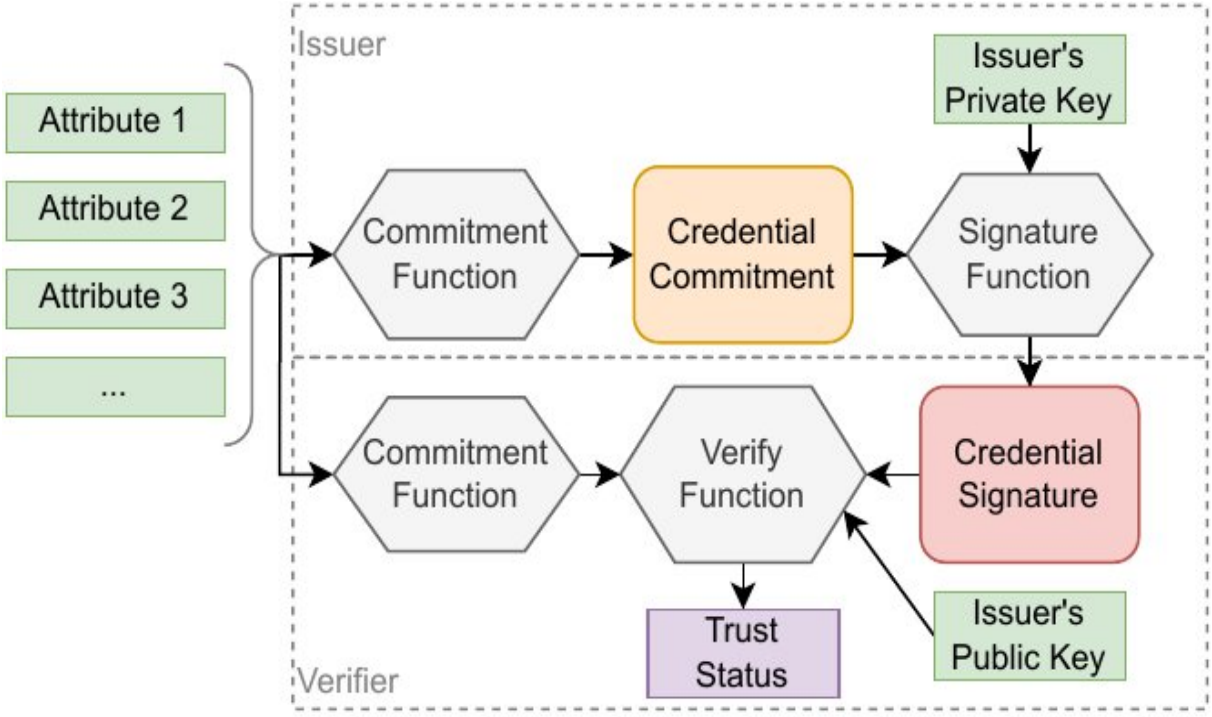

Figure 2.2: Typical credential issuing and verification process

1.1. Privacy Properties

Confidentiality refers to the protection of sensitive information from unauthorized disclosure or access. Confidentiality ensures that only authorized individuals can access the information and that it is not disclosed to others without the owner's consent.

Anonymity refers to the ability to use a service or access a resource without revealing one's identity or other personal information. Anonymity is often used to protect the privacy of users who wish to remain anonymous while accessing a service or resource.

Unlinkability refers to the ability to prevent the association of multiple transactions or interactions with the same user or entity (Pfitzmann, Andreas and Hansen, Marit, 2010). Unlinkability ensures that users can use a service or access a resource without creating a trail of their activities that can be linked together to reveal their identity or personal information. For *multi-show unlinkability*, it should be impossible for any party to tell whether two accesses to the same resource involved the same electronic credential or two different ones.

1.2. Zero-knowledge proofs and SNARKs

One of the key technologies used in privacy-enhanced access control systems are **zero-knowledge proofs** (Manuel Blum and Paul Feldman and Silvio Micali, 1988). A zero-knowledge proof is a cryptographic protocol that allows one party (the prover) to prove to another party (the verifier) that they know a particular piece of information without revealing that information itself. This means that a user can prove their eligibility for access to a resource or service without revealing any personal information about themselves.

Another technology used in privacy-enhanced access control policy systems are **SNARKs**, which stands for "Succinct Non-Interactive Argument of Knowledge" (Nir Bitansky and Ran Canetti and Alessandro Chiesa and Eran Tromer, 2012). A SNARK is a type of zero-knowledge proof that allows for the verification of a computation without having to return the computation itself. This means that a user can prove their eligibility for access to a resource or service without having to reveal any personal information, and without requiring the verifier to run the policy execution again.

SNARKs are the ideal building block for a policy system which wants to achieve both *privacy* and *expressiveness*.

1.3. Attribute-based Credentials

Attribute-based (anonymous) credentials (ABCs) are another technology used in privacy-enhanced access control policy systems (Ahmad Sabouri and Ioannis Krontiris and Kai Rannenberg, 2012). An attribute-based credential is a digital credential that contains attributes about the user, such as their age, gender, or location. The user can present their attribute-based credential to a verifier to prove their eligibility for access, without revealing any other attributes. In turn, the credentials provide authenticity of those attributes to the verifier. Attribute-based credentials allow users to prove their eligibility for access while still maintaining their privacy. Attribute-based credentials use a combination of cryptographic techniques to achieve privacy and security properties.

2. Baseline

The zkTPL system represents the baseline for our work. It introduces an approach that allows to extend any (policy-based) access control system which privacy features. This enables expressive policies that can be used in different contexts and that support generic use cases.

2.1. zkTPL

The zkTPL project presents a generic design for supporting privacy-preserving technologies in policy languages. This design prevents unnecessary disclosure of sensitive information while still allowing the formulation of expressive rules for access control. For that, zkTPL makes use of SNARKs. On a high level, the idea is that the verifier formulates an access policy and sends it to the user. The user then uses their credentials to proof that they fulfill the policy, and only send this proof to the verifier. Doing so, the verifier can be sure that the user fulfills the access policy, but learns no other information about the user's attributes. The goal is to do this in a fully automated way. The second goal is to support a policy language that can represent any access control policy, not limiting the access rules to simple statements.
Using zkTPL, the policy's author uses an existing policy system to codify which statements the user should prove and which information needs to be revealed to receive access. The author then uses the policy compiler to derive a presentation request they provide to users. The presentation request informs the user about the attributes they need to reveal and statements they need to prove. That enables users to only present the required attributes and statements, and hide the rest of the credential data. Using the compiled presentation request, all of that happens automatically. An advantage of this approach is that the same access control system can be used in various use cases without the need to re-compile the system or even modify source-code.

2.2. Establishing Trust

To establish trust in a presentation, the verifier checks if the proof was generated using a credential issued by a trustworthy issuer. The trust rules in the policy specify which issuers are trustworthy for which type of credentials. This can be done by providing a list of trusted issuers or by defining a trusted trust scheme. Depending on the defined trust scheme, the policy verifier automatically retrieves trust status information about the credential issuers from online registries. This process ensures that public and private attributes can be trusted, and therefore, all NIZK statements on these attributes are trustworthy. After the NIZK verifier and the policy verifier conclude that the presentation token is trustworthy and fulfills the user's policy, the service provider grants the user access to the service.
On a technical level, trust in a presentation is established through a trust chain from the issuer along the signature to the commitment, which is in turn linked with the proof and consequently the attributes. The proof sent to the verifier contains a (revealed) statement about the commitment-digest of the original credential. This statement proofs that all attributes (revealed and not revealed) and predicates are part of the pre-image of the commitment. In other words, if all attributes are encoded and their commitment computed, the result is the same digest than the one revealed. Since the issuer's signature was issued on this commitment, it cryptographically links the proof to the issuer. Thus, the presentation consists of the proof (including revealed attributes), the revealed commitment (as part of the proof), and the issuer's signature on the commitment.

2.3. Linkability

A disadvantage of this approach is that the signature and commitment are always sent to the verifier. These values are always the same for presentations derived from the same credential (see Figure 3.3.1) – even if different attributes are revealed, or no attributes are revealed at all. This violates the *unlinkability* property.

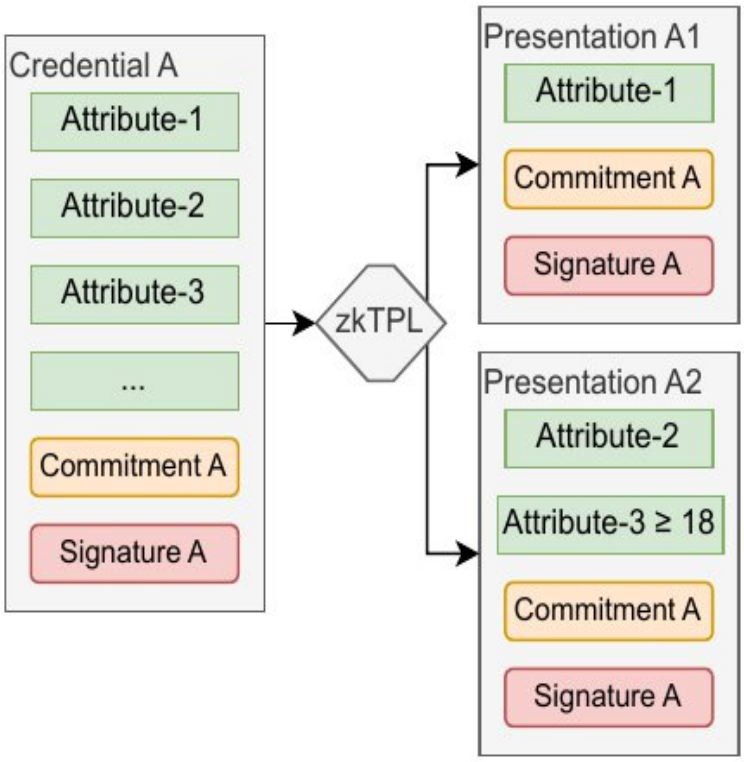

Figure 3.3.1: Example process of using zkTPL to derive two presentations from one credential

As a result, a verifier can link multiple showings of the same user, which is a problem for the user's privacy. Additionally, if the user presented different proofs to different verifiers, and those verifiers collude, then they can also link the presentations of the user. This is especially a problem if the user revealed different attributes in different showings, which can then be combined to one record – effectively reconstructing the full credential.

3. Towards unlinkability

The goal is that a user can take one credential and generate two presentations that have no information in common with the credential or each other.[2] Doing so prevents a verifier can link the showings of the two presentations. For the type of commit-then-sign credential introduced in Figure 2.2, this means that a user cannot simply send the commitment and signature to the verifier. But, the verifier requires both pieces of information to verify the authenticity of the credential. This is because the verifier uses the signature and it's issuer in a trust policy to access whether it considers the issuer trustworthy. In the zkTPL approach, this is done by linking the proof to the commitment and then using the signature on the commitment to establish trust in the attributes.

[2] An exception to this is when the user voluntarily gives up unlinkability and revealed enough attributes to allow linking.

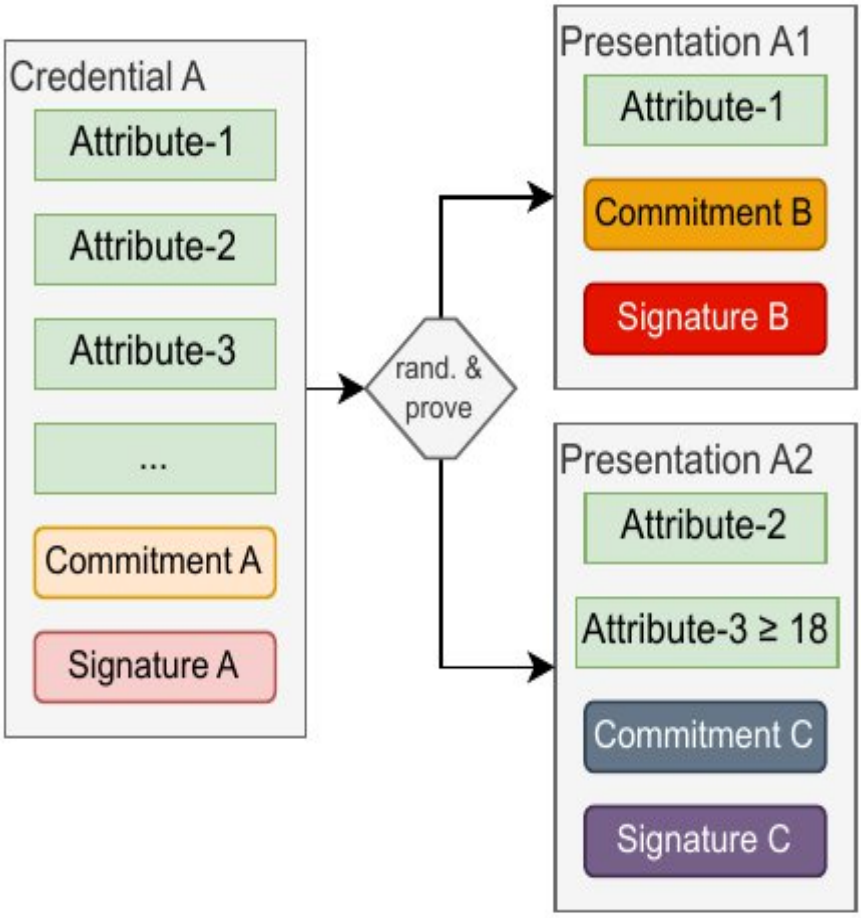

Figure 4.1: Example process of using commitment and signature randomization to derive to unlinkable presentations from one credential

A solution to this is to *randomize* the commitment and the signature.

The result is a process as shown in Figure 4.1: both presentations contain a unique commitment value and a unique signature value. If a suitable scheme is used for both values, and this randomization is done in the correct way, the signature is still valid for the data in the proof (see Figure 4.2).

The verifier then proceeds to authenticate the (randomized) signature and establishes trust in its issuer in the same way as in Section 3. As a result, the verifier can ensure that the presentation was constructed using a valid credential issued by a trustworthy issuer.

One example for such a scheme combination are Dual commitments (Torben P. Pedersen, 1991) together with Pointcheval and Sanders signatures (David Pointcheval and Olivier Sanders, 2016). Another example are Structure-Preserving Signatures on Equivalence Classes introduced by Georg Fuchsbauer and Christian Hanser and Daniel Slamanig and their corresponding set commitments (Georg Fuchsbauer and Christian Hanser and Daniel Slamanig, 2019). An alternative solution is to not reveal the commitment and signature values at all.

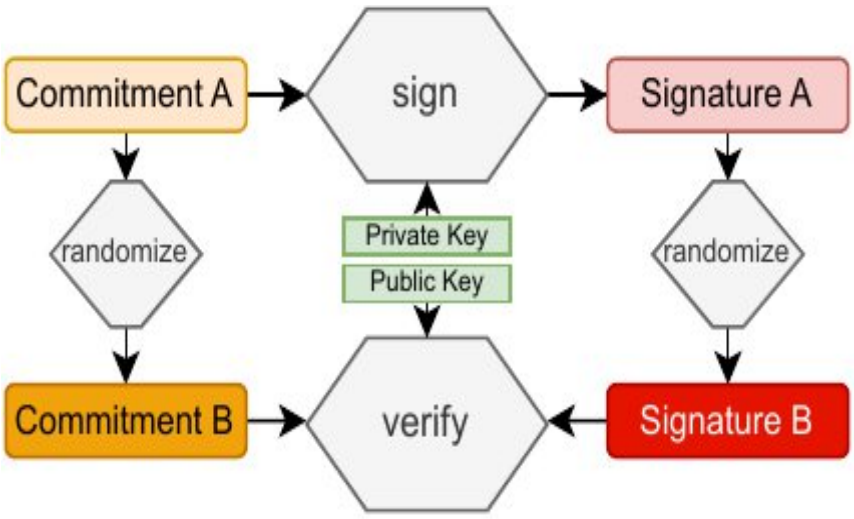

Figure 4.2: Conceptual visualization of randomizing a signature together with its message (the commitment)

3.1. Policy Integration

From the user and verifier perspective, the process flow does not change in comparison to zkTPL. First, the verifer creates a policy, formulating and encoding trust- and access-rules. It then uses this policy to compile a presentation request. This presentation request is sent to the user when they want access to some resource. In turn, the user uses the request to derive a presentation from their credential, revealing only the required attributes. On a technical level, both the presentation request and the presentation are different. But all involved steps are still automatically performed using the existing policy and credentials. Additionally, the user's system performs the necessary steps to prevent linkability, thus preserving the user's privacy.
Conceptually, there are two approaches to achieve unlinkability. We give an overview of these two approaches in Figure 4.3.1.

3.2. Approach 1: Using randomization

To avoid that two presentations contain identifiable (or linkable) information, one needs to randomize all linkable information. In our case, the information causing the (undesired) linkability between the presentations are the commitment (e.g., hash digest on the attributes) and the signature (on the commitment). The trick is to randomize both values in a clever way: First, the randomized commitment needs to be a commitment to the same data, since they are used to protect the data's integrity and bind it to the signature[3] Next, the same is true for the randomized signature, as it is later used to verify the authenticity of the commitment. Finally, both values are in relationship to each other, and they are only meaningful if this relation stays intact. It makes no sense to randomize the commitment in a different way than the signature, which would render the signature invalid. Thus, either both values need to be randomized in a compatible way. Or, there is a proof that shows this relationship. Depending on the utilized schemes, the user needs to proof the connection between the original commitment and randomized commitment (see Figure 4.3.1). We visualize this process in Figure 4.2.

3.3. Approach 2: Using a proof

An alternative approach to this is to create the link between attributes, commitment, and signatures directly in the proof.
This has the advantage that the verifier only needs to verify the proof, check if it was performed using the correct public key, and establish trust in this public key. On the other hand, a limitation is that proving the verification of a signature is very slow, or – depending on the used signature scheme – even impossible. But, if done right, the verifier can check the authenticity of the presentation and that it fulfills the policy, all while only learning who signed the original credential.[4]

---

[3] On a technical level, both commitments *open* to the same message.

[4] Hiding the issuer of a credential is also possible by extending the scheme further (Jan Bobolz and Fabian Eidens and Stephan Krenn and Sebastian Ramacher and Kai Samelin, 2021).

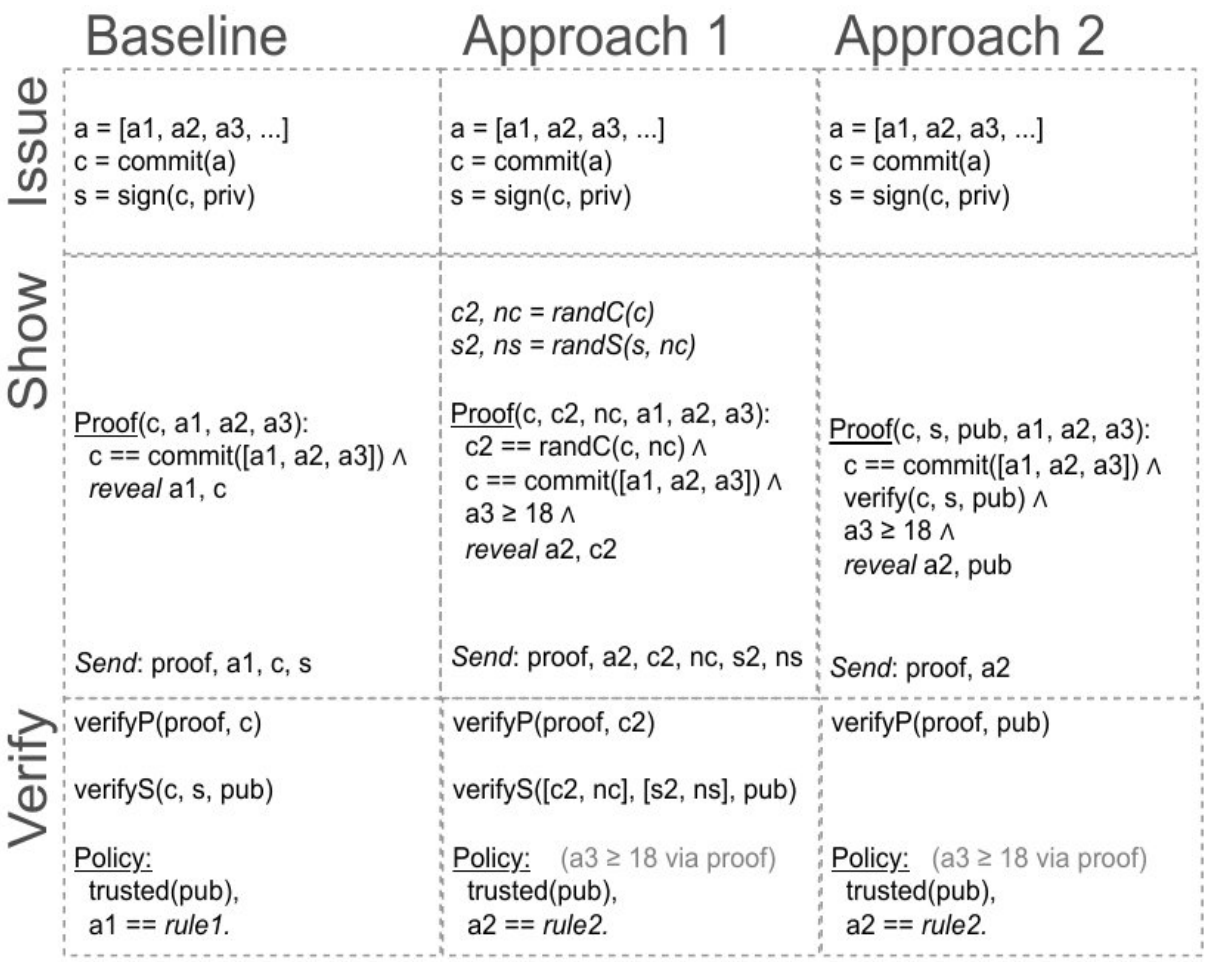

Figure 4.3.1: High-level comparison of two approaches with our baseline

4. Conclusions

In this report, we discuss several strategies on how to extend an access control system with privacy features. With the presented strategies, it is further possible to achieve this goal while also ensuring the unlinkability of multiple credentials or credential showings. This ensures the user's privacy while satisfying the verifier's policy requirements.

5. References

Ahmad Sabouri and Ioannis Krontiris and Kai Rannenberg (2012).
Attribute-Based Credentials for Trust (ABC4Trust), Springer.

David Pointcheval and Olivier Sanders (2016).
Short Randomizable Signatures, Springer.

Georg Fuchsbauer and Christian Hanser and Daniel Slamanig (2019).
Structure-Preserving Signatures on Equivalence Classes and Constant-Size Anonymous Credentials, J. Cryptol.

Jan Bobolz and Fabian Eidens and Stephan Krenn and Sebastian Ramacher and Kai Samelin (2021).
Issuer-Hiding Attribute-Based Credentials, Springer.

Manuel Blum and Paul Feldman and Silvio Micali (1988).
Non-Interactive Zero-Knowledge and Its Applications (Extended Abstract), {ACM}.

Nir Bitansky and Ran Canetti and Alessandro Chiesa and Eran Tromer (2012).
From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again, {ACM}.

Pfitzmann, Andreas and Hansen, Marit (2010).
A terminology for talking about privacy by data minimization: Anonymity, Unlinkability, Undetectability, Unobservability, Pseudonymity, and Identity Management.

Torben P. Pedersen (1991).
Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing, Springer.

6. Appendix: zkTPL demo run

The separate “abcTPL_appendix.pdf” document describes the project’s baseline by walking through an example.

In this demo[5], we use the SNARK-based approach presented above. The sequence of commands and the exchanged data does not significantly change when applying other cryptographic techniques.

[5] https://technology.a-sit.at/privatheitsbewahrende-trust-policies/