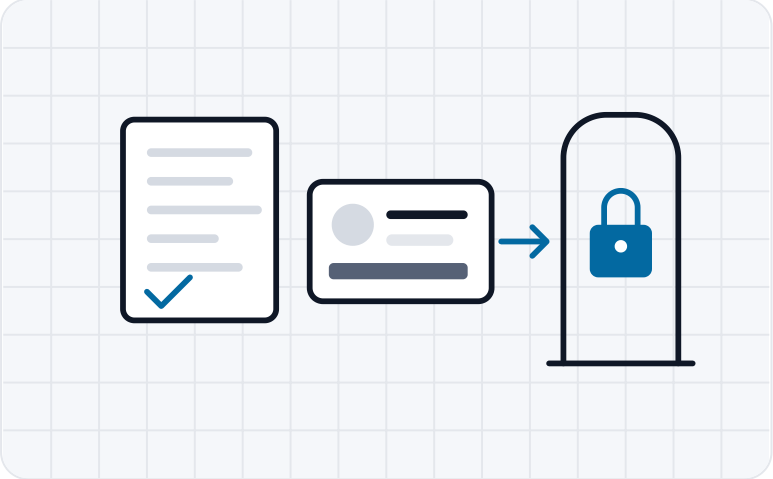
A-SIT Technology
E-Government Innovationszentrum

18. April 2023 Konzept · Digitale Identität · Privatsphäre & Krypto · 13 Min. Lesezeit

# Verbesserte Privatsphäre für flexible Zugangskontrollsysteme

Stefan More



| | |
|---|---|
| **Autorinnen und Autoren** | Stefan More |
| **Quelle** | https://technology2.a-sit.at/veroeffentlichungen/verbesserte-privatsphaere-fuer-flexible-zugangskontrollsysteme/ |
| **Hinweis** | Maschinell übersetzt. |

## Kernaussagen

- Beim zkTPL-Ansatz sind Commitment und Signatur bei jeder Presentation desselben Credentials gleich und machen diese verknüpfbar.
- Verifier können dadurch mehrere Presentations eines Benutzers verknüpfen; bei Zusammenarbeit lassen sich offengelegte Attribute zu einem Datensatz vereinen.
- Ansatz 1: Commitment und Signatur werden randomisiert, sodass die Signatur gültig bleibt, z. B. mit Pointcheval-Sanders-Signaturen.
- Ansatz 2: Die Verbindung von Attributen, Commitment und Signatur wird im Nachweis hergestellt; der Verifier erfährt nur den Issuer des Credentials.

- Beide Ansätze ermöglichen Unverknüpfbarkeit und erfüllen zugleich die Anforderungen der Policy des Verifiers.

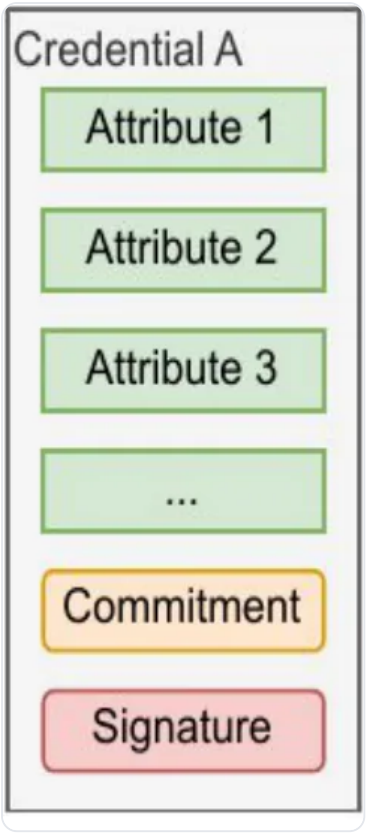
Policies für Zugriffskontrollsysteme (Access Control Policies) ermöglichen eine von der Geschäftslogik entkoppelte Regelung des Zugriffs auf Daten und andere Ressourcen. Neben der daraus entstehenden Flexibilität ist ein weiterer Vorteil, dass diese Policies von „Domain Experts“ ohne spezielle IT Expertise erstellt und gewartet werden können. Diese Policies definieren in der Regel, wer über welche Rechte auf welchen Ressourcen verfügt. Neben dieser Autorisierung kann in solchen Policies auch definiert werden, wie Benutzer authentifiziert werden. Dabei können sowohl bestehende Systeme wie eIDAS oder Enterprise Systeme zum Einsatz kommen, als auch neuere Modelle wie Self-sovereign Identity (SSI).

Eine Herausforderung beim Design von Zugriffskontrollsystemen ist die Wahrung der Privatsphäre von Benutzern. Um eine Zugriffsanfrage gegen eine Policy zu prüfen, muss ein Benutzer oft verschiedenste (oft personenbezogene) Daten bereitstellen. Wenn sich diese Daten in digitalen Dokumenten befinden, werden dabei oft mehr Daten übermittelt als eigentlich notwendig. Um dieser Herausforderung zu begegnen können privatheitsbewahrende Technologien eingesetzt werden. Die Integration dieser Technologien in Zugriffskontrollsysteme, vor allem bestehende, ist aber oft nicht einfach.

## 1. Einleitung

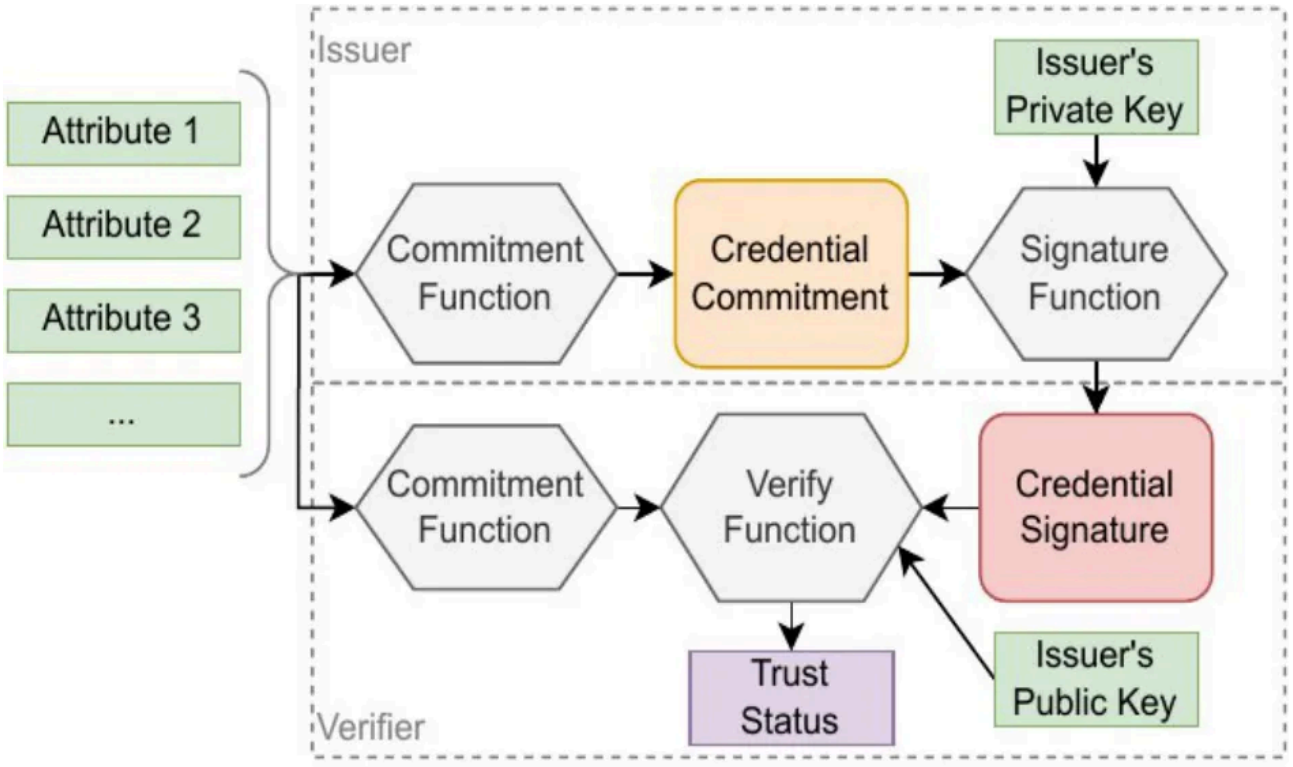
Policies für Zugriffskontrollsysteme sind ein wichtiger Aspekt moderner Computersysteme. Sie tragen dazu bei, dass nur berechtigte Benutzer auf bestimmte Ressourcen zugreifen oder bestimmte Aktionen ausführen können. Systeme setzen die Policies durch, indem sie Zugriffsanfragen auf Grundlage von Identität, Attributen oder anderen Faktoren der Benutzer gewähren oder ablehnen. Üblicherweise legt ein Benutzer dem Verifier ein Credential vor. Ein Beispiel für ein Credential zeigt Abbildung 2.1. Dieses Credential hat zuvor ein Issuer ausgestellt, indem er das Commitment auf die Attribute berechnet (üblicherweise mit einer Hashfunktion) und den resultierenden Digest signiert hat. Einen konzeptionellen Überblick über diesen Vorgang zeigt Abbildung 2.2. Der Verifier prüft anschließend diese Signatur. Dadurch sind Authentizität und Integrität der Attribute geschützt. Herkömmliche Zugriffskontrollmechanismen reichen jedoch nicht aus, um sensible und personenbezogene Informationen zu schützen. Im Gesundheitswesen kann etwa die Krankenakte einer Patientin oder eines Patienten hochsensible Informationen enthalten, auf die nur selektiv und nach dem Need-to-know-Prinzip zugegriffen werden sollte. Ein weiteres Beispiel ist der klassische Fall eines Studenten, der eine Bar besuchen möchte, die einen Altersnachweis verlangt. Der Türsteher der Bar interessiert sich nur für das Alter des Studenten,[1] doch ein staatlich ausgestellter Ausweis würde auch weitere Attribute offenlegen. In solchen Szenarien kann ein herkömmliches Policy-System für Zugriffskontrolle unzureichend sein, da es Daten an Personen offenlegen kann, die diese Informationen nicht legitim benötigen. Um dieser Herausforderung zu begegnen, beziehen privatheitsfördernde Zugriffskontroll-Policies Überlegungen zur Privatsphäre in die Entscheidung über den Zugriff ein. Diese Policies zielen darauf ab, die Privatsphäre sensibler Informationen zu schützen, indem sie die Menge der Informationen begrenzen, die während der Zugriffskontrolle offengelegt werden. Privatheitsfördernde

Zugriffskontroll-Policies lassen sich mit verschiedenen privatheitsbewahrenden Authentifizierungsprotokollen umsetzen, etwa mit Zero-Knowledge-Proofs. Die Konstruktion eines konkreten Protokolls hat jedoch enorme Auswirkungen auf die erreichten Privatheitseigenschaften.




Credential

Insgesamt bieten privatheitsfördernde Zugriffskontroll-Policies einen granulareren und feiner abgestuften Ansatz der Zugriffskontrolle, der stärkere Privatheitsgarantien für sensible Informationen bieten kann. Indem sie Überlegungen zur Privatsphäre in die Entscheidung über den Zugriff einbeziehen, können diese Policies sicherstellen, dass nur berechtigte Personen auf sensible Informationen zugreifen können, und zugleich das Risiko von Verletzungen der Privatsphäre oder unbefugten Offenlegungen minimieren.




## 1.1. Privatheitseigenschaften

**Vertraulichkeit** (Confidentiality) bezeichnet den Schutz sensibler Informationen vor unbefugter Offenlegung oder unbefugtem Zugriff. Vertraulichkeit stellt sicher, dass nur berechtigte Personen auf die Informationen zugreifen können und dass sie ohne Zustimmung der Eigentümerin oder des Eigentümers nicht an andere weitergegeben werden.

**Anonymität** (Anonymity) bezeichnet die Möglichkeit, einen Dienst zu nutzen oder auf eine Ressource zuzugreifen, ohne die eigene Identität oder andere personenbezogene Informationen preiszugeben. Anonymität dient häufig dem Schutz der Privatsphäre von Benutzern, die beim Zugriff auf einen Dienst oder eine Ressource anonym bleiben möchten.

**Unverknüpfbarkeit** (Unlinkability) bezeichnet die Möglichkeit, die Zuordnung mehrerer Transaktionen oder Interaktionen zum selben Benutzer oder zur selben Entität zu verhindern (Pfitzmann, Andreas and Hansen, Marit, 2010). Unverknüpfbarkeit stellt sicher, dass Benutzer einen Dienst nutzen oder auf eine Ressource zugreifen können, ohne eine Spur ihrer Aktivitäten zu hinterlassen, die sich verknüpfen lässt, um ihre Identität oder personenbezogene Informationen aufzudecken. Für die *Multi-Show-Unverknüpfbarkeit* (multi-show unlinkability) soll es für keine Partei möglich sein zu erkennen, ob zwei Zugriffe auf dieselbe Ressource dasselbe elektronische Credential oder zwei verschiedene Credentials verwendet haben.

## 1.2. Zero-Knowledge-Proofs und SNARKs

Eine der Schlüsseltechnologien in privatheitsfördernden Zugriffskontrollsystemen sind **Zero-Knowledge-Proofs** (Manuel Blum and Paul Feldman and Silvio Micali, 1988). Ein Zero-Knowledge-Proof ist ein kryptografisches Protokoll, mit dem eine Partei (der Prover) einer anderen Partei (dem Verifier) beweisen

kann, dass sie eine bestimmte Information kennt, ohne die Information selbst preiszugeben. Das bedeutet, dass ein Benutzer seine Berechtigung für den Zugriff auf eine Ressource oder einen Dienst nachweisen kann, ohne personenbezogene Informationen über sich offenzulegen.

Eine weitere Technologie in privatheitsfördernden Policy-Systemen für Zugriffskontrolle sind **SNARKs**. Die Abkürzung steht für „Succinct Non-Interactive Argument of Knowledge“ (Nir Bitansky and Ran Canetti and Alessandro Chiesa and Eran Tromer, 2012). Ein SNARK ist eine Art von Zero-Knowledge-Proof, der die Überprüfung einer Berechnung erlaubt, ohne dass die Berechnung selbst zurückgegeben werden muss. Das bedeutet, dass ein Benutzer seine Berechtigung für den Zugriff auf eine Ressource oder einen Dienst nachweisen kann, ohne personenbezogene Informationen offenlegen zu müssen und ohne dass der Verifier die Ausführung der Policy erneut durchführen muss.

SNARKs sind der ideale Baustein für ein Policy-System, das sowohl *Privatsphäre* als auch *Ausdrucksstärke* erreichen will.

### 1.3. Attributbasierte Credentials

Attributbasierte (anonyme) Credentials (Attribute-based Credentials, ABCs) sind eine weitere Technologie in privatheitsfördernden Policy-Systemen für Zugriffskontrolle (Ahmad Sabouri and Ioannis Krontiris and Kai Rannenberg, 2012). Ein attributbasiertes Credential ist ein digitales Credential, das Attribute über den Benutzer enthält, etwa Alter, Geschlecht oder Standort. Der Benutzer kann sein attributbasiertes Credential einem Verifier vorlegen, um seine Berechtigung für den Zugriff nachzuweisen, ohne weitere Attribute offenzulegen. Umgekehrt bestätigen die Credentials dem Verifier die Authentizität dieser Attribute. Attributbasierte Credentials erlauben es Benutzern, ihre Berechtigung für den Zugriff nachzuweisen und dabei ihre Privatsphäre zu wahren. Attributbasierte Credentials nutzen eine Kombination kryptografischer Verfahren, um Privatheits- und Sicherheitseigenschaften zu erreichen.

## 2. Ausgangsbasis

Das zkTPL-System bildet die Ausgangsbasis für unsere Arbeit. Es führt einen Ansatz ein, mit dem sich jedes (Policy-basierte) Zugriffskontrollsystem um Privatheitsfunktionen erweitern lässt. Das ermöglicht ausdrucksstarke Policies, die in unterschiedlichen Kontexten einsetzbar sind und generische Anwendungsfälle unterstützen.

### 2.1. zkTPL

Das Projekt zkTPL stellt ein generisches Design vor, um privatheitsbewahrende Technologien in Policy-Sprachen zu unterstützen. Dieses Design verhindert die unnötige Offenlegung sensibler Informationen und erlaubt dennoch, ausdrucksstarke Regeln für die Zugriffskontrolle zu formulieren. Dafür nutzt zkTPL SNARKs. Auf hoher Ebene formuliert der Verifier eine Zugriffs-Policy und sendet sie an den Benutzer. Der Benutzer weist dann mithilfe seiner Credentials nach, dass er die Policy erfüllt, und sendet nur diesen Nachweis an den Verifier. So kann der Verifier sicher sein, dass der Benutzer die Zugriffs-Policy erfüllt, erfährt aber nichts weiter über die Attribute des Benutzers. Ziel ist es, dies vollständig automatisiert zu erledigen. Das zweite Ziel ist die Unterstützung einer Policy-Sprache, die jede Zugriffskontroll-Policy abbilden kann und die Zugriffsregeln nicht auf einfache Aussagen beschränkt. Mit zkTPL nutzt die Autorin oder der Autor der Policy ein bestehendes Policy-System, um festzulegen, welche Aussagen der Benutzer beweisen muss und welche Informationen offengelegt werden müssen, um Zugriff zu erhalten.

Anschließend leitet die Autorin oder der Autor mit dem Policy-Compiler eine Presentation Request ab, die den Benutzern bereitgestellt wird. Die Presentation Request informiert den Benutzer darüber, welche Attribute er offenlegen und welche Aussagen er beweisen muss. So können Benutzer nur die erforderlichen Attribute und Aussagen vorlegen und die übrigen Daten des Credentials verbergen. Mit der kompilierten Presentation Request läuft all dies automatisch ab. Ein Vorteil dieses Ansatzes ist, dass dasselbe Zugriffskontrollsystem in verschiedenen Anwendungsfällen genutzt werden kann, ohne das System neu zu kompilieren oder gar den Quellcode zu ändern.

## 2.2. Vertrauen herstellen

Um Vertrauen in eine Presentation herzustellen, prüft der Verifier, ob der Nachweis mit einem Credential erzeugt wurde, das ein vertrauenswürdiger Issuer ausgestellt hat. Die Vertrauensregeln in der Policy legen fest, welche Issuer für welche Art von Credentials vertrauenswürdig sind. Das kann durch eine Liste vertrauenswürdiger Issuer oder durch die Definition eines vertrauenswürdigen Trust Scheme geschehen. Abhängig vom definierten Trust Scheme ruft der Policy-Verifier automatisch Informationen zum Vertrauensstatus der Credential-Issuer aus Online-Registern ab. Dieser Vorgang stellt sicher, dass öffentlichen und privaten Attributen vertraut werden kann und dass somit alle NIZK-Aussagen über diese Attribute vertrauenswürdig sind. Nachdem der NIZK-Verifier und der Policy-Verifier festgestellt haben, dass das Presentation Token vertrauenswürdig ist und die Policy des Benutzers erfüllt, gewährt der Service Provider dem Benutzer Zugriff auf den Dienst. Auf technischer Ebene entsteht das Vertrauen in eine Presentation durch eine Vertrauenskette vom Issuer über die Signatur zum Commitment, das wiederum mit dem Nachweis und folglich mit den Attributen verknüpft ist. Der an den Verifier gesendete Nachweis enthält eine (offengelegte) Aussage über den Commitment-Digest des ursprünglichen Credentials. Diese Aussage beweist, dass alle Attribute (offengelegte und nicht offengelegte) und Prädikate Teil des Urbilds des Commitments sind. Anders gesagt: Werden alle Attribute kodiert und ihr Commitment berechnet, ergibt sich derselbe Digest wie der offengelegte. Da die Signatur des Issuers auf diesem Commitment ausgestellt wurde, verknüpft sie den Nachweis kryptografisch mit dem Issuer. Die Presentation besteht somit aus dem Nachweis (einschließlich der offengelegten Attribute), dem offengelegten Commitment (als Teil des Nachweises) und der Signatur des Issuers auf dem Commitment.

## 2.3. Verknüpfbarkeit

Ein Nachteil dieses Ansatzes ist, dass Signatur und Commitment immer an den Verifier gesendet werden. Diese Werte sind für Presentations, die aus demselben Credential abgeleitet wurden, immer gleich (siehe Abbildung 3.3.1) - auch wenn unterschiedliche Attribute offengelegt werden oder gar keine Attribute offengelegt werden. Das verletzt die Eigenschaft der *Unverknüpfbarkeit*.

Dadurch kann ein Verifier mehrere Vorlagen desselben Benutzers verknüpfen, was ein Problem für die Privatsphäre des Benutzers ist. Wenn der Benutzer zudem verschiedenen Verifiern unterschiedliche Nachweise vorlegt und diese Verifier zusammenarbeiten, können sie die Presentations des Benutzers ebenfalls verknüpfen. Das ist besonders dann ein Problem, wenn der Benutzer in verschiedenen Vorlagen unterschiedliche Attribute offengelegt hat. Diese lassen sich dann zu einem Datensatz zusammenführen und rekonstruieren so faktisch das vollständige Credential.

## 3. Auf dem Weg zur Unverknüpfbarkeit

Ziel ist, dass ein Benutzer aus einem Credential zwei Presentations erzeugen kann, die weder mit dem Credential noch untereinander Informationen gemeinsam haben.[2] Dadurch wird verhindert, dass ein Verifier die Vorlagen der beiden Presentations verknüpfen kann. Für den in Abbildung 2.2 eingeführten Typ eines Commit-then-sign-Credentials bedeutet das, dass ein Benutzer Commitment und Signatur nicht einfach an den Verifier senden kann. Der Verifier benötigt jedoch beide Informationen, um die Authentizität des Credentials zu prüfen. Der Verifier nutzt nämlich die Signatur und deren Issuer in einer Trust-Policy, um zu beurteilen, ob er den Issuer für vertrauenswürdig hält. Im zkTPL-Ansatz geschieht dies, indem der Nachweis mit dem Commitment verknüpft und anschließend die Signatur auf dem Commitment genutzt wird, um Vertrauen in die Attribute herzustellen.

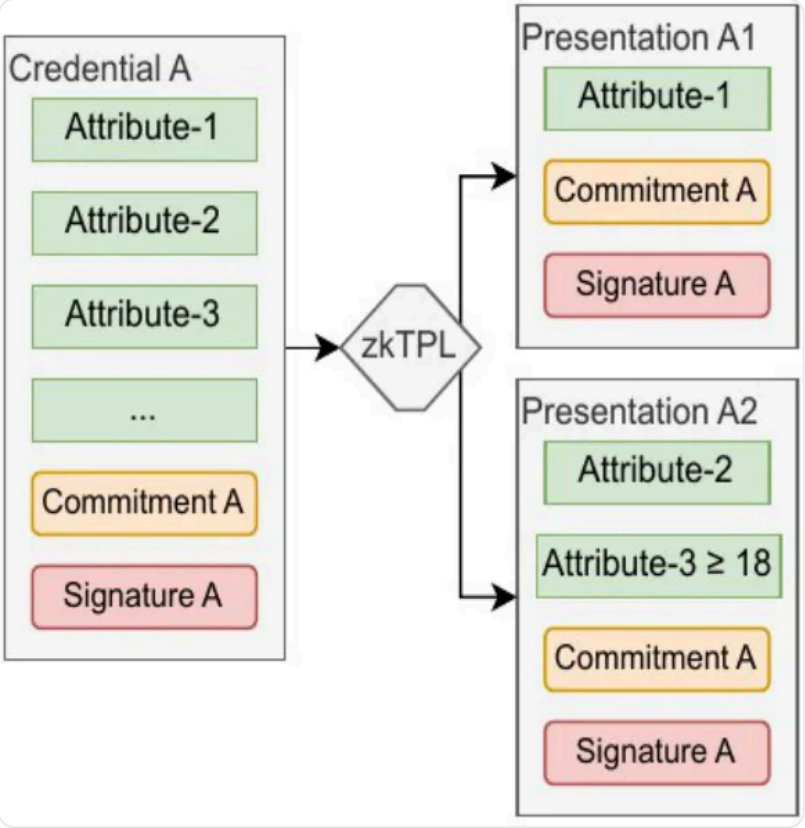
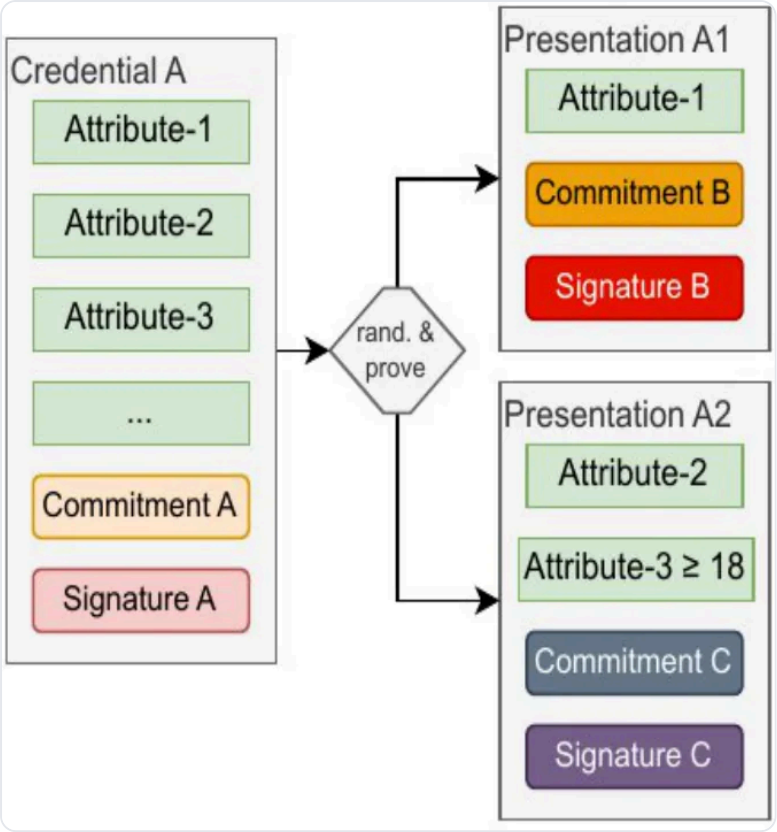



Abbildung 3.3.1: Beispielhafter Ablauf, wie mit zkTPL aus einem Credential zwei Presentations abgeleitet werden




Randomisierung, um aus einem Credential zwei unverknüpfbare Presentations abzuleiten

Eine Lösung besteht darin, das Commitment und die Signatur zu *randomisieren*.

Das Ergebnis ist ein Ablauf wie in Abbildung 4.1: Beide Presentations enthalten einen eigenen Commitment-Wert und einen eigenen Signaturwert. Wird für beide Werte ein geeignetes Verfahren verwendet und die Randomisierung korrekt durchgeführt, bleibt die Signatur für die Daten im Nachweis gültig (siehe Abbildung 4.2).

Der Verifier authentifiziert dann die (randomisierte) Signatur und stellt Vertrauen in deren Issuer auf dieselbe Weise her wie in Abschnitt 3. Dadurch kann der Verifier sicherstellen, dass die Presentation mit einem gültigen Credential eines vertrauenswürdigen Issuers erzeugt wurde.

Eine geeignete Kombination sind Dual Commitments (Torben P. Pedersen, 1991) zusammen mit Pointcheval-Sanders-Signaturen (David Pointcheval and Olivier Sanders, 2016). Ein weiteres Beispiel sind Structure-Preserving Signatures on Equivalence Classes, eingeführt von Georg Fuchsbauer, Christian Hanser und Daniel Slamanig, mit den zugehörigen Set Commitments (Georg Fuchsbauer and Christian Hanser and Daniel Slamanig, 2019). Eine alternative Lösung besteht darin, die Werte für Commitment und Signatur gar nicht offenzulegen.

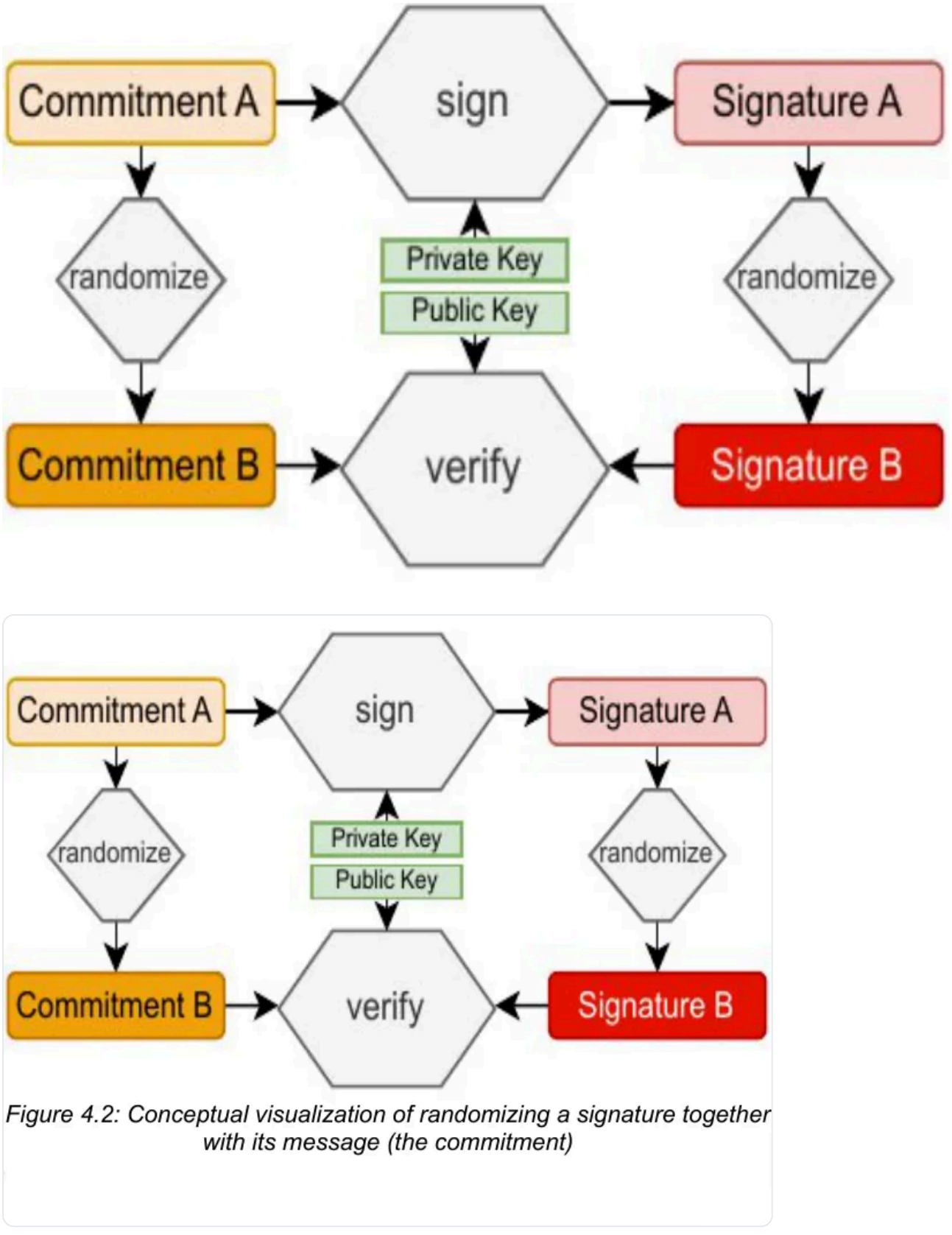



*Figure 4.2: Conceptual visualization of randomizing a signature together with its message (the commitment)*

Abbildung 4.2: Konzeptionelle Darstellung der Randomisierung einer Signatur zusammen mit ihrer Nachricht (dem Commitment)

## 3.1. Integration in die Policy

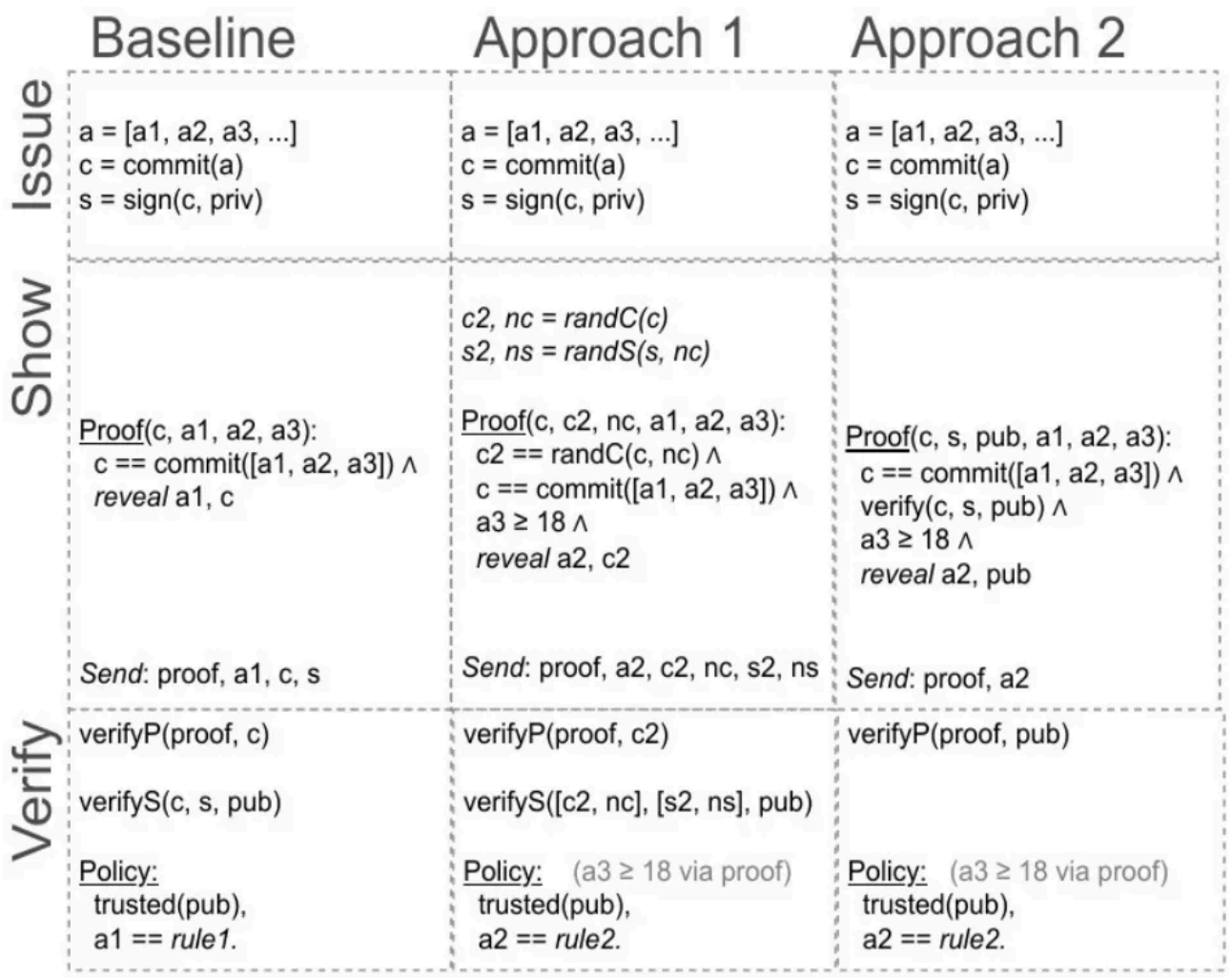
Aus Sicht von Benutzer und Verifier ändert sich der Ablauf im Vergleich zu zkTPL nicht. Zuerst erstellt der Verifier eine Policy und formuliert und kodiert darin Vertrauens- und Zugriffsregeln. Daraus kompiliert er eine Presentation Request. Diese Presentation Request wird dem Benutzer gesendet, wenn er Zugriff auf eine Ressource möchte. Der Benutzer leitet seinerseits mit der Anfrage aus seinem Credential eine Presentation ab und legt nur die erforderlichen Attribute offen. Auf technischer Ebene unterscheiden sich sowohl die Presentation Request als auch die Presentation. Alle beteiligten Schritte laufen aber weiterhin automatisch mit der bestehenden Policy und den bestehenden Credentials ab. Zusätzlich führt das System des Benutzers die notwendigen Schritte aus, um Verknüpfbarkeit zu verhindern und so die Privatsphäre des Benutzers zu wahren. Konzeptionell gibt es zwei Ansätze, um Unverknüpfbarkeit zu erreichen. Einen Überblick über diese beiden Ansätze geben wir in Abbildung 4.3.1.

## 3.2. Ansatz 1: Randomisierung verwenden

Damit zwei Presentations keine identifizierbaren (oder verknüpfbaren) Informationen enthalten, müssen alle verknüpfbaren Informationen randomisiert werden. In unserem Fall verursachen das Commitment (z. B. der Hash-Digest auf die Attribute) und die Signatur (auf dem Commitment) die (unerwünschte) Verknüpfbarkeit zwischen den Presentations. Der Trick besteht darin, beide Werte auf geschickte Weise zu randomisieren: Erstens muss das randomisierte Commitment ein Commitment auf dieselben Daten sein, da es dazu dient, die Integrität der Daten zu schützen und sie an die Signatur zu binden.[3] Zweitens gilt dasselbe für die randomisierte Signatur, da sie später genutzt wird, um die Authentizität des Commitments zu prüfen. Drittens stehen beide Werte in Beziehung zueinander und sind nur sinnvoll, wenn diese Beziehung erhalten bleibt. Es ergibt keinen Sinn, das Commitment anders zu randomisieren als die Signatur, denn das würde die Signatur ungültig machen. Daher müssen entweder beide Werte auf kompatible Weise randomisiert werden. Oder es gibt einen Nachweis, der diese Beziehung belegt. Je nach verwendeten Verfahren muss der Benutzer die Verbindung zwischen dem ursprünglichen Commitment und dem randomisierten Commitment nachweisen (siehe Abbildung 4.3.1). Diesen Ablauf stellen wir in Abbildung 4.2 dar.

## 3.3. Ansatz 2: Einen Nachweis verwenden

Eine Alternative besteht darin, die Verbindung zwischen Attributen, Commitment und Signaturen direkt im Nachweis herzustellen. Das hat den Vorteil, dass der Verifier nur den Nachweis prüfen, kontrollieren muss, ob er mit dem korrekten öffentlichen Schlüssel erstellt wurde, und Vertrauen in diesen öffentlichen Schlüssel herstellen muss. Eine Einschränkung ist dagegen, dass der Nachweis der Verifikation einer Signatur sehr langsam oder - je nach verwendetem Signaturverfahren - sogar unmöglich ist. Richtig umgesetzt kann der Verifier jedoch die Authentizität der Presentation und die Erfüllung der Policy prüfen und erfährt dabei nur, wer das ursprüngliche Credential signiert hat.[4]




## 4. Schlussfolgerungen

In diesem Bericht erörtern wir mehrere Strategien, um ein Zugriffskontrollsystem um Privatheitsfunktionen zu erweitern. Mit den vorgestellten Strategien ist es außerdem möglich, dieses Ziel zu erreichen und dabei die Unverknüpfbarkeit mehrerer Credentials oder Credential-Vorlagen sicherzustellen. Das schützt die Privatsphäre des Benutzers und erfüllt zugleich die Anforderungen der Policy des Verifiers.

## 5. Literaturverzeichnis

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. Anhang: zkTPL-Demolauf

Das separate Dokument „abcTPL_appendix.pdf“ beschreibt die Ausgangsbasis des Projekts anhand eines Beispiels.

In dieser Demo[5] verwenden wir den oben vorgestellten SNARK-basierten Ansatz. Die Abfolge der Befehle und die ausgetauschten Daten ändern sich beim Einsatz anderer kryptografischer Techniken nicht wesentlich.

---

1. Konkreter: Der Türsteher der Bar interessiert sich nur dafür, ob das Alter des Studenten über einem bestimmten Schwellenwert liegt - ein sogenanntes Prädikat auf ein Attribut. ↩
2. Eine Ausnahme ist der Fall, dass der Benutzer freiwillig auf Unverknüpfbarkeit verzichtet und genügend Attribute offenlegt, um eine Verknüpfung zu ermöglichen. ↩
3. Auf technischer Ebene *öffnen* beide Commitments zur selben Nachricht. ↩
4. Auch der Issuer eines Credentials lässt sich verbergen, indem das Verfahren weiter ausgebaut wird (Jan Bobolz and Fabian Eidens and Stephan Krenn and Sebastian Ramacher and Kai Samelin, 2021). ↩
5. https://technology.a-sit.at/veroeffentlichungen/privatheitsbewahrende-trust-policies/ ↩