Invisible Traces: Subversion Attacks on Batch-Issued Credentials

Anna-Birgitta Burmeister[1], Anna Fennig[1], Andreas Franke[1], Karla Friedrichs[2], Anja Lehmann[2], Kurt-Kester Leißering[1], Konrad Letz[1], Cavit Özbay[2]
Hasso Plattner Institute, University of Potsdam
[1]name.surname@student.hpi.de, [2]name.surname@hpi.de

Abstract

All EU member states are required to roll out a digital identity system—the European Digital Identity (EUDI) wallet—by the end of 2026. Strong privacy is at the core of the underlying regulation, which mandates the EUDI wallet to support selective disclosure and unlinkability. The wallet currently being developed relies on the batch issuance of one-time ECDSA credentials that sign attributes through individually salted hashes for selective disclosure. This solution is known to achieve only a weak form of unlinkability, where the credential issuer must be honest: a malicious issuer could trace users through the salted hashes it signs and the signature value itself. But such a tracing attack requires the issuer to store all signed data and communicate with the verifying parties for tracing, which can be argued to be too cumbersome or obvious to happen in reality. In this work, we therefore initiate the study of a more subtle type of subversion attacks. Therein, the issuer can deviate from the issuance protocol, with two goals: (i) enabling verifiers in possession of a short tracing key to de-anonymize users and (ii) keeping this deviation undetectable from users. We formalize unlinkability against such subversion attacks, and show that batch-issued credentials with salted hashes do not achieve that form of privacy. We present several undetectable subversion attacks against batch-issued ECDSA credentials and suggest lightweight mechanisms to provably mediate them.

Keywords

batch-issued credentials, unlinkability, privacy, subversion, EUDI

1 Introduction

A digital identity system represents a user's identity by a credential, which consists of a set of attributes signed by a central issuer. The user can authenticate to verifiers using this credential: to do so, they create a presentation that reveals certain necessary attributes in a verifiable manner. There is currently a strong interest in deploying such digital identity systems, including the EU Digital Identity Wallet (EUDI), which all EU member states are required to roll out by the end of 2026. Using a digital identity system and strong authentication can improve security, but its ubiquitous manner also poses a risk to privacy when implemented in the wrong way. Therefore, the eIDAS2.0 regulation [19] mandates several core privacy principles that the EUDI wallet must satisfy.

The first of these privacy properties is selective disclosure [19, §4a], enabling the user to select a subset of attributes they want to reveal in a presentation. The second is unlinkability [19, §16]: if the revealed attributes do not identify the user, verifiers should not be able to recognize if the same user is authenticating multiple times, or track their presentations across services. A stronger form of unlinkability also considers the issuer to be corrupt and to collude with the verifiers.

Salted Hashes & Batch Issuance. Solutions such as anonymous credentials [4, 13] exist that naturally satisfy all required privacy guarantees, including the strong notion of corrupt-issuer unlinkability. However, the EU has decided to not use them (yet), due to the lack of existing standards and production-ready implementations. Instead, the EUDI wallet relies on classic ECDSA-signed credentials with two modifications: salted hashes and batch issuance [18]. Salted hashes enable selective disclosure: instead of signing user attributes $a_1, \ldots, a_n$ directly, the issuer chooses random salts $s_1, \ldots, s_n$ and signs $H(a_1 \| s_1), \ldots, H(a_n \| s_n)$ with an ECDSA signature. At presentation time, the user can hide attributes by merely revealing their hash, but not the salt and attribute. For achieving honest-issuer unlinkability, the user is supposed to use every credential only once, and get a batch of one-time credentials from the issuer, each with freshly salted hashes. This solution is already standardized as mdoc (ISO/IEC 18013-5) [26] and SD-JWT VC [21, 37].

While such batch-issuance achieves the weaker form of honest-issuer unlinkability—any two presentations are independent from each other—unlinkability in the presence of corrupt issuers is elusive: both the revealed salts and the signature value link the presented and issued credential. The issuer could trivially record all credentials it issues and look up a specific salt or signature the verifier receives later. This fully de-anonymizes the user, regardless of how little the presented attributes reveal, e.g., even when solely proving that one is over 16. While devastating, this attack requires significant resources: the issuer must store all one-time (!) credentials of potentially millions of users, and exchange and compare all received presentations with the verifiers. Assuming that issuers and verifiers have an interest in avoiding detection when exploiting such an identity system for surveillance, this large-scale storage and data exchange could be considered too risky and impractical. Consequently, the current solution may be conjectured to provide "practical" corrupt-issuer unlinkability.

Tracing through Subversion Attacks? A natural question is if there are subtler and more convenient ways to track users than through storing and comparing all credentials and presentations. For instance, a corrupt issuer could include in every credential a tracing ID as a mandatory-to-reveal attribute—or just not use batch issuance. But this deviation is obvious, and users would (hopefully) stop using the system. The more interesting question is whether the issuer can deviate from the original design in a way that is

This work is licensed under the Creative Commons Attribution 4.0 International License. To view a copy of this license visit https://creativecommons.org/licenses/by/4.0/ or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.
Proceedings on Privacy Enhancing Technologies YYYY(X), 1–24
© YYYY Copyright held by the owner/author(s).
https://doi.org/XXXXXXX.XXXXXXX

- undetectable for users, i.e., they cannot notice whether the original credential system has been manipulated, and
- offline-traceable, i.e., meaning that it allows for immediate tracing by verifiers without further exchange of information with the issuer.

In fact, such subversion attacks also capture an attack vector that even honest issuers might suffer from: as credential issuance in a national identity system must be protected through secure hardware, their own infrastructure will rely on sealed hardware components that could be subject to secret subversion attacks from hardware vendors or state-level adversaries. Overall, we are interested in two aspect of such subversion attacks: (1) whether there are undetectable subversion attacks that enable convenient tracing while remaining compliant with the EUDI specification, and (2) and how to modify the currently designed credential systems so that they are resilient against such subversion attacks.

1.1 Our Contributions

In this work, we answer both questions by initiating the study of subversion attacks that allow undetectable offline tracing in credential-based identity systems. We formulate a new privacy property that sits in between honest and corrupt-issuer unlinkability and captures this threat. We show that this property is not achieved by the currently standardized solution of batch issuance and salted hashes, by giving practical attacks. Finally, we describe fixes that prevent subversion attacks on these type of constructions. More concretely, our work makes the following contributions.

Security Model. We first give a formal model to capture classic unlinkability with an honest or a corrupt issuer for batch-issued one-time credentials. Then, we propose a new, intermediate notion subversion unlinkability: therein the issuer is corrupt, but can misbehave only "minimally" in order to be undetectable. In particular, issuer and verifiers are not allowed any direct and continued communication, but can only exchange a "tracing key". The latter can be any information the issuer wants to share with the verifiers to enable offline tracing. Importantly, the tracing key is shared when the system is set up and before the users obtain credentials. Once the credential system is active, the issuer can deviate from honest issuance by running a subverted issuance algorithm. Subversion unlinkability ensures that even with subverted credentials and knowledge of the tracing key, verifiers cannot break the unlinkability of the users' presentations.

Issuers might offer such tracing keys in the form of "tracking-as-a-service" to verifiers that want to recognize returning users even from minimal disclosures (e.g., age proofs). Or, the issuer itself wants to surveil users, incentivizing verifiers into colluding and reporting back through other channels. In fact, "issuer" need not mean a government authority, but could be any qualifying organization, including large tech or industry players. Such an issuer might offer a seemingly narrow, privacy-preserving service (e.g., age-attesting credentials) while having its own commercial interest in linking user activity across the verifiers relying on its credentials.

To identify a meaningful set of attacks, we require two additional properties of a subversion attack. Undetectability ensures that credentials issued by the subverted issuer are indistinguishable from honestly created ones. Subversion unforgeability captures that while issuers may have an incentive to undermine user privacy, they typically do not wish to weaken the overall security of the digital identity system. Hence the exposure of the tracing key to corrupt verifiers should not reduce the unforgeability of credentials.

Subversion Attacks & Defenses. We show that batch-issued credentials from salted hashes do not satisfy this form of subversion unlinkability. Our first attack exploits the salts that must be revealed for every disclosed attribute. As the salts have high entropy for realizing the desired selective disclosure, they are a prime target for subversion attacks: we can simply replace them with probabilistic encryptions of a unique identifier of the user. The tracing key is then the decryption key. A more subtle way is to leverage the known anamorphic capabilities [1, 29, 36] of ECDSA signatures. That is, we can embed tracing information into the randomness of ECDSA signatures. Both attacks are clearly undetectable, enable immediate tracing, and leave unforgeability intact. As a side result, we show that measures in the SD-JWT VC standard [21, 37] aimed to increase user privacy provide further avenues for tracing; concretely, the random ordering and variable number of attributes. We discuss the practical impact of these attacks by estimating the amount of information that can be secretly embedded in a credential. We find that even a presentation that discloses a single attribute, e.g., an "over-16" attribute or expiration date, is sufficient to embed a unique tracing ID for every person on earth. To demonstrate feasibility, we provide a proof-of-concept:[1] we modify an existing implementation of the SD-JWT standard [20] such that the issuer can enable any of the identified attacks to embed tracing data, which the verifier can then extract during verification.

Finally, we propose countermeasures for all identified attacks and prove the modified credentials to provide the desired subversion unlinkability. The goal thereby is to be as lightweight as possible, making use of the leeway we get by aiming only at subversion but not corrupt-issuer unlinkability. As a result, even our modified credentials are vulnerable against corrupted issuers, and we lead the reader to prior work aiming at corrupt-issuer unlinkability for such constructions [3, 14].

1.2 Related Work

The analysis of security and privacy risks arising for the EUDI wallet has naturally already received some attention, for instance, a comprehensive threat model is presented by [34], where user linkability is identified as a threat. It is also a known fact that the current wallet version does not achieve unlinkability against a malicious issuer [4]. The novelty of our work lies in the formal and concrete study of subversion attacks, which fall into the established threat category of user linkability, but target an unexplored vulnerability of EUDI credentials, cannot be mitigated by inspection of credentials, and are more likely to elude audits of the issuer, since the deviation from honest behavior is minimal.

Our security model and analysis lean onto a long line of work on subversion resilience. The first work to model replacing honest algorithms with malicious ones is kleptography [39, 40]. After the Snowden revelations, this field got renewed interest and several

[1] https://github.com/abc013/sd-jwt-python

works studied subverting different primitives and algorithms with the goal of surveillance, e.g., [2, 5, 16, 33]. The first formal model of algorithm substitution is due to [5], strengthened by [33] and [2]. While we apply the same techniques to credential systems, the paradigm is somewhat different: the typical scenario of subversion attacks is that an honest user unknowingly executes subverted algorithms, which leak private information to the big brother. We aim instead to capture a corrupt issuer that limits its deviation from honest behavior in order to keep a low profile.

We draw from the same line of research in our attacks, where we use a subliminal channel in ECDSA signatures to embed tracing information. Interestingly, signatures have been one of the first primitives for which subliminal channels were considered [35]. Recently, this direction has been revived by anamorphic signatures [27, 29], where the hidden channel in signatures circumvents a hypothetical ban of encryption. While anamorphic properties of DSA have been recognized before [1, 29, 36], we formulate ECDSA as an anamorphic signature for the first time, which might be of independent interest. Our work gives anamorphic signatures a dark twist—so far, they rescued privacy in a world of surveillance, whereas we show how anamorphism can undermine privacy in the context of a subverted digital identity scheme.

An obvious solution to prevent subversion attacks is to use anonymous credentials [4, 13]. These natively achieve corrupt-issuer unlinkability, and thus also satisfy our weaker subversion unlinkability. The biggest obstacle to their deployment, however, is the lack of standardization and production-ready implementations. As the EUDI wallet is being built now, it needs to rely on standardized components. Our work explores how such solutions without anonymous credentials can be strengthened to provide privacy against the more powerful type of subversion attacks. Here, we build upon earlier observations: we use unique signatures to eliminate any subliminal channels as [2].

2 Preliminaries

This section introduces our notation and cryptographic primitives. A more formal definition is given in App. A.

Notation. We use $\|$ for concatenating bit strings. Empty sets are denoted as $\emptyset$ and empty lists as $()$. For sets of integers, $[a]$ is short for $\{x \in \mathbb{Z} \mid 1 \le x \le a\}$.

Hash Functions. In order to make only the minimal necessary assumptions, we consider the two properties of hash functions: collision resistance and input hiding. Collision resistance ensures that it is hard to find two messages with the same hash value; formally, we demand that finding an efficient (PPT) adversary $\mathcal{A}$ that outputs m, m' with $m \neq m' \wedge \mathsf{H}(m) = \mathsf{H}(m')$ should be hard. Input hiding ensures that hashing some message concatenated to a random salt hides the message: for this, we define a distinguishing game where the adversary as access to $O\mathsf{H}$. The oracle takes two messages m_0, m_1 as input, chooses a random $s \leftarrow\!\!\$\ \{0,1\}^\lambda$ and returns $\mathsf{H}(s\|m_b)$. For an input-hiding hash function, it must be hard for all adversaries to guess b. It is believed that standard collision-resistant hash functions are indeed also input-hiding [11, Section 8.12].

Signatures. A signature scheme consists of three algorithms, $\mathsf{KGen}(1^\lambda) \to (sk, pk)$, $\mathsf{Sign}(sk, m) \to \sigma$, and $\mathsf{Vf}(pk, m, \sigma) \to 0/1$, for key generation, signing, and signature verification, respectively. Schemes are called unforgeable if adversaries that do not hold the secret key but may learn some signatures on chosen messages cannot create a fresh signature on any other message. Furthermore, a signature scheme is *unique*, if there do not exist values $(pk, m, \sigma_1, \sigma_2)$ such that $\sigma_1 \neq \sigma_2$ and $\mathsf{Vf}(pk, m, \sigma_1) = \mathsf{Vf}(pk, m, \sigma_2) = 1$. The description of ECDSA signatures is presented in App. A for completeness.

ECDSA. ECDSA is one of the most widely used signatures. We repeat a definition following Johnson et al. [28] here, assuming that a suitable elliptic curve $\mathbb{G}$ of size $q \in [2^{\lambda-1}..2^\lambda)$ with generator G is available to all algorithms.

Definition 2.1 (ECDSA). Let H be a hash function. The ECDSA signature scheme is defined by the following algorithms:

$\mathsf{KGen}(1^\lambda)$: Sample $d \leftarrow\!\!\$\ \mathbb{Z}_q^*$. Return $(sk, pk) \leftarrow (d, dG)$.

$\mathsf{Sign}(sk, m)$: Parse $d \leftarrow sk$. First, sample $\kappa \leftarrow\!\!\$\ \mathbb{Z}_q^*$ and let $\kappa G = (x, y)$. Set r to x as an integer mod q. Then, compute $s \leftarrow \kappa^{-1}(\mathsf{H}(m) + dr) \bmod q$. If $s = 0$ or $r = 0$, return $\perp$. Else, return $\sigma \leftarrow (r, s)$.

$\mathsf{Vf}(pk, \sigma, m)$: Parse $Q \leftarrow pk$ and $(r, s) \leftarrow \sigma$. If $(r, s) \notin \mathbb{Z}_q^*$, return 0. Compute $(x, y) = \mathsf{H}(m)s^{-1}G + rs^{-1}Q$. Return 1 if $(x, y) \neq 0_\mathbb{G}$ and $x \equiv r \pmod q$, else return 0.

Pseudorandom Encryption. Finally, we employ symmetric encryption schemes that produce ciphertexts indistinguishable from random strings. Assuming that ENC encrypts $u(\lambda)$-bit plaintexts into ciphertexts $c \in C$ for a polynomial u, a formal game following Kutyłowski et al. [29] asks an adversary to distinguish between the encryption of a chosen message and a random value from C. We call an encryption scheme $\mathsf{ENC} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ pseudorandom, if distinguishing is hard for all efficient (PPT) adversaries. A pseudorandom encryption scheme can be easily constructed by relying on a PRF as shown by Kutyłowski et al. [29] and we recap this construction in App. A.

3 Batch-Issued Credentials (BC)

To assess the security of batch-issued credentials with salted hashes, as used in the upcoming EUDI Wallet, we first formalize algorithms and the security model. We mostly follow the traditional syntax for credential schemes as well as the known properties unforgeability and unlinkability. However, we adapt them for the batch-issuance that salted-hash credentials leverage for user privacy.

Batch Issuance. The issuance process of credential schemes is influenced by their unlinkability guarantees and can be defined as either multi-use or single-use. Multi-use unlinkability requires that repeated presentations of the same credential remain unlinkable, whereas single-use unlinkability guarantees unlinkability only when each credential is presented once. Batch-issued credentials issue a batch of single-use credentials at once. Our definitions reflect the batch issuance starting from the syntax level.

3.1 BC Syntax

Batch-issued credentials (BC), as all credential schemes, require algorithms for generating issuer keys, issuance of credentials, presenting, and verifying presentations. The only part we modify syntax-wise for batching is issuance.

Issuer keys are generated by KGen. The Issue algorithm now generates k credentials at once. Additionally, looking ahead, we require interactive issuance once we develop defenses against subversion attacks. In particular, it turns out useful to let the user contribute to the issuance and verify the received credentials, which results in a 2-move interaction: with ReqCreds a user requests the issuer to attest attributes $\overline{a}$, Issue lets the issuer generate a batch of credentials, and in FinCreds, the user performs final steps such as verifying the issuer's output. To authenticate themselves to a verifier, a user then picks a fresh credential from the batch, selects with $\overline{a}^*$ the attributes from $\overline{a}$ they want to disclose, and runs Present. We assume $\overline{a}^*$ is of length n just like $\overline{a}$, but has $\perp$ in place of the hidden attributes. By assuming the user to keep state of their batch, i.e., delete consumed credentials and re-fetch once a batch is used up, we may employ the classic, non-batch, presentation algorithm that works on a single credential as usual. Verifiers run VfPres with the public key announced by the issuer to check the validity of presentations, again with no particularities.

*Definition 3.1 (Batch-Issued Credentials (*BC*)).* $\mathsf{BC}_{\mathcal{M},n,k}$ on attribute space $\mathcal{M}$ with $n \in \mathbb{N}^+$ attributes and batch size $k \in \mathbb{N}^+$ is a tuple of algorithms, s.t.:

- $\mathsf{KGen}(1^\lambda) \to (isk, ipk)$: generates the issuer's key pair.
- $\mathsf{ReqCreds}(ipk, \overline{a}) \to (st, \overline{req})$: lets the user request k credentials for attributes $\overline{a}$. The vector $\overline{req}$ is sent to the issuer, while st is kept to finalize the response.
- $\mathsf{Issue}(isk, \overline{a}, \overline{req}) \to \overline{rsp}$: generates the issuer's response in order to batch-issue credentials for the attribute set $\overline{a}$.
- $\mathsf{FinCreds}(st, \overline{rsp}) \to \overline{cred}$: finalizes response $\overline{rsp}$ into a vector of credentials using the stored state st, or returns $\perp$ if finalization fails.
- $\mathsf{Present}(cred, \overline{a}^*) \to pres$: takes a credential along with the attributes $\overline{a}^*$ to disclose, and returns a presentation.
- $\mathsf{VfPres}(ipk, pres, \overline{a}^*) \to b \in \{0,1\}$: checks whether the presentation $pres$ is valid with respect to disclosures $\overline{a}^*$ and issuer verification key ipk.

3.2 Base Security Model

Next, we state the standard security notions—unforgeability and unlinkability—adapted to a credential scheme with batch issuance. We define correctness of BC in App. B.1 for completeness.

3.2.1 Security: Unforgeability. We want to capture that a malicious user should not be able to generate presentations with attributes for which they do not possess a matching credential. A simple game-based definition is given in Fig. 1. We allow the adversary to request credentials over arbitrary attributes through OIssue, which generates a batch of k credentials following our syntax. The goal of the adversary is to return a presentation over some attributes $\overline{a}^*$ that could not have been presented from a credential that OIssue output before. Note that we slightly abuse the vector notation: $\overline{a}^* \subseteq \overline{a}$ intends that $\forall a_i^* \neq \perp \in \overline{a}^* : a_i^* = a_i$.

Definition 3.2 (Unforgeability). BC is unforgeable for the experiment in Fig. 1 if $\forall$ PPT adversaries $\mathcal{A}$, the following advantage is negligible in λ:

$$\mathsf{Adv}^{\mathsf{UNF}}_{\mathcal{A},\mathsf{BC}}(\lambda) \coloneqq \Pr\left[\mathsf{Exp}^{\mathsf{UNF}}_{\mathcal{A},\mathsf{BC}}(\lambda) = 1\right]$$

3.2.2 Privacy: Unlinkability. When presenting a credential, the user should be guaranteed that the verifier learns only the disclosed attributes and nothing else, and in particular, cannot distinguish users by their presentations. One can formulate a basic guarantee selective disclosure, which ensures that a verifier learns nothing about the hidden attributes from a single presentation. A more comprehensive guarantee is that a verifier should, even after interacting many times with different users, learn nothing about the owner of a presentation beyond what they know through the disclosed attributes. This is commonly known as unlinkability, and implies selective disclosure. We therefore focus immediately on this stronger notion and defer the definition of selective disclosure to App. B.2.

Fig. 1 defines two unlinkability properties depending on whether the issuer is honest or corrupt. Both share the same underlying game which simulates honest users. The adversary can obtain presentations from the users' credentials by using the OPresent, but only once from each credential of a user. Then the adversary chooses two users u_0, u_1 with common attributes S and unused credentials i_0, i_1 of these users. It receives a presentation which discloses S from the credential i_b of the user u_b and outputs a bit b'. The adversary succeeds if the distribution of b' changes non-negligibly depending on the challenge bit b.

Honest-Issuer Unlinkability. This property also models the issuer as an honest party. The game generates the issuer key pair and sends the issuer public key to the adversary. Furthermore, the adversary gets access to OAdd and OIssue. OAdd runs the issuance honestly for a new user and outputs its index u. OIssue outputs credentials on queried credential requests, modeling the adversary's power to see the corrupt users' credentials.

Corrupt-Issuer Unlinkability. A stronger unlinkability lets the adversary fully control the issuer and select the issuer public key. Therefore, OIssue is no longer needed. For adding honest users, the adversary has control over the issuance but the request and finalization of credentials is performed by the user, so we model OReqCreds and OFinCreds. Completing the latter makes a user available to OPresent.

We define both versions of unlinkability at once. In App. B.3 we show that modeling a corrupt issuer implies the simpler version.

Definition 3.3 (Unlinkability). For $\mathsf{x} \in \{\boxed{honest}, \boxed{corrupt}\}$ and experiments in Fig. 1, BC is x-*issuer-unlinkable* if $\forall$ PPT adversaries $\mathcal{A}$, the following advantage is negligible in λ:

$$\mathsf{Adv}^{\mathsf{LINK\text{-}x}}_{\mathcal{A},\mathsf{BC}}(\lambda) \coloneqq \left| \Pr\left[\mathsf{Exp}^{\mathsf{LINK\text{-}x\text{-}0}}_{\mathcal{A},\mathsf{BC}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{LINK\text{-}x\text{-}1}}_{\mathcal{A},\mathsf{BC}}(\lambda) = 1\right] \right|$$

3.3 BC from Salted Hashes

Having defined BC and its properties, we now turn to its instantiation by salted-hash credentials. We present a construction that is an abstraction of the approaches in the standards ISO/IEC 18013-5 [26] and SD-JWT VC [21, 37] and focuses on the mechanisms relevant to

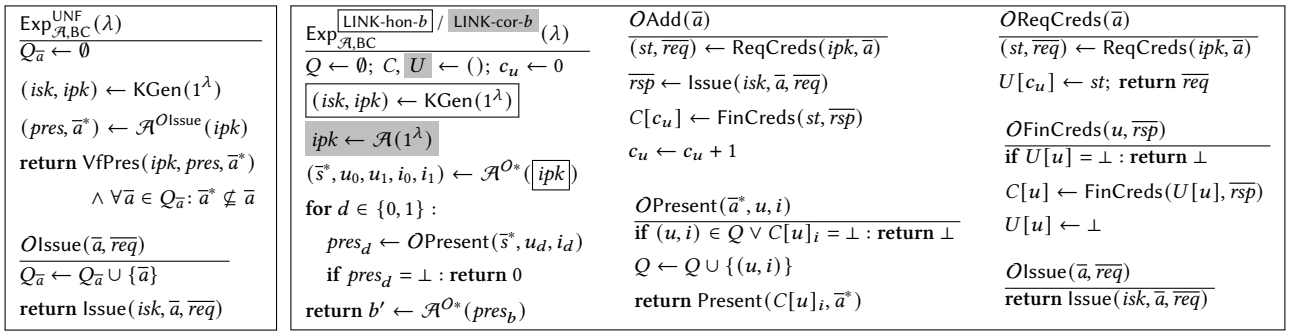


Figure 1: Left: Unforgeability. Right: Unlinkability with an honest or corrupt issuer. In game LINK-hon**, the adversary has access to oracles** $O* := (O\mathrm{Issue}, O\mathrm{Add}, O\mathrm{Present})$**. For game** LINK-cor**, it has access to** $O* := (O\mathrm{ReqCreds}, O\mathrm{FinCreds}, O\mathrm{Present})$**.**

this work. After that, we prove that unforgeability and honest-issuer unlinkability are achieved but that the approach is fundamentally insufficient for unlinkability against malicious issuers.

3.3.1 Construction. Fig. 2 shows our formalization of credentials from salted hashes (saltBC). A saltBC key pair is simply a signature key pair. Instead of signing a tuple of attributes directly, the issuer signs the hash of each attribute concatenated with a random salt, which is also given to the user. During the presentation, the user can selectively disclose some attributes by revealing their corresponding salts along with the signature and all the hashes. The verifier checks that the signature is valid on the hash values, and that the revealed attributes and their salts match the signed hashes.

Note that $\mathrm{saltBC}_{\mathrm{SIG,H}}$ is formally parameterized with SIG and H, but we omit the subscript for the sake of brevity from now on.

Simplifications. SD-JWT introduces additional mechanisms intended to improve privacy, namely, to hide the order and number of attributes contained in a credential. However, we find that, if not handled carefully, these mechanisms enable further tracing attacks. Since this result concerns only a specific standard, we present it separately in Sec. 6. Other details of the standards are omitted for simplicity. Firstly, we consider holder-binding out of scope, i.e., the binding of credentials to a user-protected key pair. Realizing such holder-binding for batch-issued credentials has its own challenges for privacy which are unrelated to the issues we attempt to solve here. Further, we do not model nested disclosures, or content that must always be disclosed.

3.3.2 Achieved Security. Below we analyze the unforgeability and unlinkability guarantees of saltBC. We only provide sketches for the proofs here and present the full proofs in App. D, that also includes the proof of correctness.

Theorem 3.4. saltBC *is (1) unforgeable if* SIG *is* UNF-CMA *and* H *is collision-resistant, and (2) honest-issuer-unlinkable if* H *is hiding. (3) It is not corrupt-issuer-unlinkable.*

Unforgeability. Unforgeability of the construction follows from the unforgeability of the signature and collision resistance of H: intuitively, a malicious party cannot create signatures valid under *ipk* on new arrays of hashes, and for the credentials it receives from the issuer, it knows only the one intended preimage to each hash. Thus, it can only convince verifiers about exactly the attributes that were attested by the issuer.

Honest-Issuer Unlinkability. The salted hashes make it hard to recover the salted preimage of a digest—specifically, if the hash function is hiding, hashing an attribute together with a random salt is indistinguishable from a truly random bit string of the same length. Furthermore, signatures in the presentations cannot be linked to each other as each signature is only presented once.

No Corrupt-Issuer Unlinkability. saltBC cannot achieve unlinkability against a corrupt issuer: since signature and salt values are revealed in a presentation, it suffices that the issuer records all credentials that it creates. Then a colluding verifier can later compare and de-anonymize any presentation it receives.

4 Modeling Subversion Attacks

The above attack on saltBC by a corrupt issuer clearly breaks unlinkability fully. However, it also requires continuous communication between issuer and verifier, which could be considered too risky for malicious issuers, who might be subjected to regular audits. This raises the question whether more subtle and convenient tracing attacks exist. We study this question by introducing a new unlinkability notion that considers malicious issuers and verifiers, but where the issuer corruption is restricted. While the issuer is still allowed to deviate from the honest issuance procedure, we prohibit any online communication between issuers and verifiers. Instead, the malicious issuer is allowed to send a single tracing key upfront to the verifier. The task of the adversary is to break the unlinkability of presentations with the help of that additional tracing key. The resulting notion subversion unlinkability is positioned between honest and corrupt-issuer unlinkability (see App. B.3).

If a scheme is not achieving that notion, it means the tracing key and the manipulated issuance enable immediate tracking by malicious verifiers. When studying such attacks, we want to focus on subtle and realistic attacks, which we capture through two additional security requirements. First, we want any subversion attack to be undetectable to users, i.e., the subverted credential must be indistinguishable from the original one, since noticeable deviations could alarm users. Thus, for honest parties that do not know the tracing key, the malicious behavior remains invisible. Second, we take into account that the issuer wants to trace, but not impair

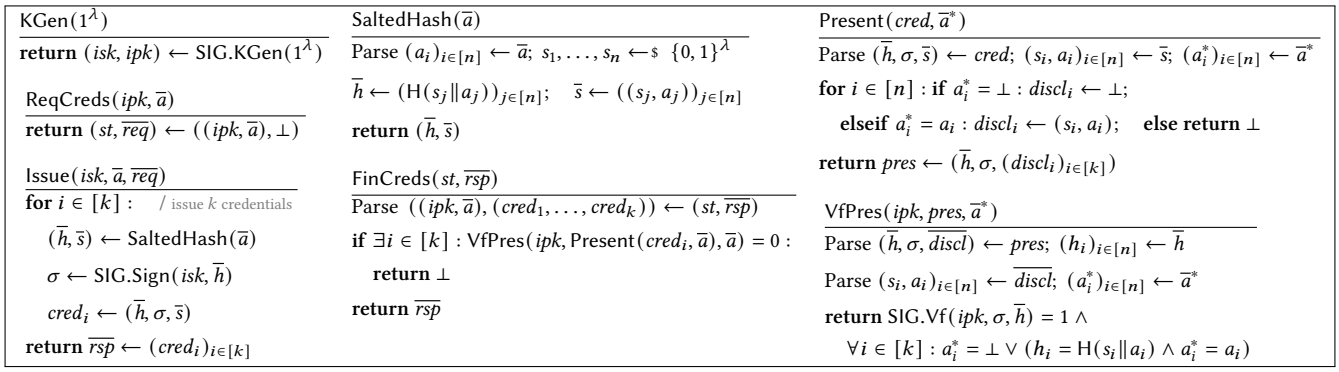


Figure 2: saltBC **algorithms with signature scheme** SIG **and hash function** H**. Recall: n is the number of attributes per credential and k the batch size.**

security, so subversion unforgeability demands that learning the tracing key must not help the adversary in forging credentials.

4.1 Subversion Unlinkability

Our new privacy notion follows the same goal as classic unlinkability, demanding that malicious verifiers receiving several presentations on the same attributes cannot tell whether they stem from the same credential or not. Here, we additionally consider the issuer to be corrupt and to be able to communicate with the verifier—but in a subverted and isolated way. To model the isolated attacks, we follow the concept of algorithm substitution or subversion [5, 39, 40]. That is, we allow the issuer's algorithms to be replaced by malicious versions KGen* and Issue*. The only communication between the issuer and verifier is through a tracing key *tk*, which the issuer generates and hands to adversary, i.e., the malicious verifiers. Note that the tracing key does not exist in the BC syntax, but appears only in the subverted algorithms. Leveraging the concept of subversion (which was invented for a different context) for this purpose might appear counter-intuitive at first, but it turns out to model exactly the limitations we wanted to impose on the issuer: it ensures that there is no direct communication channel to the verifier beyond *tk*. The static nature imposed by such a substituted issuance completely avoids that the issuer exfiltrates stateful information (e.g., the credentials it has previously issued) to the adversary via the legitimate issuance interfaces that are open to everyone.

We formally define subversion unlinkability in Fig. 3. The game relies on subverted algorithms BC* := (KGen*, Issue*). KGen* outputs a tracing key *tk*, besides an issuer key pair, which is revealed to the adversary and given to Issue*. Both KGen* and Issue* are stateless. Subversion unlinkability follows honest-issuer unlinkability in that the adversary can add arbitrary users to the system and tries to learn which user a presentation comes from; but now with the help of subverted key generation and issuance.

Definition 4.1 (Subversion Unlinkability). BC is subversion-unlinkable for the experiment in Fig. 3 if, ∀ PPT $\mathcal{A}$ and BC*, the following advantage is negligible in λ:

$$\left|\Pr\left[\mathsf{Exp}^{\text{LINK-sub-0}}_{\mathcal{A},\text{BC}^\star,\text{BC}}(\lambda)=1\right]-\Pr\left[\mathsf{Exp}^{\text{LINK-sub-1}}_{\mathcal{A},\text{BC}^\star,\text{BC}}(\lambda)=1\right]\right|$$

4.2 Undetectability

Looking at subversion unlinkability, tracing users based on storing all issued credentials (see Sec. 3.3) clearly is no longer possible. The remaining attack angle is to embed de-anonymizing information into credentials during Issue*, which the adversary can extract from presentations, with the help of the tracing key. In particular, we want to single out attacks that permit tracing but do not change the credentials in any way the user could notice, as otherwise the issuer could be detected as corrupt.

We draw once more from Bellare et al.'s [5] work on subversion attacks and model undetectability. In short, users should be unable to tell between corrupt (Issue*) and honest (Issue) issuance from received credentials. The game is shown in Fig. 3. The distinguisher, in the role of a user, interacts with $\mathcal{O}$Issue and may request as many credentials as they wish. After doing so, they must guess whether they were interacting with an honest or subverted issuer.

Definition 4.2 (Undetectability). BC* is undetectable for BC for the experiment in Fig. 3 if ∀ PPT $\mathcal{D}$, the following advantage is negligible in λ:

$$\left|\Pr\left[\mathsf{Exp}^{\text{DETECT-0}}_{\mathcal{D},\text{BC}^\star,\text{BC}}(\lambda)=1\right]-\Pr\left[\mathsf{Exp}^{\text{DETECT-1}}_{\mathcal{D},\text{BC}^\star,\text{BC}}(\lambda)=1\right]\right|$$

4.3 Subversion Unforgeability

Despite installing surveillance, the issuer might still want to preserve the overall unforgeability of credentials—also in the spirit of keeping their own corruption secret. Therefore, we impose a second restriction subversion unforgeability to eligible subversion attacks: credentials must remain unforgeable even for parties who learn the tracing key *tk*, i.e., the malicious verifiers. We adapt unforgeability accordingly in Fig. 3: the adversary obtains *tk* at the start, and issuance is subverted.

Definition 4.3 (Subversion Unforgeability). BC* is subversion-unforgeable for BC if ∀ PPT adversaries $\mathcal{A}$ the following advantage is negligible in λ for the experiment in Fig. 3:

$$\mathsf{Adv}^{\text{UNF-sub}}_{\mathcal{A},\text{BC}^\star,\text{BC}}(\lambda) := \Pr\left[\mathsf{Exp}^{\text{UNF-sub}}_{\mathcal{A},\text{BC}^\star,\text{BC}}(\lambda)=1\right]$$

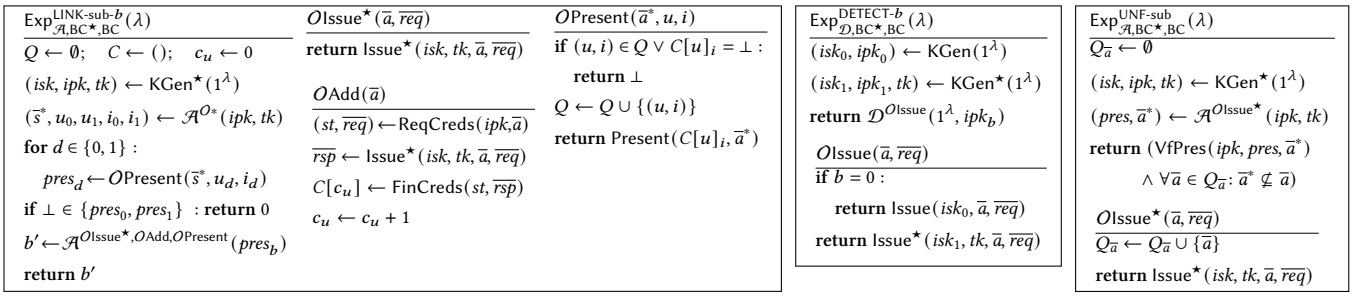


Figure 3: Modeling subversion attacks on batch-issued credentials. Left: Subversion unlinkability. Middle: Undetectability. Right: Subversion unforgeability.

5 Subversion Attacks on saltBC

This section presents two attacks against the subversion unlinkability of saltBC, both undetectable and subversion-unforgeable. The core idea is to embed user-specific identifiers into credentials' randomness. Specifically, our attacks exploit the salted hashes and the anamorphic capabilities of the underlying signature scheme, such as ECDSA. These attacks apply to any credential scheme relying on salted hashes and signatures with anamorphic capabilities.

To break subversion unlinkability, formally, even embedding a single tracing bit suffices. However, we demonstrate the severity of our attacks by estimating how much information can be embedded into a credential in total in Sec. 7.

5.1 Hiding Tracing Data in Salts

The underlying idea of the first attack is that the issuer embeds a user identifier into the salts. As an honest salt would be chosen randomly, the issuer hides its attack by encrypting the user identifier using pseudorandom encryption, and passes the encryption key to a colluding verifier via the tracing key. The verifier can then decrypt the user identifier from the salt of any disclosure, while the salts appear random to honest parties.

Attack Description. A minimal instantiation of the attack is given in Fig. 4, assuming an adversary that wants to trace users that have a special attribute a_{id}. We rely on a pseudorandom encryption scheme ENC with λ-bit ciphertexts. The subverted key generation algorithm $\mathsf{KGen}^\star$ samples an encryption key ek and includes both ek and a_{id} into the tracing key.

The subverted issuance algorithm $\mathsf{Issue}^\star$ behaves identically to the honest one except how it chooses the salts. $\mathsf{SaltedHash}^\star$ computes salts as a fresh encryption of a bit c that indicates whether the credential contains the traced attribute a_{id}, e.g., it embeds bit 1 into the salts if the user's attribute vector $\overline{a}$ contains a_{id}.

We now argue that the attack breaks subversion unlinkability but remains undetectable and subversion-unforgeable. We refer to App. D.4 for detailed proofs.

Theorem 5.1 (Salt Attack). *For the subverted credential scheme* $\mathsf{saltBC}^\star_s$ *in Fig. 4 and the attacker* $\mathcal{A}_s$ *in Fig. 4, the following holds:*

(1) $\mathsf{saltBC}^\star_s$ *is undetectable for* saltBC *if* ENC *is pseudorandom.*
(2) $\mathsf{saltBC}^\star_s$ *is subversion-unforgeable for* saltBC *if* SIG *is unforgeable and* H *is collision-resistant.*
(3) saltBC *is not subversion-unlinkable due to attacker* $\mathcal{A}_s$ *with subverted scheme* $\mathsf{saltBC}^\star_s$ *if* ENC *is correct.*

Undetectability. The sole difference between the subverted and the honest issuance is the generation of salts. In the honest scheme, salts are sampled randomly, whereas in the subverted scheme salts are ciphertexts. Thus, we can show that if the ciphertexts are pseudorandom, the subverted issuance is undetectable.

Subversion Unforgeability. $\mathsf{saltBC}^\star_s$ still computes the signatures honestly. Thus, forging a signature on a fresh payload requires to break the unforgeability of SIG. Furthermore, finding openings of a payload to distinct attribute vectors would break the collision resistance of the hash function, just as in the unforgeability proof. Thus, the subversion unforgeability property follows from the unforgeability of the signature scheme and the collision resistance of the hash function.

Subversion Unlinkability Attack. From the subverted algorithms above we define attacker $\mathcal{A}_s$ in Fig. 4. $\mathcal{A}_s$ breaks subversion unlinkability by issuing one credential that contains a_{id} and therefore encrypts 1 in the salts, and another credential without a_{id}, which hence encrypts 0 in the salts. For the challenge, $\mathcal{A}_s$ asks the game to present a shared attribute a (in Fig. 4: using the 0-th credential of the batch of either user). By decrypting the revealed salt of the presentation, $\mathcal{A}_s$ trivially determines the underlying credential under the correctness of ENC. Note that this attack assumes $n = 2$ and the attribute space $\mathcal{M}$ to have two distinct elements a_{id} and a.

5.2 Hiding Tracking Data in Signatures

Our next attack exploits a specific property of the signature scheme used in the saltBC construction. Kutyłowski et al. [29] defined anamorphism as a property for digital signatures. In a nutshell, this property provides a subverted signing algorithm that embeds an anamorphic message into signatures. These signatures are indistinguishable from the honestly created ones to any party which does not own the anamorphic double key adk.

In the context of saltBC, if the underlying signature scheme is anamorphic, the issuer can embed an identifier into the credential signature. Unsurprisingly, the subverted issuer can include the anamorphic double key as part of the tracing key, enabling the verifier to extract an identifier from the user's presentation. At the same time, the anamorphic properties ensure that undetectability and subversion-unforgeability are preserved.

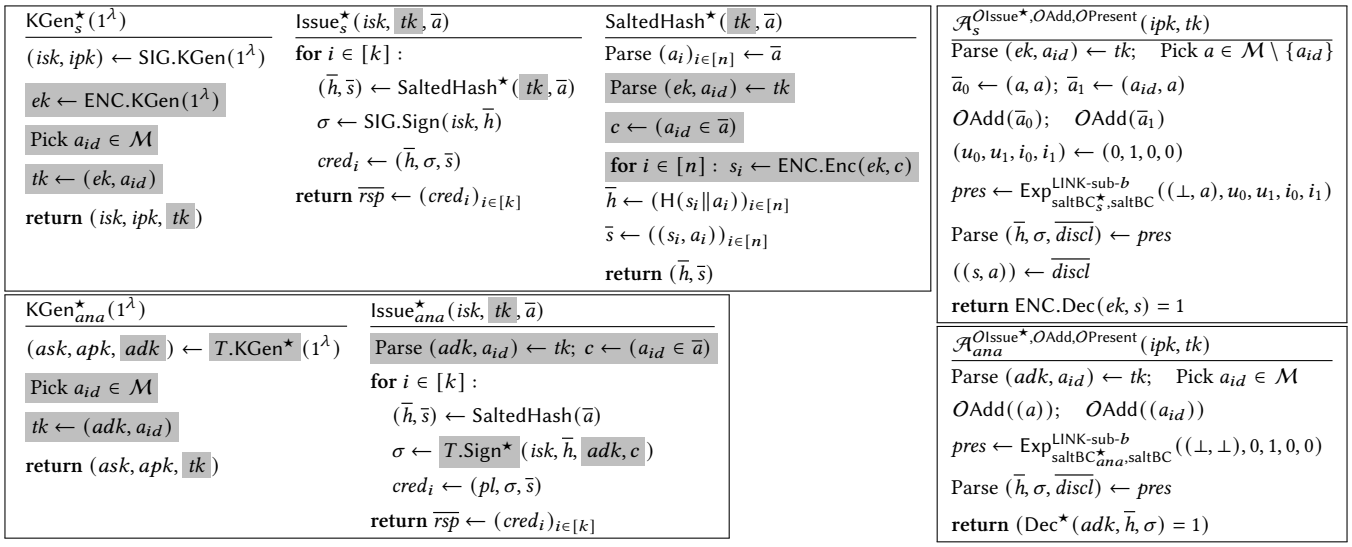


Figure 4: Left: Subverted issuer algorithms for the salt attack (top) and the signature attack (bottom). Let $\mathsf{saltBC}^\star_s = (\mathsf{KGen}^\star_s, \mathsf{Issue}^\star_s)$ **and** $\mathsf{saltBC}^\star_{ana} = (\mathsf{KGen}^\star_{ana}, \mathsf{Issue}^\star_{ana})$**. The differences to** saltBC **are highlighted. Right: Adversaries** $\mathcal{A}_s$ **and** $\mathcal{A}_{ana}$ **implementing respective attacks for** $n = 2$ **and** $k = 1$**.**

We stress that this is not a theoretic attack, as the existing standards and ongoing implementation efforts rely on ECDSA signatures [18, 26, 38] which admit subliminal channels due to the randomness used in the signature generation. We show that ECDSA is concretely anamorphic at the end of this section.

Anamorphic Signatures. Anamorphic signatures (formally introduced in App. A) are accompanied by an anamorphic triplet:

Definition 5.2 (Anamorphic Triplet). An anamorphic triplet $T = (\mathsf{KGen}^\star, \mathsf{Sign}^\star, \mathsf{Dec}^\star)$ for message space $\mathcal{M}$ and anamorphic message space $\mathcal{M}_A$ consists of three PPT algorithms:

- $\mathsf{KGen}^\star(1^\lambda) \to (ask, apk, adk)$ outputs the anamorphic signing key, public key, and double key.
- $\mathsf{Sign}^\star(ask, m, adk, am) \to \sigma$ takes an anamorphic signing key, a regular message to be signed, a double key, and an anamorphic message. It outputs the anamorphic signature.
- $\mathsf{Dec}^\star(adk, m, \sigma) \to am$ takes the double key, a message, and an anamorphic signature on that message. It outputs the anamorphic message.

Together with the regular signature scheme $\mathsf{SIG} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$, the triplet T has the following informal properties [29]:

- **Anamorphic Correctness:** The anamorphic decryption of an anamorphic signature $\sigma \leftarrow \mathsf{Sign}^\star(ask, m, adk, am)$ must output message am with overwhelming probability.
- **Anamorphism:** Anamorphic signatures are indistinguishable from the honest signatures to all parties which do not hold the double key adk.
- **Private Anamorphism:** Signatures remain unforgeable even to an adversary that knows the anamorphic double key adk.

Attack Description. Let us now demonstrate how subversion unlinkability of saltBC is broken if the underlying signature scheme is anamorphic. The subverted scheme $\mathsf{saltBC}^\star_{ana}$ is presented in Fig. 4, where we once more tracks some attribute a_{id} among users. In $\mathsf{KGen}^\star$, the key generation of SIG is replaced with the anamorphic key generation. Then the tracing key is set to the double key and a_{id}. $\mathsf{Issue}^\star$ runs the honest issuance but replaces the signature generation with anamorphic signing in order to embed a user identifier. We sketch the arguments for following theorem below and refer to App. D.5 for a detailed proof.

Theorem 5.3 (Signature Attack). *Let T be an anamorphic triplet for the signature scheme* SIG *underlying* $\mathsf{saltBC}^\star_{ana}$*. Let $\mathcal{A}_{ana}$ be as defined in Fig. 4.*

1. $\mathsf{saltBC}^\star_{ana}$ *is undetectable for* $\mathsf{saltBC}_{\mathsf{ANA}}$ *if* SIG *is anamorphic for anamorphic triplet T.*
2. $\mathsf{saltBC}^\star_{ana}$ *is subversion-unforgeable for* saltBC *if* SIG *is private anamorphic for T and* H *is collision-resistant.*
3. saltBC_{ana} *is not subversion-unlinkable against $\mathcal{A}_{ana}$ and subversion* $\mathsf{saltBC}^\star_{ana}$ *if T is anamorphically correct.*

Undetectability. Anamorphism of SIG immediately implies undetectability: it guarantees that anamorphic signatures are indistinguishable from honest ones.

Subversion Unforgeability. For subversion-unforgeability we rely on the private anamorphism of SIG. This property ensures that the double key gives no advantage to the verifier in forging signatures.

Subversion Unlinkability Attack. Assuming SIG achieves anamorphic correctness, it is straightforward to embed information about a credential into its signature. The attacker $\mathcal{A}_{ana}$ creates two users out of which only one has the special attribute a_{id}, then obtains a challenge presentation. It decrypts the signature contained in the presentation with the anamorphic double key, which immediately reveals the underlying user.

ECDSA Signatures are Anamorphic. While the potential for an anamorphism in ECDSA signatures has been recognized [22], this claim has, to our knowledge, not been proven formally. The rough idea is that ECDSA signatures sample a random scalar $\kappa \leftarrow\$ [q]$ for each signature, which can be extracted from a signature when knowing the signing key sk. Naïvely, one could replace κ by the random encryption of an anamorphic. However, this would require including sk in the double key, which is undesirable for private anamorphism. An alternative was described for ElGamal signatures by [29] and works as follows for ECDSA: we include in adk a PRF key k, then embed a message am into a signature by rejection-sampling the ECDSA signature (r, s) until $\mathsf{F}(k, r) = am$.

We parametrize T by anamorphic message length ℓ and the number of repeats t for the rejection sampling in the anamorphic signing algorithm. Furthermore, T relies on a pseudorandom function F with input space $\mathbb{G}$ and output space $\{0,1\}^\ell$.

$\mathsf{KGen}^\star(1^\lambda)$: Set $(d, Q) \leftarrow \mathsf{ECDSA.KGen}(1^\lambda)$ and get $k_\mathsf{F} \leftarrow \mathsf{F.KGen}(1^\lambda)$. Set $adk \leftarrow (k_\mathsf{F}, Q)$. Return (d, Q, adk).

$\mathsf{Sign}^\star(ask, m, adk, am)$: Parse $d \leftarrow ask$ and $(k_\mathsf{F}, Q) \leftarrow adk$. Then, sample $\kappa_1, \ldots, \kappa_t \leftarrow\$ \mathbb{Z}_q^*$. Search for $i^* \in [t]$, such that $\mathsf{F}(k_\mathsf{F}, \kappa_{i^*} G) = am$. If there is no such i^*, set $i^* = 1$. Return $\sigma \leftarrow \mathsf{ECDSA.Sign}(sk, m; \kappa_{i^*})$.

$\mathsf{Dec}^\star(adk, m, \sigma)$: Parse $(k_\mathsf{F}, Q) \leftarrow adk$ and $(r, s) \leftarrow \sigma$. If $(r, s) \notin \mathbb{Z}_q^*$, return $\perp$. Otherwise, compute $X = \mathsf{H}(m)s^{-1}G + rs^{-1}Q$ and return $\mathsf{F}(k_\mathsf{F}, X)$.

The anamorphic triplet achieves the desired anamorphic properties, which we prove formally in App. C.

Theorem 5.4 (Informal). *The anamorphic triplet T satisfies (1) anamorphic correctness and (2) anamorphism if* F *is pseudorandom, and (3) private anamorphism if ECDSA is unforgeable.*

Rejection-Sampling Hashes? Inspired by the anamorphism of ECDSA, note we could perform a similar attack on the hashes, where we re-sample salts until $\mathsf{F}(k, h) = m$ for some message m the issuer wants to embed. Since hashes are always revealed in a presentation, this is particularly beneficial for credentials that contain many attributes that must remain hidden for privacy-sensitive use cases. However, as we show in Sec. 7, the simpler embedding in the salt yields much more space for tracking information than the rejection-sampling technique, and thus we did not explore this additional attack further.

6 Subversion Attacks specific to SD-JWT

We observe further attack vectors specific to design decisions of the SD-JWT standard [21]. We stress that this is not because SD-JWT is fundamentally more vulnerable than its alternative ISO/IEC 18013-5, but we focused more on this standard simply because it is openly available such that the following claims are easy to validate. To set the stage, we first describe the vulnerable features added by the SD-JWT standard and update our saltBC construction to reflect these changes. Then, we present the arising subversion attacks.

6.1 Modeling Additional Features of SD-JWT

The SD-JWT standard aspires to implement privacy-preserving mechanisms beyond what we modeled so far. Specifically, it encompasses attribute order hiding and attribute count hiding, meaning that from user presentations, the verifier cannot tell how many attributes there are in the underlying credential, or what order the attributes are in. This is relevant to privacy because order and count can be a "side channel" to credential details of the user. To illustrate, imagine that the second attribute is "nationality", and if a user has more than one nationality, these take up the following spaces; it follows the age attribute. Then, this number of nationalities will be leaked by the overall number of attributes, or by the attribute position if the user only presents their age.

Hiding the Attribute Order. To hide the attribute order during presentations, the SD-JWT standard suggests to apply a function Order to the hashes before signing and including it in the credential. The standard permits any ordering method, as long as the original order of attributes is hidden, and suggests shuffling randomly or sorting alphanumerically [21, Section 4.2.4.1]. Looking ahead, random ordering will enable a subversion attack.

Hiding the Attribute Count. To hide how many attributes a credential contains during presentations, the standard enforces decoy digests: it assumes a maximum number of attributes n and pads smaller credentials with dummy attributes, which are simply random salts that are hashed by the same hash function as regular salted attributes. These dummy hashes are called "decoy digests" in the standard.

Credentials on Arbitrary JSON Objects. Note that the original SD-JWT specification supports issuance of credentials on arbitrary JSON objects and enables a wide range of selective disclosure mechanisms through JSON objects in different structures, e.g., using nested structures. For the purposes of this work, we abstract these capabilities away and only model the aspects necessary to authenticate an unordered set of attributes and express attribute order hiding and attribute count hiding features. It is an interesting future work to analyze whether other JSON object structures give more room for subversion attacks.

Construction sdjwtBC. We modify the saltBC construction to capture these two additional features and call the extended construction sdjwtBC. To account for the lack of ordering, most of the vectors in the syntax become now sets: attributes, salts, hashes, and the salt-attribute pairs in credentials. Semantically, n now describes the maximum number of attributes instead of a fixed one. Fig. 5 gives the modified algorithms for the sdjwtBC construction, note that, as for saltBC, the construction sdjwtBC is parametrized with a signature scheme SIG and a hash function H. It is additionally parametrized by an ordering function Order that returns a permutation of its inputs.

6.2 Hiding Tracing Data in the Digest Order

The standard allows the issuer to freely choose the order of the digests, and crucially, this order is forwarded without modification from user to verifier. We show that if sdjwtBC is instantiated with a random ordering function Order = randomOrder, which returns the digests in a random permutation, then this leeway can be exploited by the issuer for a subversion attack.

Attack Description. The core observation is that a user identifier can be encoded by the order in which the credential contains

SaltedHash(A)	Present$(cred, \overline{a}^*)$	VfPres$(ipk, pres, \overline{a}^*)$
Parse $\{a_1, \ldots, a_{\lvert A\rvert}\} \leftarrow A$; $s_1, \ldots, s_n \leftarrow\$ \{0,1\}^\lambda$	Parse $(pl, \sigma, S) \leftarrow cred$	$(pl, \sigma, Discl) \leftarrow pres$
/ Normal and decoy hashes:	**if** $\exists a_i^* \in \overline{a}^* : \forall (s,a) \in S : a_i^* \neq a$: **return** $\bot$	**return** SIG.Vf$(ipk, pl, \sigma) = 1 \wedge$
$H \leftarrow \{\mathsf{H}(s_j \Vert a_j)\}_{j\in[\lvert A\rvert]} \cup \{\mathsf{H}(s_{\lvert A\rvert+j})\}_{j\in[n-\lvert A\rvert]}$	$Discl \leftarrow \{(s,a) \in S \mid a \in \overline{a}^*\}$	$\forall (s,a) \in Discl : \mathsf{H}(s \Vert a) \in pl \wedge$
return $(pl \leftarrow \mathsf{Order}(H), S \leftarrow \{(s_j, a_j)\}_{j\in[\lvert A\rvert]})$	**return** $pres \leftarrow (pl, \sigma, Discl)$	$\forall a_i^* \in \overline{a}^* : \exists (s,a) \in Discl : a = a_i^*$

Figure 5: sdjwtBC **algorithms with signature scheme** SIG**, a hash function** H**, and an ordering function** Order**. The algorithms** KGen**,** ReqCreds**,** Issue**, and** FinCreds **are identical to Fig. 2, with the newly unordered artifacts as sets instead of vectors. Recall:** n **is the number of attributes per credential and** k **the batch size.**

its digests. A value $m \in [\varepsilon!]$ can be mapped to a permutation of ε elements for which a strict total order exists and this mapping is invertible; this is an established problem with efficient solutions [31]. We assume an algorithm MsgToPermutation$(list, \varepsilon, m)$, which re-orders the first ε elements of the *list* to encode a message $m \in [\varepsilon!]$, and a second algorithm PermutationToMsg$(list, \varepsilon)$, which decodes the order of the first ε elements of the *list* into a message $m \in [\varepsilon!]$. Note that we can assume a correct decoding, i.e., PermutationToMsg(MsgToPermutation$(list, \varepsilon, m), \varepsilon) = m$, as long as strict total order of the first ε elements exists. Since we permute a list of (w.h.p. distinct) hash values, we can use the alphanumerical order as the 'standard' total order. As a last step, to make the permuted order indistinguishable from the expected random order of randomOrder, we encrypt the user identifier with a pseudorandom encryption scheme ENC, before embedding it into the hash order. Practically, to embed ciphertexts of ENC with a ciphertext space of size $w(\lambda)!$, we need that $n \geq w(\lambda)$. The full subverted scheme sdjwtBC$^\star_{ord}$ is given in Fig. 6. A full proof of the attack is provided in App E, with the main arguments sketched in the following.

Theorem 6.1 (Digest Order Attack). *For the subverted scheme* sdjwtBC$^\star_{ord}$ *and attacker* $\mathcal{A}_{ord}$ *in Fig. 6, the following holds:*

1. sdjwtBC$^\star_{ord}$ *is undetectable for* sdjwtBC *with* randomOrder *if* ENC *is pseudorandom and* H *is collision-resistant.*
2. sdjwtBC$^\star_{ord}$ *is subversion-unforgeable for* sdjwtBC *if* SIG *is unforgeable and* H *is collision-resistant.*
3. sdjwtBC *is not subversion-unlinkable due to attacker* $\mathcal{A}_{ord}$ *with subverted scheme* sdjwtBC$^\star_{ord}$ *if* ENC *is correct and* H *is collision-resistant.*

Undetectability. Comparing the hash ordering created by our subversion attack to the random order of randomOrder, the change should be unnoticeable to an honest user, since the pseudorandom encryption creates a pseudorandom ciphertext, which is mapped bijectively to a permutation. If an adversary could distinguish between the permutation-encoding of a random string and the permutation-encoding of the ciphertext, then it would also break the pseudorandomness of ENC.

Subversion Unforgeability. The subversion does not obstruct unforgeability of the base scheme sdjwtBC, which is analogous to unforgeability of saltBC. In particular, because SIG is unforgeable and the message received by the corrupt verifier is generated independently from the signature's secret key, an adversary cannot forge a signature on a new *pl*. On the other hand, reusing a signature for a different preimage of *pl* requires finding a fresh salt-attribute preimage of either (1) an attribute digest, or (2) a decoy digest. Both is hard for a collision-resistant H.

Subversion Unlinkability Attack. $\mathcal{A}_{ord}$ distinguishes two credentials out of which one contains a special attribute a_{id}, as in prior attacks. Issue$^\star_{ord}$ encodes a bit that signifies the presence of a_{id} in the order of the first $w(\lambda)$ hash values of *pl*, using MsgToPermutation. Decoding with PermutationToMsg then succeeds if the first $w(\lambda)$ hash values in the *pl* are distinct, which happens with overwhelming probability, and decryption succeeds if ENC is correct. By retrieving the identifier, $\mathcal{A}_{ord}$ breaks subversion unlinkability.

6.3 Hiding Tracing Data in Decoy Digests

Decoy digests are hashes of random salts chosen by the issuer, yet these salts are not revealed to the user and the user has no way of verifying how the decoy digests were generated. Thus, a malicious issuer can exploit the decoy digests as a subliminal channel.

Attack Description. Similar to the salt attack, the issuer replaces the decoy digests with pseudorandom encryptions of a user identifier. As the decoy digests are included in each presentation, a verifier can extract and decrypt them even if no attribute is disclosed. The subverted issuer algorithms of the decoy-digest attack are identical to those of the order attack, except SaltedHash*.

We sketch the main arguments for the following theorem here and provide the full proofs in App. E.

Theorem 6.2 (Decoy Digest Attack). *For the subverted credential scheme* sdjwtBC$^\star_{dd}$ *in Fig. 6 and the attacker* $\mathcal{A}_{dd}$ *in Fig. 6, the following holds:*

1. sdjwtBC$^\star_{dd}$ *is undetectable for* sdjwtBC *if* ENC *is pseudorandom and* H *is a non-programmable random oracle.*
2. sdjwtBC$^\star_{dd}$ *is subversion-unforgeable for* sdjwtBC *if* SIG *is unforgeable and* H *is a non-programmable random oracle*
3. sdjwtBC *is not subversion-unlinkable due to attacker* $\mathcal{A}_{dd}$ *with subverted scheme* sdjwtBC$^\star_{dd}$ *if* ENC *is correct.*

Undetectability. The user only ever sees the decoy digests, not the underlying salts. Therefore, assuming the hash function to be a random oracle, the honest decoy digests look like perfectly random values to the adversary. In the attack, the issuer swaps the decoy digests with pseudorandom ciphertexts, which are indistinguishable from the random decoy digests.

Subversion Unforgeability. As in previous proofs, unforgeability is preserved since the tracing key is completely independent of the signing key—forging the signature of the credential remains hard. Neither can an adversary reuse a signed payload *pl* while

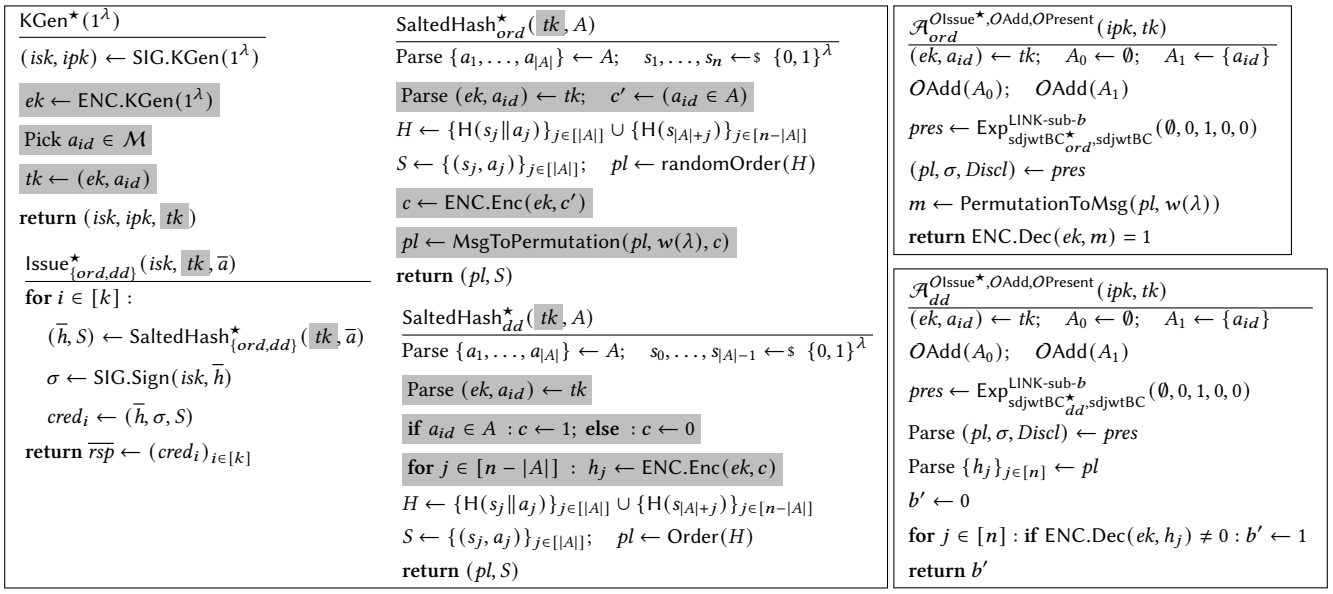


Figure 6: Left: Subverted algorithms for the order and decoy-digest attack. Let $\mathsf{sdjwtBC}^{\star}_{ord} = (\mathsf{KGen}^{\star}, \mathsf{Issue}^{\star}_{ord})$ and $\mathsf{sdjwtBC}^{\star}_{dd} = (\mathsf{KGen}^{\star}, \mathsf{Issue}^{\star}_{dd})$. The differences to the honest algorithms are highlighted in gray. Top right: Adversary $\mathcal{A}_{ord}$ implementing the order attack. Bottom right: Adversary $\mathcal{A}_{dd}$ implementing the decoy-digest attack.

presenting different attributes; this would require finding a different salt-attribute preimage to one of the regular hashes, or to one of the decoy digests. Both is hard if H is a random oracle.

Subversion Unlinkability Attack. As before, $\mathcal{A}_{dd}$ calls the oracle $\mathcal{O}\mathsf{Add}$ to issue one credential that contains a_{id} and one that does not, it extracts all hashes from the presentation received and tries to decrypt them. It returns 0 if all hashes decrypt to 0 or 1 otherwise. From the correctness of ENC follows that $\mathcal{A}_{dd}$ outputs the correct bit in either world.

7 Concrete Tracking Capabilities

For presentation purposes we so far embedded a single bit into credentials, which breaks subversion unlinkability formally. Now, we explore the practical impact by examining how much tracing data the attacks can realistically embed into a single presentation.

We instantiate the pseudorandom encryption by the PRF-based scheme PRFEnc (see App. A). This method encrypts ℓ-bit messages into 2ℓ-bit ciphertexts.

For the salt attack, recall that $\mathsf{saltBC}^{\star}_{s}$ relies on a λ-bit ciphertext, so we can encode $\lambda/2$ bits per salt and a maximum of $n\lambda/2$ bits if the verifier is able to see a presentation of all attributes. The mandatory person identification data for a natural person contains five attributes [17], but combining all mandatory and optional attributes of ISO/IEC 18013-5, there might be up to 33 attributes [26]. Using SHA-256 and ECDSA over a 256-bit curve yields $\lambda = 256$, so disclosing even one attribute allows for 128 tracing bits.

The impact of the signature attack depends on the concrete signature and anamorphic triplet. For ECDSA, a message m is encoded by rejection-sampling signatures (r, s) until $\mathsf{F}(k, r) = m$. If F is pseudorandom, the probability of finding a matching signature for a message of length ℓ is $1/2^{\ell}$. So after t tries the probability that the rejection sampling terminated is $1 - (1 - 1/2^{\ell})^{t}$. To illustrate, for embedding tracing data of length $\ell = 10$, rejection sampling has terminated with $> 1/2$ probability after 710 tries, for $\ell = 20$ after 726,818 tries, or for $\ell = 30$ after 744,261,118 tries. Importantly, this does not have to happen online but if the attributes are known, credentials are entirely pre-computable.

As for the additional, SD-JWT-specific attacks, the decoy-digest attack can encode $(n - |\bar{a}|) \cdot \frac{\lambda}{2}$ bits if there are $(n - |\bar{a}|)$-many decoy digests. The order attack only adds $\frac{n}{4}$ bits.

Observe that a unique identifier for every EU citizen, as of 2026, requires only 29 bits, and a unique identifier for every person of the world costs 33 bits.

8 Securing saltBC Against Subversion Attacks

A natural solution to avoid the subversion attacks is to never reveal the original credential, but instead let the user merely make a zero-knowledge proof of an issuer's signature on the selected subset of attributes. This is the approach of anonymous credentials, and even yields the stronger corrupt-issuer unlinkability. However, such schemes either rely on pairing-based cryptography or circuit-based SNARKS, which both do not enjoy the same level of maturity and deployment readiness as classic signatures. Therefore, we investigate how batch-issuance approaches can be lifted to achieve at least subversion unlinkability. In this section, we sketch mitigations against the attacks in Sec. 5, then present our improved subversion-resilient scheme: saltBC+, which maintains the core feature of the saltBC construction—its simplicity and relying solely on standard(ish) primitives—and provably achieves the desired

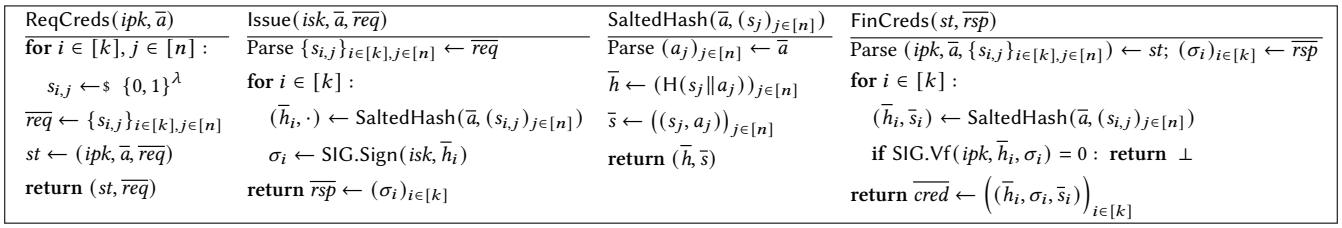


Figure 7: The saltBC+ **construction.** KGen, Present, **and** VfPres **are identical to** saltBC**.**

security and privacy goals. Finally, we also sketch fixes for the SD-JWT-specific attacks from Sec. 6.

8.1 Securing saltBC against Subversion Attacks

Preventing the Salt Attack. This attack exploits the issuer's freedom to choose the salts for attribute digests. One mitigation is to let the user choose the salts and send them as part of the credential request $\overline{req}$. A caveat of this mitigation is that it requires the user's wallet instance to be trusted; yet it has the advantage of requiring no additional round of interaction and leaving the credential structure intact. Alternatives are generating randomness jointly, or deriving randomness from a PRF. The former, however, introduces additional rounds. For the latter, the issuer would use a PRF to derive the signature randomness, then prove to the user that the PRF is correctly evaluated for each signature. But for ECDSA signatures, this would require heavy machinery such as exponent VRFs [9] or circuit-based NIZK proofs [7].

Preventing the Signature Attack. The second attack relies on randomness introduced into the signature by the issuer. In fact, as long as an issuer has multiple options for a valid signature, it can exploit this to encode information. Hence, unique signatures, proposed by Lysyanskaya [30], remedy this exact issue. In particular, a unique signature scheme enforces that there is only one signature to each *ipk*-message pair, efficiently preventing the issuer from embedding information by its choice of signature. Unique signatures are also proposed by Ateniese et al. [2] to achieve subversion resilience.

There are numerous constructions of unique signatures based on RSA [15] and pairings [10, 25, 30]. A candidate that stands out as ready-to-use is the Full-Domain-Hash RSA scheme (RSA-FDH) [15], which gives unique signatures if the signer public keys are well-formed. RSA-FDH is standardized as part of the IRTF's document on VRFs (RFC 9381) [23], and accessible on HSMs following the PKCS#11 standard [12], which gives an interface for raw RSA exponentiation through X.509 (raw) RSA. The most compact and efficient unique signature are BLS signatures [10], which also undergo standardization in the IETF [8]. They require pairing-friendly curves though. If such curves are acceptable, then anonymous credentials are also within reach, offering even stronger privacy.

Improved Construction is Secure. saltBC+ (see Fig. 7) introduces the above mitigations: in comparison to saltBC, it samples the salts on the user side, and relies on a unique signature scheme. We show that this yields the desired hardening formally in App. F.

Theorem 8.1 (Informal). saltBC+ *is (1) unforgeable if* H *is collision-resistant and* SIG *achieves* UNF-CMA*; it is (2) subversion-unlinkable if* H *is hiding and* SIG *is a unique signature scheme.*

Unforgeability. The user's increased control over the payload that is signed in a credential leaves the argument for unforgeability unchanged. This is because the signed hashes still commit to the attributes under collision resistance of the hash function. Hence, a successful saltBC+ forger has to forge a signature for the underlying signature scheme.

Subversion Unlinkability. It remains to show that our modifications indeed make the scheme subversion-unlinkable. The intuition is that since the issuer only chooses the signatures and these are unique, there also is a unique valid response $\overline{rsp}$ to each credential request $\overline{req}$. Any dishonest response from a subverted issuance algorithm is now rejected by the FinCreds algorithm. But from Theorem 3.4 we know that presentations from honestly generated credentials are indeed unlinkable.

8.2 Securing sdjwtBC Against Subversion Attacks

The attacks exploiting SD-JWT's attribute-order and attribute-count hiding are similarly prevented by removing any margin for variation in the credential issuance. For a fully secure version of sdjwtBC, the fixes from this section need to be combined with the solutions from the prior section. We omit a formal analysis, since the argumentation is analogous, and sdjwtBC was merely an auxiliary construction to showcase further attack vectors.

Preventing the Order Attack. To prevent the issuer from using a variable digest order as a subliminal channel, the standard would need to fix a single $\mathcal{A}_{ord}$ function (e.g., alphanumeric ordering) for the hashes, and have the user verify the correct ordering.

Preventing the Decoy-Digest Attack. This subversion attack is similar to the salt attack, so instead of allowing the issuer to freely choose the input salt to decoy digests, we suggest salts to be chosen by the user and sent as part of the issuance request.

Further Considerations for a Secure Implementation. It needs to be stressed that our safeguards in saltBC+ are on the cryptographic and data format level. But subliminal channels can still be introduced at other points of an implementation. For instance, Zaverucha and Shumov [41, Section 4] show that encodings of signatures are not necessarily unique, even if the underlying signature is. While abusing different encoding options for a subliminal would fortunately be detectable, this observation highlights that it remains

an intricate task to eliminate all ways to tracing users, even when considering a strongly limited issuer.

9 Conclusion and Limitations

In this work, we identify tracing attacks on batch-issued credential systems (as adopted for the first version of the EUDI wallet), where a malicious issuer can exploit subliminal channels within the credentials. Additional attacks on features of the SD-JWT standard showcase that apparent implementation details may hide additional attack vectors and concrete standards need to be carefully vetted for the use case of digital identity systems. Notably, all attacks are undetectable through inspection of the credentials, and could be launched at any point, even in an already running system.

Holder binding does not affect the subversion attacks we consider in this work. Holder binding for batch-issued credentials [26, 37] assumes that the user (or, more precisely, their device) generates a fresh holder key pair for every credential. That is, each single-use credential contains a single-use device public key as an attested attribute, and presentation requires a signature under the corresponding device secret key for a nonce sent by the verifier. Since this key and signature generation is under the user's sole control, it neither introduces additional attack vectors nor removes the ones we discuss. That said, one could extend our study to also treat the secure element within the user's device as subverted, and analyze what additional protections are possible under that stronger threat model. Further, many EU member states appear to be adopting a "remote HSM" approach to holder binding, which would likewise require a different security model and analysis. We view such a study as interesting future work.

Our proposed fix saltBC+ shows a way out of traceable credentials without sacrificing simplicity. However, we remark that while there exist minimal fixes for the salt, order, and decoy-digest attacks, preventing the subversion of ECDSA signatures is harder. To mitigate subversion, one would have to replace ECDSA with a unique signature scheme such as RSA-FDH or BLS. This is a major change on the issuer side, and thus rather unlikely to be adopted. However, it highlights signature uniqueness as a helpful property to avoid subliminal channels when they can be abused for tracking. It is an interesting open problem how to achieve this for post-quantum solutions. Our work can be seen as yet another motivation to move away from accumulating patches to the current salted-hashes design, and instead adopt anonymous credentials that natively support the desired security and privacy properties.

Acknowledgments

This research was partially funded by the HPI Research School on Systems Design. The authors thank Greg Zaverucha for a helpful comment on the unique encodings of signatures.

Ethical Considerations

This work involved no human subjects and was not submitted to an external ethics panel.

Ethics of Our Results. This work treats the topic of mass surveillance enabled by digital identity systems. We lay open the possibilities of tracing the use of personal credentials with the goal of raising awareness and educating citizens about the risks of the first EUDI version and, in general, solutions based on batch-issued, salted-hash-based credentials. By no means is this a "tutorial" for malicious issuers to enable tracing; since the presented attacks build on established techniques, we believe that the same results could have been reproduced by a resourceful issuing party. So instead of letting our privacy depend on the fact that no malicious party discovers any tracing method, we deem it more valuable to clearly and publicly describe how the current system is fundamentally vulnerable to subversion and tracking attacks, and that this can only be fixed by changing the issuance mechanism.

Open Science

The bulk of this work is of formal nature and verifiable without auxiliary material. For demonstration purposes, we provide a proof-of-concept of the attacks we identified, which can be found at https://github.com/abc013/sd-jwt-python.

AI Use

AI-based tools were not used in conceiving and writing this work. The proof-of-concept was developed without AI, but AI was used to polish the code, and write the quick-start script.

References

[1] Ross J. Anderson, Serge Vaudenay, Bart Preneel, and Kaisa Nyberg. 1996. The Newton Channel. In Proceedings of the First International Workshop on Information Hiding. Springer-Verlag, Berlin, Heidelberg, 151–156.

[2] Giuseppe Ateniese, Bernardo Magri, and Daniele Venturi. 2015. Subversion-Resilient Signature Schemes. In Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security (Denver, Colorado, USA) (CCS '15). Association for Computing Machinery, New York, NY, USA, 364–375. doi:10.1145/2810103.2813635

[3] Foteini Baldimtsi and Anna Lysyanskaya. 2013. Anonymous credentials light. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (Berlin, Germany) (CCS '13). Association for Computing Machinery, New York, NY, USA, 1087–1098. doi:10.1145/2508859.2516687

[4] Carsten Baum, Olivier Blazy, Jan Camenisch, Jaap-Henk Hoepman, Eysa Lee, Anja Lehmann, Anna Lysyanskaya, René Mayrhofer, Hart Montgomery, Ngoc Khanh Nguyen, Bart Preneel, abhi shelat, Daniel Slamanig, Stefano Tessaro, Søren Eller Thomsen, and Carmela Troncoso. 2024. Cryptographers' Feedback on the EU Digital Identity's ARF. https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/discussions/211. available online at https://github.com/user-attachments/files/15904122/cryptographers-feedback.pdf.

[5] Mihir Bellare, Kenneth G. Paterson, and Phillip Rogaway. 2014. Security of Symmetric Encryption against Mass Surveillance. In Advances in Cryptology – CRYPTO 2014, Juan A. Garay and Rosario Gennaro (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–19.

[6] Mihir Bellare and Phillip Rogaway. 1993. Random oracles are practical: A paradigm for designing efficient protocols. In Proceedings of the 1st ACM Conference on Computer and Communications Security. Association for Computing Machinery, New York, NY, USA, 62–73. https://doi.org/10.1145/168588.168596

[7] Jens-Matthias Bohli, María Isabel González Vasco, and Rainer Steinwandt. 2007. A Subliminal-Free Variant of ECDSA. In Information Hiding, Jan L. Camenisch, Christian S. Collberg, Neil F. Johnson, and Phil Sallee (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 375–387.

[8] Dan Boneh, John Bradley, Sergey Gorbunov, Riad S. Wahby, Hoeteck Wee, Christopher A. Wood, and Zhenfei Zhang. 2025. BLS Signatures. Internet-Draft draft-irtf-cfrg-bls-signature-06. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-irtf-cfrg-bls-signature/06/ Work in Progress.

[9] Dan Boneh, Iftach Haitner, Yehuda Lindell, and Gil Segev. 2025. Exponent-VRFs and Their Applications. In Advances in Cryptology – EUROCRYPT 2025: 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4–8, 2025, Proceedings, Part VII (Madrid, Spain). Springer-Verlag, Berlin, Heidelberg, 195–224. doi:10.1007/978-3-031-91098-2_8

[10] Dan Boneh, Ben Lynn, and Hovav Shacham. 2004. Short Signatures from the Weil Pairing. J. Cryptol. 17, 4 (2004), 297–319. doi:10.1007/S00145-004-0314-9

[11] Dan Boneh and Victor Shoup. 2023. A Graduate Course in Applied Cryptography.

[12] Dieter Bong and Tony Cox. 2022. PKCS #11 Specification Version 3.1. OASIS Committee Specification 01 CS01. OASIS Open. https://docs.oasis-open.org/pkcs11/pkcs11-spec/v3.1/cs01/pkcs11-spec-v3.1-cs01.html Approved as OASIS Committee Specification.

[13] Jan Camenisch and Anna Lysyanskaya. 2001. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In Advances in Cryptology — EUROCRYPT 2001, Birgit Pfitzmann (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 93–118.

[14] Rutchathon Chairattana-Apirom, Franklin Harding, Anna Lysyanskaya, and Stefano Tessaro. 2025. Server-Aided Anonymous Credentials. In Advances in Cryptology – CRYPTO 2025: 45th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 17–21, 2025, Proceedings, Part VI (Santa Barbara, CA, USA). Springer-Verlag, Berlin, Heidelberg, 291–324. doi:10.1007/978-3-032-01887-8_10

[15] Jean-Sébastien Coron. 2000. On the Exact Security of Full Domain Hash. In Proceedings of the 20th Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '00). Springer-Verlag, Berlin, Heidelberg, 229–235.

[16] Yevgeniy Dodis, Ilya Mironov, and Noah Stephens-Davidowitz. 2016. Message Transmission with Reverse Firewalls–Secure Communication on Corrupted Machines. In Proceedings, Part I, of the 36th Annual International Cryptology Conference on Advances in Cryptology — CRYPTO 2016 - Volume 9814. Springer-Verlag, Berlin, Heidelberg, 341–372. doi:10.1007/978-3-662-53018-4_13

[17] European Commission. 2024. Commission Implementing Regulation (EU) 2024/2977 of 28 November 2024 laying down rules for the application of Regulation (EU) No. 910/2014 of the European Parliament and of the Council as regards person identification data and electronic attestations of attributes issued to European Digital Identity Wallets. Official Journal of the European Union. https://eur-lex.europa.eu/eli/reg_impl/2024/2977/oj

[18] European Commission. 2026. EUDI Architecture and Reference Framework. Technical Report. European Commission. https://github.com/eu-digital-identity-wallet/eudi-doc-architecture-and-reference-framework/releases/download/v2.8.0/architecture-and-reference-framework-main-and-annexes.pdf

[19] European Parliament and Council of the European Union. 2024. Regulation (EU) 2024/1183 of the European Parliament and of the Council of 11 April 2024 amending Regulation (EU) No 910/2014 as regards establishing the European Digital Identity Framework. Official Journal of the European Union, L 2024/1183. https://data.europa.eu/eli/reg/2024/1183/oj CELEX: 32024R1183.

[20] Daniel Fett. 2026. SD-JWT Python Reference Implementation. https://github.com/openwallet-foundation-labs/sd-jwt-python.

[21] Daniel Fett, Kristina Yasuda, and Brian Campbell. 2025. Selective Disclosure for JSON Web Tokens. RFC 9901. doi:10.17487/RFC9901

[22] Davor Frkat, Robert Annessi, and Tanja Zseby. 2018. ChainChannels: Private Botnet Communication Over Public Blockchains. In 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). 1244–1252. doi:10.1109/Cybermatics_2018.2018.00219

[23] Sharon Goldberg, Leonid Reyzin, Dimitrios Papadopoulos, and Jan Včelák. 2023. Verifiable Random Functions (VRFs). RFC 9381. doi:10.17487/RFC9381

[24] Jens Groth and Victor Shoup. 2021. On the security of ECDSA with additive key derivation and presignatures. Cryptology ePrint Archive, Paper 2021/1330. https://eprint.iacr.org/2021/1330

[25] Fuchun Guo and Willy Susilo. 2022. Optimal Tightness for Chain-Based Unique Signatures. In Advances in Cryptology – EUROCRYPT 2022, Orr Dunkelman and Stefan Dziembowski (Eds.). Springer International Publishing, Cham, 553–583. doi:10.1007/978-3-031-07085-3_19

[26] ISO 18013-5:2021(E) 2021. Personal identification – ISO-compliant driving licence – Part 5: Mobile driving licence (mDL) application. Standard. ISO/IEC.

[27] Joseph Jaeger and Roy Stracovsky. 2024. Dictators? Friends? Forgers.: Breaking and Fixing Unforgeability Definitions for Anamorphic Signature Schemes. In Advances in Cryptology – ASIACRYPT 2024: 30th International Conference on the Theory and Application of Cryptology and Information Security, Kolkata, India, December 9–13, 2024, Proceedings, Part II (Kolkata, India). Springer-Verlag, Berlin, Heidelberg, 105–137. doi:10.1007/978-981-96-0888-1_4

[28] Don Johnson, Alfred Menezes, and Scott A. Vanstone. 2001. The Elliptic Curve Digital Signature Algorithm (ECDSA). Int. J. Inf. Sec. 1, 1 (2001), 36–63. doi:10.1007/S102070100002

[29] Mirosław Kutyłowski, Giuseppe Persiano, Duong Hieu Phan, Moti Yung, and Marcin Zawada. 2023. Anamorphic Signatures: Secrecy from a Dictator Who Only Permits Authentication!. In Advances in Cryptology – CRYPTO 2023: 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20–24, 2023, Proceedings, Part II (Santa Barbara, CA, USA). Springer-Verlag, Berlin, Heidelberg, 759–790. doi:10.1007/978-3-031-38545-2_25

[30] Anna Lysyanskaya. 2002. Unique Signatures and Verifiable Random Functions from the DH-DDH Separation. In Advances in Cryptology — CRYPTO 2002, Moti Yung (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 597–612.

[31] Wendy Myrvold and Frank Ruskey. 2001. Ranking and unranking permutations in linear time. Inform. Process. Lett. 79, 6 (2001), 281–284.

[32] Jesper Buus Nielsen. 2002. Separating Random Oracle Proofs from Complexity Theoretic Proofs: The Non-committing Encryption Case. In Proceedings of the 22nd Annual International Cryptology Conference on Advances in Cryptology (CRYPTO '02). Springer-Verlag, Berlin, Heidelberg, 111–126.

[33] Alexander Russell, Qiang Tang, Moti Yung, and Hong-Sheng Zhou. 2016. Cliptography: Clipping the Power of Kleptographic Attacks. In Proceedings, Part II, of the 22nd International Conference on Advances in Cryptology — ASIACRYPT 2016 - Volume 10032. Springer-Verlag, Berlin, Heidelberg, 34–64. doi:10.1007/978-3-662-53890-6_2

[34] Amir Sharif, Zahra Ebadi Ansaroudi, Giada Sciarretta, Daniela Pöhn, Majid Mollaeefar, Wolfgang Hommel, and Silvio Ranise. 2025. Protecting Digital Identity Wallet: A Threat Model in the Age of eIDAS 2.0. In Risks and Security of Internet and Systems, Simon Collart-Dutilleul, Samir Ouchani, Nora Cuppens, and Frédéric Cuppens (Eds.). Springer Nature Switzerland, Cham, 89–106.

[35] Gustavus J. Simmons. 1984. The Prisoners' Problem and the Subliminal Channel. Springer US, Boston, MA, 51–67. doi:10.1007/978-1-4684-4730-9_5

[36] Gustavus J. Simmons. 1994. Subliminal Communication is Easy Using the DSA. In Advances in Cryptology — EUROCRYPT '93, Tor Helleseth (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 218–232.

[37] Oliver Terbu, Daniel Fett, and Brian Campbell. 2026. SD-JWT-based Verifiable Digital Credentials (SD-JWT VC). Internet-Draft draft-ietf-oauth-sd-jwt-vc-14. Internet Engineering Task Force. https://datatracker.ietf.org/doc/draft-ietf-oauth-sd-jwt-vc/14/ Work in Progress.

[38] K. Yasuda, T. Lodderstedt, C. Bormann, and J. Heenan. 2025. OpenID4VC High Assurance Interoperability Profile 1.0. Technical Report. SPRIND and Authlete. Workgroup: Digital Credentials Protocols.

[39] Adam Young and Moti Yung. 1996. The Dark Side of "Black-Box" Cryptography or: Should We Trust Capstone?. In Advances in Cryptology — CRYPTO '96, Neal Koblitz (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 89–103.

[40] Adam Young and Moti Yung. 1997. Kleptography: Using Cryptography Against Cryptography. In Advances in Cryptology — EUROCRYPT '97, Walter Fumy (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 62–74.

[41] Greg Zaverucha and Dan Shumow. 2019. Are Certificate Thumbprints Unique? Cryptology ePrint Archive, Paper 2019/130. https://eprint.iacr.org/2019/130

A Further Preliminaries

Notation for the Appendix. Throughout the Appendix we additionally use the notion of power sets, denoting by $\mathcal{P}(A)$ the power set of A. We also use size-limited power sets, writing $\mathcal{P}_{\leq n}(A)$ for the set of all subsets of A with size at most n.

A.1 Non-programmable Random Oracles

Besides input-hiding and collision resistance, one attack on salted-hash-based credentials relies on a non-programmable random oracle. This is a weakening of the classic Random Oracle Model (ROM). The non-programmable version assumes the outputs of a hash function to be truly random and the inputs the adversary evaluates the function on to be observable. Yet the arguable most unrealistic assumption of the ROM, namely, the hash function to have programmable outputs to selected inputs, is left out. For further details on the ROM we refer to [6] and for a model of non-programmable random oracles to [32].

A.2 Signatures

Let us formalize the properties of signatures that appear throughout this work.

Definition A.1 (Signature). A signature scheme SIG is defined w.r.t. to message space $\mathcal{M}$ as the triple of the following algorithms:

- $\mathsf{KGen}(1^\lambda) \to (sk, pk)$: generates the secret signing key sk and a public verification key pk.
- $\mathsf{Sign}(sk, m) \to \sigma$: creates a signature σ on message m.
- $\mathsf{Vf}(pk, \sigma, m) \to 0/1$: verifies if σ is a valid signature over message m under the public key pk.

Correctness. We call a signature correct if $\forall (sk, pk) \leftarrow \mathsf{KGen}(1^\lambda)$ and $\forall m \in \mathcal{M}$, the probability of $\mathsf{Vf}(pk, \mathsf{Sign}(sk, m), m) = 0$ is negl. in λ. We call it perfectly correct should the probability be 0.

Unforgeability. The main security property of signatures is unforgeability, intuitively demanding that an adversary that does not know the secret key cannot forge fresh signatures. In the unforgeability UNF-CMA used throughout this work, the adversary can request signatures on chosen messages but must then forge a signature for a message that it queried no signature for.

Definition A.2 (Existential Unforgeability under Chosen Message Attacks). We say a signature scheme $\mathsf{SIG} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ is unforgeable, if $\forall$ PPT adversaries $\mathcal{A}$ the following advantage w.r.t. to the experiment in Fig. 8 is negligible:

$$\mathsf{Adv}^{\text{UNF-CMA}}_{\mathcal{A},\mathsf{SIG}}(\lambda) := \Pr\left[\mathsf{Exp}^{\text{UNF-CMA}}_{\mathcal{A},\mathsf{SIG}}(\lambda) = 1\right]$$

Unique Signatures. Finally, we require uniqueness, i.e., for any key and message, only a single signature value is accepted during verification. Formally:

Definition A.3 (Uniqueness). A signature scheme $\mathsf{SIG} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ is unique, if there do not exist values $(pk, m, \sigma_1, \sigma_2)$ such that $\sigma_1 \neq \sigma_2$ and

$$\mathsf{Vf}(pk, m, \sigma_1) = \mathsf{Vf}(pk, m, \sigma_2) = 1.$$

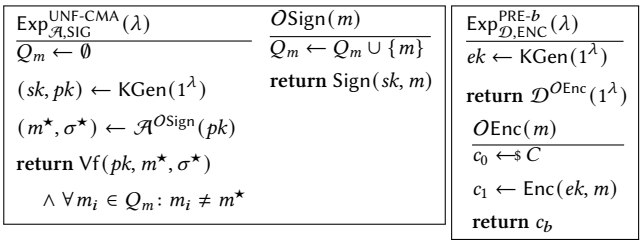


Figure 8: Left: (Existential) Unforgeability under chosen-message attacks for signatures. Right: Pseudorandom Encryption for symmetric encryption schemes.

A.3 Anamorphic Signatures

We only informally described anamorphic signatures so far, let us now give a more precise definition. For signatures that are anamorphic, there exists an anamorphic triplet as defined in Definition 5.2.

Anamorphic Correctness. This anamorphic triplet T is considered correct for a signature scheme SIG with message space $\mathcal{M}$, if the anamorphic decryption outputs the intended anamorphic message with overwhelming probability.

Definition A.4. An anamorphic triplet T for a signature scheme SIG is correct if $\forall m \in \mathcal{M}$, $am \in \mathcal{M}_A$, and $\lambda \in \mathbb{N}$,

$$\mathsf{Adv}^{\text{anaCORRECT}}_{\mathsf{SIG},T}(\lambda) = \Pr\left[\mathsf{Exp}^{\text{anaCORRECT}}_{\mathsf{SIG},T}(m, am) = 0\right]$$

is negligible in λ.

Anamorphic triplets represent an information channel within a signature. If there is such an information channel that cannot be detected, we call the signature scheme anamorphic:

Definition A.5 (Anamorphic Signatures). A signature scheme $\mathsf{SIG} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ is anamorphic for the anamorphic message space $\mathcal{M}_A$ if there exists an anamorphic triplet $T = (\mathsf{KGen}^\star, \mathsf{Sign}^\star, \mathsf{Dec}^\star)$ for the signature message space and $\mathcal{M}_A$ such that, for all PPT distinguishers $\mathcal{D}$, the following advantage is negligible in λ w.r.t. the experiment ANA in Fig. 9:

$$\mathsf{Adv}^{\text{ANA}}_{\mathcal{D},\mathsf{SIG},T}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\text{ANA-0}}_{\mathcal{D},\mathsf{SIG},T}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\text{ANA-1}}_{\mathcal{D},\mathsf{SIG},T}(\lambda) = 1\right]\right|$$

Finally, privately anamorphic signatures preserve unforgeability of the underlying signature scheme even towards adversaries who know the double key adk.

Definition A.6 (Privately Anamorphic Signatures). A signature scheme $\mathsf{SIG} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ achieves private anamorphism for the anamorphic message space $\mathcal{M}_A$ if there exists an anamorphic triplet $T = (\mathsf{KGen}^\star, \mathsf{Sign}^\star, \mathsf{Dec}^\star)$ for the signature message space and $\mathcal{M}_A$ such that, for all PPT adversaries $\mathcal{A}$, the following advantage is negligible in λ w.r.t. the experiment PANA in Fig. 9:

$$\mathsf{Adv}^{\text{PANA}}_{\mathcal{A},\mathsf{SIG},T}(\lambda) := \Pr\left[\mathsf{Exp}^{\text{PANA}}_{\mathcal{A},\mathsf{SIG},T}(\lambda) = 1\right]$$

A.4 Pseudorandom Functions

Below we define pseudorandom functions.

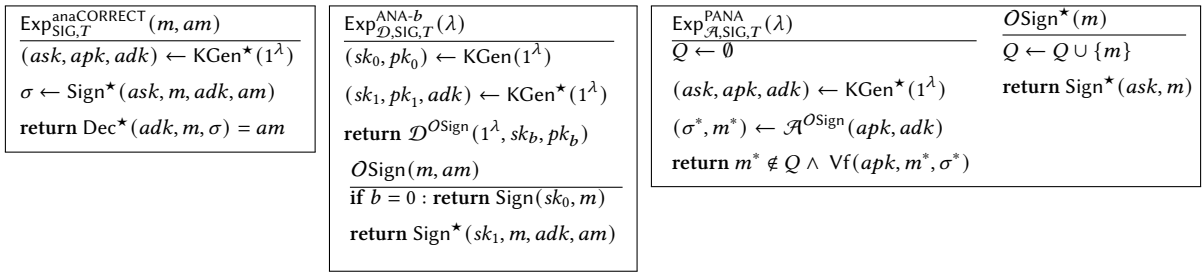


Figure 9: Experiments for anamorphic signatures. Left: Correctness for anamorphic triplets. Middle: Anamorphism game. Right: Private anamorphism game.

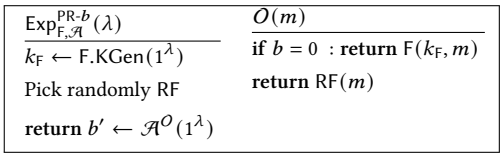


Figure 10: Pseudorandomness game description. RF is a truly random function with a matching input and output space to F, which is picked randomly from all such functions in the beginning of the game.

Definition A.7 (Pseudorandom Function). A function F is pseudorandom if for all PPT adversaries $\mathcal{A}$,

$$\mathsf{Adv}^{\mathsf{PR}}_{\mathsf{F},\mathcal{A}}(\lambda) = \left|\Pr\left[\mathsf{Exp}^{\mathsf{PR}\text{-}0}_{\mathsf{F},\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{PR}\text{-}1}_{\mathsf{F},\mathcal{A}}(\lambda) = 1\right]\right|$$

is negligible in λ for the experiment in Figure 10.

A.5 Pseudorandom Encryption

Our definition of a pseudorandom encryption scheme is based on the definition of pseudorandom ciphertexts given by [29]. It refers to a symmetric encryption scheme ENC = (KGen, Enc, Dec). Intuitively, a pseudorandom encryption scheme outputs ciphertexts that cannot be distinguished from real randomness. We assume that ENC encrypts $u(\lambda)$-bit plaintexts into ciphertexts in C for a polynomial u and a ciphertext space C.

Based on the game shown in Fig. 8, we define pseudorandomness as follows.

Definition A.8 (Pseudorandom Encryption). A symmetric encryption scheme ENC is pseudorandom if, $\forall$ PPT distinguishers $\mathcal{D}$, the following advantage is negl. in λ for the experiments in Fig. 8:

$$\mathsf{Adv}^{\mathsf{PRE}}_{\mathcal{D},\mathsf{ENC}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\mathsf{PRE}\text{-}0}_{\mathcal{D},\mathsf{ENC}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\mathsf{PRE}\text{-}1}_{\mathcal{D},\mathsf{ENC}}(\lambda) = 1\right]\right|.$$

Pseudorandom Encryption via PRFs. We present a classic pseudorandom encryption scheme that encrypts λ-bit messages into $2 \cdot \lambda$-bit ciphertexts, based on a PRF $\mathsf{F} : \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$, is given in the following. Essentially, it chooses a random value r, computes a one-time pad with the output of the PRF on r, and outputs the result together with r as the ciphertext. Pseudorandomness follows from the pseudorandomness of F.

$\mathsf{PRFEnc.KGen}(1^{\lambda})$: Return $ek \leftarrow\!\!\$\ \{0,1\}^{\lambda}$.

$\mathsf{PRFEnc.Enc}(ek, m)$: Sample a random $r \leftarrow\!\!\$\ \{0,1\}^{\lambda}$ and return $c \leftarrow (r, \mathsf{F}(ek, r) \oplus m)$.

$\mathsf{PRFEnc.Dec}(ek, c)$: Parse $(c_0, c_1) \leftarrow c$, and return as the message $m \leftarrow c_1 \oplus \mathsf{F}(ek, c_0)$.

B Omitted Definitions for Batch-Issued Credentials

B.1 Correctness

Let us formalize the notion of correctness for batch-issued credentials. Fig. 11 shows the corresponding experiment. It generates honest issuer keys, then issues a batch of credentials by chaining the honest execution of the user-issuer interaction. From the batch, one credential is presented and we demand that this presentation verifies as expected.

Definition B.1 (Correctness). A credential scheme BC for attribute space $\mathcal{M}$ and $n, k \in \mathbb{N}^+$ is η-correct if the following holds for the experiment CORRECT in Fig. 11:

$$\forall \lambda \in \mathbb{N}^+, \overline{a} \in \mathcal{M}^n, \overline{a}^* \subseteq \overline{a}, i \in [k] :$$
$$\Pr\left[\mathsf{Exp}^{\mathsf{CORRECT}}_{\mathsf{BC}}(\lambda, \overline{a}, \overline{a}^*, i) = 1\right] \geq 1 - \eta$$

If $\eta = 0$, then we call the scheme perfectly correct.

B.2 Selective Disclosure

Selective disclosure is the most basic privacy property. It requires that a single presentation does not reveal information about undisclosed attributes to a verifier. This should hold for all attributes and for all credentials in a batch. In addition, verifiers may have the chance to obtain credentials themselves and this should not impact security.

Based on this intuition, we define selective disclosure with the game given in Fig. 11. In this game, an adversary with access to an issuance oracle picks all parameters for two credentials and a shared presentation: two sets of attributes, the attribute set that is to be disclosed $\overline{s}^*$, as well as an index i that denotes which credential of the batch should be presented. For both attribute sets, a batch of k credentials is issued. One of these batches is chosen at random and the adversary obtains a presentation of the i-th credential in it which discloses the attribute set $\overline{s}^*$. To avoid trivial wins, we ensure that the issuance did not fail, and that from either credential the subset $\overline{s}^*$ can successfully be presented. The adversary must then guess for which credential the presentation was created.

$\underline{\mathsf{Exp}^{\text{CORRECT}}_{\text{BC}}(\lambda, \overline{a}, \overline{a}^*, i)}$
$(isk, ipk) \leftarrow \mathsf{KGen}(1^\lambda)$
$(st, \overline{req}) \leftarrow \mathsf{ReqCreds}(ipk, \overline{a})$
$\overline{rsp} \leftarrow \mathsf{Issue}(isk, \overline{a}, \overline{req})$
$\overline{cred} \leftarrow \mathsf{FinCreds}(st, \overline{rsp})$
$pres \leftarrow \mathsf{Present}(\overline{cred}_i, \overline{a}^*)$
return $\mathsf{VfPres}(ipk, pres, \overline{a}^*)$

$\underline{\mathsf{Exp}^{\text{SD-}b}_{\mathcal{A},\text{BC}}(\lambda)}$
$(isk, ipk) \leftarrow \mathsf{KGen}(1^\lambda)$
$(\overline{s}^*, \overline{a}_0, \overline{a}_1, i) \leftarrow \mathcal{A}^{O\mathsf{Issue}}(ipk)$
for $d \in \{0, 1\}$:
  $(st_d, \overline{req}_d) \leftarrow \mathsf{ReqCreds}(ipk, \overline{a}_d)$
  $\overline{rsp}_d \leftarrow \mathsf{Issue}(isk, \overline{a}_d, \overline{req}_d)$
  $\overline{cred}_d \leftarrow \mathsf{FinCreds}(st_d, \overline{rsp}_d)$
  if $cred_d = \bot$: **return** 0
  $pres_b \leftarrow \mathsf{Present}(cred_d, \overline{s}^*)$
  if $pres_b = \bot$: **return** 0
return $\mathcal{A}^{O\mathsf{Issue}}(pres_b)$

$\underline{O\mathsf{Issue}(\overline{a}, \overline{req})}$
return $\mathsf{Issue}(isk, \overline{a}, \overline{req})$

Figure 11: Additional experiments for BC. Left: Correctness experiment. Right: Selective Disclosure.

We formally define selective disclosure as follows.

Definition B.2 (Selective Disclosure). A credential scheme BC fulfills selective disclosure if, for all PPT adversaries $\mathcal{A}$, the following advantage is negligible in λ for the experiment SD in Fig. 11:

$$\mathsf{Adv}^{\text{SD}}_{\mathcal{A},\text{BC}}(\lambda) := \left|\Pr\left[\mathsf{Exp}^{\text{SD-0}}_{\mathcal{A},\text{BC}}(\lambda) = 1\right] - \Pr\left[\mathsf{Exp}^{\text{SD-1}}_{\mathcal{A},\text{BC}}(\lambda) = 1\right]\right|$$

Note that the attacker only observes a single presentation. As such, the following attacks do not constitute violations of selective disclosure:

(1) Inferring hidden attributes from multiple presentations in a single batch.
(2) Distinguishing the batch size a presentation stems from.
(3) Identifying the index of the credential used to issue a presentation.

Security against such attacks is covered by unlinkability.

B.3 Hierarchy of Privacy Properties

The four iterations of privacy selective disclosure, honest-issuer unlinkability, subversion unlinkability, and corrupt-issuer unlinkability continuously increase the attacker's abilities, as summarized in Fig. 12. We want to state and prove these relations in the following.

B.3.1 Selective Disclosure vs. Honest-Issuer Unlinkability. Observe that if a scheme achieves honest-issuer unlinkability, then even after seeing arbitrarily many presentations of different users, a malicious verifier cannot link presentations to their underlying credential beyond the disclosed attributes. Then, it is intuitive that also selective disclosure is achieved, i.e. the adversary evidently learns nothing about hidden attributes.

Lemma B.3. *Any honest-issuer-unlinkable credential scheme* BC *fulfills selective disclosure.*

Proof. We proceed via contraposition. Let $\mathcal{A}_{\text{SD}}$ be a PPT adversary with non-negligible advantage in the selective disclosure game for BC. Using $\mathcal{A}_{\text{SD}}$, we construct the adversary $\mathcal{A}_{\text{LINK-hon}}$ for the honest-issuer unlinkability game in Fig. 13.

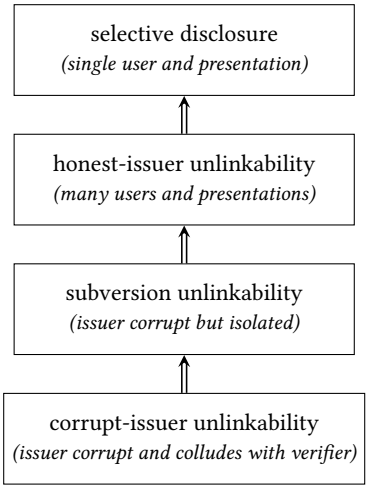


Figure 12: Overview of our privacy properties. In all properties the verifier is malicious. Arrows indicate implications: any credential scheme that satisfies a property also satisfies all properties above it. Abilities of malicious parties increase from top to bottom.

$\underline{\mathcal{A}^{O\mathsf{Issue},O\mathsf{Add},O\mathsf{Present}}_{\text{LINK-hon}}(ipk)}$
$(\overline{s}^*, \overline{a}_0, \overline{a}_1, i) \leftarrow \mathcal{A}^{O\mathsf{Issue}'}_{\text{SD}}(ipk)$
$O\mathsf{Add}(\overline{a}_0); \quad O\mathsf{Add}(\overline{a}_1)$
$pres \leftarrow \mathsf{Exp}^{\text{LINK-hon-}b}_{\text{BC}}(\overline{s}^*, 0, 1, i, i)$
return $\mathcal{A}^{O\mathsf{Issue}'}_{\text{SD}}(pres)$

$\underline{O\mathsf{Issue}'(\overline{a}, \overline{req})}$
/ forward input unaltered
return $O\mathsf{Issue}(\overline{a}, \overline{req})$

Figure 13: Adversary $\mathcal{A}_{\text{LINK-hon}}$ for the proof of Lemma B.3.

We claim that $\mathcal{A}_{\text{LINK-hon}}$ has non-negligible advantage in the honest-issuer unlinkability game for BC. The environment expected by $\mathcal{A}_{\text{SD}}$ is simulated perfectly as we first pass a verification key and then a presentation of either the i-th credential of the batch for $\overline{a}_0$ or $\overline{a}_1$ with equal probability. This mirrors the challenger's behavior in the selective disclosure game. Note that the issuance oracle in both games is identical and can thus simply be forwarded. It follows that $\mathcal{A}_{\text{LINK-hon}}$ succeeds if and only if $\mathcal{A}_{\text{SD}}$ does. As the latter's advantage is not negligible in λ, the same holds true for our new adversary. In addition, $\mathcal{A}_{\text{LINK-hon}}$ must be PPT since $\mathcal{A}_{\text{SD}}$ is. Hence, BC is not honest-issuer-unlinkable for BC* as defined above, which concludes the proof. □

B.3.2 Honest-Issuer vs. Subversion Unlinkability. If we set the subverted algorithms $\mathsf{KGen}^\star$ and $\mathsf{Issue}^\star$ to exactly the honest algorithms, keeping the tracing key empty, then subversion unlinkability is equivalent to the honest-issuer version. The proof of the following is straightforward.

Lemma B.4. *Any subversion-unlinkable credential scheme* BC *is honest-issuer unlinkable.*

$\mathcal{A}^{O\mathsf{Issue}^\star, O\mathsf{Add}, O\mathsf{Present}}_{\mathsf{LINK\text{-}sub}}(ipk, tk)$
$(\vec{s}^*, u_0, u_1, i_0, i_1) \leftarrow \mathcal{A}^{O\mathsf{Issue}, O\mathsf{Add}', O\mathsf{Present}'}_{\mathsf{LINK\text{-}hon}}(ipk)$
$pres \leftarrow \mathsf{Exp}^{\mathsf{LINK\text{-}hon}\text{-}b}_{\mathsf{BC}}(\vec{s}^*, u_0, u_1, i_0, i_1)$
$b' \leftarrow \mathcal{A}^{O\mathsf{Issue}, O\mathsf{Add}', O\mathsf{Present}'}_{\mathsf{LINK\text{-}hon}}(pres)$
return b'

$O\mathsf{Issue}(\overline{a}, \overline{req})$
return $O\mathsf{Issue}^\star(\overline{a}, \overline{req})$

$O\mathsf{Add}'(\overline{a})$
return $O\mathsf{Add}(\overline{a})$

$O\mathsf{Present}'(\overline{a}^*, u, i)$
return $O\mathsf{Present}(\overline{a}^*, u, i)$

Figure 14: Adversary $\mathcal{A}_{\mathsf{LINK\text{-}sub}}$ for the proof of Lemma B.4.

Proof. We proceed via contraposition, so let $\mathcal{A}_{\mathsf{LINK\text{-}hon}}$ be a PPT adversary with non-negligible advantage in the honest-issuer unlinkability game for BC. We construct an adversary for the subversion unlinkability game as shown in Fig. 14. The corrupt issuer acts just as the honest issuer, i.e. we set $\mathsf{Issue}^\star = \mathsf{Issue}$ and $\mathsf{KGen}^\star = \mathsf{KGen}$ with an empty tk.

We claim that $\mathcal{A}_{\mathsf{LINK\text{-}sub}}$ has non-negligible advantage in the subversion unlinkability game for BC. The environment expected by $\mathcal{A}_{\mathsf{LINK\text{-}hon}}$ is simulated perfectly as we first pass the honestly generated verification key and then a presentation of one of the selected credentials with equal probability. This mirrors the challenger's behavior in the honest-issuer game. Note that all of the oracles in both games are identical and can thus simply be forwarded. It follows that $\mathcal{A}_{\mathsf{LINK\text{-}sub}}$ succeeds if and only if $\mathcal{A}_{\mathsf{LINK\text{-}hon}}$ does. As the latter's advantage is not negligible in λ, the same holds true for our new adversary. In addition, $\mathcal{A}_{\mathsf{LINK\text{-}sub}}$ must be PPT since $\mathcal{A}_{\mathsf{LINK\text{-}hon}}$ is. Thus, BC cannot be subversion-unlinkable if it is not honest-issuer-unlinkable. □

B.3.3 Subversion vs. Corrupt-Issuer Unlinkability. Again, corrupt-issuer unlinkability evidently strengthens the subversion version, since the adversary becomes more powerful while the goal remains the same. Any malicious behavior encoded by subverted algorithms can be executed by the adversary in LINK-cor itself; it has full control over the Issue algorithm. There is no need for a tracing key, as LINK-cor models corrupt issuer and verifier as the same entity.

Lemma B.5. *Any corrupt-issuer-unlinkable credential scheme* BC *is subversion-unlinkable.*

Proof. We again show this by contraposition. Let a PPT adversary $\mathcal{A}_{\mathsf{LINK\text{-}sub}}$ and $\mathsf{BC}^\star = (\mathsf{KGen}^\star, \mathsf{Issue}^\star)$ have non-negligible advantage in the subversion unlinkability game for BC. We construct a reduction $\mathcal{A}_{\mathsf{LINK\text{-}cor}}$ for the corrupt-issuer unlinkability game in Fig. 15.

We claim that $\mathcal{A}_{\mathsf{LINK\text{-}cor}}$ has a non-negligible advantage in the corrupt-issuer unlinkability game for BC. The environment expected by $\mathcal{A}_{\mathsf{LINK\text{-}sub}}$ is simulated perfectly as we now show.

First, the issue oracle is implemented exactly as in the subversion unlinkability game, including how keys are generated. Next, $O\mathsf{Add}$ is implemented such that it interacts with $O\mathsf{ReqCreds}$ and $O\mathsf{FinCreds}$ to create a credential and store it into C exactly as is done in the subversion unlinkability game. Hence, $O\mathsf{Present}$, which is forwarded unchanged, also behaves as expected.

Finally, the reduction sends (ipk, tk) and then $pres$ to $\mathcal{A}_{\mathsf{LINK\text{-}sub}}$, the credential for which is selected identically in both games.

$\mathcal{A}^{O\mathsf{ReqCreds}, O\mathsf{FinCreds}, O\mathsf{Present}}_{\mathsf{LINK\text{-}cor}}(1^\lambda)$
$(isk, ipk, tk) \leftarrow \mathsf{KGen}^\star(1^\lambda)$
$c_u \leftarrow 0$ / Counter for users
$\mathsf{Exp}^{\mathsf{LINK\text{-}cor}\text{-}b}_{\mathsf{BC}}(ipk)$ / Announce ipk
$(\vec{s}^*, u_0, u_1, i_0, i_1) \leftarrow \mathcal{A}^{O*}_{\mathsf{LINK\text{-}sub}}(ipk, tk)$
$pres \leftarrow \mathsf{Exp}^{\mathsf{LINK\text{-}cor}\text{-}b}_{\mathsf{BC}}(\vec{s}^*, u_0, u_1, i_0, i_1)$
return $\mathcal{A}^{O\mathsf{Issue}^\star, O\mathsf{Add}, O\mathsf{Present}'}_{\mathsf{LINK\text{-}sub}}(pres)$

$O\mathsf{Issue}^\star(\overline{a}, \overline{req})$
return $\mathsf{Issue}^\star(isk, tk, \overline{a}, \overline{req})$

$O\mathsf{Add}(\overline{a})$
$\overline{req} \leftarrow O\mathsf{ReqCreds}(\overline{a})$
$\overline{rsp} \leftarrow \mathsf{Issue}^\star(isk, tk, \overline{a}, \overline{req})$
$O\mathsf{FinCreds}(c_u, \overline{rsp})$
$c_u \leftarrow c_u + 1$

$O\mathsf{Present}'(\overline{a}^*, u, i)$
return $O\mathsf{Present}(\overline{a}^*, u, i)$

Figure 15: Adversary $\mathcal{A}_{\mathsf{LINK\text{-}cor}}$ for the proof of Lemma B.5.

This mirrors the challenger's behavior in the subversion unlinkability game. It follows that $\mathcal{A}_{\mathsf{LINK\text{-}cor}}$ succeeds if and only if $\mathcal{A}_{\mathsf{LINK\text{-}sub}}$ does. As the latter's advantage is not negligible in λ, the same holds true for our new adversary. In addition, $\mathcal{A}_{\mathsf{LINK\text{-}cor}}$ must be PPT since $\mathcal{A}_{\mathsf{LINK\text{-}sub}}$ is. For this, it is of particular note that $\mathsf{Issue}^\star$ and $\mathsf{KGen}^\star$ are PPT algorithms. Hence, BC is not corrupt-issuer-unlinkable, which concludes the reduction. □

C ECDSA Anamorphism

Let us now support our claim about anamorphic ECDSA signature from Sec. 5.2 by a formal analysis. We prove each anamorphic property separately, together with a more formal theorem.

C.1 Anamorphic Correctness

Theorem C.1. *The anamorphic triplet T above satisfies anamorphic correctness with the advantage*

$$\mathsf{Adv}^{\mathsf{anaCORRECT}}_{\mathsf{ECDSA},T}(\lambda) \leq \left(\mathsf{Adv}^{PRF}_{\mathsf{F}}(\lambda) + (8 + t(t+1))/(q-1) + (1-2^{-\ell})^t\right).$$

Proof. We first change the original game to use a truly random function instead of F. This changes the success probability of the anamorphic correctness game only negligibly. Next we argue the failing probability of the game.

The decryption fails in two cases: if the ECDSA signature generation fails or if the rejection sampling fails to find a matching i^*. It is known that the former occurs with probability $\leq 8/(q-1)$ [24]. Let us analyze the latter. The sampled κ_i values collude only with probability $t(t+1)/(q-1)$, so we can assume that κ_i values are distinct with overwhelming probability. This means that we receive t random ℓ-bit strings from the truly random function evaluation, since the inputs κ_i are distinct. The probability that none of these random strings matches the anamorphic message am is only $(1-2^{-\ell})^t$, concluding our proof. □

C.2 Anamorphism

Theorem C.2. *The anamorphic triplet T above satisfies anamorphism for* ECDSA *if* F *is a pseudorandom function. Concretely, with Q_S being the number of signing queries made by the adversary,*

$$\mathsf{Adv}^{\mathsf{ANA}}_{\mathsf{ECDSA},T}(\lambda) = \mathsf{Adv}^{PRF}_{\mathsf{F}}(\lambda) + Q_S(Q_S+1)/(q-1).$$

Proof. We provide a short sequence of indistinguishable games that starts with ANA-1 and ends with ANA-0. Let Game_0 be identical to ANA-1, which uses the anamorphic signing algorithm.

Game_1. This game replaces the pseudorandom function F with a truly random function. This change is indistinguishable under the pseudorandomness of F.

Game_2. We replace the anamorphic signing algorithm with the honest ECDSA signing algorithm. For the cases that Game_2 always outputs signatures with distinct κ values, this change is perfectly indistinguishable. However, when the same κ is used multiple times when answering the signing queries with different anamorphic messages, this change would be distinguishable as the random function maps κG to a single anamorphic message. Luckily, such collision only occurs with probability $Q_S(Q_S+1)/(q-1)$ where Q_S is the number of signing queries that the anamorphism adversary makes. Thus the games are indistinguishable.

Game_3. This game runs the honest key generation instead of the anamorphic one. ECDSA key pair's distribution remains identical. Furthermore, as the PRF key k_F is not used by the game anymore, absence of it does not change the distribution. Thus, Game_3 is identically distributed to Game_2.

Notice that Game_3 is identical to ANA-0, concluding our proof. □

C.3 Private Anamorphism

Theorem C.3. *The anamorphic triplet T above satisfies private anamorphism if* ECDSA *is unforgeable with the advantage*

$$\mathsf{Adv}^{\mathsf{PANA}}_{\mathsf{ECDSA},T}(\lambda) \leq \mathsf{Adv}^{\mathsf{UNF\text{-}CMA}}_{\mathsf{ECDSA}}(\lambda).$$

Proof. If there is an efficient private anamorphism adversary $\mathcal{A}_{\mathsf{PANA}}$, then we can build an ECDSA forger $\mathcal{A}_{\mathsf{UNF\text{-}CMA}}$ as follows. $\mathcal{A}_{\mathsf{UNF\text{-}CMA}}$ receives an ECDSA public key and sends it to $\mathcal{A}_{\mathsf{PANA}}$ together with a PRF key k_F it honestly samples. $\mathcal{A}_{\mathsf{UNF\text{-}CMA}}$ simulates the signing queries on (m, am) of $\mathcal{A}_{\mathsf{PANA}}$ by making t signing queries on m to the ECDSA unforgeability challenger, re-computing X values as in the ECDSA verification, and searching for a match $\mathsf{F}(k_\mathsf{F}, X) = am$. If the computation of X fails, it treats this as if no match occurred and it outputs the first signature output by the unforgeability game for this batch of signing queries on m. If it finds a match, it outputs the corresponding signature to $\mathcal{A}_{\mathsf{PANA}}$. Finally, when $\mathcal{A}_{\mathsf{PANA}}$ returns a forgery (σ^*, m^*), it passes this as its forgery to the unforgeability challenger.

The view of $\mathcal{A}_{\mathsf{PANA}}$ is simulated perfectly. Also, $\mathcal{A}_{\mathsf{UNF\text{-}CMA}}$ makes only polynomially many signing queries and makes no signing queries on messages distinct from those sent by $\mathcal{A}_{\mathsf{PANA}}$. As a result, if (σ^*, m^*) is a non-trivial PANA forgery, then $\mathcal{A}_{\mathsf{UNF\text{-}CMA}}$ also breaks ECDSA unforgeability. This concludes our proof. □

D Full Proofs for saltBC

Let us state the full theorems and give more detail of the security proofs for saltBC, sketched in Sec. 3.3. Note that saltBC works for any message space $\mathcal{M}$ as long as there exists a matching hash function. In the following we use an abstract $\mathcal{M}$.

D.1 Correctness

We first state correctness formally:

Theorem D.1. saltBC *is a correct credential scheme for $\mathcal{M}$ and any $n, k \in \mathbb{N}^+$, a correct signature scheme* SIG*, and hash function* H.

The proof is straightforward and largely follows from the correctness of the employed primitives.

Proof. Let $\lambda, n, k \in \mathbb{N}^+, \overline{a} \in \mathcal{M}^n$ be arbitrary and consider the correctness experiment in Fig. 11.

First note that we have $\left|\overline{rsp}\right| = k$, since Issue generates exactly k credentials. It now suffices to show that for all $\overline{a}^* \subseteq \overline{a}$ and $i \in [k]$ we have

$$\mathsf{VfPres}(ipk, \mathsf{Present}(\overline{rsp}_i, \overline{a}^*), \overline{a}^*) = 1.$$

From there, it is easy to derive that FinCreds does not return $\perp$. We consider the two conditions in VfPres separately.

(1) $\mathsf{SIG.Vf}(ipk, \sigma, \overline{h}) = 1$. The signature σ is created by Issue as $\mathsf{SIG.Sign}(isk, \overline{h})$ and so this condition holds by the correctness of SIG.
(2) $\forall i \in [k] : a_i^* = \perp \vee (h_i = \mathsf{H}(s_i \| a_i) \wedge a_i^* = a_i)$. If $a_i^* = \perp$, then the vector of disclosures $\overline{discl}$ is empty at this point, too. Else, it contains the s_i, a_i at index i that are the preimage of h_i by how the digests were computed in SaltedHash. Similarly, a_i and a_i^* must match.

Hence, saltBC is a correct credential scheme. □

D.2 Theorem 3.4(1): Unforgeability

Recall that unforgeability of saltBC follows from the fact that the adversary can neither forge a signature on fresh hashes nor find a different preimage for an array of hashes attested by the issuer.

Proof. Assume there is a PPT adversary $\mathcal{A}_{\mathsf{UNF}}$ that wins the unforgeability game with non-negligible probability. In order to do so, it returns $(pres, \overline{a}^*)$ such that the issuance oracle never issued a credential for a superset of $\overline{a}^*$ and $pres$ is a valid presentation for the attribute set $\overline{a}^*$. Recall the presentation $pres$ is made up of hashes $\overline{h}$, signature σ and disclosures $\overline{discl}$. Since VfPres checks that the signature σ is valid for the $\overline{h}$, the adversary must either have forged the signature or reuse a $\overline{h}$ of one of the credentials issued by the issuance oracle.

The former is possible only with negligible probability by the unforgeability of SIG. Specifically, one can build a reduction $\mathcal{A}_{\mathsf{SIG}}$ by relying on $\mathcal{A}_{\mathsf{UNF}}$. $\mathcal{A}_{\mathsf{SIG}}$ simulates the BC unforgeability game against $\mathcal{A}_{\mathsf{UNF}}$ identically by following the saltBC algorithms, except that it obtains the challenge ipk from the signature unforgeability challenger and creating signatures during the issuance by calling its signing oracle.

Now consider the latter case that $pres$ has the same $\overline{h}$ as a credential $cred_i$ that was previously issued for an attribute set $\overline{a}$ by the issuance oracle. As $\overline{a}^* \not\subseteq \overline{a}$, there has to be an attribute $a_i^* \in \overline{a}^*$ with $a_i^* \notin \overline{a}$. For this attribute a_i^*, the adversary must have found a salt s such that the hash value $h = \mathsf{H}(s \| a_i^*)$ is included in the $\overline{h}$ of $cred_i$. But as h is the salted hash $h = H(s_j \| a_j)$ for an attribute $a_j \in \overline{a}$, its preimage $s_j \| a_j$ must be different from $s \| a_i^*$ because it contains $a \neq a_i^*$. So we found a hash collision, which is only possible with negligible probability due to the collision resistance of H.

Hence, any PPT adversary can win the unforgeability game only with negligible probability. □

D.3 Theorem 3.4(2): Honest-Issuer Unlinkability

The construction achieves the weakest privacy property we modeled, honest-issuer unlinkability. This is enabled by a hiding hash function, where an adversary learns nothing about a freshly salted preimage when given a hash value.

Proof. We prove this theorem via the following sequence of indistinguishable games: LINK-hon-0, LINK-hon-0′, LINK-hon-1′, and LINK-hon-1, which are shown in Fig. 16. We show for $b \in \{0,1\}$ that the winning probability of any adversary in LINK-hon-b' is the same as in LINK-hon-b. By demonstrating that an adversary cannot distinguish LINK-hon-0′ and LINK-hon-1′, it follows that the advantage $\mathsf{Adv}^{\text{LINK-hon}}_{\mathcal{A},\text{saltBC}}(\lambda)$ of any adversary $\mathcal{A}$ is negligible.

LINK-hon-0 → LINK-hon-0′. The LINK-hon-0′ game differs from the LINK-hon-0 game by issuing a new credential for the same attribute set to create the presentation *pres* instead of using $C[u_0]_{i_0}$. Additionally, the attribute sets used for issuance are explicitly stored during the Add oracle calls. As both credentials are issued for the same attribute set and the adversary never received $C[u_0]_{i_0}$, it cannot distinguish the newly issued credential from the one issued by the Add oracle. Furthermore, the changes to the issuance process do not alter the adversary's view and therefore remain hidden from them. For this reason, LINK-hon-0 and LINK-hon-0′ do not differ from the adversary's perspective and it must output 1 with equal probability.

LINK-hon-1 → LINK-hon-1′. Following an analogous argument, LINK-hon-1 and LINK-hon-1′ are identically distributed.

LINK-hon-0′ → LINK-hon-1′. These games are indistinguishable under the hiding property of the hash function. Formally, we show that we can build an adversary $\mathcal{A}_{\text{HID}}$ against the hiding game of H if the winning probabilities of an adversary $\mathcal{A}_{\text{LINK-hon}}$ in LINK-hon-0′ and LINK-hon-1′ differ non-negligibly.

We construct $\mathcal{A}_{\text{HID}}$ as in Fig. 16. It simulates the oracles of $\mathcal{A}_{\text{LINK-hon}}$ exactly as in the altered games, which is possible since $\mathcal{A}_{\text{HID}}$ generates the secret key *isk* itself. Then, $\mathcal{A}_{\text{HID}}$ creates a challenge presentation either for $\overline{a}_0$ or for $\overline{a}_1$. To do so, it constructs the hashes of the disclosed attributes $\overline{s}^*$ as in SaltedHash. The hashes of undisclosed attributes, however, are obtained from the OH oracle of the hiding game. This means that the hiding game determines which presentation is generated for $\mathcal{A}_{\text{LINK-hon}}$. $\mathcal{A}_{\text{HID}}$ simulates either LINK-hon-0′ or LINK-hon-1′, depending on the challenge bit of the hiding game. We conclude that the game hop between LINK-hon-0′ and LINK-hon-1′ has a negligible bound if H is hiding.

Overall, the advantage of any adversary in the honest-issuer unlinkability game must be negligible. □

D.4 Theorem 5.1: Salt Attack

In the following, we give the full proofs of undetectability and subversion unforgeability of attack captured by $\text{saltBC}^\star_s$, and show how $\mathcal{A}_s$ uses these subverted algorithms to trace users and break subversion unlinkability of saltBC. Find the specification of $\text{saltBC}^\star_s$ and $\mathcal{A}_s$ in Fig. 4.

D.4.1 Theorem 5.1(1): Undetectability. Assuming pseudorandomness of the encryption scheme ENC, the salts chosen in subverted $\text{saltBC}^\star_s$.Issue* are indistinguishable from honest salts.

Proof. We want to show that if a distinguisher can tell apart honest and malicious salts, it breaks the pseudorandomness of ENC. Assume there is a distinguisher $\mathcal{D}_{\text{DETECT}}$ with non-negligible advantage in the undetectability game for $\text{saltBC}^\star_s$. Using this distinguisher, we define a distinguisher $\mathcal{D}_{\text{PRE}}$. $\mathcal{D}_{\text{PRE}}$ runs $\mathcal{D}_{\text{DETECT}}$ internally and simulates the oracle OIssue for it. This oracle issues credentials exactly as Issue* would do, except that it calls the oracle $O\text{Enc}(\cdot)$ instead of the encryption ENC.Enc$(ek, \cdot)$. For this reason, $\mathcal{D}_{\text{PRE}}$ does not need to generate an encryption key *ek*. Except for this, it generates *isk*, *ipk*, and *tk* just as $\text{saltBC}^\star_s$.KGen* would do. Finally, $\mathcal{D}_{\text{PRE}}$ outputs the same bit b as $\mathcal{D}_{\text{DETECT}}$ whenever $\mathcal{D}_{\text{DETECT}}$ outputs its final guess.

The runtime of the oracle OIssue is linear in n and k. Thus, $\mathcal{D}_{\text{PRE}}$ is a PPT algorithm since it only calls SIG.KGen, generates *tk*, and simulates OIssue.

We are now going to prove that $\mathcal{D}_{\text{PRE}}$ has a non-negligible advantage in the pseudorandom encryption game for the encryption scheme ENC. In this game, $\mathcal{D}_{\text{PRE}}$ receives either (1) the encryption of the requested message or (2) random values of length λ from the oracle OEnc. In the first case, $\mathcal{D}_{\text{PRE}}$ simulates $\mathsf{Exp}^{\text{DETECT-1}}(\lambda)$ perfectly for $\mathcal{D}_{\text{DETECT}}$, as it only replaces ENC.Enc in SaltedHash* with calling OEnc and generates *isk* and *ipk* exactly as KGen* would do. As the distinguisher never receives *tk*, it is not visible to $\mathcal{D}_{\text{DETECT}}$ that *tk* does not contain an encryption key *ek*. In the second case, OEnc returns random values of length λ. This means that all salts in SaltedHash* are generated as random values. Since *ipk* and all credentials are generated exactly as Issue and KGen would do, $\mathcal{D}_{\text{PRE}}$ simulates $\mathsf{Exp}^{\text{DETECT-0}}(\lambda)$ perfectly in this case.

As $\mathcal{D}_{\text{PRE}}$ perfectly simulates DETECT-1 if it plays PRE-1 and DETECT-0 if it plays PRE-0, its advantage equals that of $\mathcal{D}_{\text{DETECT}}$ in the undetectability game:

$$\mathsf{Adv}^{\text{PRE}}_{\mathcal{D}_{\text{PRE}},\text{ENC}}(\lambda) = \mathsf{Adv}^{\text{DETECT}}_{\mathcal{D}_{\text{DETECT}},\text{saltBC}^\star_s,\text{saltBC}}(\lambda).$$

Since ENC is pseudorandom, this advantage is negligible and the subverted scheme $\text{saltBC}^\star_s$ is undetectable for saltBC. □

D.4.2 Theorem 5.1(2): Subversion-Unforgeability. Unforgeability is not impaired by the salt attack, since the tracing key is generated entirely independent of the signing key.

Proof. Assume there is a PPT adversary $\mathcal{A}_{\text{UNF-sub}}$ that wins the subversion-unforgeability game with non-negligible probability.

As with regular BC unforgeability and because the message it receives is entirely independent of *isk*, $\mathcal{A}_{\text{UNF-sub}}$ can forge a new signature only with negligible probability by the unforgeability of SIG.

To reuse a $\overline{h}$ while presenting different attributes, $\mathcal{A}_{\text{UNF-sub}}$ must find a pair of salt and attribute with the same hash as one of the attribute digests. Thus, just as in the unforgeability of saltBC, the

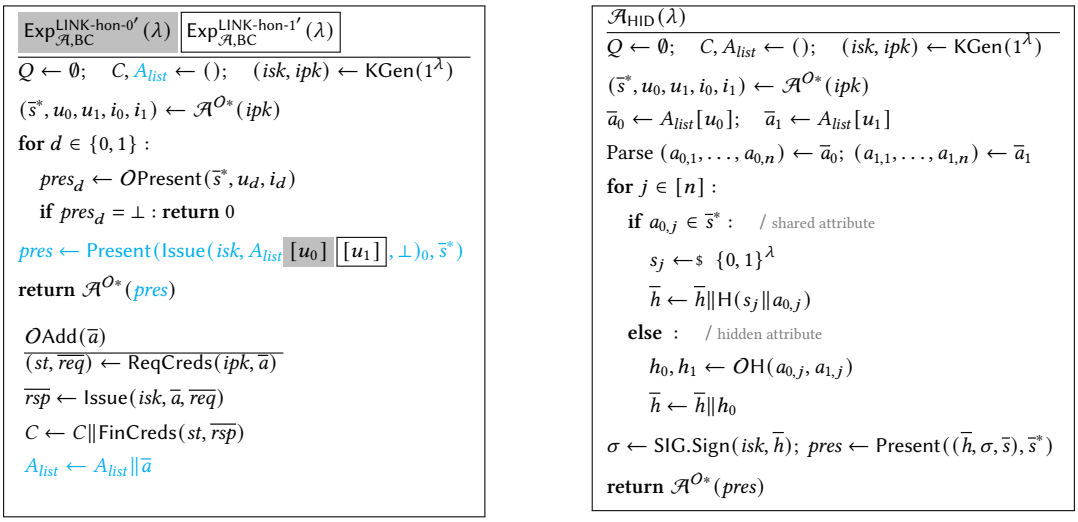


Figure 16: Games for proving honest-issuer unlinkability of saltBC**. Changes to the honest-issuer unlinkability game are highlighted in cyan. The symbols in gray and solid boxes are only included in games** LINK-hon-0′ **and** LINK-hon-1′**, respectively. The remaining oracles are defined as in the original game.**

adversary here has to find a hash collision to reuse a previously signed $\vec{h}$, which is hard for a collision-resistant H. □

D.4.3 Theorem 5.1(3): Subversion Unlinkability Attack. Finally, let us show that $\mathcal{A}_s$ indeed breaks subversion unlinkability using $\text{saltBC}^\star_s$.

Proof. In the subversion unlinkability game, $\mathcal{A}_s$ calls the oracle $O\mathsf{Add}$ to issue one credential for the attribute set $\vec{a}_0$ and one for $\vec{a}_1$. Then, it gets a presentation for either the first or the second credential. If a credential is to be issued for an attribute set containing a_{id}, the malicious issuance generates the salts as an encryption of 1. Otherwise, the salts encrypt 0. As $\mathcal{A}_s$ only requests to disclose attribute a, the presentation includes a single disclosure (s, a) from which $\mathcal{A}_s$ extracts the salt s. Then, $\mathcal{A}_s$ returns 1 if this salt decrypts to 1 and 0 otherwise.

$\mathcal{A}_s$ is efficient, as it only generates the attribute sets, calls the oracle $O\mathsf{Add}$ twice and decrypts one salt. Note that it is indeed possible to choose the attribute a as a distinct value from a_{id} in PPT if $\mathcal{M}$ contains at least two elements.

To analyze the winning probability of $\mathcal{A}_s$, we distinguish two cases based on the challenge bit b. If $b = 0$, $\mathcal{A}_s$ gets a presentation of the credential containing $\vec{a}_0$. As $\vec{a}_0$ does not contain a_{id}, the salt s is generated as $\mathsf{ENC.Enc}(ek, 0)$. From the perfect correctness of ENC, it follows that $\mathsf{ENC.Dec}(ek, s) = 0$, so $\mathcal{A}_s$ returns 0 and:

$$\Pr\left[\mathsf{Exp}^{\text{LINK-sub-0}}_{\mathcal{A}_s,\text{saltBC}^\star_s,\text{saltBC}}(\lambda) = 1\right] = 0.$$

If $b = 1$, $\mathcal{A}_s$ gets a presentation of the credential containing $\vec{a}_1$. As $\vec{a}_1$ contains a_{id}, the salt s is generated as $\mathsf{ENC.Enc}(ek, 1)$. From the correctness of ENC follows that $\mathsf{ENC.Dec}(ek, s) = 1$, meaning $\mathcal{A}_s$ returns 1. Therefore:

$$\Pr\left[\mathsf{Exp}^{\text{LINK-sub-1}}_{\mathcal{A}_s,\text{saltBC}^\star_s,\text{saltBC}}(\lambda) = 1\right] = 1.$$

The overall advantage of $\mathcal{A}_s$ in the subversion unlinkability game is hence not negligible: $\mathsf{Adv}^{\text{LINK-sub}}_{\mathcal{A}_s,\text{saltBC}^\star_s,\text{saltBC}}(\lambda) = 1$. □

D.5 Theorem 5.3: Signature Attack

Given the subverted algorithm and adversary used in the signature attack as shown in Fig. 4, we prove undetectability, subversion unforgeability, and the attack on subversion unlinkability.

D.5.1 Theorem 5.3(1): Undetectability. The attack is undetectable by the analogous property anamorphism of the anamorphic triplet, which intends that subverted signatures are indistinguishable from honest ones.

Proof. Let $\mathcal{D}_{\mathsf{DETECT}}$ be a PPT distinguisher with non-negligible advantage in the undetectability game. We construct a distinguisher $\mathcal{D}_{\mathsf{ANA}}$ against the anamorphism game of SIG as shown in Fig. 17.

$\mathcal{D}_{\mathsf{ANA}}$ acts as an issuer, that uses a signing oracle rather than having the issuer secret key itself. Because the only difference between saltBC and $\text{saltBC}^\star_{ana}$ is the signature generation in Issue, $\mathcal{D}_{\mathsf{ANA}}$ simulates either an honest or dishonest issuer depending on whether is is calling an honest or anamorphic signer.

$\mathcal{D}_{\mathsf{ANA}}$ is PPT since the oracles can be simulated in polynomial time and $\mathcal{D}_{\mathsf{DETECT}}$ is itself PPT. When the anamorphism challenger uses $b_{\mathsf{ANA}} = 0$, then $\mathcal{D}_{\mathsf{ANA}}$ simulates the honest world of the undetectability game. Otherwise, $\mathcal{D}_{\mathsf{ANA}}$ simulates the subverted world of the undetectability game. Thus, we conclude that the advantage of $\mathcal{D}_{\mathsf{ANA}}$ is equal to the advantage of $\mathcal{D}_{\mathsf{DETECT}}$.

□

D.5.2 Theorem 5.3(2): Subversion Unforgeability. This proof follows the saltBC unforgeability proof closely. However, instead of the unforgeability of SIG, it relies on the private anamorphism of SIG.

$\mathcal{D}_{\mathsf{ANA}}^{O\mathsf{Sign}}(1^\lambda, apk)$

Pick $a_{id} \in \mathcal{M}$

return $\mathcal{D}_{\mathsf{DETECT}}^{O\mathsf{Issue}}(1^\lambda, apk)$

$O\mathsf{Issue}(\overline{a}, \overline{req} = \bot)$

$c \leftarrow (a_{id} \in \overline{a})$

for $i \in [k]$:

$(\overline{h}_i, \overline{s}_i) \leftarrow \mathsf{SaltedHash}(\overline{a})$

$\sigma_i \leftarrow O\mathsf{Sign}(\overline{h}_i, c)$

return $\overline{rsp} \leftarrow \left(cred_i \leftarrow (\overline{h}_i, \sigma_i, \overline{s}_i)\right)_{i\in[k]}$

Figure 17: Distinguisher $\mathcal{D}_{\mathsf{ANA}}$ for the anamorphism game

Proof. Assume there is a PPT adversary $\mathcal{A}_{\mathsf{UNF}}$ that wins the unforgeability game with non-negligible probability. In order to do so, it returns $(pres, \overline{a}^*)$ such that the issuance oracle never issued a credential for a superset of $\overline{a}^*$ and *pres* is a valid presentation for the attribute set $\overline{a}^*$. Recall the presentation *pres* is made up of hashes $\overline{h}$, signature σ and disclosures $\overline{discl}$. Since VfPres checks that the signature σ is valid for the $\overline{h}$, the adversary must either have forged the signature or reuse a $\overline{h}$ of one of the credentials issued by the issuance oracle.

The former is possible only with negligible probability by the private anamorphism of T for SIG. Specifically, one can build a reduction $\mathcal{A}_{\mathsf{SIG}}$ by relying on $\mathcal{A}_{\mathsf{UNF}}$. $\mathcal{A}_{\mathsf{SIG}}$ simulates the BC subversion-unforgeability game against $\mathcal{A}_{\mathsf{UNF}}$ identically by following the saltBC algorithms, except that it obtains the challenge *ipk* and *adk* from the private anamorphism challenger and creates signatures during the issuance by calling its signing oracle.

Now consider the latter case that *pres* has the same $\overline{h}$ as a credential $cred_i$ that was previously issued for an attribute set $\overline{a}$ by the issuance oracle. As $\overline{a}^* \not\subseteq \overline{a}$, there has to be an attribute $a_i^* \in \overline{a}^*$ with $a_i^* \notin \overline{a}$. For this attribute a_i^*, the adversary must have found a salt s such that the hash value $h = \mathsf{H}(s\|a_i^*)$ is included in the $\overline{h}$ of $cred_i$. But as h is the salted hash $h = H(s_j\|a_j)$ for an attribute $a_j \in \overline{a}$, its preimage $s_j\|a_j$ must be different from $s\|a_i^*$ because it contains $a \neq a_i^*$. So we found a hash collision, which is only possible with negligible probability due to the collision resistance of H. Hence, any PPT adversary can win the unforgeability game only with negligible probability. □

D.5.3 Theorem 5.3(3): Subversion Unlinkability Attack. If the employed anamorphic triplet is correct, then the verifier will be able to extract the embedded tracing information from the signature.

Proof. We show that the adversary $\mathcal{A}_{ana}$ has non-negligible advantage in the subversion unlinkability game for the credential scheme saltBC_{ana} and the subverted credential scheme $\mathsf{saltBC}^\star_{ana}$ that uses the algorithms from Fig. 4.

Suppose $b = 0$. Then the issued credential does not contain a_{id} and the signature is created with the anamorphic message 0. From the correctness of T it follows that $\mathcal{A}_{ana}$ decrypts this message incorrectly with negligible probability and returns 0 otherwise. The situation is analogous for $b = 1$. The anamorphic message 1 is incorrectly decrypted with only negligible probability and $\mathcal{A}_{ana}$ returns 1 otherwise. Thus, we conclude that

$$\mathsf{Adv}^{\mathsf{LINK\text{-}sub}}_{\mathcal{A}_{ana},\mathsf{saltBC}^\star_{ana},\mathsf{saltBC}_{ana}}(\lambda) \geq 1 - 2 \cdot \mathsf{Adv}^{\mathsf{anaCORRECT}}_{\mathsf{ANA},T}(\lambda).$$

Hence, $\mathcal{A}_{ana}$ succeeds with non-negligible probability. □

E Full Proofs for SD-JWT-specific Attacks

Here, we extend the proof sketches for the SD-JWT-specific subversion attacks given in Sec. 6.

E.1 Hiding Tracing Data in the Digest Order

In the following, we provide the full proofs for undetectability and subversion unlinkability as sketched in Sec. 6.2. (The proof for subversion unforgeability is sufficiently clear from the sketch and the referenced similar proofs.)

E.1.1 Theorem 6.1(1): Undetectability.

Theorem E.1. *If* $\mathsf{sdjwtBC}^\star_{ord} = (\mathsf{KGen}^\star, \mathsf{Issue}^\star)$ *is instantiated with a pseudorandom encryption scheme* ENC *and a collision-resistant hash function* H*, it is undetectable for* sdjwtBC.

Proof. We want to show that a distinguisher, which can detect whether malicious credentials are issued instead of honest ones, can also distinguish between ciphertexts of the pseudorandom encryption scheme ENC and true randomness. To do so, we define a distinguisher $\mathcal{D}_{\mathsf{PRE}}$ that plays the pseudorandom encryption game and simulates the undetectability game for $\mathcal{D}_{\mathsf{DETECT}}$.

$\mathcal{D}_{\mathsf{PRE}}$ replicates the subverted credential scheme $\mathsf{sdjwtBC}^\star_{ord}$ exactly but instead of encrypting the bit c', it sends c' to the oracle $O\mathsf{Enc}$. Clearly, the distinguisher is PPT.

We now analyze the winning probability of $\mathcal{D}_{\mathsf{PRE}}$. If $\mathcal{D}_{\mathsf{PRE}}$ plays PRE-1, it simulates LINK-sub-1 perfectly for $\mathcal{D}_{\mathsf{LINK\text{-}sub}}$, since the oracle $O\mathsf{Enc}$ returns an encryption under ENC in this case. On the other hand, if $\mathcal{D}_{\mathsf{PRE}}$ plays PRE-0, it receives a random value from $O\mathsf{Enc}$. Hence, the resulting order of the values in *pl* is random: the algorithm MsgToPermutation is bijective, so on input of a truly random value, one of the orders is output at random. This means that in this case, $\mathcal{D}_{\mathsf{PRE}}$ simulates LINK-sub-0 perfectly. We conclude that $\mathcal{D}_{\mathsf{PRE}}$ has the same advantage in the pseudorandom encryption game as $\mathcal{D}_{\mathsf{DETECT}}$ in the undetectability game:

$$\mathsf{Adv}^{\mathsf{PRE}}_{\mathcal{D}_{\mathsf{PRE}},\mathsf{ENC}}(\lambda) = \mathsf{Adv}^{\mathsf{DETECT}}_{\mathcal{D}_{\mathsf{DETECT}},\mathsf{sdjwtBC}^\star_{ord},\mathsf{sdjwtBC}}(\lambda)$$

As this advantage is negligible, the subverted credential scheme $\mathsf{sdjwtBC}^\star_{ord}$ is undetectable for sdjwtBC. □

Theorem 6.1(3): Subversion Unlinkability.

Theorem E.2. *If* ENC *is a correct encryption scheme and* H *is a collision-resistant hash function, then* sdjwtBC *is not subversion-unlinkable for the subverted credential scheme* $\mathsf{sdjwtBC}^\star_{ord}$.

Proof. We give an adversary $\mathcal{A}_{ord}$ against subversion unlinkability in Fig. 6 and describe it in the following. Similar to previous attacks, the adversary makes use of the subverted issuance to embed either 1 or 0 as a user identifier into a credential, depending on whether a_{id} is present or not. To do so, $\mathcal{A}_{ord}$ calls $O\mathsf{Add}$ twice, once with an attribute set containing a_{id}, once with an empty attribute set. Then, it attempts to decode and decrypt the user identifier contained in the order of the first $w(\lambda)$ elements of the payload of the received presentation.

$\mathcal{A}_{ord}$ is a PPT algorithm, as it only calls the oracle OAdd twice, parses a presentation and runs PermutationToMsg on it, which is also a PPT algorithm.

To analyze the winning probability of $\mathcal{A}_{ord}$, we first note that the algorithm PermutationToMsg returns the value encoded by MsgToPermutation if the first $w(\lambda)$ hash values in the pl are distinct. Then it follows from the correctness of ENC that $\mathcal{A}_{ord}$ decrypts the correct identifier encoded in the credential and hence wins the subversion unlinkability game.

It remains to be shown that the first $w(\lambda)$ hash values contain a duplicate only with negligible probability. To do so, we assume that there are at least two identical hash values. These hash values are either equal because the input to the hash function was identical or because a hash collision was found. As the hash function H is collision-resistant, the latter only happens with negligible probability. Further, the input to the hash function contains for each of the hash values a random salt s of length λ. According to the birthday paradox, the probability that at least two of these $w(\lambda)$ inputs are identical is

$$\Pr[X \geq 1] \leq \frac{w(\lambda)^2}{2^\lambda},$$

if the random variable X is the number of collisions within the $w(\lambda)$ inputs. Therefore, the probability that the first $w(\lambda)$ hash values are not distinct is negligible. This means that $\mathcal{A}_{ord}$ has a non-negligible advantage in the subversion unlinkability game. □

E.2 Hiding Tracing Data in Decoy Digests

In the following, we provide the full proofs for undetectability, subversion unforgeability, and subversion unlinkability as sketched in Sec. 6.3.

E.2.1 Theorem 6.2(1): Undetectability. Undetectability relies on the indistinguishability of pseudorandom ciphertexts and hash values.

Proof. As in the undetectability proof for the salt attack, we want to show that for a distinguisher to distinguish malicious and honest credentials, it has to break the pseudorandomness of the encryption scheme ENC. Below we present a short sequence of indistinguishable games that starts with DETECT-0 and ends up with DETECT-1. Let Game_0 be identical to DETECT-0.

Game_1. This game immediately returns 0 without waiting for the adversary's guessed bit b' if any of the salts chosen during the honest issuance collide. For Q_I being the number of issuance queries that the adversary makes, there is at most $Q_I \cdot k \cdot n$ salt values. Thus, the change in Game_1 occurs only with probability $Q_I \cdot k \cdot n \cdot (Q_I \cdot k \cdot n + 1)/2^\lambda$.

Game_2. This game immediately returns 0 if the adversary makes a random oracle query with any salts that are used in decoy digest computation. Notice that the game does not use decoy digest salts beyond making random oracle queries with them, so it leaks no information on these salts. As a result, for each decoy digest, the adversary can make a random oracle query with its salt only with probability $Q_H/2^\lambda$ for Q_H being the number of random oracle queries that the adversary makes. As there are at most $Q_I \cdot k \cdot n$ decoy digest, the adversary's advantage in this game differs from Game_1 with probability at most $Q_I \cdot k \cdot n \cdot Q_H/2^\lambda$.

Game_3. For all computed decoy digest hashes during the issuance queries, this game randomly chooses the decoy digest instead of making a random oracle query. As the prior games ensure that random oracle queries for decoy digests are made on distinct inputs that are unknown to the adversary, they are perfectly indistinguishable from randomly chosen values. Thus, this game does not change the adversary's view.

Finally, we argue that now Game_3 is indistinguishable from DETECT-1. Notice that Game_3 only differs from ENC by randomly choosing decoy digests instead of setting them as pseudorandom ciphertexts. Thus, one can build an efficient distinguisher against the pseudorandomness of ENC if there is an efficient distinguisher between Game_3 and DETECT-1. This concludes our proof. □

E.2.2 Theorem 6.2(2): Subversion Unforgeability. As in previous proofs, unforgeability is preserved since the tracing key is completely independent of the signing key.

Proof. Assume there is a PPT adversary $\mathcal{A}_{\text{UNF-sub}}$ that wins the subversion unforgeability game with non-negligible probability. As with the salt attack, the issuer message tk is independent of the signature scheme's secret key and so forging a signature is possible only with negligible probability given that SIG is unforgeable.

To reuse some pl while presenting different attributes, $\mathcal{A}_{\text{UNF-sub}}$ must find a pair of salt and attribute with the same hash as either (1) one of the attribute digests, or (2) one of the decoy digests.

Considering (1), in the ROM, such a collision occurs only with probability $Q_H(Q_H + 1)/2^\lambda$, where Q_H is the number of random oracle queries made by the adversary.

Considering (2), recall that the decoy digests are generated by ENC.Enc. This means that $\mathcal{A}_{\text{UNF-sub}}$ finds a collision of ENC.Enc and H such that $H(s\|a) = \text{ENC.Enc}(ek, a_i^*)$. For Q_I being the number of issuance queries made by the adversary, there are at most $Q_I \cdot k \cdot n$ decoy digests computed by the game. Each random oracle query's output collides with one of these decoy digests with probability at most $Q_I \cdot k \cdot n/2^\lambda$. For Q_H random oracle queries, the probability that a case (2) collision occurs is $Q_H \cdot Q_I \cdot k \cdot n/2^\lambda$. This is negligible and concludes our proof. □

E.2.3 Theorem 6.2(3): Subversion Unlinkability Attack. Let us show that the subversion makes for a successful attack on unlinkability.

Proof. The malicious issuance generates the decoy digests as an encryption of 1 if a credential is to be issued for an attribute set containing a_{id}. Otherwise, all decoy digests encrypt 0. The adversary $\mathcal{A}_{dd}$ calls the oracle OAdd to issue one credential that contains a_{id} and one that does not. Then, it extracts all hashes from the presentation received and tries to decrypt them. It returns 0 if all hashes decrypt to 0 or 1 otherwise.

$\mathcal{A}_{dd}$ is a PPT algorithm as it only generates the attribute sets, calls the oracle OAdd twice, iterates over the hashes in the payload and tries to decrypt them.

To analyze the winning probability of $\mathcal{A}_{dd}$, we distinguish two cases: (1) $\mathcal{A}_{dd}$ plays the game LINK-sub-0, or (2) $\mathcal{A}_{dd}$ plays the game LINK-sub-1. In the first case, $\mathcal{A}_{dd}$ gets a presentation of the credential containing the empty attribute set. Hence, pl only contains encryptions of 0 instead of real hashes. Thus, $\mathcal{A}_{dd}$ returns 0

in this case under the perfect correctness of the ENC and we get:

$$\Pr\left[\mathsf{Exp}^{\text{LINK-sub-0}}_{\mathcal{A}_{dd},\text{sdjwtBC}^{\star}_{dd},\text{sdjwtBC}}(\lambda) = 1\right] = 0.$$

In the second case,$\mathcal{A}_{dd}$ gets a presentation of the credential containing a_{id}. This means that all $n-1$ decoy digests were generated as ENC.Enc$(ek, 1)$. From the correctness of the encryption scheme ENC follows that they also decrypt to 1, meaning $\mathcal{A}_{dd}$ returns 1. Thus,

$$\Pr\left[\mathsf{Exp}^{\text{LINK-sub-1}}_{\mathcal{A}_{dd},\text{sdjwtBC}^{\star}_{dd},\text{sdjwtBC}}(\lambda) = 1\right] = 1.$$

The overall advantage of $\mathcal{A}_{dd}$ in the subversion unlinkability game is hence $\mathsf{Adv}^{\text{LINK-sub}}_{\mathcal{A}_{dd},\text{sdjwtBC}}(\lambda) = 1$. □

F Security of saltBC+

We give the full proofs of security and privacy of our improved salted-hash construction saltBC+.

F.1 Theorem 8.1(1): Unforgeability

The fixes we added to saltBC in order to prevent subversion attacks do not interfere with unforgeability. Hence, the proof follows an analogous argumentation to Theorem 3.4(1).

Proof. This proof follows the saltBC unforgeability proof. In particular, the only change between the two constructions is that the salts are chosen by the user in saltBC+. However, the unforgeability proof of saltBC does not rely on the randomly chosen salts—ensuring the correct computation of the signed hashes is enough. Thus, the argumentation in the proof of Theorem 3.4(1) identically follows for saltBC+. □

F.2 Theorem 8.1(2): Subversion Unlinkability

We introduces mechanisms to combat subversion attacks that force the issuer to generate a unique credential when given some request and attribute vector. In particular, salts and attributes are fixed as per these inputs, and a signature scheme is employed that has a unique valid output.

Proof. We prove the theorem via a sequence of games. Let Game_0 be identical to LINK-sub-0. Below we present a sequence of indistinguishable hybrids which end up in a game identical to LINK-sub-1.

Game_1. This game changes how the challenge presentations are created. Namely, after computing $pres_0$ and $pres_1$, instead of outputting $pres_b$, this game re-computes a credential batch honestly from the attributes and request vector $\overline{req}$ corresponding to the credential the adversary indicated (i.e., the u_0-th call to OAdd). It returns a presentation $pres^*_0$ of the i_0-th credential from this batch. This leaves the output of the oracle identical as long as SIG is a unique signature scheme. In particular, if the adversary used any other payload for the signature than the attributes and salts of the request, FinCreds would have rejected the credentials and the game would have returned 0. Similarly, due to the uniqueness of SIG and the corresponding check in FinCreds, any signature other than the one we created would have been rejected.

Game_2. This game no longer uses salts from the request $\overline{req}$ to create the challenge presentation, but samples fresh ones. This change is indistinguishable, since $\overline{req}$ originates from ReqCreds, which samples salts in the same way.

Game_3. This game creates a presentation of the attribute set 1 instead of 0. Let, $pres^*_d = (\overline{h}_d, \sigma_d, discl_d)$ for $d \in \{0,1\}$. As both presentations disclose the same attributes, the distribution of $discl_d$ is identical for both $d \in \{0,1\}$. Similarly, σ_d is a unique valid signature on $\overline{h}_d$, so it is determined directly by σ_d and ipk. Hence the only remaining task is to show that $\overline{h}_0$ and $\overline{h}_1$ are indistinguishable.

We show that this is indeed the case if H is hiding. Specifically, one can build an adversary $\mathcal{A}_{\mathsf{HID}}$ against the hiding property of H by using an efficient distinguisher between Game_2 and Game_3. $\mathcal{A}_{\mathsf{HID}}$ runs Game_2 identically except the following changes. When computing the challenge presentation $pres^*_0$, for the hidden attributes according to $\overline{s}^*$, it makes queries to the hiding game challenger with the attributes of the challenged users to compute these hash values. The hash values for the disclosed attributes are computed as in Game_2. $\mathcal{A}_{\mathsf{HID}}$ returns the guess of the distinguisher between Game_2 and Game_3 identically to the hiding property challenger. Notice that $\mathcal{A}_{\mathsf{HID}}$ simulates wither Game_2 or Game_3 depending on which attributes that the hiding challenger uses in its hash computations. Thus, its advantage is identical to the success probability of the distinguisher between games Game_2 and Game_3.

Game_4. We take back the changes of Game_2 and start using the salts from the request $\overline{req}$ for the u_1-th call to OAdd. This is indistinguishable with the same reasoning as for hop $\mathsf{Game}_1 \rightarrow \mathsf{Game}_2$.

Game_5. Finally, we revert the change in Game_1: we once more output the presentation based on a credential created by Issue*, this time $pres_1$. Again, this change is indistinguishable if SIG is a unique signature scheme. Notice that Game_5 is identical to LINK-sub-1, and we have shown it to be indistinguishable from LINK-sub-0 if SIG is unique and H is hiding, concluding our proof. □