Rebooting the Web of Trust

Designing the Future of Decentralized Self-sovereign Identity

A White Paper from RWOT XI: The Hague

Taking out the CRUD: Five Fabulous DID Attacks

by Shannon Appelcline, Cihan Saglam, Kate Sills, Carsten Stöcker

RWOT XI Gold Sponsors:

Abstract

Decentralized identity solutions, such as DID methods, tend to be designed to protect against certain attacks, but the purpose of that design usually is not explicitly stated in any architectural description or threat documentation. In particular, some DID methods have costly on-chain requirements that must have had a reasoning behind their requirement. We can today see that these DID methods were purposefully shaped, but it's not clear why such decisions were made. The purpose of this paper is to describe a few colorful attacks on DID methods so that we can better understand what threats a system might be vulnerable to.

Although we derived the examples in this paper by examining current DID methods, we believe these attack vectors are more general, even for systems not using DIDs. The goal is to support engineers and developers who are developing decentralized identity solutions to safeguard their work and make it secure and compliant.

Introduction: Why Threat Analysis is Important

Security, as part of the software development process, is an ongoing process involving people and practices. It ensures application confidentiality, integrity, and availability. Secure software is the result of security-aware software development processes where security is built in and software is thus developed with security in mind.[1]

Security is most effective if planned, managed, tested, and controlled throughout every stage of the Software Development Life Cycle (SDLC), especially for critical applications concerning health, safety, regulatory compliance, or critical infrastructures. Government agencies and industries such as travel and transport, energy, or defense all adhere to formal and strict SDLC processes. A cousin of SDLC is Computerized System Validation (CSV) which is widely used in the Pharmaceutical, Life Sciences, and BioTech industries.

Both SDLC and CSV focus on the process of testing/validating/qualifying a regulated computerized system to ensure that it does exactly what it is designed to do in a consistent and reproducible manner that is as safe, secure, and reliable. The state of validated software systems and their electronic digital records and signatures shall be maintained, tested, and controlled through the entire life cycle until the software is end of life and the retention period is fully expired.

These processes include the following activities:

1. System classification with regard to legal compliance applicability and a risk-based classification with impact on product quality, safety, and data integrity.
2. Creation of a test/validation/qualification plan based on the risk-based categorization.
3. SLDC requirement definition and implementation.
4. Formal test verification, quality assurance, and documentation.

1 The Open Web Application Security Project, Securing Enterprise Web Applications at the Source: An Application Security Perspective

It shall be understood that there are multiple legal compliance requirements that affect validated software. Examples are: eIDAS, GDPR, GxP, HIPAA, or PCI. These compliance requirements define controls and non-functional requirements (NFRs). Common attributes of the NFRs shall be in scope of security testing[2]. These common attributes include:

- authentication,
- authorization,
- confidentiality including data privacy,
- availability,
- integrity,
- attributability,
- non-repudiation, and
- resilience.

The emergence of decentralized identity technology will soon be a foundational part of secure and compliant software development and security testing. Digital decentralization can support many of these attributes as well as non-functional requirements for identity governance, signing, hashing, encryption, and key management/rotation. (We would like to highlight that key rotation is a common non-functional requirement which was introduced years ago for encryption keys and which is now gaining more and more importance when system engineers move towards signed data.)

> *"When it is your job to keep billions of people and devices safe online, you have to live and breathe and see the internet [as well as the underlying identity & trust infrastructure] just like the attackers do."*
>
> *—Hacking Google* [3]

A critical part of security-test planning is threat modeling. In threat modeling a data flow model is created and vulnerabilities from the perspective of attackers are analyzed among different software components as well as internal and external system and human user interfaces. Such a data flow model touches all aspects and modules of the digital identity system and its interfaces. Attackers only need a single entry point to inject malicious code or conduct an identity theft attack. Each interface or code dependency thus presents another unique opportunity for threat actors to attack.

This paper focuses on a subset of the data flow model of an identity system that is related to the data flow interactions with regards to creating, reading, updating, and deleting (CRUD) DID documents. It describes threat scenario categories of DID-methods and determines their vulnerabilities. We introduce user stories for each threat scenario from the perspective of the victim and attacker.

As we are only focusing on the attack surface of DID document operations, the larger field of secure Key Management Systems (KMS) and Random Number Generator attacks are out of scope, but the interface between the KMS and the software that facilitates the DID operations is in scope of the paper.

Following are the Five Fabulous DID Attacks.

2 Wikipedia, Security Testing

3 Youtube, Hacking Google Documentary

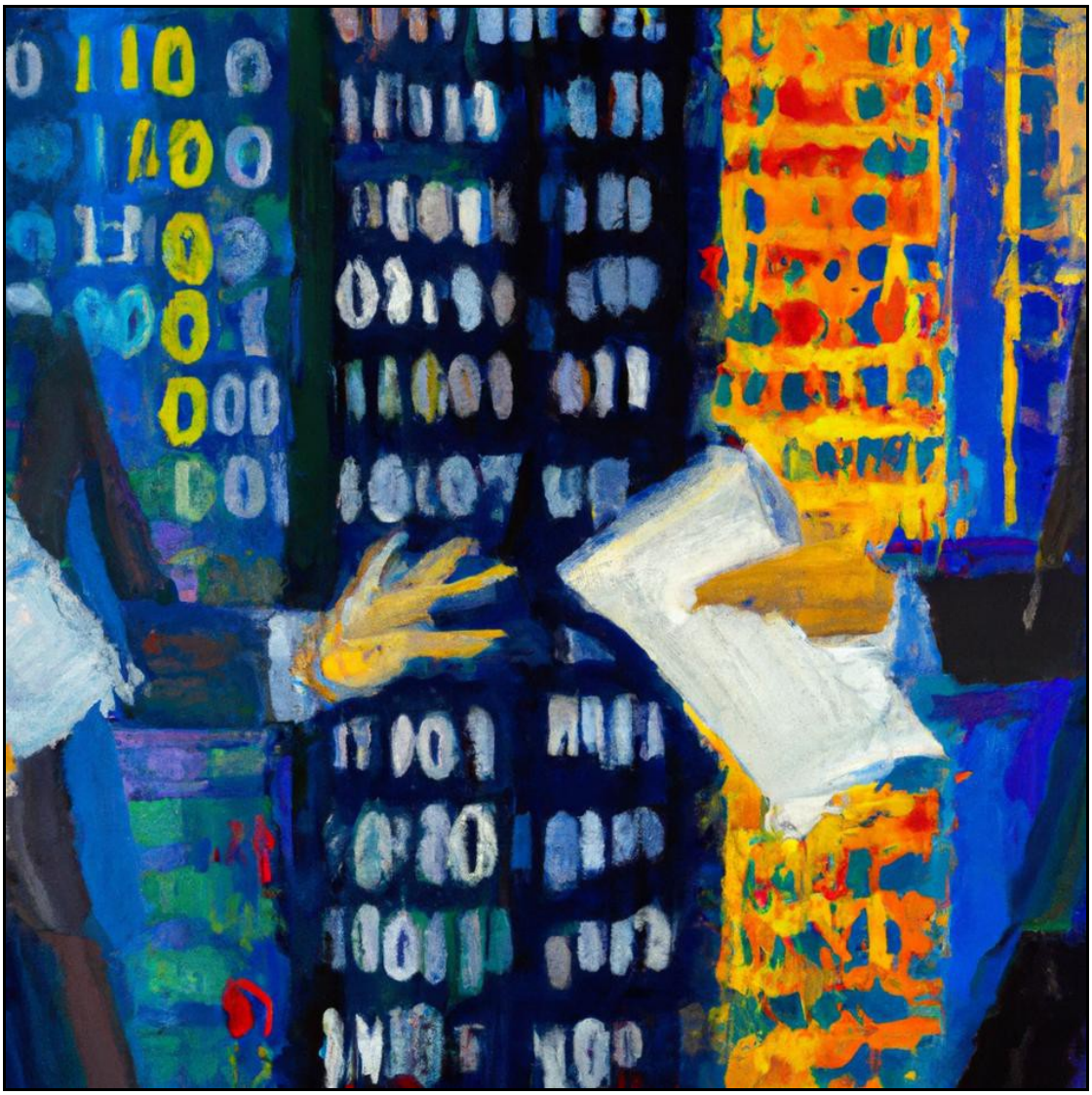
1. Create: The DID Creation Switcharoo

Keywords: DID create, DID derivation from public key, interface with KMS, malicious derivation code injection, replacement of DIDs, secure SW supply chain

NFR Attributes: authentication, authorization, integrity, attributability, non-repudiation

Summary: Using the attack surface of DID CRUD operations, malicious DID derivation code is injected in an open or closed source repo. This switches the derived DID output with a malicious DID when the DID derivation algorithm is executed. Victims will not recognize that their malicious DIDs do not match the public keys generated by their secure KMS unless they double check their derived DIDs or sign transactions with their corresponding keys. Meanwhile, the malicious derived DIDs are used in the victim's business processes and exploited by an attacker.

Examples: DID:key or DID:ethr created in a way other than expected by the controller.



Assumptions: Crypto suite and KMS are secure (e.g. HMS or secure element).

User Story: Anastasia is a darknet hacker. She has acquired illegitimate access to 'DID-Mania', a popular open-source software repo for managing the DID life-cycle for a variety of different methods. When she attained access she injected a tiny piece of malicious code into the DID creation feature.

Quentin, a senior software developer at 'Quick and Cheap SW Solutions', loves the DID-Mania library because it does all the DID management work for him. DID-Mania is integrated into all his signature Trustziod ERP suites, which he sells to Fortune 500 companies.

'Fortune favours the Brave' (F²B) is one of those companies. It's a global market leader in DeFI, with its success built on a solid foundation of security, privacy, and compliance. F²B is proud to be the first DeFi company in the world to have ISO 22301:2019, ISO/IEC 27701:2019, ISO/IEC 27001:2013 and PCIDSS v3.2.1 Level 1 compliance and is independently assessed at Tier 4, the highest level for both NIST Cybersecurity and Privacy Frameworks, as well as SOC2 compliance.

F²B has used Quentin's Trustziod ERP suites for many years ... which means they're now using Anastasia's malicious code. That code wakes up when F²B derives a new DID for an Accounts-Receivable process from a public key that was created with F²B's world-class PKI infrastructure.

Anastasia's code does not apply the proper DID derivation algorithm. Instead, her code replaces the output of the deviation function with a DID that is controlled by Anastasia. A few hours later, F²B triggers the Accounts-Receivable process using the new DID in one data field of a legacy invoicing protocol.

Matt, a wealthy celebrity and happy F²B customer, parses the invoice message, resolves the (malicious) DID and sends his payment to Anastasia's fake payment endpoint.

The user story described above is not a theoretical threat. Attacks on companies like SolarWinds [4], Colonial Pipeline [5] and Codecov [6] showed how threat actors could compromise organizations on a mass scale by planting malware in software updates from trusted vendors.

Risks:

1. **Blitz Identity Theft.** Victims create public/private key pairs and derive DID identifiers from their corresponding public keys. They assume they control the DIDs with their private keys stored in a secure KMS. Victims might activate business processes or service endpoints with their DIDs or share their DIDs with external parties. They do not detect that an attacker switched the DIDs with a code snippet injected into the DID derivation algorithm until they sign a data payload with their corresponding private keys and the signature in the proof is being verified. The time between DID creation and the detection of the 'DID Creation Switcharoo' is the window of opportunity for 'identity theft' for the threat actor.
2. **Service Endpoint Attack.** Fresh DIDs that are shared with supply chain partners can be configured by the attacker. Attackers can configure key material and service end-points. This configuration enables them to reroute authenticated and encrypted communication to their own endpoints.

4 Dark Reading, August 2022, Nearly 3 Years Later, SolarWinds CISO Shares 3 Lessons From the Infamous Attack

5 Dark Reading, May 2022, Colonial Pipeline 1 Year Later: What Has Yet to Change?

6 Dark Reading, April 2021, Attackers Compromised Code-Checking Vendor's Tool for Two Months

3. **Compliance Attack**. Threat actors can immediately start signing on behalf of the victim and create forged compliance data and audit logs.
4. **Advanced Attack involving KMS**. Threat actors manipulate 'key to DID' mappings and reroute trigger events for creating and signing operations for a given DID to a malicious signing module. Such an approach opens an extended window of opportunity for the threat actor to sign data on behalf of the victim. This advanced attack vector stays undetected until the victim detects maliciously signed data sets and responds to the exploit with deactivating the DID (unless the threat actor removed the victim's ability to deactivate; such an attack would result in a major problem for victims as victims can neither revoke verifiable credentials for their malicious DIDs).

Mitigations:

1. **Secure the CI/CD Pipeline.** Securing the CI/CD pipeline includes security scanning tools for vulnerabilities for both open-source SW and commercial off the shelf (COTS) SW components.
2. **Require Audit by Third Party SW Provider.** Auditing the Information Security Management System (ISMS, ISO 27001) and SDLC processes of COTS SW development partners or solution providers allows a systematic methodology for discovering malicious injection of this sort.
3. **Double Check the DID Derivation**. A user of a DID software package would derive a DID with an independent, validated software tool.
4. **Sign with DIDs Prior to any Internal or External Usage.** Checking a new derived DID by signing a transaction and verifying the signature allows another methodology for double-checking the derivation.
5. **Establish Ecosystem Conformance Criteria.** A first conformance criterion can define a valid DID document (and corresponding DID) when the owner has performed at least one signed DID document operation. A second conformance criterion can establish a control requiring that DID owners must perform DID document operations and check that the DID documents were signed with the correct corresponding private key in their secure KMS systems. As a consequence, default DID documents in on-chain DID registries shall not be accepted as a valid DID (e.g. default DID documents of DID:ethr).
6. **Establish DID to Key Mapping Controls.** Further security tests and controls can be established to validate that the mapping between DIDs and key material stored in the KMS system is secure and correct. These controls address the attack vector of rerouting signing transactions to a malicious key management system.
7. **Incorporate Key Attestation.** Key attestation mechanisms can be used as a control to verify that public private key pairs were generated in a secure hardware module. Such a control protects against malicious software-based signing modules.

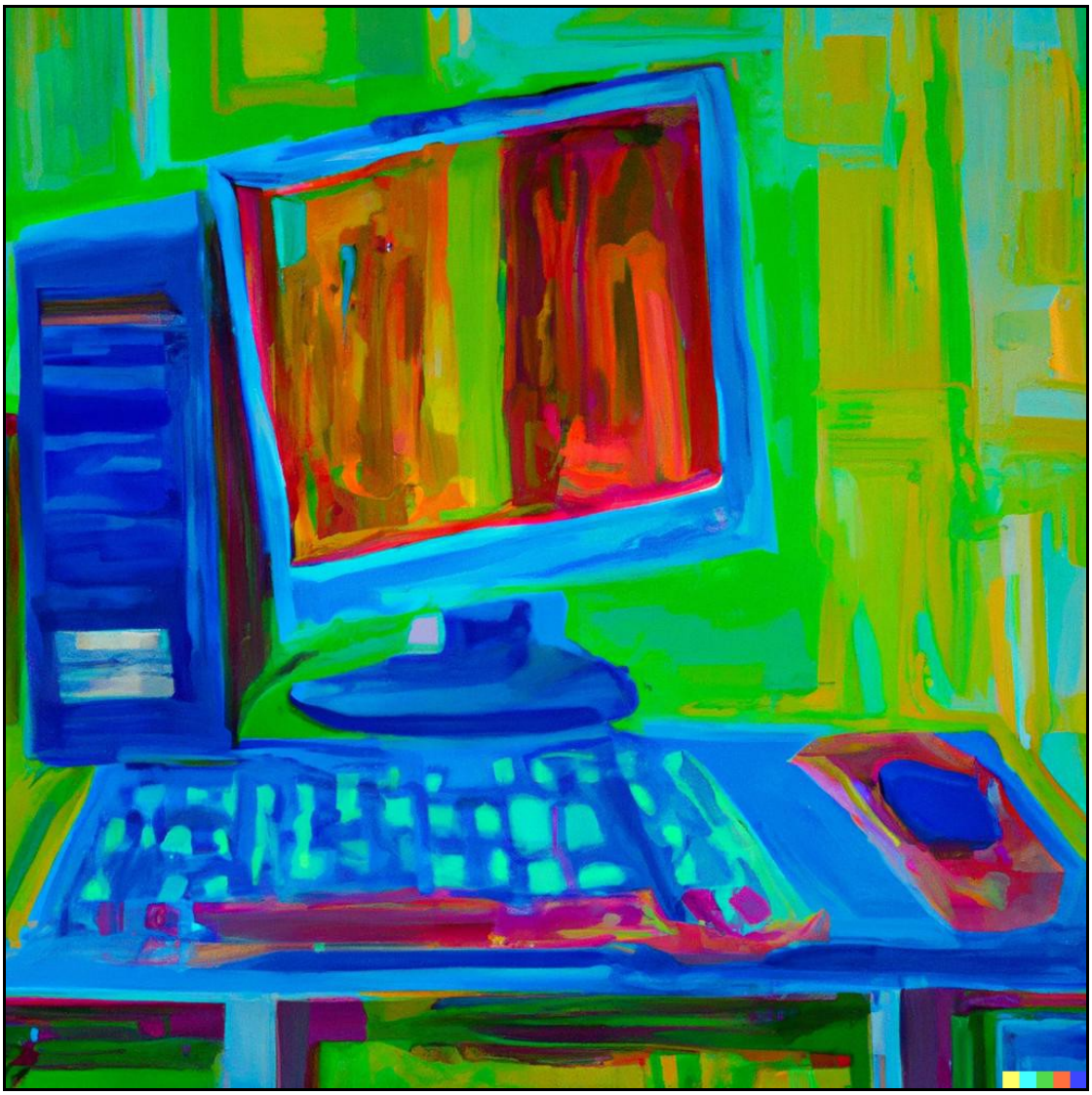
2. Read: The Poop-Emoji DID Doc

Keywords: DID read, duplicity, fake identities, mutable DID documents, nonattributability, nonverifiability

NFR Attributes: attributability, availability, integrity

Summary: Web-based DID documents are obviously mutable over time, but they can also be mutable based on the location of a viewer, creating serious problems with integrity.

Examples: DID documents for DID:btcr 0.1.



User Story: Mark is running for political office. He's a well-known businessman but he has no political experience. He'd do a lot better if he could obtain an endorsement from a polished politician!

Cliff Clipse is a politician without strong business experience. So he in turn could use an endorsement from a businessman, and he knows that Mark would happily offer him an exchange. The problem is that Cliff hates everything that Mark stands for. Fortunately, he has a plan to get an endorsement while cheating Mark out of his!

Cliff sets up a DID using a method that encodes its DID doc on a web server. He then extends Moxie's trick of creating a Poop-Emoji NFT to the DID world: his DID document looks different when viewed from Mark's IP addresses (those registered to his political campaign and seen in his email) and when seen by the rest of the world! Cliff signs the endorsement for Mark with the key shown to Mark, but not to the rest of the world.

Mark is able to see Cliff's endorsement, so he stands by his own endorsement to Cliff. The rest of the world sees a different public key in the DID document, and so they don't think that Cliff endorsed Mark, and so Mark receives no advantage from it.

(Cliff ultimately wins his political race; Mark does not.)

Risks:

1. **Mutable Information.** Any DID method that doesn't in some way timestamp or otherwise record its DID doc is vulnerable to unauditable changes to the document over time.
2. **Simultaneously Mutable Information.** DID methods that store their DID docs on web sites are vulnerable to further attacks such as the ability to show different pages to different people based on location, individual identification, software, or other factors. This can cause considerable compliance issues by destroying attributability among multiple parties.
3. **Centralization.** In a situation where everyone is served a different DID doc, no one can cross-verify information. This creates a *de facto* centralized authority. It could even be done openly, making everyone beholden to that single source.
4. **Fake Identities.** DID docs that are simultaneously mutable can also allow the creation of fake identities, where a fake identity is presented to the world, but hidden from someone who would recognize that it is fake.

Mitigations:

1. **Hash DID Doc.** DID Documents can be hashed and that hash can then be stored in an immutable ledger, such as a blockchain. This allows anyone to verify the current state of a DID doc, and also provides timestamping of when it was last changed.

3. Read/Update: Don't Talk about Fight Club (unless you want to compare DIDs)

Keywords: DID update, DID read, bad reputation, canonical update, double-spend identity, forked identity, malicious controller attack

NFR Attributes: attributability, non-repudiation

Summary: If a malicious identity controller anticipates negative feedback from a user or customer, the controller can give the user a DID which is not the identity that other people look to for reputation, thus dodging the bad review.

From the user's perspective, this attack is difficult to detect in cases where public knowledge is not common (dark markets, black markets, fight clubs, etc.). It can simply look like the controller has updated their DID to a new identifier when using a DID method that doesn't support canonical updates (e.g. rotatation of keys in key-based DID methods).

Examples: did:key and similar DID methods if they loosened their restrictions and allowed updates.

Assumptions: The victim has reason not to publicly claim involvement with the identity controller.

User Story: George joins Fight Club, an underground men's group. At first, he's thrilled to be included but as time goes on he realizes that Tyler, the leader, is abusive. He leaves the group, but that's not enough: he wants to warn others. He writes a negative review of Tyler with Tyler's DID as the subject of the review.

Unfortunately, unbeknownst to George, no one will ever read his review of Tyler. When people hear about Fight Club and want to check it out, the invitees see a different DID, one that has only good reviews.

Tyler was able to make sure that George's review would be dropped because he chose to use a DID method that allowed updates without having to publicly announce them and without deactivating the old DID. George was given the updated DID; everyone else was given the original. Thus, from George's perspective, nothing is wrong: Tyler just did a regular key rotation. But the new invitees never hear of any update to a new DID and never know that George's review might be pertinent to them.

Because no one in Fight Club talks about Fight Club, the split identity is not discovered.

Risks:

1. **Bad Reputation Suppression**: Reputation usually requires the collection of information from multiple sources about the same subject. However, if the subject is able to selectively use a different identifier, software that naively correlates only based on identifier (and not on the entity that is the subject) will miss the bad reviews.
2. **Repudiation**: Being able to carry over reputation to a new identity and then drop the identity at will without anyone noticing creates a repudiation problem. How does a verifier know that you controlled the newly created DID if the update was not recorded anywhere?

Mitigations:

1. **Require Canonical Updates, such as Transactions on a Blockchain:** If all updates for a particular DID must be recorded in a particular location where the updates are checked for contradiction, then the subject cannot split off a new DID specifically for bad reviews while retaining their original DID as their main identity. While the subject can of course maintain multiple DIDs from the very beginning, by enforcing canonical updates, we can at least prevent the victim from seeing the new DID as a recent update of the main identity, carrying over the main identity's reputation.
2. **Require Canonical Bindings of DIDs to Real-world Entities:** For particular use cases, it might be appropriate for a new kind of DID method where the identifier is not random. In this case, the identifier matches the human-readable reference — for example, the business name or address of a restaurant. However, this is similar to a "real name" policy, which takes away the ability of the subject to create multiple identifiers or identify by a different name than determined by authorities. Additionally, this approach leads to vulnerability to spoofing attacks and requires potentially costly governance of the identifier namespace.
3. **Use Petnames and Web of Trust:** A more decentralized approach is to share knowledge of identity bindings. For example, if the Fight Club members shared their nicknames (petnames) for the DIDs, they would have learned that George had a different identifier for Tyler. Moreover, with a Web of Trust-style solution, George would be able to know that no one else acknowledged the DID he had for Tyler, because no one had signed over it. However, the requirement not to talk about Fight Club might weaken this approach.

4. Update: Harmer in the Dwell Latency Attack

Keywords: DID update, advanced persistent threat, dwell time, frontrunning, latency, timing

NFR Attributes: authentication, authorization, availability, confidentiality, data privacy

Summary: High latency in identity updates, combined with the ability to foresee those updates, can allow an attacker to make changes in advance of a victim.

Examples: Updates to DID:btcr occur via Bitcoin transactions and so can be seen in advance in the Bitcoin mempool.

User Story: Victor is a system administrator working for OVER, a critical infrastructure company that provides energy for a large portion of a country. The private key for his identity has been compromised, and he doesn't know it.

The key was compromised by Arthur, a spy for a nearby country. They've been using the private key to engage in data theft, such as accessing the blueprints to the entire energy system, making it vulnerable to directed attacks.

Every six months, Victor rotates his key in accordance with compliance requirements. When he does this, following the theft by Arthur, Arthur is able to spot the rotation in advance because the DID method that Victor uses has a latency in its update process where the update can be seen as a pending transaction.

Arthur is able to push a key rotation of his own before Victor's rotation goes through and takes total full control of the identity. Because his key rotation will expose him, Arthur is now willing to engage in much more explicit attacks. He takes down the entire IT/OT computer network that Victor has access to, also darkening much of the energy grid. The foreign country immediately attacks, initially targeting the critical infrastructural locations they've already discovered due to the previous data theft.

Risks:

1. **Frontrunning.** The latency of updates or deletions to an identity network can create the opportunity for frontrunning, where an attacker can insert their own transactions ahead of the victims.
2. **Advanced Persistent Threat.** Because the attacker has foreknowledge of when they would otherwise lose access to an identity, they can increase their dwell time and potentially do more damage for a more extended period before beginning to engage in more explicit attacks.

Mitigations:

1. **Pre-rotate with Update Keys.** If rotations are defined in advance with update keys, then an attacker cannot frontrun a key rotation. (This presumes that the update private keys are actually secure and even then does not stop an attacker from escalating their attacks during the latency period.)
2. **Reduce Latency.** An instant blockchain confirmation methodology would end any frontrunning and attack escalation by instantly changing the key without warning. Of course, not all latency is technical: a compliance policy such as one that requires rotation *precisely* every six months could similarly offer advance notice of likely rotation.
3. **Use Hash Commitments.** Frontrunning can also be foiled through hash commitments, which commit a transaction to a block without revealing it until a later block.

5. Delete: The Dishonorable DID Deletion

Keywords: DID delete, cthulhu, repudiation, spec compliance, trust

Summary: If a DID method defines off-chain creation and allows update/deactivate operations, those should be provably documented. Solely removing the corresponding private key to a publicly resolvable DID neither equates to deactivating nor complies with the DID Core specs.

Examples: The ethereum-based DID method, "did:ethr", defines two ways to deactivate a DID. According to its specs, deactivation can be done off-chain by removing the private key when a DID has no update on the verifiable data registry.

User Story: Bob was a proud member of Pleasantville, a virtual world in Metaverse. He enjoyed the relationships with other members, but more and more, it showed him how terrible Real Life was. While everybody smiled at his tall-dark-and-handsome avatar in Pleasantville, not one of his building complex neighbors liked him. While he had a weekly date in Pleasantville, he hadn't been outside his real-life apartment in almost two years.

Bob was beyond exasperation. The pain became ubiquitous. There was no solution. Bob decided to "delete" his Metaverse profile. On the last evening, he opened his GallacticWallet and clicked "deactivate". A quick "Yes, I am sure" at the prompt, and the process was done. There were no checks or writing to the ledger; the GallacticWallet followed the deactivation process and removed the private key of Bob's DID.

Another couple of glasses of wine later, and before going further, Bob felt a jolt of panic and wanted to see Pleasantville again. He imported the key back to his wallet, and — lo and behold — his beautiful, tanned, toned avatar was there, incognizant of any troubles of Real Life, happy and surrounded by cheering fans. His Metaverse identity resolved to the initial DID state and simply loaded all the associated perks. His moment of weakness, his "Yes, I am sure", never invalidated the credentials that he had collected so far. He realized there is no "death" in Metaverse as long as there is no on-chain deactivation.

He was invincible. He could take risks nobody dared to take. He lived through others' compliments as his virtual know-how compounded, and soon, he was the most experienced and capable person in Metaverse. He became the first Cthulhu of the Metaverse — a mighty creature who lurked in the off-chain darkness and exploited the weakness in the DID method spec. Bob was the original Meta-god. Many followed after him, leading the world to an apocalypse.

Risks:

1. **Repudiation, Non-Definitive DID Deletion:** An attacker can claim the deactivation did happen but keep using the DID. Off-chain deactivations are not distinguishable from the initial state.

Mitigations:

1. **Require Provable Change History:** DIDs resolving from public verifiable data registries should have all the state changes on-chain.
2. **Intitiate Conformance Checks for DID Methods:** If a DID method supports a deactivation operation, the way to perform it shouldn't cause any ambiguity, nor should it rely on the honesty of the controllers. It should conform to the requirements of DID Core specs and return a deactivation state upon resolution.

Conclusion

This paper limited its scope by addressing a small component of a decentralized identity system through a focus on the vulnerabilities of CRUD operations for managing DID documents. It analyzed the data flows for CRUD operations only and identified different attack vectors. The authors summarized the attack vectors and described risks and mitigation measures. It shall be understood that the authors did not apply a formal threat analysis methodology.

Even though the scope of this paper is narrow and formal methodologies are not applied, the outcome of the paper clearly demonstrates that there are a variety of different attack factors in digital identity systems, and decentralized identity systems in particular.

This paper makes the case that developers shall apply formal threat modeling methodologies and security testing within a secure development life cycle from inception to end of life of an identity software solution component. As a result of the threat modeling and security testing, risks must be analyzed, mitigation measures implemented, and formal controls established.

> *"The only way to stop a hacker is to think like one."*
>
> *—Hacking Google* [7]

It shall be understood that identity systems must be embedded into a trust framework that is linked to a 'Level of Assurance' (LoA) model. The level of assurance is measured by the strength and rigor of the identity proofing process, the strength of the token used to authenticate the identity claim, and the management processes the identity provider applies to it.

To establish trust, solution providers shall implement strong PKI-based identity infrastructure and cryptographic tokens to authenticate identity claims. In decentralized identity systems the PKI identity infrastructure is built upon DID documents.

Therefore developers shall implement the mitigation measures described above and cannot implement the DID CRUD risk mitigation measures in isolation. Mitigation measures must be part of a set of ecosystem-wide conformance criteria that are endorsed, audited, and enforced by the entire use-case ecosystem and the respective identity & trust governance structure. If mitigation measures are implemented in isolation or inconsistently across different solution providers or their conformance is not enforced, a consistent, reliable LoA cannot be achieved. Hence, trust is broken in systems with weak identity conformance governance or less secure software development practices.

7 Youtube, Hacking Google Documentary

Glossary

#	Acronym	Definition
1	APT	Advanced Persistent Threat
2	CI/CD	Continuous Integration / Continuous Deployment
3	CRUD	Create, Read, Update, Deactivate
4	COTS	Commercial Off the Shelf
5	CSV	Computerized System Validation
6	DID	Decentralized Identifier
7	HMS	Hardware Security Module (for Key Management)
8	ISMS	Information Security Management System
9	KMS	Key Management System
10	LoA	Level of Assurance
11	OWASP	Open Web Application Security Project
12	RNG	Random Number Generator
13	SDLC	Secure (SW) Development Life Cycle
14	VC	Verifiable Credential

Additional Credits

Lead Author: Kate Sills (Independent, work funded by Digital Contract Design)

Authors: Shannon Appelcline (Blockchain Commons), Cihan Saglam (Danube Tech), Kate Sills (Independent, work funded by Digital Contract Design), Carsten Stöcker (Spherity)

Artwork: DALL·E 2

Sample APA Citation

Appelcline, S., Saglam C., Sills, K., and Stöcker, C. (2022). Taking out the CRUD: Five Fabulous DID Attacks. Rebooting the Web of Trust XI. Retrieved from https://github.com/WebOfTrustInfo/rwot11-the-hague/blob/master/final-documents/taking-out-the-crud-five-fabulous-did-attacks.pdf.

About Rebooting the Web of Trust

This paper was produced as part of the Rebooting the Web of Trust XI design workshop. On September 26th to 30th, 2022, over 60 tech visionaries came together in The Hague, The Netherlands to talk about the future of decentralized trust on the internet with the goal of writing at least 5 white papers and specs. This is one of them.

- **RWOT Board of Directors:** Christopher Allen, Joe Andrieu, Erica Connell.
- **RWOT11 Coordination Team:** Will Abramson, Christopher Allen, Joe Andrieu, Shannon Appelcline, Erica Connell, Eric Schuh, Carsten Stöcker.
- **Workshop Credits:** Will Abramson (Producer), Christopher Allen (Founder), Shannon Appelcline (Editor-in-Chief), Erica Connell (Host), Amy Guy (Ombudsperson), Willemijn Lambert (Graphic Recorder), Eric Schuh (Ombudsperson), Carsten Stöcker (Co-Producer, Demo Organizer), Dorothy Zablah (Facilitator).
- **Gold Sponsors:** The City of the Hague, Digital Contract Design, Dutch Blockchain Coalition, The Hague University of Applied Sciences, eSSIF-Lab.
- **Contributing Sponsors:** Blockchain Commons, Legendary Requirements, Spherity.

Thanks to all our attendees and other contributors!

What's Next?

The design workshop and this paper are just starting points for Rebooting the Web of Trust. If you have any comments, thoughts, or expansions on this paper, please post them to our GitHub issues page:

https://github.com/WebOfTrustInfo/rwot11/issues

The twelfth Rebooting the Web of Trust design workshop is scheduled for early 2022 in Los Angeles, California. If you'd like to be involved or would like to help sponsor the event, email: Leadership@WebOfTrust.info