Xia *et al. Cybersecurity* (2026) 9:97
https://doi.org/10.1186/s42400-025-00544-5

Cybersecurity

RESEARCH Open Access

Revocable signature: handling valid but unauthorized Non-Fungible Token through Auxiliary Embedded Key

Yuxin Xia[1], Ziyang Ji[1], Jie Zhang[1*], Wanxin Li[1], Ka Lok Man[1,2], Steven Guan[1] and Dominik Wojtczak[3]

Abstract

Non-Fungible Token (NFT) creators use digital signatures to ensure the ownership, authenticity, integrity, and non-repudiation of their digital works. However, if the private key is compromised, an attacker can generate unauthorized NFTs by using the creator's private key to issue valid signatures. These valid but unauthorized signatures will be accepted in the NFT market and cannot be revoked. Even if the NFT creators update their private-public key pairs, they cannot deny the NFTs generated by the attacker. To mitigate these risks, we propose revocable signature by introducing commitment mechanism and an Auxiliary Embedded Key (*AEK*) into the signature, while the regular verification process does not involve this *AEK*. If a valid but unauthorized signature is detected and needs to be revoked, *AEK* will be disclosed to perform the revocation operation. To illustrate the application of revocable signatures in NFT, we design and implement a revocable Elliptic Curve Digital Signature Algorithm (ECDSA) scheme with provable security. Experimental evaluations on the FIPS-recommended elliptic curves show that the performance of revocable ECDSA is comparable to the basic ECDSA, with additional 0.0303 s (P-256 curve) and 0.15 USD gas fee in Remix VM for revoking a signature.

Keywords Digital signature, Blockchain, Non-Fungible Token, Private key leakage, Revocability, ECDSA

Introduction

Motivation

The Non-Fungible Token (NFT) has emerged as a cornerstone of digital ownership, enabling creators to authenticate and trade unique digital assets through blockchain technology (Hammi et al. 2023; Wang et al. 2021). It relies on digital signature technology to ensure uniqueness of ownership, authenticity, non-tamperability, and verifiability of the creator's identity. Thus, the private key of digital signature scheme plays an important role in NFT application scenarios. As illustrated in Fig. 1, the creator needs to use his private key to sign both minting requests for new NFTs and transaction information for listing them.

If the private key is compromised, an attacker can generate legitimate and irrevocable NFTs by generating valid signatures on the minting requests using the private key. Specifically, the attacker can send requests to the smart contract. Then the smart contract mints NFTs, associates them with attacker's metadata, and registers these tokens on-chain under the identity of the legitimate creator. Furthermore, the attacker can authorize the list by signing the transaction information using the leaked private key, which will pass the verification on the blockchain. Since NFTs are often traded on decentralized marketplaces and immediately accessible to collectors or automated agents, valid but unauthorized tokens may be sold before the

*Correspondence:
Jie Zhang
Jie.Zhang01@xjtlu.edu.cn
[1] School of Advanced Technology, Xi'an Jiaotong-Liverpool University, Suzhou, China
[2] Kazimieras Simonavicius University, Lithuania and Vytautas Magnus University, Vilniaus, Lithuania
[3] School of Electrical Engineering, Electronics and Computer Science, University of Liverpool, Liverpool L69 3GJ, UK

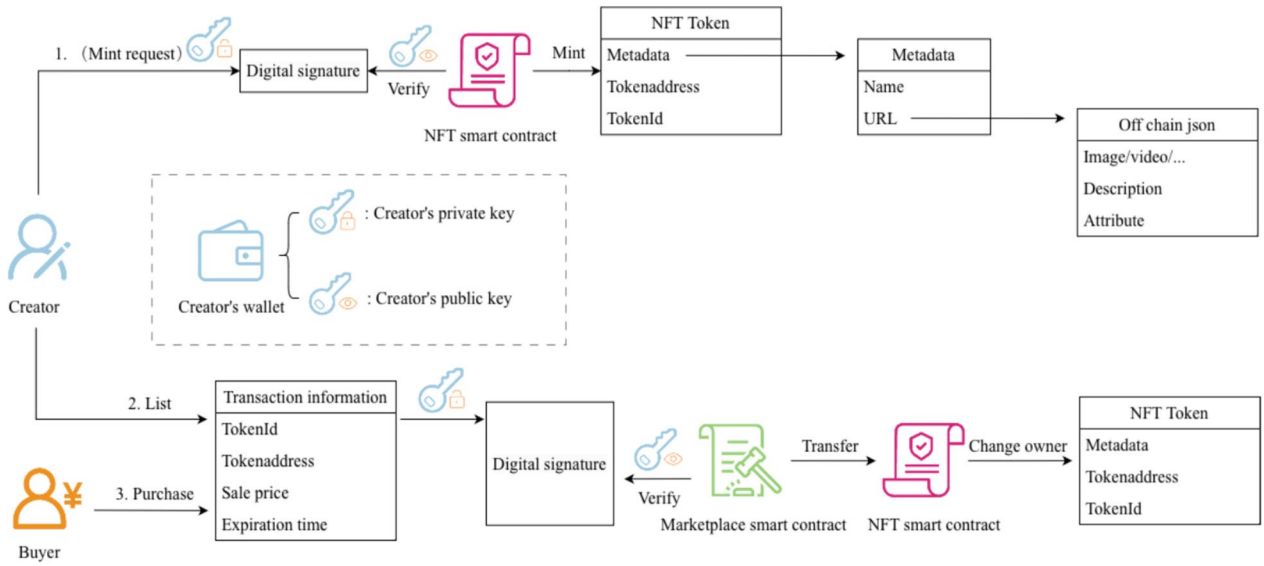


Fig. 1 Digital signature in the NFT system. It is involved from Phase 1 when the creator mints an NFT to Phase 2 and 3 when the NFT is listed in the marketplace and then purchased by a buyer

real creator becomes aware of the breach. The economic value, reputation, and provenance guarantees associated with NFTs are therefore critically undermined, and given the immutable nature of blockchain records, dealing with such an incident is extremely difficult without additional cryptographic mechanisms.

For instance, in 2020, after the death of digital artist Qing Han, fraudsters seized the opportunity to sell her works of art as NFTs, the most popular sold for $69 million (Kwan 2020). In September 2022, the website of renowned graffiti artist Banksy was hacked, leading to a page promoting and selling his NFT works, which resulted in a collector paying $336,000 for a valid but unauthorized digital art (Davidson 2022). Tyler Hobbs, the creator of the Art Blocks project "Fidenza", publicly criticized the SolBlocks platform for selling copies of his work using his code (Quarmby 2021). Additionally, the art of Derek Laufman was sold by a fraudulent account impersonating his identity (Stephen 2021). Such scams are numerous and widespread.

Although there exist two types of methods that specifically focus on the security of private keys, none of the existing solutions can effectively address the revocation of signatures issued by attackers with leaked private keys. Traditional key rotation (Everspaugh et al. 2017; Vijayakumar et al. 2011) limits the exposure window of private keys by periodically updating them, however, it cannot revoke signatures issued by attackers under leaked private keys during the interval from the leaking to updating of private keys. Threshold cryptography (Desmedt 1994) reduces the risk of a single point of compromise by splitting the private key into multiple segments and dynamically updating the segments during key rotation. However, if shares are eventually reconstructed (e.g., through collusion or attack persistence), the private key will be leaked ultimately. Threshold schemes focus on preventing compromise, not revoking its consequences. In general, applying these schemes directly to digital signatures only mitigates the future risk of private key compromise but cannot solve the problem of misuse of leaked private keys. In short, how to revoke signatures issued by leaked private keys remains a significant and unaddressed problem.

Our work

To address the problem, this paper proposes a new type of signature scheme, named as revocable signature.

Generic construction of revocable signature. Compared to existing methods that focus on preventing future compromises but fail to address the legacy risks of past key exposures, the revocable signature scheme enables signature revoking by integrating commitment and revocation mechanisms in a digital signature scheme.

The basic signature scheme includes three algorithms: key generation algorithm, signing algorithm, and verification algorithm. Revocable signature introduces a revoke algorithm into a basic signature scheme. During key generation, an Auxiliary Embedded Key (*AEK*) is generated in addition to a pair of public and private keys. Additionally, a public commitment of *AEK* is released as *AEK-com*. The signing algorithm embeds *AEK* into signatures such that valid but unauthorized signatures

(generated without *AEK*) can be distinguished from legal ones and thereby revoked via the revoke algorithm. The verification algorithm remains the same as in a basic signature scheme.

It is crucial to note that only the legitimate owner is capable of running the revoke algorithm, since the revoke algorithm requires the input *AEK*, which is never disclosed publicly. Since the private key-leakage attacker does not have *AEK*, they can only issue signatures that are checked to be valid via the verification algorithm but detected as unauthorized through the revoke algorithm. These signatures are thereby referred as "valid but unauthorized" signatures.

Even if an attacker is aware of the existence of *AEK* and its role in the revocation mechanism, the incentive to generate valid but unauthorized signatures remains strong in many real-world settings. This is because such signatures are immediately accepted as valid by standard verification algorithms and can be used to trigger irreversible operations before revocation takes effect. In time-sensitive or high-value contexts, such as the NFT markets in this paper, the attacker may profit from transient validity before revocation is executed, making forgery attempts still attractive despite the risk of later exposure and invalidation.

Instantiation of revocable signature. Following the idea of revocable signature, this paper designs a concrete Revocable Elliptic Curve Digital Signature Algorithm (R-ECDSA) scheme from the basic ECDSA (Johnson et al. 2001). The reason for choosing ECDSA is that it has been widely adopted by mainstream blockchain systems, which serve as the underlying infrastructure for NFT trading marketplaces (e.g., OpenSea). The proposed R-ECDSA ensures that even if an attacker is able to steal the private key and generate a valid but unauthorized signature that can be accepted in the NFT marketplaces, the true owner of the private key can deny and revoke it. The existential unforgeability of R-ECDSA is proved based on the assumption of ECDSA's existential unforgeability against chosen-message attacks (EU-CMA). A prototype of the scheme is realized on an ordinary laptop (Apple M2 pro chip, 16.00-GB memory). The computational overhead of R-ECDSA and basic ECDSA is compared on the basis of three elliptic curves P-256, P-384 and P-521.

For the most widely used NIST P-256 elliptic curve, the time of key generation algorithm is 0.0103 s of R-ECDSA vs 0.0087 s of basic ECDSA. The time of signing algorithm is 0.0073 s vs 0.0149 s, and the time of verification algorithm is 0.0149 s vs 0.0147 s. The results show that the efficiency of the three basic algorithms in R-ECDSA is comparable to those in basic ECDSA within the margin of error. The revocation algorithm takes only 0.0303 s, which is also acceptable for real-world deployment. We also test the gas consumption of the revocation algorithm as it will be performed by smart contracts in real-world scenarios. The result is 87,566 gas, that is 0.15 USD according to the exchange rate of August 9, 2025.

In summary, our primary contributions are concluded as following:

1. We formally define the revocable signature scheme, whose core idea is to decouple cryptographic validity from true authorization, enabling the signer to revoke valid but unauthorized signatures generated by a leaked private key. We further introduce a new security model, existential unforgeability under chosen-message and private-key leakage attacks (EU-CM&PLA).
2. We design R-ECDSA, a practical instantiation of the revocable signature built on ECDSA and fully compatible with existing verification logic for seamless adoption.
3. We provide a formal security analysis of R-ECDSA based on the standard EU-CMA security assumption of ECDSA.
4. We implement a full prototype and demonstrate its real-world feasibility. Our evaluation confirms that R-ECDSA adds negligible computational overhead and economically viable gas cost, making it a practical solution for securing NFT ecosystems.

Organization

The remainder of this paper is organized as follows. Section Related works provides an overview of related work. Section Preliminary presents the mathematical, cryptographic and smart contract background. Section Generic constructions presents the formal definitions of revocable signature. Section Instantiation of revocable signature presents the R-ECDSA scheme, proves its security, and evaluates its performance, respectively. Section Use case illustrates the application of revocable signature in the NFT system. Finally, Section Conclusion concludes the paper and proposes future work.

Related works

This section reviews current revocation schemes in the blockchain and existing revocation schemes in the field of digital signature.

Revocation in blockchain

Although immutability is a core feature of blockchain, it presents certain challenges in practical applications, such as the inability to revoke or update transactions that have already been completed, even when errors are identified. To address this problem, revocation in blockchain has attracted attentions in this field. Several specific methods tailored to different revocation needs have been proposed. Existing methods can be classified as arbitration-based revocation, rollback-based revocation and escrow-based revocation. Below we review some representative approaches and compare them in Table 1.

Arbitration-based revocation (Deuber et al. 2019; Ren et al. 2022) is a mechanism to achieve authority/decision revocation through a pre-determined arbitration agreement, where the revocation is triggered by a multi-party verification or authoritative ruling. For example, Lesaege et al. (2019) is a decentralized application built on top of Ethereum. It relies on game theoretic incentives to have jurors rule cases correctly. However, its core mechanism relies on a network of on-chain jurors rather than creator-autonomous execution of operations, which fundamentally conflicts with the creator autonomy that the NFT ecosystem values. More critically, the dispute resolution protocol may result in high latency, which prove fatal in the fast-paced NFT marketplace. An unauthorized NFT could be minted, sold, and traded multiple times before arbitration yields a verdict, by which point the attacker will have already secured their profits and exited. Finally, the "fairness"of arbitration requires additional mechanisms to ensure, making it vulnerable to attacks or manipulation.

Rollback-based revocation (Liao et al. 2023; Peng and Xu 2022) is the revocation of a recent operation or privilege change after a security threat has been detected, restoring the state of the system to a previous point in time. For example, some customized NFT contracts (Groschopf et al. 2021; Kubilay et al. 2019; Hu et al. 2020) add control permissions on top of the basic standard to support the rollback of erroneous transactions. However, this requires the revocation logic to be predefined in the smart contract, which increases the complexity of the operations. Additionally, rollback can mistakenly hurt other legitimate transactions signed with the same private key. For instance, if a creator legitimately mints 10 NFTs and an attacker fraudulently mints an 11th, a rollback would indiscriminately jeopardize all 11 NFTs, thereby punishing innocent collectors who purchased the legitimate assets. Furthermore, if the legitimate NFTs have already been worked as collateral or used in games, the rollback would trigger cascading failures.

Escrow-based revocation (Kumar and Tripathi 2019; Rajalakshmi et al. 2018; Guo et al. 2023) is the revocation of a transaction if a disputed issue arises prior to closing, which can be completed by the escrow provider. In certain transactions, assets are held in an escrow account until all parties confirm the transaction's validity. If any anomalies are detected, particularly when disputes, errors, or the need for intermediary intervention arise, the escrow provider can stop or reverse the transaction. Escrow-based revocation is designed to mediate the transaction phase (e.g., a buyer's payment and seller's transfer). However, a core problem we address is fraudulent minting, where an attacker uses a leaked private key to call the mint function directly. This minting operation entirely bypasses any escrow. Consequently, escrow schemes are completely ineffective in preventing the creation of a valid but unauthorized NFT. Additionally, The revocation of a transaction in an escrow account typically relies on pre-agreed conditions and the intervention of intermediaries. As the process becomes more complex, it can impact user experience.

In summary, arbitration-based revocation imposes substantial governance overhead and latency in decision-making due to its reliance on third-party adjudication processes, contravening the foundational principle of decentralized autonomy. Rollback-based revocation forces a state reset that destroys the finality of the transaction and triggers an on-chain historical credibility crisis. Escrow-based revocation relies on a third-party

Table 1 Existing revocation solutions in blockchain

Schemes	Revocation method	Limitations (or characteristics)
Arbitration-based revocation (Deuber et al. 2019; Ren et al. 2022; Lesaege et al. 2019)	Multi-vote decision	Slow processing, possible vote manipulation
Rollback-based revocation (Liao et al. 2023; Peng and Xu 2022; Groschopf et al. 2021; Kubilay et al. 2019; Hu et al. 2020)	Contract logic decision	May affect normal trading
Escrow-based revocation (Kumar and Tripathi 2019; Rajalakshmi et al. 2018; Guo et al. 2023)	Third-party intermediary decision	Intermediaries may have security risks
Proposed scheme	*AEK* decision	No third-party dependency, conditional revocation

escrow agent, which carries the risk of a single point of failure and requires preset conditions and intermediary interventions, which is not flexible enough. None of these methods can revoke already-minted NFTs signed under leaked private keys because the verification logic on-chain cannot distinguish legitimate from unauthorized signatures. Our solution of revocable signature eliminates unnecessary steps by enabling revocation through verification of *AEK* when needed. It requires no additional entities, as the true owner of the private key can independently perform the revocation operation.

Revocation in signature schemes

Revocation has been explored in digital signature schemes such as time-deniable signature, ring signature, attribute-based signature, group signature and so on. Among these schemes, some use the term "denial", while others adopt "revocation". To maintain consistency, we unify this concept as "revocation"in our paper, where both terms convey the same semantic meaning in our context. While at the first glance revocation (or denial) in these signature schemes is close to the revocation in our work, the problems they address are totally different.

"Denial" in time-deniable signatures (TDS) (Beck et al. 2023) is used to refer to denying the creation of older signature content. It restricts signature authenticity to a specific time period. Beyond this period, signatures that do not provide strong authentication become easier to forge, enabling the signer to plausibly deny their creation. While TDS is effective in applications like Domain Keys Identified Mail (DKIM)-signed emails, it cannot revoke valid but unauthorized signatures caused by the leakage of private keys within any period.

The term "denial"and "revocation"are also used in some ring signature, attribute-based signature and certificateless signature schemes. However, "denial"and "revocation"in these works are not about the revocation of signature. In traditional ring signature schemes, the complete anonymity of signers may potentially allow malicious signers to shift blame onto other group members. To address this issue, Komano et al. (2006) and Gao et al. (2019) propose deniable ring signature schemes (DRS), which is designed to enable signers to demonstrate their non-involvement in generating specific signatures through zero-knowledge proofs. This mechanism allows legitimate members to proactively deny illegitimate signatures via interactive verification protocols, thereby ensuring that innocent participants can effectively exonerate themselves from false accusations while preserving the fundamental properties of signer anonymity. The schemes of Liu et al. (2007) and Bresson and Stern (2001) grant a designated set of authorities the ability to revoke the signer's anonymity. Authorities can reveal the true identity of the signer when necessary, prevent abuse of anonymity, and enforce accountability for malicious behavior. Escala et al. (2011) also to prevent anonymous abuse, propose a revocable attribute-based signature based on commitment program and zero-knowledge proofs. Sun et al. (2013) try to address membership revocation by improving the composition of the signer's private key thus better managing signers.

In summary, while it seems that the terms of "denial"and "revocation"in these schemes are close to the revocation of signatures in this paper, they deal with different research questions. Existing deniable and revocable signature schemes provide no solution to address the problem of revoking valid but unauthorized signatures caused by private key leakage.

Preliminary

This section reviews the cryptographic and mathematical foundations of revocable signature, then introduces the smart contract. Symbols used in this paper are explained in Table 2.

Elliptic Curve Cryptography (ECC)

Let F_p be a finite field of p elements, where p is a prime. An elliptic curve E that is suitable for cryptography is defined by the equation in the variables x and y of the form:

$$y^2 = x^3 + ax + b \mod p \tag{1}$$

where $a, b \in F_p, 4a^3 + 27b^2 \neq 0 \mod p$. For detailed information about elliptic curves, we refer the reader to Silverman's book (Silverman 1986).

An elliptic curve group $\mathbb{G}$ is defined by all points on the elliptic curve plus a point $\mathcal{O}$ at infinity and addition operation $+$ among these points. The exponentiation in $\mathbb{G}$ is defined as

Table 2 Notations

Symbol	Description
F_p	A finite field with prime p
E	An elliptic curve over a prime field F_p
$\mathbb{G}$	An elliptic curve group
G	A base point on elliptic curve group $\mathbb{G}$
q	The order of $\mathbb{G}$
Z_q^*	The set of integers $\{1, ..., q-1\}$
$\leftarrow$	Sampling an instance from a distribution
$\perp$	Halt

$$n \cdot Q = \underbrace{Q + \cdots + Q}_{n}$$

for any point $Q \in \mathbb{G}$ and a positive integer n, which is also known as scalar multiplication.

The security of ECC is based on the difficulty of Elliptic Curve Discrete Logarithm Problem (ECDLP).

Definition 1 (ECDLP) Given a point $M \in \mathbb{G}$ of order q, and a point $N = l \cdot M$ where $l \in Z_q^*$, determine l.

Digital signature

Formal definitions

A digital signature scheme is defined by a tuple of probabilistic polynomial time (PPT) algorithms (*Gen*, *Sign*, *Verify*) such that:

- $(pk, sk) \leftarrow Gen(1^\lambda)$: Given a security parameter λ, the *Gen* algorithm outputs a public-private key pair.
- $\sigma \leftarrow Sign(m, sk)$: Given a private key sk and a message m, the *Sign* algorithm outputs the signature σ.
- $b \in \{0, 1\} \leftarrow Verify(m, \sigma, pk)$: Given a message m, signature σ, and public key pk, the *Verify* algorithm outputs 1 indicating that the signature is valid or 0 otherwise.

The correctness of a digital signature scheme is defined as follows.

Definition 2 (Correctness of Digital Signature) For any pair of $(pk, sk) \leftarrow Gen(1^\lambda)$, if $\sigma \leftarrow Sign(m, sk)$, then $1 \leftarrow Verify(m, \sigma, pk)$.

Security of digital signature

EU-CMA is a widely used security model for digital signature schemes. It is defined through the EU-CMA security game, where an adversary $\mathcal{A}$ interacts with a challenger $\mathcal{C}$ and tries to break the signature scheme. The game proceeds as follows:

Setup. Let λ be the system parameters. $\mathcal{C}$ runs the key generation algorithm to generate a key pair (pk, sk) and sends pk to $\mathcal{A}$. $\mathcal{C}$ keeps sk to respond to signature queries from $\mathcal{A}$.

Query. $\mathcal{A}$ makes signature queries on messages that are adaptively chosen by $\mathcal{A}$ itself. For a signature query on the message m_i, $\mathcal{C}$ runs the sign algorithm to compute σ_i and then sends it to $\mathcal{A}$.

Forgery. $\mathcal{A}$ returns a valid but unauthorized signature σ^* on some m^* and wins the game if σ^* is a valid signature and a signature of m^* has not been queried in the query phase.

The advantage ε of $\mathcal{A}$ winning the game is the probability of returning a valid signature.

Definition 3 (EU-CMA) A signature scheme is (t, q_s, ε)-secure in the EU-CMA security model if there exists no adversary who can win the above game in time t with advantage ε after it has made q_s signature queries.

ECDSA

The widely adopted ECDSA signature scheme is parameterized by an elliptic curve group $\mathbb{G}$ generated by a base point G. The algorithm makes use of a hash function $H : \{0, 1\}^* \rightarrow Z_q^*$. The three algorithms of ECDSA are explained as follows:

- Key generation algorithm:
  1. Select a random integer $d \in Z_q^*$ as the private key, where q is the order of the elliptic curve group.
  2. Compute $h = d \cdot G$ as the public key.
- Signing algorithm:
  1. Select a random integer $k \in Z_q^*$.
  2. Compute $k \cdot G = (x_1, y_1)$.
  3. Compute $r = x_1 \mod q$. If $r = 0$ then go to step 1.
  4. Compute $s = k^{-1}(H(m) + dr) \mod q$. If $s = 0$ then go to step 1.
  5. Output the signature (r, s).
- Verification algorithm:
  1. Verify that r and s are integers in the interval $[1, q - 1]$.
  2. Compute $H(m)$.
  3. Compute $w = s^{-1} \mod q$.
  4. Compute $u_1 = H(m)w \mod q$ and $u_2 = rw \mod q$.
  5. Compute $u_1 \cdot G + u_2 \cdot h = (x_1', y_1')$.
  6. Compute $r' = x_1' \mod q$.
  7. Accept the signature if and only if $r' = r$.

The EU-CMA security of ECDSA has been proved in Brown (2005). We will use it as an assumption in the security proof of R-ECDSA.

Smart contract

The functionalities of an NFT are based on the underlying smart contracts. The smart contract encodes the rules and guidelines that define the token and its specific properties. When specific criteria are satisfied, a smart contract will automatically run and carry out its intended function. Digital contracts regulate token ownership and exchanges in NFTs.

Generic constructions

This section presents system overview, threat model and security model of revocable signature. Then we discuss the motivations of the adversary.

System overview

A revocable signature scheme is defined by a tuple of four PPT algorithms (`KeyGen, Sign, Verify, Revoke`). An epoch e in our paper is formally defined as a discrete temporal interval bounded by an execution of `KeyGen` and `Revoke`. Four algorithms are described as follows:

- $(sk_e, pk_e, AEK_e, AEK\text{-}com_e) \leftarrow \texttt{KeyGen}(\lambda, n)$: On input security parameter λ and the maximum number of epochs $n \in O(2\lambda)$, the generation algorithm outputs a secret and public key pair (sk_e, pk_e), and an Auxiliary Embedded Key pair $(AEK_e, AEK\text{-}com_e)$ in epoch $e \in [n]$.
- $\sigma_e \leftarrow \texttt{Sign}(sk_e, AEK_e, m)$: On input private key sk_e and Auxiliary Embedded Key AEK_e for epoch $e \in [n]$ and a message $m \in M$, the signing algorithm outputs a signature σ_e.
- $b \in \{0,1\} \leftarrow \texttt{Verify}(pk_e, m, \sigma_e)$: On input public key pk_e, a message m, and a signature σ_e for epoch $e \in [n]$, the verification algorithm outputs 1 indicating that the signature is valid or 0 otherwise.
- $b' \in \{0,1\} \leftarrow \texttt{Revoke}(\sigma_e, AEK_e, AEK\text{-}com_e)$: On input a signature σ_e and Auxiliary Embedded Key pair $(AEK_e, AEK\text{-}com_e)$ for epoch $e \in [n]$, the revocation algorithm outputs 1 indicating that the signature is unauthorized (need to revoke) or 0 otherwise.

The four algorithms are also illustrated in Fig. 2. The correctness of revocable signature is defined as follows:

Definition 4 (Correctness of revocable signature) For any pair of $(sk_e, pk_e, AEK_e, AEK\text{-}com_e) \leftarrow \texttt{KeyGen}(\lambda, n)$, if $\sigma_e \leftarrow \texttt{Sign}(sk_e, AEK_e, m)$, then $1 \leftarrow \texttt{Verify}(pk_e, m, \sigma_e)$ and $0 \leftarrow \texttt{Revoke}(\sigma_e, AEK_e, AEK\text{-}com_e)$.

Security definition

Security definition includes the definition of threat model and security model.

Threat model

We assume the following powers of adversary against a revocable signature scheme.

First, the adversary is able to request the signer to sign certain messages and obtain the corresponding signature pairs (messages and their corresponding signatures). This allows the adversary to accumulate information about the signer and the signing process. The adversary can then use the obtained signature pairs to analyze and attempt to forge signatures for other messages, with the ultimate goal of generating a valid signature for a message that the adversary has not previously requested.

In addition to the above powers inherited from adversaries' against a basic signature scheme, we further assume that the adversary in the revocable signature scheme is able to leak the private key. However, in this case the goal of adversary is to forge a valid signature which can pass both the verify (i.e., 1 is output) and the revoke algorithm (i.e., 0 is output).

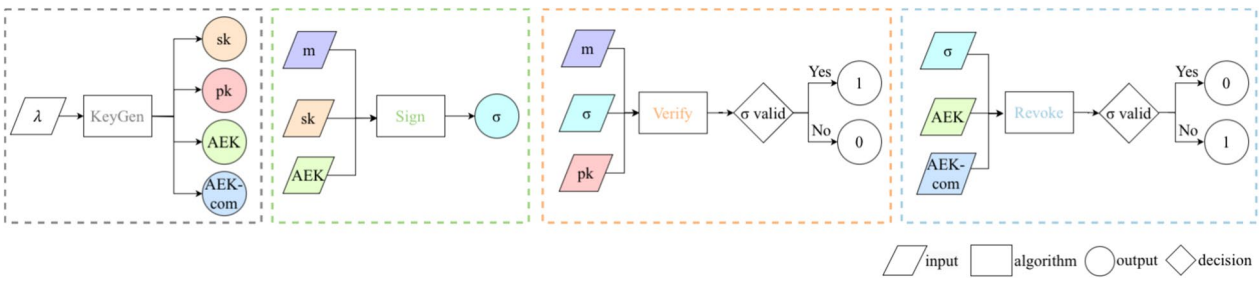


Fig. 2 General workflow of revocable signature of an epoch, which shows the algorithms of generating keys, signing messages, verifying signatures, and revoking them if necessary, focusing on how digital signatures ensure the authenticity and integrity of the message. An epoch starts with the `KeyGen` algorithm and ends with the `Revoke` algorithm

Security model

Based on the above threat model and the EU-CMA security model of ECDSA, we present the security model of revocable signature, denoted as EU-CM&PLA.

We first describe the security model of EU-CM&PLA, where security is defined by a game between a challenger and an adversary. This model formalizes the adversary's ability to make signature queries on adaptively chosen messages, as seen in the security model for an ECDSA scheme. It additionally formulates the adversary's power of leaking private keys by allowing the adversary to perform one private key leakage query in an epoch. The EU-CM&PLA security game is defined as follows:

Setup. The challenger runs `KeyGen` to generate public and private key pairs $((pk_1, sk_1), ..., (pk_n, sk_n))$, and Auxiliary Embedded Key pairs $((AEK_1, AEK\text{-}com_1), ..., (AEK_n, AEK\text{-}com_n))$ using the key generation algorithm and shares $\{pk_1, ..., pk_n\}$ and $\{AEK\text{-}com_1, ..., AEK\text{-}com_n\}$ with the adversary, while keeping $\{sk_1, ..., sk_n\}$ and $\{AEK_1, ..., AEK_n\}$ confidential to respond to the adversary's signature queries.

Query. In each epoch $e \in [n]$, the adversary can make several signature queries on messages of its own choosing and one private key leakage query. For a signature query on message m_i, the challenger computes the corresponding signature σ_{m_i} through running `Sign` and provides this signature to the adversary. For a private key leakage query in epoch $[1, n-1]$, the challenger sends sk to the adversary. In total, the adversary can query n epochs.

Forgery. The adversary submits a valid but unauthorized signature σ_{m^*} on some message m^* and wins the game if:

- σ_{m^*} is a valid signature of message m^*.
- No signature on m^* was previously requested during the query phase.

Definition 5 (EU-CM&PLA) A signature scheme is $(t, q_s, n, q_l, \varepsilon)$-secure in the EU-CM&PLA security model if no adversary can win the game described above within time t, making $q_s * n$ signature queries for all epochs involved, q_l private key leakage queries across n epochs (one private key leakage query per epoch) and with advantage ε.

Discussions

The discussions includes the discussion of assumption used in the threat model and security model and the discussion of attackers' motivation.

Discussion of model assumption

A crucial assumption in our threat model is that the private key sk_e and the Auxiliary Embedded Key AEK_e are not compromised simultaneously. Our security definition of EU-CM&PLA models the leakage of sk_e but assumes AEK_e remains secret. If an attacker were to compromise both sk_e and AEK_e, they would be indistinguishable from the legitimate owner and could generate valid and authorized signatures that can pass the `Verify` algorithm and cannot pass the `Revoke` algorithm. Our scheme relies on mitigating this risk through practical implementation, as discussed in Section Implementation considerations. We explicitly recommend storing AEK in a separate and secure manner from sk (e.g., in isolated offline storage like a USB device or NFC card), making a simultaneous compromise highly unlikely.

Discussion of attackers' motivation

One may doubt the motivation of attackers, given that they are aware that valid but unauthorized signature which does not embed the AEK could be revoked later. However, the attacks of generating unauthorized NFTs still make sense in our scenario.

Exploiting the latency for irreversible gains. The main motivation of the attacker is financial gain, made possible by the delay between a successful transaction and any revocation. Since blockchain guarantees immutability, once a transaction is confirmed, it cannot be reversed. An attacker can exploit this property (Kshetri 2022). The marketplace's smart contract, acting as a decentralized escrow, checks the signature σ_{m^*} using the public key pk. If the signature is valid, it completes the trade and records the result on-chain. By the time the true owner notices the unauthorized sale and triggers the `Revoke` to reveal AEK_e and revoke σ_{m^*}, the NFT has already been transferred. Due to blockchain immutability, this loss is permanent. The revocation only acts as a public notice of forgery, offering no way to recover the asset. This highlights that even a short delay can allow an attacker to extract the full economic value.

Inflicting irreparable reputation harm. Even after the true owner revokes the valid but unauthorized NFT, the NFT itself, along with its transaction history, remains a permanent artifact on the blockchain. The revocation proves the NFT is unauthorized, but it cannot erase the public record or the memory of the event. This incident can permanently tarnish the artist's brand equity by creating doubt among collectors about the authenticity and governance of their collection. This is exemplified by the

cases of Davidson (2022) and Quarmby (2021) involved in our introduction.

Degradation of system availability. During the execution of the `Revoke` algorithm, AEK_e needs to be reveled publicly. Then (sk_e, pk_e) and $(AEK_e, AEK\text{-}com_e)$ need to be updated. By repeatedly introducing forgeries, the attacker can compel the project to execute this process continuously. It degrades system availability, imposes significant operational overhead on the administrators, and erodes the marketplace's stability. The attacker effectively turns the system's own security feature into a vector for a resource-exhaustion attack (Luu et al. 2016).

Overall, private key-leakage attackers still have their motivation to issue valid but unauthorized NFTs for financial or other purposes. The proposed revocable signature cannot address all consequences caused by private key leakage, but provides effective measures to recouping losses to some extent and preventing the valid but unauthorized NFT from flooding the market.

Instantiation of revocable signature

This section presents a concrete revocable signature scheme based on the basic ECDSA, denoted as R-ECDSA = (`KeyGen, Sign, Verify, Revoke`). We first present the detailed procedure for each algorithm of an epoch and then prove the correctness of R-ECDSA according to Definition 4. Then we prove the EU-CM&PLA security of R-ECDSA scheme according to Definition 5. Finally, we evaluate the performance of R-ECDSA in terms of computation overhead and storage overhead.

R-ECDSA construction

R-ECDSA consists of four algorithms: the key generation algorithm `KeyGen`, the signing algorithm `Sign`, the verification algorithm `Verify` and the revocation algorithm `Revoke`, which are described as follows:

- `KeyGen`: This algorithm generates a standard ECDSA key pair (sk_e, pk_e) and, in parallel, a secret AEK_e and its public commitment $AEK\text{-}com_e$, which is simply its hash.
- `Sign`: Instead of using a random number directly, this algorithm derives a nonce c by hashing a timestamp ts and combining it with the secret AEK_e. This c is then used to compute the rest of the signature (r, s), and the final output is $\sigma_e = (r, s, ts)$.
- `Verify`: This algorithm is identical to standard ECDSA verification.
- `Revoke`: This algorithm takes the secret AEK_e and the signature σ_e as input. It recomputes the expected c' and r'. If the recomputed r' matches the signature's r, it means the AEK_e was included, so the signature is authorized. If they do not match, the signature is unauthorized. An attacker with only sk_e cannot produce a signature (r, s, ts) that passes this check.

Fig. 3 presents the detailed procedure of R-ECDSA in an epoch. After completing the revocation of the valid but unauthorized signatures, an epoch is ended. The key generation algorithm `KeyGen` will be invoked to initiate a new epoch.

We prove the correctness of R-ECDSA according to Definition 4.

Proof

For any $(sk_e, pk_e, AEK_e, AEK\text{-}com_e) \leftarrow$ `KeyGen`(λ, n), *and* $\sigma_e = (r, s, ts^*) \leftarrow$ `Sign`(sk_e, AEK_e, m),

we can compute

$$
\begin{aligned}
&z = H(m),\\
&w = s^{-1} \mod q = \frac{c}{z + r \cdot sk_e} \mod q,\\
&u_1 = zw \mod q = \frac{zc}{z + r \cdot sk_e} \mod q,\\
&u_2 = rw \mod q = \frac{rc}{z + r \cdot sk_e} \mod q.
\end{aligned}
$$

The verification point P is calculated as:

$$
\begin{aligned}
P &= u_1 \cdot G + u_2 \cdot pk_e\\
&= \frac{z \cdot c \cdot G}{z + r \cdot sk_e} + \frac{r \cdot c \cdot pk_e}{z + r \cdot sk_e}\\
&= \frac{z \cdot c \cdot G}{z + r \cdot sk_e} + \frac{r \cdot c \cdot sk_e \cdot G}{z + r \cdot sk_e}\\
&= \frac{c \cdot G(z + r \cdot sk_e)}{z + r \cdot sk_e}\\
&= c \cdot G
\end{aligned}
$$

So $(c \cdot G)_x$ is identical to r.

Furthermore, we can compute

$$
\begin{aligned}
&AEK - com'_e = H(AEK_e),\\
&c' = (ts^* + AEK_e) \mod q,\\
&r' = (c' \cdot G)_x.
\end{aligned}
$$

So r' is identical to r.

Therefore,

- `KeyGen`
  1. Choose $sk_e, AEK_e \in Z_q^*$
  2. Compute $pk_e = sk_e \cdot G$, $AEK\text{-}com_e = H(AEK_e)$
  3. return (sk_e, pk_e)
     return $(AEK_e, AEK\text{-}com_e)$
- `Sign`
  1. Compute $z = H(m)$
  2. Compute $ts^* = H(ts)$
  3. Compute $c = (ts^* + AEK_e) \bmod q$
  4. Compute $R = c \cdot G$
  5. Compute $r = R_x \bmod q$
  6. Compute $s = c^{-1}(z + r \cdot sk_e) \bmod q$
  7. $\sigma_e \leftarrow (r, s, ts)$
  8. return σ_e
- `Verify`
  1. Parse $\sigma_e = (r, s, ts)$
  2. Verify that r and s are integers in the interval $[1, q-1]$
  3. Compute $z' = H(m)$
  4. Compute $w = s^{-1} \bmod q$
  5. Compute $u_1 = z'w \bmod q$ and $u_2 = rw \bmod q$
  6. if $(u_1 \cdot G + u_2 \cdot pk_e)_x = r$ then
  7. return 1
  8. else return 0
- `Revoke`
  1. Compute $AEK\text{-}com'_e = H(AEK_e)$
  2. if $AEK\text{-}com'_e \neq AEK\text{-}com_e$, it outputs $\perp$
  3. else parse $\sigma = (r, s, ts)$
  4. Compute $ts^* = H(ts)$
  5. Compute $c' = (ts^* + AEK_e) \bmod q$
  6. Compute $r' = (c' \cdot G)_x$
  7. if $r \neq r'$ then
  8. return 1
  9. else return 0

Fig. 3 Revocable signature from the ECDSA

$$1 \leftarrow \texttt{Verify}(pk_e, m, \sigma_e),$$
$$0 \leftarrow \texttt{Revoke}(\sigma_e, AEK_e, AEK\text{-}com_e)$$

which indicates the correctness of verification algorithm and revocation algorithm. Therefore, we can prove the correctness of the R-ECDSA. □

Security proof of R-ECDSA

We prove the EU-CM&PLA security of R-ECDSA scheme based on ECDSA's EU-CMA security. Specifically, we first claim the R-ECDSA's EU-CM&PLA security in Theorem 1. Then we prove the theorem by reducing the security of R-ECDSA scheme to the security of the basic ECDSA scheme.

Theorem 1

R-ECDSA is $(t, q_s, n, q_l, \varepsilon)$-secure in the EU-CM&PLA model given that the ECDSA scheme is EU-CMA secure. Specifically, if there exists a PPT adversary $\mathcal{A}$ winning the EU-CM&PLA game with an advantage of ε, then there exists a PPT adversary $\mathcal{B}$ winning the EU-CMA game with an advantage of no less than ε.

Proof

Let $\mathcal{A}$ be a PPT adversary against R-ECDSA. $\mathcal{A}$ plays the EU-CM&PLA game with a challenger $\mathcal{B}$ who is also a PPT adversary against the ECDSA signature scheme. Thus, $\mathcal{B}$ plays an EU-CMA game with an ECDSA challenger $\mathcal{C}$. The interactions among the three entities are also illustrated in Fig. 4.

In the EU-CM&PLA game, $\mathcal{A}$ makes queries and $\mathcal{B}$ responses. To embed the EU-CMA game into the

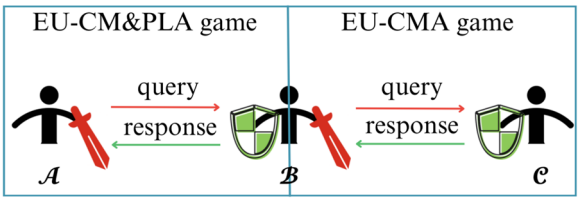


Fig. 4 Security proof method

EU-CM&PLA game, $\mathcal{B}$ forwards every signature query from $\mathcal{A}$ to $\mathcal{C}$ and then forwards the corresponding response from $\mathcal{C}$ to $\mathcal{A}$. Specifically, the interactions among $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ in the EU-CM&PLA game which embeds the EU-CMA game are explained as follows.

- **Setup:** $\mathcal{C}$ initializes the system parameters and generates a public and private key pair (pk, sk), pk is given to $\mathcal{B}$ and sk is kept secretly by $\mathcal{C}$. Then, $\mathcal{B}$ initializes the system parameters and generates public and private key pairs $((pk_1, sk_1), ..., (pk_{n-1}, sk_{n-1}))$, and Auxiliary Embedded Key pairs $((AEK_1, AEK\text{-}com_1), ..., (AEK_n, AEK\text{-}com_n))$. Moreover, $\mathcal{B}$ lets $pk_n = pk$, $\{pk_1, pk_2, ..., pk_n\}$ and $\{AEK\text{-}com_1, AEK\text{-}com_2, ..., AEK\text{-}com_n\}$ are given to $\mathcal{A}$, and $\{sk_1, sk_2, ..., sk_{n-1}\}$ and $\{AEK_1, AEK_2, ..., AEK_n\}$ are kept secretly by $\mathcal{B}$.
- **Query:** $\mathcal{A}$ can make signature queries to $\mathcal{B}$ at any epoch $e \in [1, n]$, and private key leakage queries to $\mathcal{B}$ in epoch $e \in [1, n-1]$. $\mathcal{B}$ responses to the queries as follows:
  - Signature queries: $\mathcal{A}$ submits a message m and queries $\mathcal{B}$ the signature on m. For epoch $e \in [1, n-1]$, $\mathcal{B}$ runs $\mathtt{Sign}(sk, AEK, m)$ to compute the signature σ on m and responses σ to $\mathcal{A}$. For epoch n, $\mathcal{B}$ submits m to $\mathcal{C}$ and queries the signature. $\mathcal{C}$ uses sk to compute the signature σ on m and responses σ to $\mathcal{B}$. $\mathcal{B}$ forwards σ to $\mathcal{A}$.
  - Private key leakage queries: $\mathcal{A}$ submits a private key query to $\mathcal{B}$. $\mathcal{B}$ responses sk_e to $\mathcal{A}$.
- **Forgery:** After q_s signature queries, and $q_l < t$ private key leakage queries, $\mathcal{A}$ returns to $\mathcal{B}$ a valid signature σ^* on a message m^* that has not been queried in the query phase. $\mathcal{B}$ returns (σ^*, m^*) to $\mathcal{C}$.

A wins the EU-CM&PLA game if σ^* is a valid signature for m^*. In this case, $\mathcal{B}$ wins the EU-CMA game. Therefore, the advantage of $\mathcal{B}$ winning the EU-CMA game is not less than the advantage of $\mathcal{A}$ winning the EU-CM&PLA game. Since the ECDSA scheme is EU-CMA secure provided that ECDLP is hard in G, which means the advantage of $\mathcal{B}$ winning the EU-CMA game is negligible. Therefore, ε is negligible and R-ECDSA is $(t, q_s, n, q_l, \varepsilon)$-secure in the EU-CM&PLA security model, provided that ECDLP is hard in G and $q_l < t$. Furthermore, we can draw the conclusion that R-ECDSA has EU-CM&PLA security. □

Performance of R-ECDSA

We evaluate the performance of R-ECDSA and compare it with the basic ECDSA in terms of computation overhead and storage overhead.

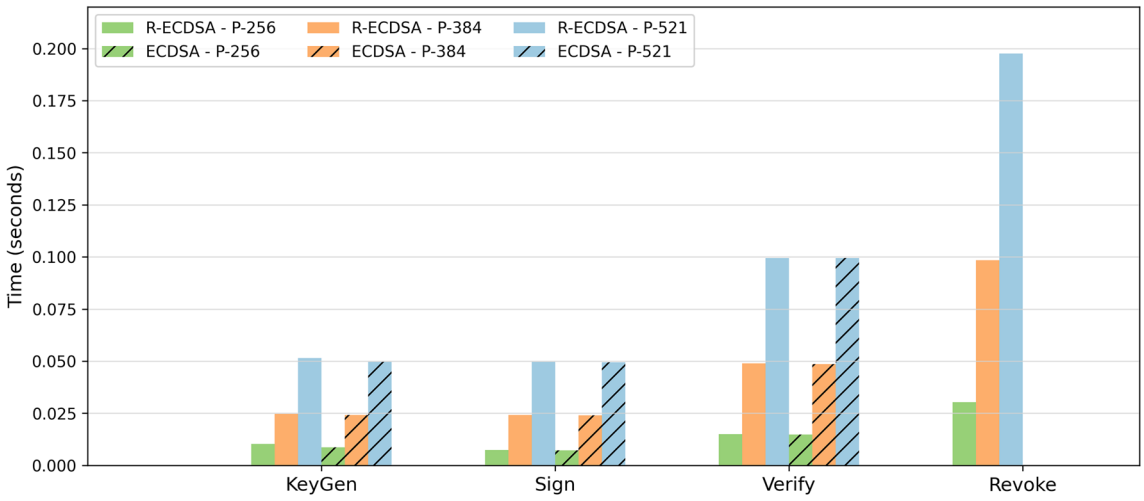
Computation overhead

We have realized a prototype for the scheme using Python programming language, and run it to test the average run time. The hardware and software environments are specified in Table 3.

In the process of implementation using Python, the hashlib module is employed for various secure hashing and message digest algorithms, along with several other modules. The curves used in the experiment are the FIPS standard curve P-256, P-384 and P-521 (Chen et al. 2019). In the experiment on the most widely used P-256 curve, we run the prototype ten times. The average computation time is 0.0414 s. The computation time on common devices in the real world is ranging from 0.0402 s to 0.0427 s, which is a very short time slot and is acceptable for time-sensitive scenarios. Additionally, we count the runtime of R-ECDSA and the basic ECDSA scheme on the four algorithms (`KeyGen`, `Sign`, `Verify`, `Revoke`). Fig. 5 illustrates the performance comparison. The results indicate that R-ECDSA is comparable with those in basic ECDSA in the common three phases on all tested curves. The runtime of `Revoke` algorithm ranges from 0.0303 s to

Table 3 Execution environment

Item	Implementation specification
Computing device	MacBook pro with apple M2 pro chip, macOS Ventura operating system, a 16.00-GB Memory, 512.00-GB Storage
Compiler environment	PyCharm
Elliptic curve	FIPS standard curve P-256, P-384, and P-521
$H()$	use SHA-2 series (SHA-256, SHA-384, and SHA-512) to generate binary data, convert the binary data to string, then convert the string to integer and modulo q



	R-ECDSA	ECDSA	R-ECDSA	ECDSA	R-ECDSA	ECDSA	R-ECDSA	ECDSA
P-256	0.0103	0.0087	0.0073	0.0072	0.0149	0.0147	0.0303	/
P-384	0.0247	0.0241	0.0241	0.0240	0.0489	0.0486	0.0984	/
P-521	0.0514	0.0497	0.0496	0.0494	0.0995	0.0994	0.1976	/

Fig. 5 Comparison of computation overhead between R-ECDSA and basic ECDSA on different elliptic curves (P-256, P-384 and P-521) for `KeyGen`, `Sign`, `Verify`, and `Revoke` algorithms

Table 4 Stored key size (in byte)

	R-ECDSA		Basic ECDSA	
	Signer	Verifier	Signer	Verifier
P-256	65	130	32	64
P-384	92	184	48	96
P-521	121	242	66	132

0.1976s, which means the algorithm itself is lightweight in terms of computing burden.

Storage overhead

The storage overhead is evaluated in terms of both the size of the stored keys and the size of the compiled files.

Key size. Table 4 compares the key size of R-ECDSA and basic ECDSA based on three different curves. In R-ECDSA, the signer needs to store the private key sk and the Auxiliary Embedded Key AEK, the verifier needs to store the public key pk and the public commitment $AEK\text{-}com$. In basic ECDSA, signer needs to store the private key sk and verifier needs to store the public key pk.

Therefore, for P-256, the storage overhead of R-ECDSA is calculated as (32 + 33) + (64 + 66) = 195 bytes. The storage overhead of basic ECDSA is calculated as 32 + 64 = 96 bytes. Similarly, for P-384, R-ECDSA requires 276 bytes compared to 144 bytes in basic ECDSA, and for P-521, the respective storage requirements are 363 bytes and 198 bytes. Despite the additional overhead introduced by R-ECDSA, the storage increase remains moderate, ranging from 195 bytes to 363 bytes. Given modern computational resources, this increase is relatively small and does not impose a significant storage burden.

Compiled file size. Moreover, we test the size of compiled file, .pyc file. The size of compiled file needs to be stored ranges from 4 KB to 5 KB, which has a compact storage footprint.

Use case

Application background

We consider an implementation within the Ethereum Virtual Machine (EVM) ecosystem. The EVM is a widely adopted decentralized computation environment that defines the runtime behavior of smart contracts. Numerous marketplaces and applications such as Ethereum, Polygon and OpenSea adopt EVM-compatible architectures. Among the most prominent standards built on EVM is ERC-721, a widely used interface for NFTs. ERC-721 defines a standard API for smart contracts to manage ownership, transfer, and many other functions of NFTs.

However, ECDSA deployed in EVM is a little different from basic ECDSA. Specifically, basic ECDSA sign algorithm outputs $\sigma = (r, s)$, while a recovery bit v is additionally output in EVM. During the verification algorithm, basic ECDSA inputs $\sigma = (r, s)$ and pk to check if the signature is valid; however, in EVM, the public key is first

derived from (r, s, v) as $pk = r^{-1}(s \cdot R - z \cdot G) \mod q$, where $r \cdot G = \{R, R'\}$ and v is used to determine the correct point R from the two symmetry points on the curve. Finally, the intermediate value pk is used to derive the address of wallet which will be compared with the one published and recorded on the blockchain. This process is essentially a transformation of formulas in the verification algorithm of basic ECDSA. Our implementation strictly follows the specification of EVM.

Assume that Alice is an artist in the NFT system. She generates a pair of public and private keys (sk_c, pk_c), where sk_c will be stored in her wallet. The pk_c is used to generate the address of wallet, which will be published and recorded on the blockchain. When she wants to create NFTs, the sk_c is used to sign the request proving that Alice is the true creator of those NFTs. The smart contract of NFT uses the pk_c to verify the signature. If the signature is valid, the smart contract of NFT will mint NFTs. When Alice wants to list NFTs on the marketplace, she must sign the sale information using the sk_c. If a buyer Bob wants to purchase this NFT, the smart contract of marketplace uses the pk_c to verify the legitimacy of the signature. If the signature is valid, the transaction will go ahead, and the ownership of the NFTs will be transferred from Alice to Bob, this operation is executed by the smart contract of NFT.

We consider the case where Alice's sk_c is stolen by an attacker Mallory, which does happen in practice. In current NFT marketplaces, Mallory can use sk_c to sign the request of minting NFTs and list them on the marketplace. Since Mallory is using Alice's sk_c for signing, the signature verification will succeed, and the marketplace will consider it a valid transaction.

The proposed revocable signature can address the above problem. Fig. 6 shows the workflow of applying R-ECDSA into the system. In R-ECDSA, upon suspecting the leakage of sk_c and finding valid but unauthorized NFTs occurring in the market, Alice can initiate the `Revoke` algorithm in the system to mark valid but

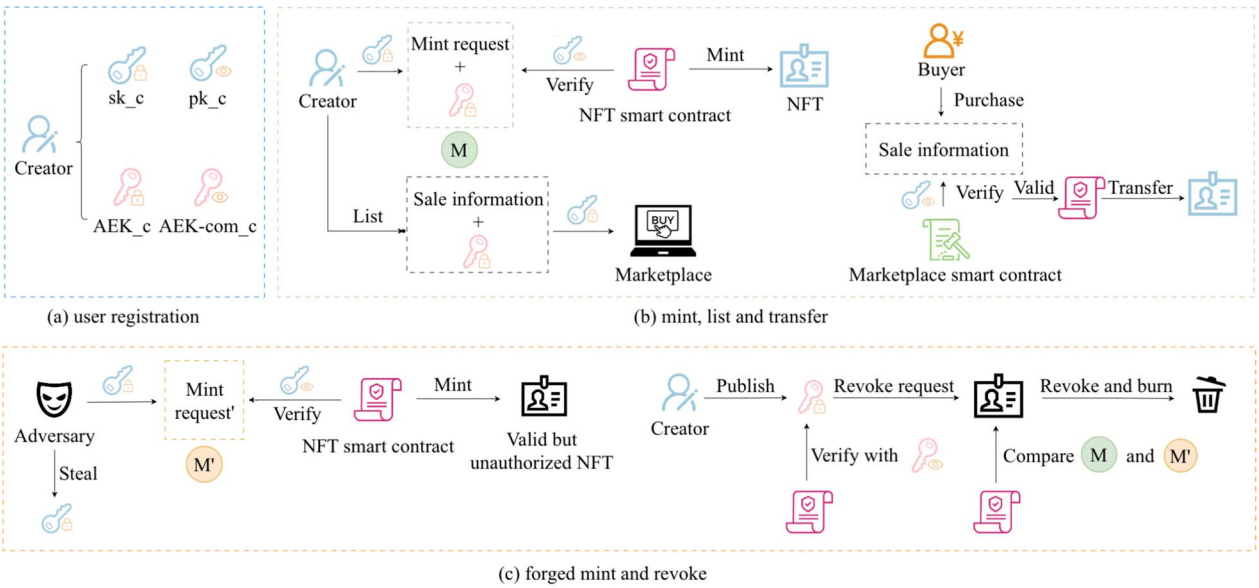


Fig. 6 Application of R-ECDSA in the NFT system

Table 5 Key functions

Function	Input	Output	Description
ecrecover ()	σ	address	Recover an address from a signature
_safeMint ()	*tokenId*, metadata	token	Safely mint NFTs
approve ()	address, *tokenId*	/	Give permission to transfer NFTs to another account
ownerOf ()	*tokenId*	address	Verify the current owner of the NFTs
safeTransferFrom ()	addresses, *tokenId*	/	Safely transfer NFTs
_burn ()	*tokenId*	/	Destroy *tokenId*

unauthorized NFTs. The following subsections will show how to deploy R-ECDSA in the NFT system and how the consequences of private key leakage is alleviated. Table 5 shows the key functions and their input, output and description used in the application.

User registration

When users register or log in to the marketplace, they establish a secure communication channel by connecting their wallet. This allows the marketplace to send requests to the user's wallet and query information such as the NFTs and transaction history through the wallet address, and display a personalized interface for the user.

Alice runs `KeyGen` algorithm to initialize system parameter: Input a security parameter into wallet to generate a key pair (public key pk_c and private key sk_c) and an Auxiliary Embedded Key pair (AEK_c, $AEK\text{-}com_c$).

The sk_c is kept secret in the wallet, and the AEK_c can be stored in the meta-info of the wallet or a portable plug-and-play offline storage (USB, NFC cards, etc.), assuming they are not leaked simultaneously. The pk_c is used to generate the address of the wallet, which will be published on the chain. The $AEK\text{-}com_c$ will be deposited into a mapping on the chain during the process of minting NFTs.

Bob runs `KeyGen` algorithm to initialize system parameter and generate a key pair (public key pk_b and private key sk_b), where sk_b is kept secret in the wallet, and the wallet address corresponding to pk_b will be published on the chain.

It is worth noting that the introduction of *AEK* does not interfere with the standard public key to address derivation mechanism. Wallet addresses remain solely determined by *pk*, preserving full compatibility with existing blockchain infrastructures.

Mint, list and transfer

Main operations in NFT system include mint, list and transfer. After installing R-ECDSA in the blockchain, the following procedures can be executed.

- Mint: Alice runs `Sign` algorithm to generate the request of minting NFTs. The smart contract of NFT runs `Verify` algorithm to verify the request and then mints NFTs.

  1. Alice inputs sk_c, AEK_c and request to generate a signature σ_{mint}.
  2. The smart contract of the NFT uses *ecrecover* () to recover address from σ_{mint} and compares it with the address of Alice. The `Verify` algorithm returns "1", which means that σ_{mint} is a valid signature.
  3. If σ_{mint} is valid, the smart contract of the NFT calls the function *_safeMint* () to mint the NFT and associate it with Alice's address.

- List: Alice runs `Sign` algorithm and calls *approve*() to list NFTs on the marketplace. Alice inputs sk_c, AEK_c and the sale information to generate a signature σ_{list}, where the sale information includes the sale price of NFTs, the expiration time, the *tokenId* and *tokenAddress* (the address of the smart contract of NFT). Then Alice calls *approve* () to give the smart contract of the marketplace permission to operate NFTs.
- Transfer: Bob sends the request of purchase, then the smart contract of marketplace runs `Verify` algorithm to verify σ_{list}. If valid, the smart contract of NFT transfer NFTs from Alice to Bob.

  1. Bob proposes the request of purchase, where the request includes the sale price of NFTs, the *tokenId* and the *tokenAddress*.
  2. The mart contract of marketplace calls *_ownerOf* () to check current owner. In addition, the NFTs will be checked if it has expired.
  3. Then the smart contract of marketplace uses *ecrecover* () to recover address from σ_{list} and compares it with the address of Alice. The `Verify` algorithm returns "1", which means that σ_{list} is a valid signature.
  4. After the validation passes, the smart contract of NFT updates the owner of the NFTs to Bob's address through the function *safeTransferFrom* () .

If Mallory obtains Alice's private key illegally, the above operation executed by Alice can easily be implemented by Mallory. Specifically, Mallory can use sk_c to sign the request of minting valid but unauthorized NFTs and make them successfully uploaded to the marketplace. Then, the valid but unauthorized NFTs can be traded normally. Mallory also can construct a fake transaction to transfer Alice's NFTs to an address under his control. The NFT marketplace will use Alice's pk_c to verify the validity of the digital signature. If the signature verification passes, the node will assume that the transaction was authorized by Alice. Since Mallory is using Alice's sk_c for signing, the signature verification will succeed, and the marketplace will consider it a valid transaction.

Revoke

If Alice discovers the valid but unauthorized NFTs in the marketplace, then she triggers a revocation operation.

The smart contract runs `Revoke` algorithm to deal with the valid but unauthorized NFTs.

1. Alice extracts AEK_c from her wallet, inputs AEK_c into the smart contract of NFT and proposes a revocation request.
2. The smart contract of NFT calculates if $AEK\text{-}com_c = hask(AEK_c)$ to ensure that the request comes from Alice herself.
3. The smart contract of NFT runs the `Revoke` algorithm. The output returns "1", which means that σ_{mint} and σ_{list} are valid but unauthorized signatures. Then the smart contract of NFT executes the revocation operation and these corresponding NFTs will be marked as "burned" in the metadata of NFTs. These corresponding *tokenId* will be destroyed through *_burn* () and cannot perform any further transactions.

Once a revocation process is completed, the system must ensure that the compromised key is no longer usable in any future operation. The marketplace will update the wallet address and unlink the old wallet address from the user's account. To continue using the system, Alice needs to restart the process from user registration. However, the specific steps go beyond the scope of this paper and will therefore not be described in detail.

Discussion

This subsection introduces two different methods of revocation, the specific steps of *_burn* () and implementation considerations.

The method of revocation

Modification of existing smart contract. This approach involves introducing a mapping into the existing smart contract to track the revocation status of each NFT. Specifically, each *tokenId* is associated with a boolean flag indicating whether the token has been marked as revoked. Once an NFT is identified as unauthorized, the contract can prevent any further transfer operations by setting the corresponding revoked flag. This mechanism offers a lightweight and effective solution, enhances asset protection for legitimate owners, and fosters greater trust within the marketplace. However, it requires modification of the contract and integration support from existing marketplaces. More critically, the revocation cannot retroactively invalidate transactions that have already been finalized on-chain.

Algorithm 1 Revocation Mechanism in Modification of Existing Smart Contract

```
mapping (uint256 ⇒ bool) public burnedTokens;
function reportAndMarkAsBurned(tokenId)
    isMarkedAsBurned[tokenId] = true;
end function
function burnUnauthorizedNFT(tokenId)
    require(isMarkedAsBurned[tokenId]);
    _burn(tokenId);
end function
```

Table 6 Gas consumption for transaction operations (as of August 9, 2025)

Operation	Transaction gas cost	Gas cost (in Gwei)	Gas cost (in Ether)	Gas cost (in USD)
Mint	125,512	51459.92	0.00005145992	0.21
Transfer	62,887	25783.67	0.00002578367	0.11
Revoke	87,566	35902.06	0.00003590206	0.15

Table 7 Comparison of gas cost between basic NFT transaction process and our scheme

Operation	Transaction gas cost		Gas cost (in USD)	
	Basic process	Our scheme	Basic process	Our scheme
Mint	103,449	125,512	0.18	0.21
Transfer	72,318	62,887	0.12	0.11
Revoke	/	87,566	/	0.15

Deploying a specialized smart contract. This approach deploys a revocation registry contract on-chain to store the revocation status of multiple NFTs. This solution not only integrates seamlessly with the original NFT contracts but also allows third-party marketplaces (such as OpenSea) to leverage the registry. Any marketplace that supports Web3 can query this registry to verify the validity of an NFT, as the revocation records are maintained in a standalone contract, independent of the specific NFT contract implementation. However, the storage of revocation data in an external contract adds complexity to the deployment. Additionally, each transfer operation requires interaction with the revocation registry contract, which could incur higher transaction costs.

Comparative analysis. We compare the above two methods across three dimensions:

- Security: The above methods present a trade-off between isolated risk and coupled risk. The specialized-contract method offers stronger isolation. Because the revocation logic resides in an independent contract, a compromise of a single NFT contract does not endanger the integrity of the ecosystem's central registry. However, this architecture introduces a new, centralized point of failure: the entire ecosystem's revocation capability becomes dependent on the security and governance of this single registry. Conversely, modified-contract method avoids this central dependency but couples the revocation logic with the token's core management functions. This creates a larger, monolithic attack surface for the individual contract, where a single bug in either the NFT logic or the revocation routine could compromise the entire asset.
- Performance: The modified-contract method is more lightweight and effective. Because the revocation status is stored locally, verifying it requires only a single operation. Conversely, the specialized-contract method architecture incurs significant gas cost, as this method is defined by its reliance on interaction with an external contract. Specifically, every transfer operation must interact with the separate revocation registry to verify an asset's status. This mandatory, cross-contract interaction effectively imposes a persistent gas overhead on all users, increasing transaction costs.
- Ecosystem compatibility: The specialized-contract method offers better compatibility. Because the revocation contract functions as an external module, it can interoperate with multiple NFT standards and support existing collections without code modification. This enables post-hoc integration of revocation capabilities into deployed ecosystems while preserving blockchain immutability and user trust. By contrast, the modified-contract method is suitable only for new NFT projects that can incorporate revocation logic during design time. It offers no practical upgrade path for legacy tokens due to the immutability of deployed smart contracts. Consequently, its applicability is limited in diverse NFT ecosystems.

The specific steps of _burn ()

In practical applications, *_burn* () achieves the goal of permanently invalidating the NFT logically and economically by clearing all ownership records and issuing a standardized public event. The specific implementation steps are as follows:

1. Use *_ownerOf* (*tokenId*) to confirm that the *tokenId* to be destroyed actually exists.
2. Find the current owner and decrease the owner's holdings by 1 in the internal balance ledger.
3. Clear the mapping relationship between *tokenId* and its owner, and set the address to address(0).
4. Use *emitTransfer* () to publish an announcement: This NFT has been permanently burned. The trading marketplaces and various wallet applications will capture this event. When they see that the to address is address(0), they will remove this NFT from the front-end interface or mark it as "burned".

Implementation considerations

Applying R-ECDSA to the current NFT ecosystem can enhance its unauthorized signature revocation capabilities. Signers can still prove that a signature was not generated by them even after their private key has been compromised, and the proof mechanism can be publicly verified. However, to optimize the utilization of R-ECDSA, the following two considerations need be carefully implemented in practical.

First, *AEK* must be properly protected. *AEK* is key to R-ECDSA as *AEK* serves as the exclusive credential for triggering the revocation procedure. Possession of *AEK* is therefore restricted to the true owner of *sk*. Given its critical role, *AEK* must be stored separately from *sk* and in a secure manner. One option is to embed it within the wallet's meta-info, which remains invisible to other blockchain participants. Alternatively, *AEK* can be kept in isolated offline storage, such as a hardware token, USB device, or NFC card, ensuring that the compromise of a single storage location does not simultaneously expose both *AEK* and *sk*. This separation of trust significantly reduces the risk of an attacker gaining full control over the revocation capability.

Second, check the marketplace frequently to quickly identify NFTs under the user's identity but not created by themselves. As one of the core characteristics of blockchain is its immutability. Once a transaction is included in a block and confirmed by the network, it becomes an immutable historical record. This implies that once a transaction is recorded on the blockchain, it cannot be undone, and any subsequent attempts to alter the confirmed transaction state are futile. Therefore, user should check the marketplace frequently, or set up some automatically notifying mechanism to send SMS/email whenever there are new NFTs listed in the marketplace under their identity.

Blockchain cost

To evaluate the performance and cost of R-ECDSA in a decentralized environment, we deploy our scheme in the Remix VM environment, including the complete workflow from minting a NFT token to revoking a valid but unauthorized NFT token. We test three times and record the average value. Gas consumption for each operation of a single NFT is summarized in Table 6, using the exchange rate 1 Ether = 4156.5 USD (as of August 9, 2025) and a gas price of 0.41 Gwei. Gas prices fluctuate based on network congestion. During periods of high congestion, gas prices tend to rise. To calculate the current gas cost, we use the medium-priority gas price in Gwei (the smallest unit of Ether) as reported by Etherscan. We then convert the gas cost from Gwei to Ether, and finally, multiply the amount of Ether by the current ETH price to calculate the cost in USD.

The operation of mint requires 125,512 gas, which is approximately 0.21 USD. This is a relatively gas-intensive operation, as minting NFTs usually involves more complex computational and storage operations, and therefore its cost is higher. The operation of transfer requires 62,887 gas, which is approximately 0.11 USD, and the operation of revoke requires 87,566 gas, which is approximately 0.15 USD. The operation of transfer has the lowest gas cost because it only involves updating the data and does not require complex computations or large-scale storage operations. The gas costs for revoke is intermediate, involving the mark and burn of valid but unauthorized NFTs. This operation is relatively less complex. These gas costs indicate that our implementation achieves a practical balance between security enhancement and economic efficiency in NFT system.

Table 7 illustrates the performance comparison between basic NFT transaction process and our scheme. Basic NFT transaction process only have the operations of mint and transfer, and the whole process doesn't involve *AEK*. The results indicate that our scheme is comparable with those in basic NFT transaction process. The gas cost of revoke operation only needs 0.15 USD.

The above data is a cost analysis of operations performed on one token. Figure 7 shows the comparison of the gas cost of each operation based on different number of tokens (from 1 to 100). Where x-axis represents

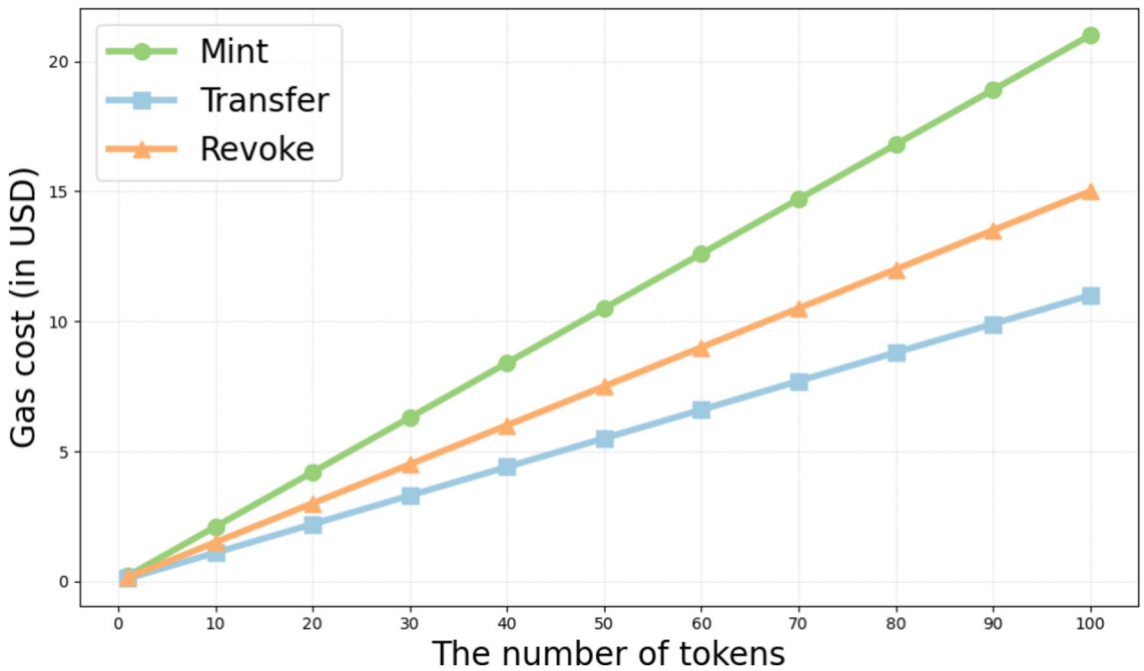


Fig. 7 Comparison of the gas cost of each operation based on different number of tokens

the number of tokens and y-axis represents the gas cost in USD. It is clear from the figure that gas costs increase linearly with the number of tokens.

Conclusion

In this paper, we addressed critical security challenges in the rapidly evolving NFT ecosystem and digital signature, with a particular focus on the risks associated with private key leakage. Given the potential for private key compromises to lead to the creation of valid but unauthorized NFTs, which can be accepted in the market, we proposed a novel revocable signature scheme aimed at enhancing the security resilience of NFT systems.

To demonstrate the effectiveness of R-ECDSA, we introduced a general system overview for implementing the scheme and designed a concrete ECDSA-based construction. The existential unforgeability of R-ECDSA is proved based on the assumption of ECDSA's EU-CMA. Additionally, we showed that the performance of R-ECDSA is comparable to traditional schemes, offering efficient performance without sacrificing security.

Beyond the technical details, R-ECDSA contributes to addressing the growing concerns around private key security in the context of NFTs, offering a practical and scalable solution for safeguarding digital assets. Moreover, R-ECDSA opens the door for future research into the broader application of revocable signatures in blockchain-based systems.

While R-ECDSA can effectively alleviate the consequences of private key leakage in NFT system, we acknowledge several limitations. First, the current design primarily operates before the phase of trading. Once an NFT transaction is finalized and revenue enters circulation, revocation becomes challenging due to the irreversible nature of blockchain-based asset transfers. However, this limitation stems from broader economic and regulatory complexities rather than cryptographic weaknesses. Second, our current scheme requires the AEK_e to be publicly revealed to the smart contract during revocation. This forces the user to execute a full key rotation (for sk_e, pk_e, and AEK_e) after every incident, imposing operational overhead. Finally, our security model fundamentally relies on the assumption that sk_e and AEK_e are not compromised simultaneously.

Future research aims to bridge the gap and lay the foundation for a more secure and resilient NFT ecosystem, where creators' assets and identities are better protected against the growing threat of private key leakage. A primary goal is to eliminate the public revelation of AEK_e. We plan to design a new mechanism using Zero-Knowledge Proof (ZKP), which would allow an owner to prove an unauthorized signature without revealing the AEK_e, thus removing the need for key rotation. Furthermore, future research will focus on extending R-ECDSA to address more complex and practical threat models. Specifically, we plan to investigate cross-contract replay attacks, in which revoked signatures might be maliciously reused across interoperable NFT marketplaces, and delayed revocation abuse, where attackers exploit the temporal gap between signature compromise detection and on-chain revocation execution. Furthermore, we aim to explore on-chain revocation oracles and incentive-compatible mechanisms that enable post-trade mitigation through collaboration between cryptographic design and blockchain governance. These directions are expected to bridge the gap between theoretical security guarantees and real-world operational resilience, paving the way toward a more comprehensive defense framework for NFT ecosystems.

Acknowledgements

This work is partially funded by the Suzhou Municipal Key Laboratory for Intelligent Virtual Engineering (SZS2022004). It is also supported by 2024 Suzhou Innovation Consortium Construction Project (LHT202406), Suzhou Data Innovation Application Laboratory and SIP High level innovation platform. This work is also supported by the Jiangsu Province Science and Technology Youth Talent Promotion Program under the Grant No. JSTJ-2025-144, and the XJTLU Research Development Fund under the Grant No. RDF-22-02-106. We thank Yuji Dong and Haotian Yin at XJTLU for discussing the attackers' motivations.

Authors' contributions

Yuxin Xia: Conceptualization, Investigation, Methodology, Software, Validation, Formal analysis, Writing-original draft. Ziyang Ji: Conceptualization, Investigation, Software, Validation, Formal analysis, Writing-review & editing. Jie Zhang: Conceptualization, Funding acquisition, Investigation, Methodology, Supervision,Writing - review & editing. Wanxin Li: Investigation, Supervision, Writing - review & editing. Ka Lok Man: Funding acquisition, Resources, Supervision, Writing - review & editing. Steven Guan: Resources, Supervision, Writing - review & editing. Dominik Wojtczak: Resources, Supervision, Writing - review & editing.

Declarations

Competing interest

We declare that we have no conflict of interest.

Received: 3 September 2025 Accepted: 18 December 2025
Published online: 09 January 2026

References

Beck G, Choudhuri AR, Green M, Jain A, Tiwari PR (2023) Time-deniable signatures. Proceedings on Privacy Enhancing Technologies

Bresson E, Stern J (2001) Efficient revocation in group signatures. In: Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography. PKC '01, pp. 190–206. Springer, Berlin, Heidelberg

Brown DRL (2005) Generic groups, collision resistance, and ecdsa. Designs Codes & Cryptography 35(1):119–152

Chen L, Moody D, Regenscheid A, Randall K (2019) Recommendations for discrete logarithm-based cryptography: Elliptic curve domain parameters. Technical report, National Institute of Standards and Technology

Davidson J (2022) Banksy's Website Hacked, Fan Scammed \$336,000 and Got Money Back

Desmedt YG (1994) Threshold cryptography. Eur Trans Telecommun 5(4):449–458

Deuber D, Magri B, Thyagarajan SAK (2019) Redactable blockchain in the permissionless setting. In: 2019 IEEE Symposium on Security and Privacy (SP), pp. 124–138 . IEEE
Escala A, Herranz J, Morillo P (2011) Revocable attribute-based signatures with adaptive security in the standard model. In: Nitaj A, Pointcheval D (eds) Progress in Cryptology - AFRICACRYPT 2011. Springer, Berlin, Heidelberg, pp 224–241
Everspaugh A, Paterson K, Ristenpart T, Scott S (2017) Key rotation for authenticated encryption. In: Katz J, Shacham H (eds) Advances in Cryptology - CRYPTO 2017. Springer, Cham, pp 98–129
Gao W, Chen L, Hu Y, Newton CJ, Wang B, Chen J (2019) Lattice-based deniable ring signatures. Int J Inf Secur 18:355–370
Groschopf W, Dobrovnik M, Herneth C (2021) Smart contracts for sustainable supply chain management: conceptual frameworks for supply chain maturity evaluation and smart contract sustainability assessment. Front Blockchain 4:506436
Guo Y, Lu Z, Ge H, Li J (2023) Revocable blockchain-aided attribute-based encryption with escrow-free in cloud storage. IEEE Trans Comput 72(7):1901–1912. https://doi.org/10.1109/TC.2023.3234210
Hammi B, Zeadally S, Perez AJ (2023) Non-fungible tokens: a review. IEEE Internet Things Magazine 6(1):46–50. https://doi.org/10.1109/IOTM.001.2200244
Hu Y, Yin D, Huang C (2020) Bbsf: A blockchain based secure framework for the internet of things with user revocation. In: 2020 IEEE International Conference on Progress in Informatics and Computing (PIC), pp. 358–362 . IEEE
Johnson D, Menezes A, Vanstone S (2001) The elliptic curve digital signature algorithm (ecdsa). Int J Inf Secur 1:36–63
Komano Y, Ohta K, Shimbo A, Kawamura S (2006) Toward the fair anonymous signatures: Deniable ring signatures. Cryptographers'Track at the RSA Conference. Springer, pp 174–191
Kshetri N (2022) Scams, frauds, and crimes in the nonfungible token market. Computer 55(4):60–64
Kubilay MY, Kiraz MS, Mantar HA (2019) Certledger: a new pki model with certificate transparency based on blockchain. Comput& Security 85:333–352
Kumar R, Tripathi R (2019) Implementation of distributed file storage and access framework using ipfs and blockchain. In: 2019 Fifth International Conference on Image Information Processing (ICIIP), pp. 246–251 . IEEE
Kwan J (2020) An Artist Died. Then Thieves Made NFTs of Her Work
Lesaege C, Ast F, George W (2019) Kleros Whitepaper. Whitepaper, Kleros . [Online]. https://kleros.io/static/whitepaper_en-8bd3a0480b45c39899787e17049ded26.pdf
Liao Z, Hao S, Nan Y, Zheng Z (2023) Smartstate: Detecting state-reverting vulnerabilities in smart contracts via fine-grained state-dependency analysis. In: Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis. ISSTA 2023, pp. 980–991. Association for Computing Machinery, New York, NY, USA . https://doi.org/10.1145/3597926.3598111
Liu DY, Liu JK, Mu Y, Susilo W, Wong DS (2007) Revocable ring signature. J Comput Sci Technol 22:785–794
Luu L, Chu DH, Olickel H, Saxena P, Hobor A (2016) Making smart contracts smarter. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, pp. 254–269
Peng C, Xu H (2022) Redactable blockchain with fine-grained autonomy and transaction rollback. In: Su C, Sakurai K, Liu F (eds) Sci Cyber Security. Springer, Cham, pp 68–84
Quarmby B (2021) Fidenza Artist Slams Knock-off NFT Project from 'honest Pirates' on Solana
Rajalakshmi A, Lakshmy K, Sindhu M, Amritha P (2018) A blockchain and IPFS based framework for secure research record keeping. Int J Pure Appl Math 119(15):1437–1442
Ren Y, Zhu F, Wang J, Sharma PK, Ghosh U (2022) Novel vote scheme for decision-making feedback based on blockchain in internet of vehicles. IEEE Trans Intell Transp Syst 23(2):1639–1648. https://doi.org/10.1109/TITS.2021.3100103
Silverman JH (1986) The arithmetic of elliptic curves. In: Graduate Texts in Mathematics . https://api.semanticscholar.org/CorpusID:117121125
Stephen B (2021) NFT Mania Is Here, and so Are the Scammers: Artists Are Seeing Their Work Showing up in NFTs They Did Not Mint Themselves
Sun Y, Zhang F, Shen L, Deng RH (2013) A revocable certificateless signature scheme. Cryptology ePrint Archive
Vijayakumar P, Bose S, Kannan A (2011) Rotation based secure multicast key management for batch rekeying operations. Networking Science
Wang Q, Li R, Wang Q, Chen S (2021) Non-fungible token (nft): Overview, evaluation, opportunities and challenges. arXiv preprint arXiv:2105.07447

Publisher's Note