# Eksperimen Signature Malleability pada ECDSA dalam Verifikasi Transaksi Blockchain dan Mitigasinya

Muhammad Aymar Barkhaya - 18223051
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail: 18223051@std.stei.itb.ac.id, aymarbarkhayaa@gmail.com

***Abstrak*—Elliptic Curve Digital Signature Algorithm (ECDSA) merupakan skema tanda tangan digital yang banyak digunakan pada sistem *blockchain* dan transaksi digital. Di balik adopsinya yang luas, ECDSA memiliki kerentanan melekat berupa *signature malleability*, yaitu kemampuan menghasilkan tanda tangan lain yang tetap valid atas pesan yang sama tanpa mengetahui *private key*. Makalah ini menganalisis mekanisme kerentanan tersebut, mendemonstrasikannya secara eksperimental, serta menguji efektivitas mitigasinya. Melalui implementasi pada kurva secp256k1, ditunjukkan bahwa sebuah tanda tangan valid (*r, s*) dapat diubah menjadi *(r, n−s)* yang tetap lolos verifikasi hanya dengan satu operasi modular, tanpa mengubah isi pesan. Simulasi sederhana memperlihatkan bahwa modifikasi ini berhasil mengubah pengenal transaksi. Pengujian mitigasi *low-S enforcement* membuktikan bahwa pembatasan tanda tangan pada satu representasi berhasil menggagalkan serangan ini. Hasil penelitian menegaskan bahwa *malleability* bersumber dari interaksi antara properti matematis ECDSA dan asumsi sistem di atasnya, sehingga penanganannya menuntut mitigasi pada lebih dari satu lapisan.**

***Kata kunci—ECDSA; signature malleability; blockchain; low-s enforcement; secp256k1***

## I. PENDAHULUAN

Tanda tangan digital merupakan mekanisme autentikasi pesan yang digunakan untuk membuktikan bahwa suatu pesan benar-benar berasal dari pengirim tertentu, mirip seperti tanda tangan pada dokumen kertas [1]. Mekanisme ini menawarkan jaminan keaslian identitas pengirim, integritas dokumen, anti penyangkalan, serta memiliki kekuatan hukum penuh yang sah diakui oleh peraturan perundang-undangan (UU ITE). Pada era di mana informasi digital dapat dengan mudah disimpan, diubah, dan dikirimkan, keamanan informasi menjadi sebuah isu yang nyata. Transaksi informasi digital tidak dapat menjamin privasi, autentikasi, dan integritas data tanpa prosedur keamanan yang memadai [2]. Maka dari itu, dalam berbagai ekosistem teknologi modern, tanda tangan digital telah menjadi salah satu mekanisme penting yang mendukung keamanan sistem.

Selama perkembangan teknologi kriptografi kunci asimetris, telah muncul berbagai algoritma tanda tangan digital, salah satu contoh yang paling banyak digunakan di industri adalah skema *Elliptic Curve Digital Signature Algorithm* (ECDSA). Algoritma ini dikembangkan dari standar *Digital Signature Algorithm* (DSA) buatan NIST yang diimplementasikan pada *Elliptic Curve Cryptography* (ECC) [3]. ECDSA sudah distandarisasi dan diterima oleh berbagai lembaga standar seperti ISO, ANSI, NIST, dan IEEE. Adopsi ECDSA di industri sangatlah luas, termasuk di dalamnya yang menggunakan ECDSA secara ekstensif adalah industri *blockchain* dan transaksi digital seperti *cryptocurrency*, *smart contract & decentralized finance (DeFi)*.

Industri *blockchain* dan transaksi digital secara langsung melibatkan penanganan aset finansial dan berperan sebagai salah satu infrastruktur perekonomian digital. Berbeda dengan sistem informasi pada umumnya, kegagalan keamanan pada sistem industri ini dapat langsung menimbulkan kerugian finansial yang nyata dan bersifat permanen karena karakteristik transaksi *blockchain* yang umumnya *irreversible*. Hal tersebut menuntut keamanan maksimal dari kerja sistem. Namun, dibalik standar ECDSA, terdapat suatu kerentanan yang melekat akibat properti matematisnya, yaitu *Signature Malleability. Signature Malleability* adalah kerentanan yang menyebabkan penyerang dapat memodifikasi suatu tanda tangan digital menjadi tanda tangan lain yang tetap valid untuk sebuah pesan tanpa harus mengetahui *private key* tanda tangan aslinya [4].

Pada Februari 2014, Mt. Gox, salah satu platform *Bitcoin exchange* terbesar yang memegang 70% keseluruhan transaksi Bitcoin, mengajukan pailit akibat kerugian dengan estimasi melebihi 450 juta dolar US [5]. Mt. Gox mengklaim kerugian ini disebabkan eksploitasi kerentanan *Signature Malleability* yang berhasil menghilangkan sekitar 850.000 bitcoins dari *wallet*-nya. Meski terdapat studi lanjutan yang mencoba membantah klaim tersebut [6], *malleability* terbukti tetap merupakan ancaman kerentanan yang nyata hingga mendorong perubahan protokol Bitcoin (BIP-62, BIP-146) dan Ethereum (EIP-2). Walaupun saat ini sudah terdapat langkah mitigasi dalam menghadapinya, tetap diperlukan pemahaman mendalam dan dokumentasi eksperimental tentang mekanisme serangan dan efektivitas mitigasinya.

Makalah ini bertujuan menganalisis kerentanan *Signature Malleability* dalam konteks verifikasi transaksi *blockchain*, mendemonstrasikan serangan, menguji metode mitigasi, dan memverifikasi efektivitas metode tersebut serta relevansi kerentanan ini di sistem modern. Kontribusi utama makalah ini adalah implementasi dan demonstrasi eksperimental serangan signature malleability pada ECDSA beserta pengujian efektivitas mitigasi secara langsung. Bagian 2 akan menjelaskan dasar teori dan analisis kerentanan. Bagian 3

menguraikan langkah implementasi dan eksperimen. Bagian 4 membahas hasil eksperimen. Ditutup dengan kesimpulan pada Bagian 5.

## II. Dasar teori dan analisis

### A. Elliptic Curve Cryptography (ECC)

ECC merupakan pendekatan kriptografi kunci publik yang dibangun di atas struktur aljabar titik-titik pada sebuah kurva eliptik di atas lapangan hingga (*finite field*). Keamanan ECC bersandar pada kesulitan memecahkan Elliptic Curve Discrete Logarithm Problem (ECDLP). Diberikan sebuah titik basis $G$ dan titik publik $Q = dG$, persoalan ECDLP adalah menemukan skalar $d$ apabila hanya diketahui $G$ dan $Q$. Selama tidak ada algoritma efisien untuk menyelesaikan persoalan ini pada kurva yang dipilih dengan baik, maka private key $d$ tetap aman. Karakteristik inilah yang membuat ECC mampu menawarkan tingkat keamanan setara dengan algoritma kunci publik lain seperti RSA, namun dengan ukuran kunci yang jauh lebih kecil sehingga lebih efisien secara komputasi maupun penyimpanan [7].

Dalam praktiknya, sebuah sistem ECC beroperasi di atas sekumpulan parameter domain yang sudah ditetapkan, yaitu ($p, a, b, G, n, h$), dengan $n$ adalah orde dari titik basis $G$ dan $h$ adalah *cofactor*. Salah satu kurva standar yang banyak digunakan pada ekosistem blockchain adalah secp256k1, kurva yang didefinisikan oleh Standards for Efficient Cryptography Group (SECG) [8]. Seluruh parameter secp256k1 telah ditetapkan secara baku, sehingga setiap implementasi yang menggunakannya beroperasi di atas kurva yang persis sama.

### B. Elliptic Curve Digital Signature Algorithm (ECDSA)

ECDSA adalah analog kurva eliptik dari Digital Signature Algorithm (DSA) yang distandarkan dalam Digital Signature Standard (DSS) oleh NIST [3]. Skema ini terdiri atas tiga prosedur utama, yaitu pembangkitan kunci (*key generation*), penandatanganan (*signing*), dan verifikasi (*verification*). Seluruh operasi aritmetika dilakukan dalam modulo n, orde dari titik basis G. [9]

**Pembangkitan kunci.** Penanda tangan memilih sebuah bilangan acak $d$ pada rentang $[1, n-1]$ sebagai *private key*, lalu menghitung titik publik $Q = dG$ sebagai *public key*. Karena memperoleh $d$ dari $Q$ memerlukan pemecahan ECDLP, *public key* dapat disebarkan secara bebas tanpa membahayakan *private key*.

**Penandatanganan.** Untuk menandatangani sebuah pesan m, penanda tangan melakukan langkah-langkah berikut:

1. Hitung nilai hash pesan, $z = H(m)$, dengan $H$ adalah fungsi hash kriptografi seperti SHA-256.
2. Pilih sebuah *nonce* acak $k$ pada rentang $[1, n-1]$.
3. Hitung titik $kG = (x_1, y_1)$, lalu tetapkan $r = x_1 \bmod n$. Jika $r = 0$, ulangi dari langkah 2.
4. Hitung $s = k^{-1}(z + rd) \bmod n$. Jika $s = 0$, ulangi dari langkah 2.
5. Tanda tangan untuk pesan m adalah pasangan bilangan (r, s).

**Verifikasi.** Penerima yang memegang public key $Q$ memverifikasi tanda tangan $(r, s)$ atas pesan $m$ dengan langkah-langkah berikut:

1. Pastikan $r$ dan $s$ berada pada rentang $[1, n-1]$; jika tidak, tanda tangan ditolak.
2. Hitung $z = H(m)$ menggunakan fungsi hash yang sama.
3. Hitung $w = s^{-1} \bmod n$.
4. Hitung $u_1 = zw \bmod n$ dan $u_2 = rw \bmod n$.
5. Hitung titik $(x_2, y_2) = u_1G + u_2Q$.
6. Tanda tangan dinyatakan valid jika dan hanya jika $r \equiv x_2 \pmod n$.

### C. Aritmatika Modular dan Invers Aditif

Seluruh komponen tanda tangan ECDSA hidup di dalam aritmetika modulo $n$. Dalam struktur ini, setiap elemen $s$ memiliki sebuah invers aditif, yaitu nilai unik $-s \bmod n$ yang setara dengan $n - s$, sehingga berlaku $s + (n - s) \equiv 0 \pmod n$. Properti ini merupakan sifat dasar yang melekat dan bukan suatu kelemahan implementasi tertentu.

Sifat ini menjadi penting karena, sebagaimana akan ditunjukkan, persamaan verifikasi ECDSA ternyata bersifat simetris terhadap $s$ dan $n - s$. Akibatnya, sebuah nilai $s$ yang valid dapat digantikan oleh invers aditifnya tanpa membuat tanda tangan menjadi tidak valid. Interaksi antara persamaan verifikasi dan properti invers aditif inilah yang menjadi akar dari kerentanan *signature malleability* pada ECDSA.

### D. Signature Malleability

*Signature malleability* adalah kondisi di mana dari sebuah tanda tangan valid $\sigma$ atas pesan $m$, seseorang dapat menghasilkan tanda tangan lain $\sigma' \neq \sigma$ yang juga tetap valid atas pesan $m$ yang sama, tanpa perlu mengetahui *private key* penanda tangan [4]. Perlu ditegaskan bahwa *malleability* berbeda dari pemalsuan tanda tangan. Pada pemalsuan, penyerang berupaya menghasilkan tanda tangan valid untuk pesan baru yang belum pernah ditandatangani. Sebaliknya, pada *malleability*, pesan sama sekali tidak berubah, yang berubah hanyalah representasi tanda tangannya. Karena pesan dan validitasnya tetap terjaga, *malleability* terkadang keliru dianggap tidak berbahaya. Namun, dampaknya menjadi nyata pada sistem yang memperlakukan tanda tangan sebagai pengidentifikasi unik dalam logika keamanannya.

Mekanisme malleability pada ECDSA dapat diturunkan langsung dari persamaan verifikasi. Misalkan $(r, s)$ adalah tanda tangan valid atas pesan $m$. Tinjau tanda tangan alternatif $(r, s')$ dengan $s' = n - s$, yakni invers aditif dari $s$ dalam modulo $n$. Pada proses verifikasi, nilai $w' = (s')^{-1} = (-s)^{-1} = -(s^{-1}) \bmod n$. Akibatnya, kedua skalar verifikasi ikut berubah tanda, yaitu $u_1' = -u_1$ dan $u_2' = -u_2 \pmod n$. Titik yang dihasilkan menjadi $u_1'G + u_2'Q = -(u_1G + u_2Q) = -(x_2, y_2)$. Pada kurva eliptik, negasi sebuah titik hanya mengubah tanda koordinat-y, sementara koordinat-x tetap sama, sehingga $-(x_2,$

$y_2$) = ($x_2$, $-y_2$). Karena verifikasi ECDSA hanya membandingkan koordinat-x dengan $r$, maka pemeriksaan $r \equiv x_2 \pmod{n}$ tetap terpenuhi. Dengan demikian, $(r, n - s)$ merupakan tanda tangan yang sama-sama valid atas pesan $m$.

Konsekuensinya, dari setiap tanda tangan ECDSA terdapat tepat dua nilai $s$ yang valid, yaitu $s$ dan $n - s$. Seorang penyerang dapat mengubah satu menjadi yang lain hanya dengan satu operasi pengurangan, tanpa mengakses *private key* dan tanpa menyentuh isi pesan.

### E. *Implikasi pada Sistem dengan Tanda Tangan sebagai Identifier*

Selama sebuah sistem hanya memeriksa validitas tanda tangan, *malleability* tidak menimbulkan kerugian karena kedua bentuk tanda tangan memang sama-sama sah. Persoalan muncul ketika sistem menggunakan data tanda tangan sebagai *identifier*. Dalam kondisi ini, mengubah $s$ menjadi $n - s$ akan mengubah *identifier*, padahal transaksi atau aksi yang diwakilinya secara substansi tidak berubah.

Skenario ini tepat menggambarkan apa yang terjadi pada sistem *blockchain*, di mana *identifier* transaksi diturunkan dengan melakukan *hashing* terhadap keseluruhan data transaksi, termasuk tanda tangannya. Karena tanda tangan ikut menjadi masukan fungsi *hash*, maka satu transaksi yang identik dapat memiliki dua pengenal yang berbeda apabila tanda tangannya dimodifikasi.

## III. IMPLEMENTASI DAN EKSPERIMEN

### A. *Lingkungan dan Rancangan Eksperimen*

Eksperimen ini dirancang untuk mendemonstrasikan tiga hal secara berurutan, yaitu (1) keberhasilan serangan *signature malleability* dalam menghasilkan tanda tangan kedua yang tetap valid, (2) bahwa serangan tersebut tidak mengubah pesan sehingga berbeda dari pemalsuan, serta (3) efektivitas mitigasi *low-S enforcement* dalam menggagalkan serangan. Untuk menjelaskan hubungannya dalam konteks transaksi *blockchain*, ditambahkan pula simulasi sederhana perubahan *Transaction ID* (TXID) akibat *malleability*.

Implementasi ditulis dalam bahasa Python dengan memanfaatkan pustaka *ecdsa* yang beroperasi pada kurva secp256k1, kurva yang digunakan Bitcoin. Pustaka *ecdsa* dipilih karena menyediakan akses langsung terhadap komponen tanda tangan $r$ dan $s$ dalam bentuk bilangan bulat, sehingga manipulasi nilai $s$ dapat dilakukan secara eksplisit Sebagai pembanding, pustaka *cryptography* sempat dipertimbangkan karena lebih umum dipakai pada lingkungan produksi. Namun pustaka tersebut menyajikan tanda tangan dalam format terenkode, sehingga setiap manipulasi terhadap s menuntut langkah pembongkaran yang menambah kerumitan demonstrasi tanpa memberi manfaat lebih.

```
def sign_message(sk, message: bytes):
    sig = sk.sign(message,
hashfunc=hashlib.sha256,
sigencode=util.sigencode_strings)
    r = int.from_bytes(sig[0], "big")
    s = int.from_bytes(sig[1], "big")
    return r, s

def verify_rs(vk, message: bytes, r: int, s:
int) -> bool:
    try:
        sig = util.sigencode_strings(r, s, n)
        return vk.verify(sig, message,
hashfunc=hashlib.sha256,

 sigdecode=util.sigdecode_strings)
    except Exception:
        return False
```

Listing 1 Fungsi helper

Perlu diperhatikan juga bahwa simulasi TXID pada eksperimen ini merupakan model sederhana dan bukan replikasi dari mekanisme transaksi Bitcoin yang sebenarnya. Tujuannya hanya menunjukkan bagaimana tanda tangan yang dimasukkan ke fungsi *hash* dapat menyebabkan perubahan *identifier* transaksi dan bukan untuk meniru keseluruhan proses validasi transaksi pada jaringan asli Bitcoin.

### B. *Demonstrasi Serangan: Konstruksi (r, n-s)*

```
sk = SigningKey.generate(curve=SECP256k1)
vk = sk.get_verifying_key()
message = b"Transfer 10 BTC to Alice"

r, s = sign_message(sk, message)
```

Listing 2 Pembangkitan pasangan kunci, pesan, dan tanda tangan asli

Tahap pertama adalah membangkitkan sepasang kunci pada kurva secp256k1, lalu menandatangani sebuah pesan uji, dalam hal ini pesan berupa string "Transfer 10 BTC to Alice". Proses penandatanganan menghasilkan tanda tangan asli berupa pasangan $(r, s)$ yang dikonfirmasi valid oleh prosedur verifikasi standar.

```
s_mall = n - s
valid = verify_rs(vk, message, r, s)
valid_mall = verify_rs(vk, message, r,
s_mall)
```

Listing 3 Konstruksi tanda tangan *malleable* dan validasi

| Tanda tangan | r | s | Pesan | Valid? |
|---|---|---|---|---|
| Asli (r, s) | 0xbfea...a458 | 0x86f8...1988 | "Transfer 10 BTC to Alice" | True |
| Malleable (r, n – s) | 0xbfea...a458 | 0x9707...27b9 | "Transfer 10 BTC to Alice" | True |

Table 1 Perbandingan tanda tangan asli dan *malleable*

Serangan kemudian dilakukan dengan menghitung tanda tangan alternatif $(r, s')$, dengan $s' = n - s$, tanpa mengakses *private key* sama sekali. Hasil menunjukkan bahwa nilai $s'$ yang dihasilkan berbeda dari $s$ asli, namun pasangan (r, s′) tetap dinyatakan valid oleh prosedur verifikasi yang sama. Temuan ini mengonfirmasi bahwa invers aditif dari $s$ dalam

modulo $n$ menghasilkan tanda tangan kedua yang sama sahnya. Dengan demikian, satu operasi pengurangan sederhana cukup untuk menghasilkan tanda tangan berbeda yang lolos verifikasi.

### C. Verifikasi bahwa Pesan Tidak Berubah

Pengujian berikutnya memverifikasi kedua tanda tangan, yaitu $(r, s)$ dan $(r, n-s)$, terhadap pesan yang sama persis. Seperti dapat dilihat pada Table 1, hasil eksperimen menunjukkan bahwa keduanya valid untuk pesan "Transfer 10 BTC to Alice" tanpa ada perubahan apa pun pada isi pesan. Hal ini membuktikan bahwa penyerang tidak memalsukan pesan baru, yang berubah hanyalah representasi tanda tangannya. Perbedaan ini penting karena menjelaskan mengapa *malleability* sering dianggap tidak berbahaya pada awalnya, padahal dampak yang diberikan cukup berbahaya pada lapisan sistem yang menggunakan tanda tangan sebagai bagian dari logikanya.

### D. Simulasi Perubahan Transaction ID

```
def simulate_txid(message: bytes, r: int, s:
int) -> str:
    payload = message + r.to_bytes(32, "big")
+ s.to_bytes(32, "big")
    return
hashlib.sha256(hashlib.sha256(payload).digest
()).hexdigest()

txid_orig = simulate_txid(message, r, s)
txid_mall = simulate_txid(message, r, s_mall)
```

Listing 4 Simulasi penurunan TXID dari pesan dan tanda tangan

| Tanda tangan | TXID |
|---|---|
| Asli (r, s) | 6998...19ef |
| Malleable (r, n – s) | 9783...bf2d |

Table 2 Hasil simulasi Transaction ID

Tahap ini memodelkan cara Bitcoin menurunkan TXID, yaitu dengan melakukan *double hashing* SHA-256 terhadap data transaksi yang telah digabung dengan komponen tanda tangan. Pada eksperimen, fungsi simulasi TXID didefinisikan sebagai SHA-256(SHA-256(pesan | $r$ | $s$)). Dengan menggunakan tanda tangan asli $(r, s)$ dan tanda tangan hasil *malleability* $(r, n-s)$ sebagai input, dihasilkan dua nilai TXID yang sepenuhnya berbeda. Hasil ini menunjukkan bahwa sebuah transaksi yang identik isinya dapat memiliki dua *identifier* berbeda hanya karena tanda tangannya dimodifikasi. Pada sistem yang mengandalkan TXID sebagai acuan status transaksi, perbedaan pengenal inilah yang membuka peluang penyalahgunaan.

### E. Implementasi dan Pengujian Mitigasi Low-S Enforcement

Mitigasi *low-S enforcement* diterapkan dengan cara menambahkan satu aturan pada prosedur verifikasi, yaitu menolak setiap tanda tangan yang nilai $s$-nya melebihi $n/2$. Aturan ini didasari kenyataan bahwa antara $s$ dan $n - s$, hanya satu di antaranya bernilai tidak lebih dari $n/2$. Representasi inilah yang ditetapkan sebagai bentuk yang sah, sehingga sistem hanya menerima satu wujud tanda tangan untuk setiap penandatanganan.

```
def verify_low_s(vk, message: bytes, r: int,
s: int) -> bool:
    if s > n // 2:
        return False  # tolak high-S
    return verify_rs(vk, message, r, s)
s_low = s if s <= n // 2 else n - s
s_high = n - s_low
```

Listing 5 Implementasi verifier dengan aturan *low-S enforcement*

| Representasi | $S \leq n/2$? | Standard Verified? | Low-s Verified? |
|---|---|---|---|
| Low-s (asli) | True | True | True |
| High-s (malleable) | False | True | False |

Table 3 Hasil Pengujian Verifier Standar vs Verifier *Low-s Enforcement*

Pada pengujian, sebuah tanda tangan dinormalisasi menjadi bentuk *low-S* ($s \leq n/2$) dan bentuk *high-S* ($s > n/2$) sebagai pasangan *malleable*-nya. *Verifier* standar yang hanya memeriksa validitas matematis menerima kedua bentuk tersebut sebagai tanda tangan yang sah. Sebaliknya, *verifier* dengan *low-S enforcement* hanya menerima bentuk *low-S*. Hasil ini menunjukkan bahwa meskipun kedua tanda tangan tetap valid secara matematis, penerapan aturan *low-S* berhasil membatasi tanda tangan yang diterima hanya pada satu representasi. Dengan demikian, serangan berbasis perubahan $s$ menjadi $n - s$ tidak lagi menghasilkan tanda tangan yang dapat diterima sistem.

## IV. PEMBAHASAN

### A. Analisis Hasil Eksperimen

Rangkaian eksperimen pada bagian sebelumnya menunjukkan bahwa *signature malleability* pada ECDSA merupakan kerentanan yang dapat direproduksi dengan langkah sederhana. Penyerang tidak memerlukan akses terhadap *private key*, tidak perlu memecahkan ECDLP, dan tidak menyentuh isi pesan. Cukup dengan operasi pengurangan $n - s$, sebuah tanda tangan kedua yang sah dapat dihasilkan. Hal ini menunjukkan bahwa biaya serangan nyaris nol, sementara prasyaratnya hanya berupa akses terhadap tanda tangan yang memang bersifat publik pada sebagian besar sistem.

Temuan ini sekaligus memunculkan persoalan yang sebenarnya. Validitas kedua tanda tangan bukanlah kecacatan penuh dari keamanan ECDSA, sebab integritas atau ketahanan dari pemalsuan tetap terjaga. Persoalan baru muncul dari asumsi sistem bahwa setiap tanda tangan memiliki representasi yang tunggal dan tetap. Eksperimen simulasi TXID memperlihatkan konsekuensi langsung dari kerusakan atas asumsi tersebut. Dengan kata lain, kerentanan ini lahir akibat ketidaksesuaian antara properti matematis ECDSA dengan sistem yang dibangun di atasnya.

### B. Efektivitas dan Keterbatasan Low-s

Hasil pengujian menunjukkan bahwa *low-S enforcement* efektif menutup bentuk *malleability* yang paling mendasar. Dengan menetapkan bentuk $s \leq n/2$ sebagai satu-satunya representasi tanda tangan yang sah, sistem berhasil mencegah eksploitasi. Penyerang yang mencoba menggunakan bentuk *high-S* akan langsung ditolak sehingga manipulasi $s$ menjadi $n - s$ tidak lagi bekerja. Keunggulan pendekatan ini terletak pada kesederhanaannya yang hanya menambahkan satu pemeriksaan ringan tanpa mengubah algoritma inti dari ECDSA.

Meski begitu, perlu dipahami bahwa *low-S enforcement* hanya mengatasi satu sumber *malleability* yang spesifik, yaitu

yang bersumber dari invers aditif nilai *s*. Pada konteks Bitcoin, *malleability* sesungguhnya memiliki beberapa sumber lain di luar komponen tanda tangan, misalnya yang berasal dari fleksibilitas pengodean *script* maupun representasi data lain dalam transaksi. Dengan demikian, *low-S enforcement* merupakan langkah yang penting namun tidak dapat berdiri sendiri. Ia perlu dipandang sebagai salah satu komponen dari rangkaian mitigasi yang lebih menyeluruh, bukan sebagai solusi tunggal yang menuntaskan seluruh varian *malleability*.

### C. Mitigasi pada Tingkat Desain Sistem

Selain *low-s enforcement* yang mengatasi *malleability* di tingkat tanda tangan, persoalan ini juga dapat ditekan pada tingkat desain sistem. Sebagaimana ditunjukkan eksperimen TXID, *malleability* hanya berdampak ketika sistem menjadikan bentuk tanda tangan sebagai bagian dari *identifier* transaksi. Dari sudut pandang ini, salah satu mitigasi yang paling mudah adalah merancang sistem agar tidak menurunkan *identifier* dari data tanda tangan, sehingga modifikasi pada *s* tidak lagi berpengaruh.

Pendekatan inilah yang ditempuh Bitcoin melalui Segregated Witness (SegWit). Dengan memisahkan data tanda tangan (*witness*) dari data inti transaksi yang digunakan untuk menghitung TXID, perubahan pada tanda tangan tidak lagi mengubah *identifier* transaksi. Pendekatan low-S enforcement dan pemisahan pengenal ini bersifat saling melengkapi. Kombinasi keduanya menunjukkan bahwa penanganan kerentanan kriptografis yang efektif terkadang menuntut intervensi hingga ke lapisan sistem, tidak cukup hanya pada lapisan algoritma.

### D. Relevansi Malleability di Sistem Modern

Pada sistem modern, sebagian besar varian *malleability* sudah ditangani pada tingkat protokol. Bitcoin mewajibkan bentuk *low-S* sebagai aturan baku dan memperkenalkan SegWit, sementara Ethereum menetapkan keharusan *low-S* pada pemrosesan tanda tangan melalui EIP-2. Dengan adanya langkah-langkah tersebut, serangan *malleability* dalam bentuk paling dasar kini sulit dilakukan terhadap implementasi terbaru.

Meskipun demikian, relevansi pemahaman atas *malleability* tidak serta-merta menjadi usang. Kerentanan ini tetap mengintai pada sistem yang dibangun di atas ECDSA namun tidak menerapkan aturan-aturan keamanan terbaru secara konsisten. Selain itu, kasus *malleability* menyimpan pelajaran yang lebih luas yaitu bahwa sebuah properti matematis yang tampak tidak berbahaya dapat berubah menjadi kerentanan serius ketika berinteraksi dengan asumsi yang keliru pada lapisan sistem. Pemahaman atas mekanisme seperti ini menjadi bekal penting dalam merancang sistem berbasis kriptografi yang aman.

## V. KESIMPULAN

Makalah ini telah menganalisis dan mendemonstrasikan kerentanan *signature malleability* pada ECDSA dalam konteks verifikasi transaksi blockchain. Dari sisi mekanisme, ditunjukkan baik secara matematis maupun eksperimental bahwa kerentanan ini bersumber dari properti invers aditif pada aritmetika modular, yang menyebabkan pasangan (*r, s*) dan (*r, n−s*) sama-sama valid atas pesan yang sama. Serangan terbukti dapat dilakukan dengan biaya nyaris nol, tanpa akses terhadap *private key*, dan tanpa mengubah isi pesan, sehingga membedakannya dari pemalsuan tanda tangan.

Melalui simulasi, diperlihatkan pula bagaimana *malleability* dapat mengubah pengenal transaksi pada sistem yang menurunkan pengenal dari data yang memuat tanda tangan. Pengujian terhadap mitigasi *low-S enforcement* menunjukkan bahwa pembatasan tanda tangan pada satu representasi efektif menggagalkan serangan dalam bentuk dasarnya, meskipun mitigasi ini perlu dilengkapi dengan pendekatan pada tingkat desain sistem, seperti pemisahan tanda tangan dari pengenal transaksi yang ditempuh melalui SegWit.

Secara keseluruhan, temuan ini menegaskan bahwa *malleability* bukanlah cacat pada ketahanan kriptografis ECDSA, melainkan akibat ketidaksesuaian antara sifat matematis algoritma dan asumsi milik sistem di atasnya. Penanganan yang efektif menuntut intervensi pada lebih dari satu lapisan, mulai dari lapisan algoritma hingga lapisan desain sistem. Untuk pengembangan selanjutnya, eksperimen ini dapat diperluas dengan meninjau sumber *malleability* lain di luar komponen tanda tangan, serta menguji kerentanan pada implementasi lain yang belum dilindungi prosedur keamanan memadai.

## PRANALA VIDEO YOUTUBE

https://youtu.be/ZnsROODZ3nQ


## UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Bapak Rinaldi Munir selaku dosen pengampu mata kuliah II4021 Kriptografi atas bimbingan, ilmu, dan arahan yang diberikan selama perkuliahan. Ucapan terima kasih juga disampaikan kepada orang tua juga keluarga penulis atas dukungan dan doa yang senantiasa diberikan, serta kepada teman-teman penulis yang telah menjadi rekan diskusi selama proses pembelajaran dan penulisan makalah ini.


## REFERENCES


[1] CGI Group Inc., "Public Key Encryption and Digital Signature: How do they work?", 2004.

[2] R. Kaur and A. Kaur, "Digital Signature," in *2012 International Conference on Computing Sciences*, Phagwara, India: IEEE, Sep. 2012, pp. 295–301. doi: 10.1109/ICCS.2012.25.

[3] National Institute of Standards and Technology (US), "Digital Signature Standard (DSS)," National Institute of Standards and Technology (U.S.), Washington, D.C., NIST FIPS 186-5, Feb. 2023. doi: 10.6028/NIST.FIPS.186-5.

[4] "SCWE-054: Signature Malleability - OWASP Smart Contract Security." Accessed: Jun. 18, 2026. [Online]. Available: https://scs.owasp.org/SCWE/SCSVS-CRYPTO/SCWE-054/

[5] U. Rajput, F. Abbas, R. Hussain, H. Eun, and H. Oh, "A Simple Yet Efficient Approach to Combat Transaction Malleability in Bitcoin," in *Information Security Applications*, vol. 8909, K.-H. Rhee and J. H. Yi, Eds., in Lecture Notes in Computer Science, vol. 8909. , Cham: Springer International Publishing, 2015, pp. 27–37. doi: 10.1007/978-3-319-15087-1_3.

[6] C. Decker and R. Wattenhofer, "Bitcoin Transaction Malleability and MtGox," vol. 8713, 2014, pp. 313–326. doi: 10.1007/978-3-319-11212-1_18.

[7] D. Mahto and D. K. Yadav, “RSA and ECC: A Comparative Analysis,” vol. 12, no. 19, 2017.

[8] D. R. L. Brown, “SEC 2: Recommended Elliptic Curve Domain Parameters”.

[9] D. Johnson, A. Menezes, and S. Vanstone, “The Elliptic Curve Digital Signature Algorithm (ECDSA),” *IJIS*, vol. 1, no. 1, pp. 36–63, Aug. 2001, doi: 10.1007/s102070100002.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 18 Juni 2026

Muhammad Aymar Barkhaya - 18223051