Bit-Flipping Attack Exploration and Countermeasure in 5G Network

Joon Kim
Department of EECS
UC Berkeley
Berkeley, CA
joonkim1@berkeley.edu

Chengwei Duan
Department of ECE
University of Florida
Gainesville, FL
duan.c@ufl.edu

Sandip Ray
Department of ECE
University of Florida
Gainesville, FL
sandip@ece.ufl.edu

Abstract—5G communication technology has become a vital component in a wide range of applications due to its unique advantages such as high data rate and low latency. While much of the existing research has focused on optimizing its efficiency and performance, security considerations have not received comparable attention, potentially leaving critical vulnerabilities unexplored. In this work, we investigate the vulnerability of 5G systems to bit-flipping attacks, which is an integrity attack where an adversary intercepts 5G network traffic and modifies specific fields of an encrypted message without decryption, thus mutating the message while remaining valid to the receiver. Notably, these attacks do not require the attacker to know the plaintext, and only the semantic meaning or position of certain fields would be enough to effect targeted modifications. We conduct our analysis on OpenAirInterface (OAI), an open-source 5G platform that follows the 3GPP Technical Specifications, to rigorously test the real-world feasibility and impact of bit-flipping attacks under current 5G encryption mechanisms. Finally, we propose a keystream-based shuffling defense mechanism to mitigate the effect of such attacks by raising the difficulty of manipulating specific encrypted fields, while introducing no additional communication overhead compared to the NAS Integrity Algorithm (NIA) in 5G. Our findings reveal that enhancements to 5G security are needed to better protect against attacks that alter data during transmission at the network level.

I. INTRODUCTION

The deployment of the fifth-generation (5G) wireless network marks a transformative leap in communication technology, offering high data rates, ultra-reliable low-latency communications, and massive connectivity. These advancements make 5G a foundational technology for a wide range of emerging applications for Internet of Things (IoT), smart city, industrial automation, etc. [1] [2] Notably, 5G is expected to play a crucial role in enabling connected and autonomous vehicle (CAV) applications [3], where its capabilities can support real-time vehicular communication and safe automated driving operations.

Research on 5G has emphasized improving communication efficiency. However, studies on its security aspects remain comparatively limited. At the network level of 5G (defined as layers above the physical layer but below the application layer) where the data is processed as a digital bitstream, two main security mechanisms are used after authentication: NAS Encryption Algorithm (NEA) for confidentiality and NAS Integrity Algorithm (NIA) for integrity. Although NEA employs strong encryption algorithms and generates a fresh keystream for every packet, it operates via a simple XOR with the plaintext therefore introducing vulnerabilities. Theoretically, a bit-flipping attack, where an adversary can alter specific bits in the ciphertext, can exploit this vulnerability to produce predictable mutations to the plaintext without decryption. NIA integrity checks are effective at detecting such tampering, but it is optional for user-plane data as specified in 3GPP Technical Specifications [4]. This is because NIA introduces additional overhead by appending a 32-bit message authentication code to each packet, increasing both bandwidth usage and processing demands on the transmission entities. Moreover, not all 5G applications require perfect message fidelity. In some cases, occasional isolated mutations have negligible impact. For instance, research shows that in Cooperative Adaptive Cruise Control (CACC) applications, which can be operated under 5G, isolated and discrete mutation attacks on the transmitted acceleration values produce little negative effect compared to the benign, unattacked case [5]. This suggests that for certain 5G applications, mandatory integrity protection with NIA may not be necessary to address bit-flipping attacks.

In this work, we empirically test whether bit-flipping attacks under encryption can truly mutate transmitted values as expected. To achieve this, we avoid building a simplified or toy implementation of 5G network. Instead, we conduct our experiments using OpenAirInterface (OAI), which is a widely used open-source software platform that implements the 5G new radio cellular network closely following the 3GPP Technical Specifications. Additionally, we make use of the one-time pad nature of NEA keystreams to design a keystream-based shuffling algorithm, which significantly reduces the likelihood that a bit-flipping attack could successfully mutate a message while keeping it valid with no additional communication overhead.

II. BACKGROUND AND RELATED WORK

A. Fundamental of User Data Transmission in 5G

Currently, most 5G applications still use IP-based communication for compatibility with existing Internet infrastructure and widely used protocols. The overall process is illustrated in Fig. 1. When an application needs to transmit some data to others, such as a numeric value, it first converts the data
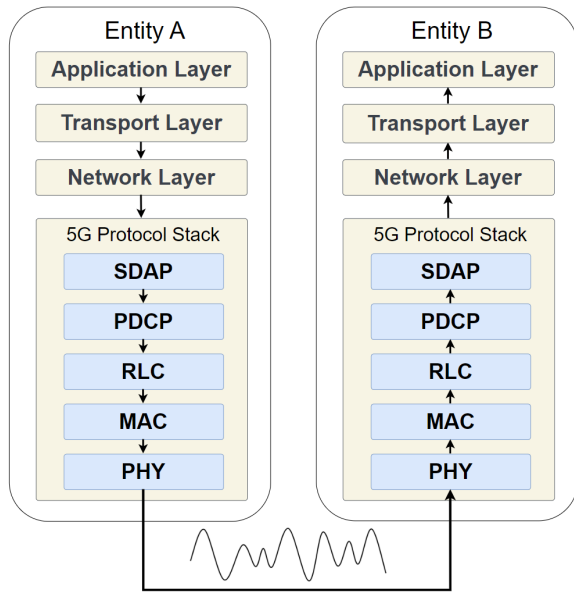
Fig. 1: Layered Architecture of User-plane Data Transmission in 5G

Fig. 2: 5G Ciphering Process

into a bitstream using a standardized format (e.g., IEEE 754 for floating-point numbers). This bitstream then moves down the protocol stack from the transport layer to the 5G protocol stack. At the physical layer, the data is converted into electromagnetic signals and sent over the air to the receiver. On reception, this process is reversed through each layer, ultimately reconstructing the original bitstream so the application layer can recover the transmitted value[1].

At the network level, three main mechanisms are employed for safeguarding data confidentiality and integrity during transmission:

1) Checksum Error Detection: The transport and network layers use checksum in their headers that allow the receiver to detect errors introduced during transmission. Only the transport layer checksum covers the user data payload to verify the entire segment and protect it. For the transport layer protocols TCP and UDP, the data is divided into 2-byte words for checksum calculation as specified by IETF standards. If the length of the data is not a multiple of 2 bytes, zeros would be padded to the end to complete the final 2-byte word. By summing these 2-byte words and storing the one's complement of the sum, the protocol enables one-sided error detection. Any mismatch at the receiver side indicates that bits were corrupted in transit, while a match does not necessarily guarantee the absence of corruption.

2) Encryption Algorithm for 5G (NEA): Confidentiality of data transmitted over 5G is enforced at the PDCP layer using the NEA algorithm, as shown in Fig. 2 [2]. On the sender side, NEA uses the secret key established during the
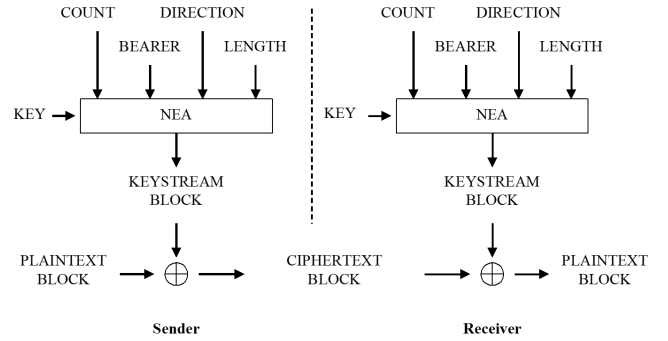
5G authentication and key agreement process (AKA), and control parameters derived from synchronized protocol state or unencrypted framing to generate a keystream. This keystream is then XORed with the plaintext bitstream to produce the ciphertext. On the receiver's side, the same keystream is generated using the same key and synchronized parameters, and XORed again with the ciphertext to recover the original plaintext. This mechanism ensures that a third party cannot understand the intercepted data without access to the secret key and relevant control information.

3) Integrity Algorithm for 5G (NIA): Integrity protection in 5G is achieved using the NIA algorithm. The sender generates a 32-bit message authentication code (MAC-I) by applying NIA to the message, key, and relevant control parameters, and appends this MAC to the transmitted data. Upon reception, the receiver uses the same key and parameters to compute the expected MAC (after decryption if encryption is used) and compares it to the received MAC-I. A match confirms data integrity, while a mismatch indicates possible tampering or transmission errors, prompting the receiver to discard the message. This method prevents unauthorized modifications to the data at the cost of adding a 4-byte authentication tag to each transmission.

B. Related Work

Bit-flipping attacks have been studied in various network environments. Paterson et al. demonstrated bit-flipping attacks on IPSec-protected datagrams and analyzed their impacts [6]. The same type of attacks has also been examined in Long-Range Wide-Area Network (LoRaWAN). Lee et al. conducted a risk analysis of bit-flipping in LoRaWAN and proposed a key-based shuffling mechanism using circular shifts and swaps to mitigate such threats [7]. In the context of cellular networks, Rupprecht et al. introduced the ALTER attack on LTE, which manipulates data payloads by adding a manipulation mask that selectively flips bits in the message. In their work, altering fields such as the destination IP address required the attacker to know the original plaintext to compute the correct manipulation mask, leading the authors to argue that robust, mandatory integrity protection in 5G is a necessary countermeasure [8]. Tan et al. discussed bit-flipping attacks on 5G data plane

---

[1]For clarity, this discussion abstracts away certain elements, such as the 5G core network, since they are not central to our research focus.

[2]Fig 2 is adapted from 3GPP TS 33.501 [4].

packets, focusing on parts of the message with predictable content, such as IP headers or retransmission indicators. They similarly noted that an attacker typically needs to know the original plaintext to forge a targeted modification via bit-flipping [9]. However, our study challenges this assumption, showing that knowledge of the message format structure alone is often sufficient to conduct successful bit-flipping attacks and achieve predictable results. Additionally, previous bit-flipping research in 4G and 5G has primarily targeted control fields where the plaintext is known, rather than exploring the potential impact of bit-flipping attacks on less predictable data payloads, which is an area that our work addresses.

III. BIT-FLIPPING ATTACK IN 5G

A. Threat Model

We define key terms for bit-flipping attacks and the adversary capabilities in this subsection. In a 5G communication setting, let $\mathcal{S}$ denote the sender, $\mathcal{R}$ denote the receiver, and $\mathcal{A}$ denote the adversary. In a benign scenario, $\mathcal{S}$ transmits a message over 5G, and $\mathcal{R}$ reliably receives the correct message. However, the addition of $\mathcal{A}$ no longer guarantees reliable communication as $\mathcal{A}$'s goal is to mutate as many messages as possible. We assume the followings for our threat model:

- $\mathcal{A}$ acts as a Man-in-the-Middle (MITM) attacker that is able to intercept the physical-layer signal from $\mathcal{S}$ to $\mathcal{R}$.
- $\mathcal{A}$ possesses the capability to decode the intercepted signal, reconstruct the PDCP-layer ciphertext bitstream, and flip any number of bits at arbitrary positions correspond to the field of checksum and data payload.
- After mutation, $\mathcal{A}$ can re-encode and forward the modified message to $\mathcal{R}$.
- $\mathcal{A}$ cannot decrypt the NEA encrypted ciphertext.

B. Bit-Flipping Attack

Even if $\mathcal{A}$ cannot decrypt the message or learn its plaintext, it can still strategically flip selected bits to maximize its chance to bypass the checksum error detection. Consider the bit-flipping attack illustrated by the red arrows (1) in Fig. 3: $\mathcal{S}$ sends a value 2.0 to $\mathcal{R}$. $\mathcal{A}$ flips two bits, one in the checksum field, and the other in the data payload field that aligns with it in the 2-byte words. The attack will successfully bypass the checksum verification and mutate one bit in the data payload if and only if the parity of the two flipped bits is even in plaintext, as illustrated in Fig. 4 (a). The plaintext is then deciphered to a value of 4.0. This yields an attack success rate around 50% for any message since the checksum field is almost pairwise independent of any one aligned bit in the data payload. Such independence arises because checksum computation incorporates not just the data payload, but also other fields like source and destination ports, message length, etc., which may vary across applications. We refer to such attacks as "*Checksum Bit-Flipping Attacks*". Another attack strategy is to flip two aligned bits within the data payload, which we call "*Payload Bit-Flipping Attacks*" illustrated by the pink arrows (2) in Fig. 3. In this case, the attack succeeds when the parity of the flipped bits is odd in plaintext, as
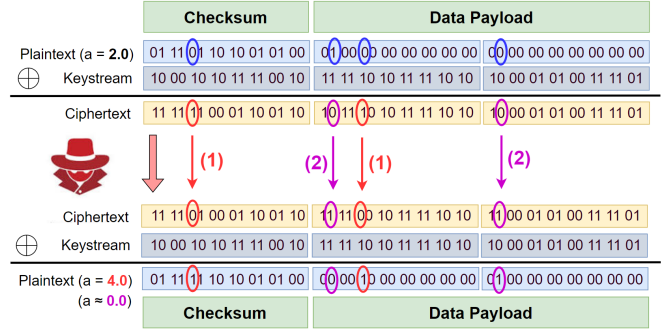

Fig. 3: Bit-Flipping Attack

Fig. 4: Success-Failure Matrix of (a) Checksum Bit-Flipping Attack, (b) Payload Bit-Flipping Attack. The two flipped bits must be aligned in the same column when divided into two-byte chunks for checksum calculation.

illustrated in Fig. 4 (b), and the value is mutated to a small value close to 0.0. While this allows $\mathcal{A}$ to mutate two bits, its success depends directly on the underlying plaintext values, and the attack does not benefit from the pairwise independence property seen in the checksum bit-flipping case.

As a MITM attacker, there is no limitation on the number of bits $\mathcal{A}$ could flip. The only issue is that there is a stark trade-off between the number of bit flips and the mutation success rate. For checksum bit-flipping attacks on separate checksum bits, we empirically show that there is an exponentially decaying trend of success rates with respect to the number of flipped bits. Payload bit-flipping attacks are difficult to analyze due to their high dependence on the specific value of the data payload plaintext. Nonetheless, we expect them to also deteriorate in success rate quickly as the number of flips increase.

IV. KEYSTREAM-BASED SHUFFLING DEFENSE

In this section, we propose a general defense mechanism against bit-flipping attacks via shuffling. We first observe that the plaintext and the ciphertext have the same columnwise alignment; i.e. flipping a position in the ciphertext directly corresponds to flipping that same position in the plaintext itself. However, if the ciphertext is shuffled unpredictably, $\mathcal{A}$ would have much less confidence in being able to pinpoint specific positions to attack. The challenge is to ensure that $\mathcal{A}$

Algorithm 1 Keystream-based Shuffling Implementation

Shared: Cipher Key k, control parameters $paras$ for each transmit message, Pseudorandom Permutation Generator $PRP()$

1: **procedure** SENDER(Plaintext P)
2:     $K \leftarrow NEA(k, paras)$ ▷ Generate Keystream
3:     $C \leftarrow P \oplus K$ ▷ XOR Ciphering
4:     $T \leftarrow PRP(K)$ ▷ Permutation Table
5:     create C_s s.t. $|C_s| = |C|$
6:     **for** i in $len(C)$ **do**
7:         $C_s[T[i]] = C[i]$ ▷ Shuffle according to T
8:     **end for**
9:     $send(C_S)$
10: **end procedure**

11: **procedure** RECEIVER(Shuffled Ciphertext C_s)
12:     $K \leftarrow NEA(k, paras)$
13:     $T \leftarrow PRP(K)$
14:     create C_u s.t. $|C_u| = |C_s|$
15:     **for** i in $len(C_s)$ **do**
16:         $C_u[i] = C_s[T[i]]$ ▷ Invert the shuffle
17:     **end for**
18:     $P \leftarrow C_u \oplus K$ ▷ XOR($\cdot, K$) is its own inverse
19: **end procedure**

cannot deduce the mechanism of the shuffling while enabling $\mathcal{R}$ to perform the inverse of the shuffle deterministically.

Theoretically, this is impossible to achieve without establishing some form of private information between $\mathcal{S}$ and $\mathcal{R}$. Fortunately, 5G NEA lets $\mathcal{S}$ and $\mathcal{R}$ generate a unique, secret keystream for each message transmitted between them. The idea is to then reuse the private keystream generated for a message to seed a pseudorandom permutation function that shuffles the checksum and data payload fields. $\mathcal{A}$, without the knowledge of the cipher key and keystream, would be cryptographically unable to guess each bit's original position better than random. On the contrary, it is easy for $\mathcal{R}$ to construct an inverse shuffling procedure to obtain the original plaintext.

Our defense mechanism leverages the Fisher-Yates shuffling algorithm, using the keystream generated for each transmission to determine the shuffle order, and applies this process directly to the ciphertext. The receiver would construct the forward permutation table with the same keystream, then invert it to produce the inverse permutation table. The plaintext is extracted after reversing the effects of the shuffle. Algorithm 1 generalizes the implementation over different choices of pseudorandom permutation generators.

The keystream-based shuffling proposes an alternative defense mechanism to NIA, and there are inherent trade-offs. NIA is a strong integrity protection algorithm that can detect any tampering of the data payload with near zero failure. However, it requires redundant bytes for each message. Shuffling,

on the other hand, does not append any additional bits to the transmitted message, gaining an advantage in latency and communication overhead. The sacrifice is that shuffling is not deterministic and has a chance to fail to defend. Theoretically, for 2-bit flips, it is expected that the attack still succeeds with probability close to $\frac{1}{16} \cdot \frac{1}{2} = \frac{1}{32} = 0.03125$. This is because there is only a $\frac{1}{16}$ chance that two randomly selected positions will be aligned in the two bytes for checksum, and if we reasonably assume that no two bits are highly correlated with each other, there is a uniform $\frac{1}{2}$ chance that they match in parity. While not perfect, it is still a significant improvement over deactivating NIA and only using checksum, which has a $\frac{1}{2}$ failure rate. For applications such as CACC, we expect that such low mutation success rate will be enough to disallow contiguous attacks for extended numbers of timesteps. Finally, shuffling has the unique trait that it also obfuscates the position of the bits, effectively randomizing the position of the bit flips attempted by $\mathcal{A}$. Even if $\mathcal{A}$ successfully mutates a payload, it has no control over which bit was actually attacked, which eliminates its capabilities to generate intentional attacks on specific bit fields.

V. EXPERIMENT RESULTS

In this section, we first assess the feasibility of bit-flipping attacks on 5G networks using the OAI platform. Then, to evaluate practical implications, we simulate a scenario in which vehicle A communicates its position, velocity, and acceleration to vehicle B through 5G. For this purpose, we select three vehicles at random from the Next Generation Simulation (NGSIM) dataset, collected by the U.S. Department of Transportation [10], and use their real-world trajectories as the shared states. We then analyze the impact of checksum bit-flipping attacks and the mitigation performance of our proposed shuffling-based defense in this context. Finally, we extend our analysis to examine the effects of payload bit-flipping attacks on a fourth vehicle.

A. Bit-Flipping Attacks Simulation

We simulate the bit-flipping attacks using OAI, a reputable open-source simulation software platform used for 5G networking research. Since our threat model assumes that the attacker $\mathcal{A}$ can intercept and decode transmissions to the ciphertext level, we implement the bit-flipping attacks on the sender side for convenience. The attacks are implemented by modifying the OAI source code in the *openair2/LAYER2/nr_pdcp* directory, specifically by editing the *deliver_pdu_drb_ue* function in *nr_pdcp_oai_api.c*. Our keystream-based shuffling defense is also implemented at the PDCP layer in OAI, with modifications to the functions *nr_pdcp_entity_process_sdu* and *nr_pdcp_entity_recv_pdu* in *nr_pdcp_entity.c*[3].

---

[3]The code associated with this paper can be found at https://github.com/DerekDuan615/5G-Bit-Flipping-Attack-Exploration (branch: keystream-based-shuffle).
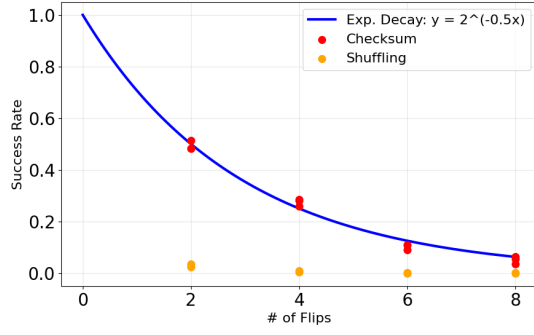
Fig. 5: Mutation Success Rate of Checksum Bit-Flipping Attack with and without Shuffling Defense

TABLE I: Mutation Success Rate of Checksum Bit-Flipping Attacks

# of flips Trajectory	2-bits	4-bits	6-bits	8-bits
Vehicle1	0.514	0.280	0.110	0.064
Vehicle2	0.482	0.258	0.090	0.034
Vehicle3	0.482	0.284	0.108	0.054

We verify the effectiveness of bit-flipping attack in a scenario where vehicle A sends its position, velocity, and acceleration (set to 300.0, 25.0, and 2.0) to vehicle B. Results for both checksum and payload bit-flipping attacks are shown in Fig. 6 and Fig. 7, with blue dashes marking checksum locations and blue boxes indicating bit-flip and mutated values. In the checksum bit-flipping attack, the adversary alters the ciphered checksum field from `0xc354` to `0xc3d4`, and the ciphered acceleration value from `0xe2c2ce32` to `0xe242ce32`. As a result, the receiver decrypts the mutated message, which passes the checksum verification and results in a received acceleration value of 4.0, as displayed in the upper-right terminal in Fig. 6. It is important to note that OAI's checksum implementation inverts all checksum bits after calculation, so the observed success-failure pattern is exactly the opposite of what is shown in Fig. 4 (a). For the payload bit-flipping attack, the ciphertexts corresponding to the position and velocity values are manipulated, causing the receiver to obtain position and velocity values of 268.0 and 27.0, respectively, which align with the success-failure matrix expected in Fig. 4 (b).

B. Attack and Defense Effect Analysis

To ensure that our approach targets realistic environments, we used the NGSIM Open Data for vehicle trajectories. Each vehicle's acceleration, velocity, and X-coordinate data were extracted and used as the transmitted message for a single timestep. Each individual experiment consisted of the success rate of 500 data points of a single vehicle.

Our main experimental results for checksum bit-flipping attacks are summarized in Figure 5 and numerically reported in Tables I and II. The red dots show the success rate of a checksum bit-flipping attack when NIA is disabled and the checksum is the sole defense against bit-flipping. While we cannot say that the relationship between the payload and

TABLE II: Mutation Success Rate of Checksum Bit-Flipping Attacks with Shuffling Defense

# of flips Trajectory	2-bits	4-bits	6-bits	8-bits
Vehicle1	0.036	0.004	0.000	0.002
Vehicle2	0.022	0.008	0.000	0.000
Vehicle3	0.028	0.004	0.002	0.000

TABLE III: Mutation Success Rate of 2-bit Flipping Attacks at Varying Payload Positions

Trajectory Locations	Vehicle 4
Checksum and Acc	0.530
Acc, Vel Sign	0.412
Acc, Vel Exp	0.276
Acc, Vel Mant	0.436

checksum bits being flipped are truly pairwise independent, our experiment empirically proves the exponential decaying trend for the mutation success rate in checksum bit-flipping attacks. In practice, we can estimate that every two additional flips halves the mutation success rate, following the blue line on the graph ($y = 2^{-x/2}$) very closely. Shuffling, on the other hand, maintains a low mutation success rate even when the number of bit flips is only 2. Indeed, the empirical success rate is close to the theoretical probability of 0.03125 we suggested in the previous section. The success rate rapidly declines as the number of flips increase, and for more than 6 flips, the success rate is almost negligible.

As proof for the non-independence of the payload attacks, we run multiple payload bit-flipping attacks that vary in location. Specifically, we attack the sign bit, the second most significant bit (MSB) of the exponent field, and the MSB of the mantissa field in both acceleration and velocity floats. A checksum bit-flipping attack is added for comparison. As expected, only the checksum bit-flipping attack shows mutation success rate near 50%, whereas other three attacks hover significantly below. Nonetheless, this does not suggest that payload bit-flipping attacks should always have lower success rate than the checksum attack; the values might as well have been far exceeding 50%. There might be potential to exploit this fact to increase the attack success rate beyond 50%. However, this discussion is out of scope for this study.

VI. CONCLUSION

We identify a feasible MITM bit-flipping attack in 5G networks and provide an alternative shuffling defense mechanism that does not require the redundancy bit overhead with the cost of slightly increased probability of failure compared to NIA. Our key findings through experiments verify that bit-flipping acts according to theoretical expectations and can be successful at mutating one bit with 50% probability if applied in particular positions. However, since the attack depends on knowing the exact alignment of the bits with respect to the checksum, shuffling effectively disallows the adversary to consistently attack particular positions of the data payload. Ultimately, shuffling is an appealing choice for 5G applications

Fig. 6: Checksum Bit-Flipping Attack (left: vehicle A; right: vehicle B)

Fig. 7: Payload Bit-Flipping Attack (left: vehicle A; right: vehicle B)

such as CACC, where the small defense failure probability is worth to trade for the additional overhead incurred by NIA. Future work would include analyzing the impact of bit-flipping attacks in CACC applications and developing more consistent yet robust defense mechanisms than shuffling.

REFERENCES

[1] M. Series, "Imt vision–framework and overall objectives of the future development of imt for 2020 and beyond," *Recommendation ITU*, vol. 2083, no. 0, pp. 1–21, 2015.

[2] J. G. Andrews, S. Buzzi, W. Choi, S. V. Hanly, A. Lozano, A. C. Soong, and J. C. Zhang, "What will 5g be?" *IEEE Journal on selected areas in communications*, vol. 32, no. 6, pp. 1065–1082, 2014.

[3] C. Campolo, A. Molinaro, A. Iera, and F. Menichella, "5g network slicing for vehicle-to-everything services," *IEEE Wireless Communications*, vol. 24, no. 6, pp. 38–45, 2018.

[4] *TS 33.501, Security architecture and procedures for 5G system*, 3rd Generation Partnership Project, 7 2025, v19.3.0.

[5] S. Boddupalli, A. S. Rao, and S. Ray, "Resilient cooperative adaptive cruise control for autonomous vehicles using machine learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 9, pp. 15 655–15 672, 2022.

[6] K. G. Paterson and A. K. Yau, "Cryptography in theory and practice: The case of encryption in ipsec," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2006, pp. 12–29.

[7] J. Lee, D. Hwang, J. Park, and K.-H. Kim, "Risk analysis and countermeasure for bit-flipping attack in lorawan," in *2017 International conference on information networking (ICOIN)*. IEEE, 2017, pp. 549–551.

[8] D. Rupprecht, K. Kohls, T. Holz, and C. Pöpper, "Breaking lte on layer two," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 1121–1136.

[9] Z. Tan, *System Security in 5G/4G/xG Mobile Networks: New Attacks and Countermeasures*. University of California, Los Angeles, 2022.

[10] U.S. Department of Transportation Federal Highway Administration, "Next generation simulation (ngsim) vehicle trajectories and supporting data," Dataset provided by ITS DataHub through Data.transportation.gov, 2016, accessed 2025-07-28. [Online]. Available: http://doi.org/10.21949/1504477