

Vulnerabilities and Attacks in Bluetooth: a Survey

Mingrui Ai¹, Kaiping Xue¹ ✉, Lutong Chen², Xuanbo Huang², and David S.L. Wei²

¹*School of Cyber Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China;*

²*Department of Computer and Information Science, Fordham University, Bronx, NY 10458 USA*

✉ Correspondence: Kaiping Xue, E-mail: kpxue@ustc.edu.cn

© 2026 The Author(s). This is an open access article under the CC BY-NC-ND 4.0 license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).

Graphical abstract



Public summary

- Presents a longitudinal and specification-centric analysis of Bluetooth security, synthesizing two decades of evolution across Bluetooth Classic and BLE to establish a unified understanding of their security and privacy properties.
- Constructs a rigorous taxonomy that systematizes over 100 publicly disclosed vulnerabilities and attacks, enabling structured reasoning across protocol layers, system components, and implementation boundaries.
- Provides a comparative assessment of attack capabilities, underlying root causes, and defensive limitations, offering analytically grounded insights into the systemic weaknesses of current Bluetooth ecosystems.
- Identifies unresolved research challenges and articulates forward-looking research directions, laying the methodological foundation for future Bluetooth security studies and the development of principled mitigation strategies.

Vulnerabilities and Attacks in Bluetooth: a Survey

Mingrui Ai¹, Kaiping Xue¹ ✉, Lutong Chen², Xuanbo Huang², and David S.L. Wei²

¹*School of Cyber Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China;*

²*Department of Computer and Information Science, Fordham University, Bronx, NY 10458 USA*

✉ Correspondence: Kaiping Xue, E-mail: kpxue@ustc.edu.cn

© 2026 The Author(s). This is an open access article under the CC BY-NC-ND 4.0 license (<http://creativecommons.org/licenses/by-nc-nd/4.0/>).



Cite This: *JUSTC*, 2026, 56(X): (17pp)



Read Online



Supporting Information

Abstract: Bluetooth is a short-range wireless communication technology widely used for commands and data exchange among billions of mobile, wearable, and IoT devices. Despite significant improvements in Bluetooth security mechanisms with the upgrade of the Bluetooth Specification, various security vulnerabilities have continued to be discovered over the past few decades, leading to attacks that impact a huge number of devices. In contrast to the enthusiasm for research on specific Bluetooth security issues, a systematic understanding of this field is still lacking, especially concerning the security and privacy properties of different versions and features, including both Bluetooth Classic and Bluetooth Low Energy. This gap hinders the development of a systematic methodology in Bluetooth security research. In this paper, we review the evolution of the Bluetooth Specification and its security mechanisms over the past two decades. We then systematically provide a comprehensive summary of more than 100 vulnerabilities and attacks in Bluetooth communication using our well-defined taxonomy. Through this taxonomy, we categorize vulnerabilities and attacks based on affected system levels and locations in the Bluetooth protocols and stack. We also offer insights into the strengths and limitations of these attacks. Based on our systematic review, we propose promising directions to inspire future research in Bluetooth security.

Keywords: Bluetooth security; vulnerability; attack

CLC number:

Document code: A

I Introduction

Invented by Ericsson in 1994, Bluetooth has gradually evolved into one of the most widely used low-power, short-range wireless communication technologies. Billions of smartphones, laptops, tablets, headsets, keyboards, mice, fitness trackers, and other types of electronic devices utilize Bluetooth technology for data transmission and device control. The Bluetooth specifications, standardized by the Bluetooth Special Interest Group (SIG), consist of two partly incompatible protocols: Bluetooth Classic (BC) and Bluetooth Low Energy (BLE). It's important to note that Bluetooth mesh is a network-layer protocol based on BLE. Bluetooth Classic is mainly used for continuous communication with high traffic, such as audio streaming and file transfer, while Bluetooth Low Energy is primarily used for short burst communication, such as control signaling, and for the large-scale networking of Bluetooth devices through the Bluetooth mesh protocol.

Since Bluetooth is designed for and widely deployed in billions of wearable, mobile, and personal computing devices, a significant amount of sensitive information is transmitted over Bluetooth connections, and many sensitive devices are controlled via Bluetooth. Therefore, it is crucial to ensure the security of Bluetooth specifications and implementations. Despite concerted efforts by the Bluetooth SIG, vendors, and security researchers have been devoted to improving Bluetooth security, vulnerabilities continue to be discovered, and their numbers are growing much faster in recent years.

Frequent vulnerability disclosures not only indicate an increasing concern over Bluetooth technology security but also highlight that the security of Bluetooth protocols and implementations is still imperfect. According to known vulnerabilities and attacks, adversaries can eavesdrop on calls, steal sensitive data like SMS verification codes and contracts, track victims' activities, make phone calls, and even gain complete control of victim devices. Therefore, understanding these vulnerabilities and attacks in detail is crucial for advancing the security of Bluetooth.

Our work aims to provide an up-to-date and comprehensive survey of the security vulnerabilities and attacks on current Bluetooth specifications and implementations to help general Bluetooth users be aware of the risks in Bluetooth. We conduct an extensive investigation on various Bluetooth vulnerabilities and attacks, including but not limited to those from the literature, and categorize them based on their location in Bluetooth technology. We also analyze these vulnerabilities and attacks and propose practical methods to mitigate them.

The rest of the paper is organized as follows: In Section 2, we briefly introduce Bluetooth specifications and security mechanisms. Section 3 presents our taxonomy of Bluetooth vulnerabilities and attacks, categorizing them accordingly. Vulnerabilities and attacks at the algorithm level are categorized and described in detail in Section 4, while those at the protocol level are discussed in Section 5. Implementation-level flaws and attacks are detailed in Section 6, those at the

application level are analyzed in Section 7, and privacy-related issues are covered in Section 8. We provide some future directions in Section 9. Finally, we summarize the paper in Section 10.

II Background

A Bluetooth Overview

Ericsson introduced Bluetooth technology in 1994 and established the Bluetooth Special Interest Group (SIG) with IBM, Intel, Nokia, and Toshiba for Bluetooth development. Bluetooth SIG released Bluetooth 1.0 in 1999 and updated it to versions 1.1 (2001) and 1.2 (2003). In 2002, the IEEE 802.15 Working Group standardized Bluetooth as IEEE 802.15.1-2002. In 2004, Bluetooth was updated to version 2.0, which introduced Enhanced Data Rate (EDR) technology to improve Bluetooth transmission capability. In 2007, Secure Simple Pairing (SSP) was introduced in Bluetooth 2.1 to strengthen pairing security. Bluetooth was upgraded to version 3.0 in 2009, introducing Alternate MAC/PHYs (AMP) technology to further improve Bluetooth transmission speed. In 2010, Bluetooth 4.0 introduced the new Bluetooth Low Energy (BLE) stack, while the previous Bluetooth stack was called Bluetooth Classic (BC) stack. Bluetooth SIG proposed Bluetooth 4.1 in 2013 and 4.2 in 2014, each introducing Secure Connections (SC) in BC and BLE, respectively, to enhance Bluetooth security. Bluetooth 5.0 was introduced in 2016 to improve the BLE transmission rate and reduce power consumption. In 2017, Bluetooth SIG proposed Bluetooth mesh technology based on the BLE stack for large-scale multi-hop networking. Subsequent updates to Bluetooth 5.0 resulted in four sub-versions, released in 2019, 2021, and 2023. Bluetooth 6.0^[1] is the latest version as of now.

Bluetooth operates in the 2.4 GHz ISM frequency band, with a typical communication range of 1 to 100 meters. Bluetooth Classic (BC) divides this band into 79 channels (each 1 MHz wide), while Bluetooth Low Energy (BLE) uses 40 channels (spaced 2 MHz apart), including 3 advertising channels and 37 data channels. The maximum theoretical data rate for BC is 3 Mbps, and for BLE, it is 2 Mbps, though actual rates may vary.

Bluetooth devices use a 48-bit Bluetooth device address as their unique identifier. The high 24 bits are assigned to device manufacturers by the Bluetooth SIG, and the low 24 bits are assigned to different Bluetooth devices by their manufacturers. BC does not have an additional privacy protection mechanism for the Bluetooth device address, except that the high 24 bits of the address are not transmitted after connection establishment. In contrast, BLE allows the use of a private device address to conceal the device's real address, thereby preventing BLE device tracking. User-friendly Bluetooth device names, which can be modified by users, are designed to facilitate device identification.

Bluetooth devices play different roles when communicating. In BC, the roles are divided into Master and Slave, where the Master initiates connections, and the Slave receives them. For BLE devices, the roles are divided into Central and Peripheral, with the Central device establishing connections based on the advertising packets from the Peripheral device. Note

that in BC, the roles of Master and Slave can be exchanged between devices during a connection. In contrast, the roles of BLE devices are fixed.

B Bluetooth Architecture

The Bluetooth system consists of two parts: the Bluetooth core stack and the application, as illustrated in Fig. 1. The core stack is divided into the Bluetooth controller and the Bluetooth host, interacting via the Host Controller Interface (HCI). The Bluetooth controller, typically implemented in the Bluetooth chip, handles lower-level operations including Physical and Link layer protocols. The Bluetooth host, running in the device's operating system (OS), is responsible for higher-level operations including the L2CAP layer and other application-related protocols. Bluetooth profiles, defined by the Bluetooth SIG, specify necessary functions in the core stack and other required settings. To date, there are numerous Bluetooth profiles, with common ones including the Hands-Free Profile (HFP) for voice connections between smartphones and Bluetooth headsets, the Phone Book Access Profile (PBAP) for transferring contacts, and the Human Interface Device Profile (HID) for devices like mice and keyboards. Additionally, the Bluetooth mesh profile, based on the BLE stack, is a networking protocol offering many-to-many communications.

C Bluetooth Security Mechanism

Various security mechanisms are designed to ensure the security of Bluetooth communication. Fig. 2 illustrates the connection establishment process for BC and BLE. When two Bluetooth devices establish a secure connection for the first time, they negotiate a shared key through the pairing process. After successful pairing, the devices store the shared key (Link Key for BC and Long Term Key for BLE) to form a bonding relationship. BC supports two pairing methods: PIN Pairing for devices prior to version 2.1 and Secure Simple Pairing (SSP) for version 2.1 onward. PIN Pairing requires two devices to enter the same PIN code for key derivation and mutual authentication. SSP, introduced in Bluetooth 2.1, util-

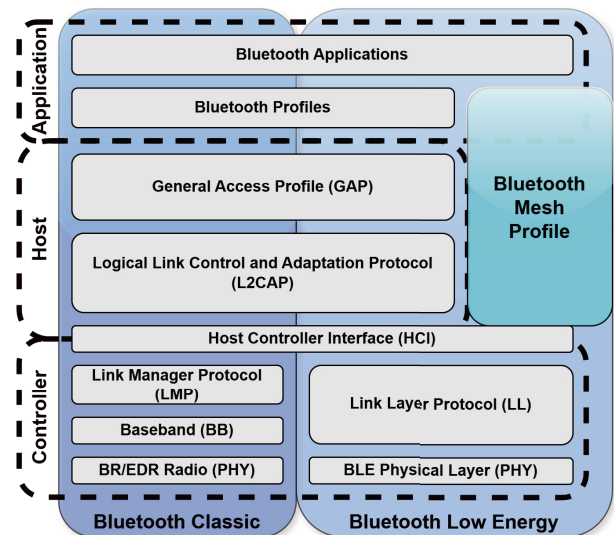


Fig. 1. Architecture of Bluetooth Classic and Bluetooth Low Energy

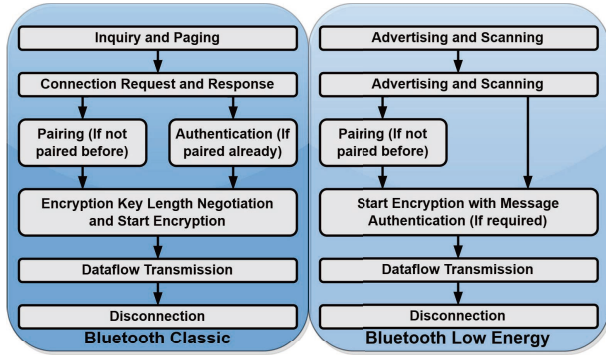


Fig. 2. Overview diagram for connection establishment of BC and BLE

izes the Elliptic Curve Diffie-Hellman (ECDH) algorithm based on the P-192 Elliptic curve for key derivation, which was later upgraded to the P-256 Elliptic curve in Secure Connections (introduced in Bluetooth 4.1) to enhance security. SSP defines four association models: Just Works (JW), Numeric Comparison (NC), Passkey Entry (PE), and Out of Band (OOB), catering to different device I/O capabilities. BLE also supports two pairing methods: LE Legacy Pairing (LEP) for devices prior to version 4.2 and Secure Connections Pairing (SCP) for version 4.2 onward. The association models in SCP are similar to those in SSP, but LEP differs notably, including the absence of the NC model.

Designing a unified mechanism for Bluetooth device pairing and user confirmation is challenging due to the wide range of Bluetooth devices and their varying user-device interaction interfaces. As a result, Bluetooth devices are categorized based on their input/output capabilities (I/O capabilities), as shown in Table 1. In terms of input capability, devices can be classified into three categories: “No input”, “Yes/No”, and “Keyboard”. “No input” devices cannot receive any user input, “Yes/No” devices allow users to confirm or reject pairing requests, and “Keyboard” devices are capable of receiving numeric input. Regarding output capability, there are two classifications: “No output” and “Numeric output”. “No output” devices lack a display, while “Numeric output” devices can show numeric strings. Combining these input and output capabilities, the Bluetooth Specification further categorizes devices into five types: “NoInputNoOutput (NINO)”, “DisplayOnly (DO)”, “DisplayYesNo (DYN)”, “KeyboardOnly (KO)”, and “KeyboardDisplay (KD)”. “NINO” devices neither receive user input nor display information. “DO” devices can only show numeric strings. “DYN” devices allow users to confirm or reject requests based on displayed numeric strings. “KO” devices can receive numeric input but have no display. “KD” devices can both receive numeric input and show numeric strings on the screen. It is important to note that the “KD” category is, as of the current specifications, unique to BLE, though this may change as technology evolves.

Furthermore, the Bluetooth standard determines the appropriate association model for pairing based on the I/O capabilities of the initiator and responder, as outlined in Table 2. In BLE Legacy Pairing, the “Just Works” model is employed if either the initiator or responder has “NINO” capabilities, or if each device is either a “DO” or “DYN” device. The “Passkey Entry” model is used when one device is “KO” or “KD”, and

Table 1. I/O capabilities mapping of BC and BLE

Input cap.	Output cap.	
	No output	Numeric output
No input	NoInputNoOutput	DisplayOnly
Yes/No	NoInputNoOutput	DisplayYesNo
Keyboard	KeyboardOnly	KeyboardDisplay (BLE)/ DisplayYesNo (BC)

Table 2. I/O capabilities mapping to association models

Responder	Initiator				
	Display Only	Display YesNo	Keyboard Only	NoInput NoOutput	Keyboard Display
Display Only	JW	JW	PE	JW	PE
Display YesNo	JW	JW (LEP) NC (SSP/SCP)	PE	JW	PE (LEP) NC (SCP)
Keyboard Only	PE	PE	PE	JW	PE
NoInput NoOutput	JW	JW	JW	JW	JW
Keyboard Display	PE	PE (LEP) NC (SCP)	PE	JW	PE (LEP) NC (SCP)

the other is not “NINO”. For BLE Secure Connections Pairing, “Numeric Comparison” is employed if both devices are either “KO” or “KD”. In other cases, the association models for BLE Secure Connections align with those of BLE Legacy Pairing. In BC Secure Simple Pairing/Secure Connections, “Numeric Comparison” is used if both the initiator and responder are “DYN” devices. For all other cases, the association models comply with those of BLE Legacy Pairing.

Once two Bluetooth devices are bonded, the pairing procedure is no longer necessary for future secure connections. Secure connection establishments in BC require device authentication and the negotiation of encryption key length. There are two versions of BC Authentication: Legacy Authentication (prior to 4.1) and Secure Authentication (4.1 onward). Legacy Authentication involves unilateral authentication using the E1 algorithm, whereas Secure Authentication employs mutual authentication using the HMAC-SHA-256 algorithm. If authentication is successful, the two devices negotiate the encryption key length, which ranges from 1 to 16 bytes, before initiating encrypted communication. If both devices support Secure Connections, Secure Authentication and AES-CCM are used for authentication and encryption, respectively. If not, Legacy Authentication and the E0 stream cipher are employed for authentication and encryption. For BLE, the secure connection establishment does not involve an authentication process. Instead, AES-CCM is utilized for confidentiality, integrity, and data authentication.

The Bluetooth Core Specification incorporates various security algorithms for different purposes, including the E0 algorithm for encryption, the SAFER+ algorithm for pairing and key generation, and the E1 algorithm for authentication. However, these self-developed algorithms are not approved by the Federal Information Processing Standards (FIPS). As the Bluetooth standard has evolved, these algorithms are being gradually replaced by FIPS-approved algorithms such as AES-CCM, HMAC-SHA-256, and P-256 ECDH.

III Bluetooth Vulnerability and Attack Classification

According to the Glossary of Key Information Security Terms [2] by the National Institute of Standards and Technology (NIST), vulnerability is defined as “Weakness in a system, system security procedures, internal controls, or implementation that could be exploited or triggered by a threat”, while an attack is defined as “Any kind of malicious activity that attempts to collect, disrupt, deny, degrade, or destroy information system resources or the information itself”. In general, attacks utilize specific vulnerabilities, and vulnerability disclosures always accompany corresponding attacks. Therefore, it is essential to analyze the combination of vulnerabilities and attacks to be thoroughly aware of the resulting security threats.

To study existing Bluetooth vulnerabilities and attacks and summarize them, we first investigate all papers related to Bluetooth security and privacy published between 2000 and April 2025 at the big four top-tier security conferences (i.e., CCS, USENIX Security, SP, and NDSS). We also select papers from journals and conferences associated with Bluetooth security and privacy. To ensure the comprehensiveness of our survey, we search for Bluetooth-related vulnerability and attack reports from renowned security industry conferences such as Black Hat. Finally, we include vulnerabilities and attack incidents reported by The Hacker News [3] that are not covered in the above research.

To better assess the threats posed by vulnerabilities and the capabilities of attacks, we classify the Bluetooth vulnerabilities and attacks according to the affected levels. Vertically, Bluetooth security can be divided into four levels from bottom to top: algorithm level, protocol level, implementation level, and application level. Specifically, all Bluetooth security features rely on underlying cryptographic algorithms. Based on cryptographic primitives, the Bluetooth Core Specification proposes various security protocols to protect the confidentiality, integrity, and authentication of Bluetooth communications. Bluetooth device manufacturers implement corresponding security features according to the Bluetooth protocols. Application developers utilize the security capabil-

ities of the Bluetooth protocol stack to protect Bluetooth applications. Security vulnerabilities may exist at each level of Bluetooth and could be exploited for attacks. Horizontally, in addition to the above security issues, Bluetooth also faces privacy-related vulnerabilities and attacks. Therefore, we classify Bluetooth vulnerabilities and attacks into five categories: algorithm level, protocol level, implementation level, application level, and privacy-related. For each category, we further divide it into subcategories based on the affected components, which will be discussed from Section 4 to Section 8.

In Table 3, we summarize the distribution of current Bluetooth vulnerabilities and attacks under our classification. It is evident that most Bluetooth security research is focused on the protocol level and implementation level. The Fig. 4 illustrates research trends in Bluetooth vulnerabilities and attacks over the past 25 years. According to the figure, there is a small peak of research activity in the first decade, followed by a stable period. After 2018, a new wave of research began and continues to the present.

Vulnerability discovery and attack implementation methods constantly evolve with Bluetooth version updates. The security protocol design and implementation before Bluetooth 2.1 were incomplete, leading to numerous primitive vulnerabilities. Vulnerability mining in this period mainly relied on manual analysis and experience summaries, and the attack methods were straightforward. However, Bluetooth was not an extremely attractive target for attackers and security researchers at that time, given the limited functionalities and

Table 3. Bluetooth vulnerabilities and attacks classified statistic

Level	Vulnerability&Attack Type	Number
Algorithm	Encryption algorithm cracking	8
	Pairing protocol vulnerability	13
	Authentication protocol vulnerability	2
Protocol	Key negotiation and generation protocol vulnerability	8
	Miscellaneous protocol vulnerability	1
Implementation	Logic-level vulnerability	15
	Binary-level vulnerability	13
Application	Inappropriate use	10
Privacy	Communication sniffing	2
	Device tracking	8

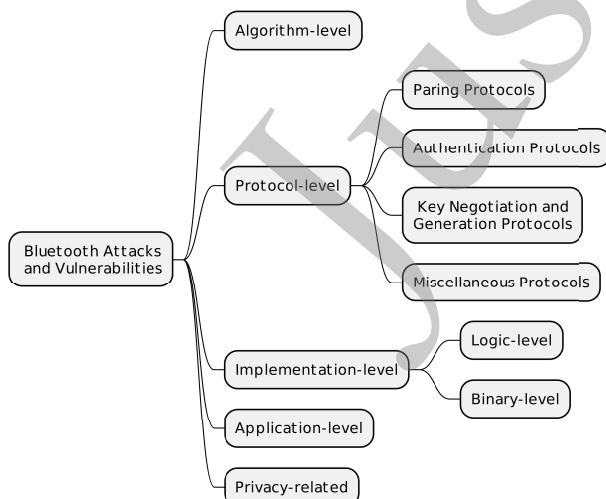


Fig. 3. Classification of Bluetooth vulnerabilities and attacks

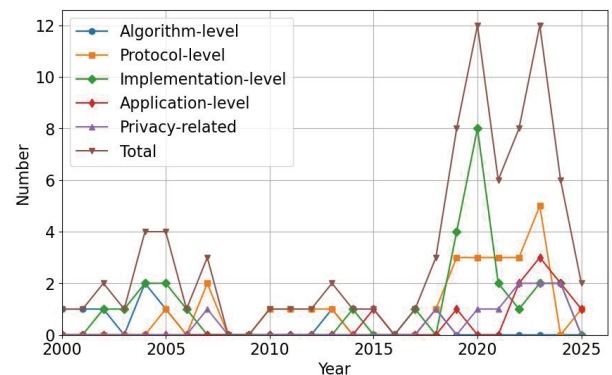


Fig. 4. Bluetooth vulnerability and attack research trends

user base. With the updates of the Bluetooth standard, especially the introduction of BLE in Bluetooth 4.0, the functions of Bluetooth greatly expanded, and the widespread use of mobile/wearable smart devices dramatically increased the user base. The growing amount of sensitive data and privileges related to Bluetooth has raised significant concerns about Bluetooth security. In recent years, security researchers have employed automated analysis methods, such as formal analysis and fuzzing, to hunt for vulnerabilities in Bluetooth protocols and implementations. Meanwhile, the security mechanisms of Bluetooth have continued to be upgraded, posing increased challenges for launching attacks on Bluetooth.

IV Algorithm-level Vulnerabilities and Attacks

Algorithms here refer to the various security functions, such as those for encryption, key derivation, authentication, pairing, etc., utilized by Bluetooth technology. Since these security algorithms are fundamental modules of Bluetooth security, implemented and executed on all Bluetooth devices regardless of the manufacturer, a vulnerability in one of these algorithms can potentially put all Bluetooth devices on the market at risk. Even more concerning is that a vulnerability in a security algorithm often indicates inherent design flaws, which are almost impossible to patch in theory. The only way to address such a vulnerability effectively is to replace the flawed algorithm with a new, secure one. This replacement necessitates a major revision of the Bluetooth standard and may result in a prolonged period of insecurity. Therefore, vulnerabilities and attacks at the algorithm level pose the greatest security threat to Bluetooth technology among all types of vulnerabilities and attacks.

Fortunately, no theoretical vulnerabilities have been found in self-developed security algorithms except for the E0 algorithm. AES-CCM replaced the E0 algorithm after Bluetooth 4.1, and attacks on the E0 algorithm had a limited impact on Bluetooth security. The E0 algorithm is a stream cipher that generates the ciphertext by exclusive-OR-ing the plaintext with the keystream. It was utilized for Bluetooth Classic communication encryption before version 4.1. Note that BC devices after version 4.2 still support the E0 algorithm for backward compatibility with older devices unless the newer devices are in Secure Connections Only Mode. The keystream of the E0 algorithm is generated using a cryptographic algorithm based on Linear Feedback Shift Registers (LFSRs) with a 128-bit Link Key. The E0 stream cipher is considered a relatively weak encryption algorithm in cryptography, according to the National Institute of Standards and Technology (NIST) [4]. Hermelin and Nyberg [5] analyzed the correlation properties of the E0 algorithm for the first time, proving that it is significant to limit the keystream length to avoid being broken. Following that research, a number of attacks on the E0 cipher keep coming up [6]–[9]. Still, they are not practical since the frames required for analysis are more than the maximum frame number 2^{26} using the same key, as specified by the Bluetooth Core Specification. Lu *et al.* [10] proposed the first practical attack on the E0 algorithm, which can recover the encryption key with 2^{38} computations and $2^{23.8}$

frames. Zhang *et al.* [11] proposed a more efficient attack requiring 2^{27} computations and $2^{22.7}$ frames. However, all attacks mentioned above are known-plaintext cryptanalysis, which cannot crack the real Bluetooth ciphertext under the ciphertext-only condition. Zhang *et al.* [12] further improved their previous attack and presented the first practical ciphertext-only attack against the E0 algorithm, using 2^{26} frames with 2^{26} computations to recover the encryption key in a few seconds.

Summary. Due to the comprehensive security evaluations of the E0 algorithm and the adoption of FIPS-approved cryptographic algorithms in the security protocols designed after Bluetooth 4.1, there are few research analyzing Bluetooth security vulnerabilities from an algorithmic perspective over the past decade. It is also unlikely to become a popular research focus for Bluetooth security in the future.

V Protocol-level Vulnerabilities and Attacks

Protocol here refers to the various protocols designed by the Bluetooth SIG, including pairing protocols, authentication protocols, key negotiation and generation protocols, etc. Note that third-party application-layer security protocols designed by device manufacturers, such as Tesla's Bluetooth car key authentication protocol, are excluded. As Bluetooth devices produced by various manufacturers comply with the same Bluetooth protocols, all Bluetooth products are threatened by the vulnerabilities existing in their adopted protocols. Mitigating such principal flaws requires the Bluetooth SIG to revise its protocols and all Bluetooth manufacturers to patch their implementations, which takes a long period and is impossible for upgrading legacy Bluetooth device protocols. In practice, device manufacturers mitigate protocol-level vulnerabilities by patching their firmware rather than revising protocols, which does not solve all security issues. The threats posed by protocol-level vulnerabilities and attacks are only second to those at the algorithm level. Vulnerabilities and attacks, according to the protocol type, are classified in Fig. 5 and summarized in Table 4.

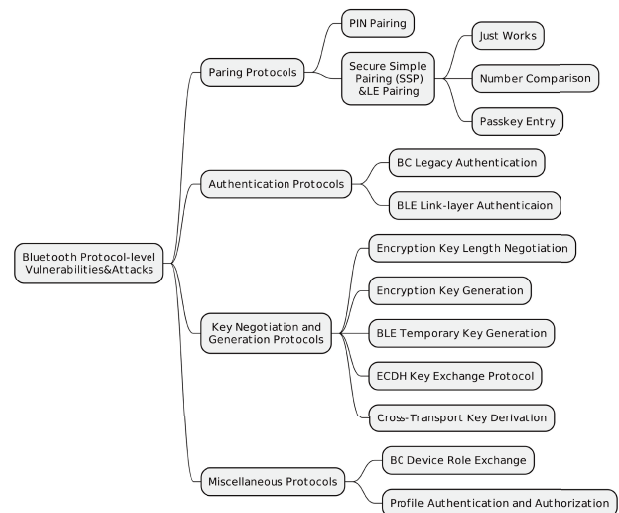


Fig. 5. Classification of Bluetooth protocol-level vulnerabilities and attacks

Table 4. Protocol-level vulnerabilities and attacks summary

Protocol Type	Attack Type	Vulnerability&Attack	Effect	Affected Protocol	Affected Stack	Affected Version	Mitigation
Pairing	Key Guessing	PIN Brute-force ^[13]	Recover PIN	PIN Pairing	BC	1.0-6.0	SSP
		Group Guessing Attack ^[16]	Recover PassKey	SSP/LEP/SCP PE	BC&BLE	2.1-6.0	Spec. Update
		BT-Nino-MITM Attack ^{[14],[15]}	Impersonate Both Sides	SSP JW	BC	2.1-6.0	Spec. Update
	Man in the Middle	Passkey Reuse ^{[17],[18]}	Impersonate Both Sides	SSP/LEP/SCP PE	BC&BLE	2.1-5.0	Spec. Update
		Method Confusion Attack ^[19]	Impersonate Both Sides	SSP/LEP/SCP PE&NC	BC&BLE	2.1-5.2	Spec. Update
		Pairing Confusion Attack ^[20]	Impersonate Both Sides	SSP/LEP/SCP PE	BC&BLE	1.0-5.3	Spec. Update
		BlueMirror ^[21]	Impersonate Initiator	SSP/LEP/SCP PE	BC&BLE	2.1-5.2	Spec. Update
		Ghost Attack ^[16]	Impersonate Responder	SSP/LEP/SCP PE	BC&BLE	2.1-6.0	Spec. Update
		Misbinding Attack ^[22]	Impersonate Responder	SSP/LEP/SCP NC	BC&BLE	2.1-6.0	Spec. Update
		Denial of Service	Keysize Confusion Attack ^[23]	SSP/LEP/SCP	BC&BLE	2.1-6.0	Spec. Update
Authentication	Authentication Bypass	BIAS ^[24]	Impersonate Bonded Device	BC Authentication	BC	2.1-6.0	Spec. Update
		BLESA ^[25]	Impersonate Peripheral	BLE Authentication	BLE	4.0-6.0	Spec. Update
	Breaking Key Negotiation	KNOB ^{[26],[27]}	Encryption Key Brute-force	Key Length Negotiation	BC&BLE	1.0-6.0	Spec. Update
		BLUFFS ^[28]	Encryption Key Reuse	Key Length Negotiation	BC	2.1-6.0	Spec. Update
Key negotiation and generation protocol vulnerability	Breaking Key Generation	TK Cracking ^[29]	Encryption Key Brute-force	LEP	BLE	4.0-4.1	SCP
		Invalid Curve Attack ^{[30],[31]}	Encryption Key Brute-force	ECDH Key Exchange	BC&BLE	2.1-5.0	Spec. Update
	Breaking Key Derivation	BLUR ^[32]	Overwrite Shared Key	Cross-Transport Key Derivation	BC&BLE	4.2-5.2	Spec. Update
		CSIA ^[33]	Overwrite Shared Key	Cross-Transport Key Derivation	BC&BLE	4.2-5.2	Spec. Update
Miscellaneous	Proactive Connection	Blacktooth Attack ^[34]	Illegal Connection	Role Exchange	BC	2.1-6.0	Spec. Update
	Privilege Escalation	Blacktooth Attack ^[34]	Illegal Service Access	Profile Authen.&Author.	BC	2.1-6.0	Spec. Update

A Vulnerabilities and Attacks on Pairing Protocols

Two Bluetooth devices utilize pairing protocols to establish a trust relationship. During pairing, the devices authenticate each other and generate shared secret keys. Once the pairing process is complete, the paired devices trust each other and store these shared keys for future secure connections. Since pairing protocols vary among Bluetooth standard versions and between BC and BLE, vulnerabilities and attacks can be classified according to the versions of the pairing protocols.

Bluetooth PIN Pairing requires the user to enter the same secret PIN code into both Bluetooth devices for authentication. The PIN length is between 1 and 16 bytes, but in practice, the vast majority of PINs consist of 4 decimal digits, meaning the PIN can be brute-forced in at most 10000 attempts. In addition, the Bluetooth Core Specification does not specify a method for PIN generation, and the randomness of the PIN is not guaranteed. Even worse, some Bluetooth devices (like wireless headsets) do not have input capabilities and use fixed default PINs (like 0000 and 1234) that cannot be modified by users. All these vulnerabilities make the PIN easy to guess. Shaked and Wool ^[13] introduced an efficient

brute-forcing PIN cracking algorithm, capable of cracking any 4-digit PIN in 63 ms with a Pentium IV 3GHz HT computer. They also proposed the Re-Pairing attack, which allows an attacker to force a victim's device to forget a paired device and then pair with the attacker's device using the cracked PIN. However, this attack may raise the user's suspicion, as the pairing request originates from a device that should already be paired, and user confirmation is needed to complete the process, making the attack less stealthy and practical.

SSP/LE Pairing Just Works (JW) model is designed for cases where at least one of the pairing devices has neither input nor output capabilities, or in cases where both devices lack input capabilities. In theory, a malicious device may stealthily pair with a victim device in JW model since this mode does not require any manual user confirmation. Although central devices like smartphones and laptops require user confirmation when pairing with other devices using the JW model to avoid silent pairing, such pop-up confirmation dialogs are not mandated by the Bluetooth standard. Furthermore, user confirmation is impossible on a NINO device, meaning silent pairing is still feasible. Since no information is

available for device verification in the JW model, users cannot distinguish an authentic pairing device from a forged one. The Bluetooth Core Specification also points out that the JW model provides no protection against man-in-the-middle (MITM) attacks. Additionally, the input/output capability exchange messages are not integrity-protected, allowing an attacker to force the victim device into pairing in JW model by falsifying these exchange messages. Haataja *et al.* [14,15] proposed the BT-Nino-MITM attack, exploiting the aforementioned vulnerability. The adversary clones the Bluetooth device addresses of both pairing devices. When the two victim devices are pairing, the forged devices, impersonating the victim devices to each other, send NINO capability messages during the I/O capability exchange to force both devices into using the JW model. This enables the attacker to silently pair a malicious device with the victim device, thereafter eavesdropping on communications and sending malicious or deceptive messages to the victim devices. In fact, numerous subsequent attacks (e.g., BlueDoor, BLUR, BadBluetooth, BlueRepli, etc.) utilize vulnerabilities in the JW model as part of their attack strategies.

SSP/LEP/SCP Number Comparison (NC) model is designed for cases where both pairing devices can display a 6-digit number and accept a “yes” or “no” response from a user. A user should compare the passkeys displayed on both devices and confirm the pairing requests if the passkeys are the same. In contrast, the SSP/LEP/SCP Passkey Entry (PE) model is designed for cases where one pairing device has input capability, while the other only has output capability. A user should enter the 6-digit passkey, displayed on one device, into the other, and then both devices check if the passkeys are equal. Sun *et al.* [17] pointed out that the MITM protection of the PE model will be compromised once devices reuse the passkey for pairing and provided a method to recover the passkey from pairing exchanges. Barnickel *et al.* [18] created Bluetrial to launch MITM attacks under passkey reuse conditions in the PE model. Claverie and Esteves [21] proposed the BlueMirror attack against the PE model. When two devices are pairing, a malicious device first impersonates the Responder and reflects all pairing messages to the Initiator to pass the authentication process. The 6-digit passkey can be recovered from the pairing exchanges. Then, the malicious device impersonates the Initiator and pairs with the Responder using the PE model with the recovered passkey. In this way, the malicious device can replace the Initiator to pair with the Responder. However, the Initiator under the BlueMirror attack cannot communicate with the Responder since they are not paired. The user easily becomes aware of the attack because of the availability anomaly, which limits the threat in practice.

Some researchers notice that certain pairing models have similar UIs, which may lead to user mistakes. Von Tschirschnitz *et al.* [19] pointed out that since both NC and PE are related to 6-digit passkeys for user verification, general users who are not familiar with the Bluetooth pairing mechanism may get confused when distinguishing between NC and PE. Exploiting this vulnerability, they proposed a MITM attack named the Method Confusion Attack. An attacker uses two malicious devices to impersonate two pairing devices, re-

spectively, by copying their Bluetooth device addresses. When the two victim devices are pairing, the attacker interferes with the pairing process and forces one victim device to pair with one of the malicious devices using the NC model, while the other uses the PE model simultaneously. Then the attacker asks the user to enter the passkey generated in the NC pairing process for the PE pairing process. If the user enters the correct passkey on one device and clicks “yes” on the other, both victim devices pair with the malicious devices, respectively, rather than with each other. The attacker can then eavesdrop and modify communications. They also performed a user study and found that most users thought their devices were securely paired using the PE model, and few users recognized irregularities during the attack. Claverie *et al.* [20] discovered more method-confusion-type vulnerabilities and proposed Pairing Confusion Attack. Von Tschirschnitz *et al.* [35] further evaluated the migration proposed by Shi *et al.* [23] and identified new vulnerabilities in previous solution proposals.

Vulnerability discoveries in the pairing protocols presented above rely on manual protocol analysis, which can be inefficient and less comprehensive. To automate protocol analysis and improve the efficiency of vulnerability discovery, formal analysis tools (e.g., ProVerif) have been introduced for Bluetooth pairing protocol analysis. Chang and Shmatikov [36] conducted a formal analysis of the SSP NC model with ProVerif and found that a Bluetooth device might not be capable of matching user confirmation with pairing sessions when multiple SSP sessions are executed concurrently. This issue causes the passkey for verification to potentially not belong to the session requiring authentication. Sethi *et al.* [22] formally analyzed the SSP protocol using ProVerif and proposed the Misbinding Attack against the NC model. During the pairing of two devices, an attacker uses a malicious device to impersonate the Responder, compromised with malware, to execute the pairing process with the Initiator in the NC model. The malicious device forwards all pairing communication from the Initiator to the Responder, and the attacker can obtain the NC passkey via the malware in the Responder. Then, the malicious device is authenticated by the Initiator using the passkey, successfully pairing with the Initiator. Note that since the Responder pairs with neither the Initiator nor the malicious device, the user may become aware of the attack due to the unavailability of the Responder’s Bluetooth connection.

Wu *et al.* [33] conducted a comprehensive formal analysis of pairing protocols in BC and BLE, rediscovering five known vulnerabilities/attacks, including Device Mis-Bonding (DMB) [37], Co-located App [38], BadBluetooth [39], BLESAs [25], and Method Confusion Attack [19], as well as a new attack named Cross Stack Illegal Access (CSIA), similar to BLUR [32]. Jangid *et al.* [16] conducted a formal analysis of the PE model, uncovering three known vulnerabilities, including Passkey Reusing [17], BlueMirror [21], and Method Confusion Attack [19], and two new attacks named Group Guessing Attack and Ghost Attack. The Group Guessing Attack, exploiting the vulnerability that the passkey generation is non-thread-safe and the same passkey is used in all concurrent sessions, allows an attacker to crack the passkey by pairing with the vic-

tim device using several devices (no more than 20) concurrently. Then the attacker can pair with the victim device using the recovered passkey. The Ghost Attack is a kind of MITM attack that exploits the vulnerability of a general user being unable to determine whether the device sending the authentication message is the device being paired. Assuming a laptop and a keyboard are already paired and not in use currently, an attacker establishes a new pairing relationship with the keyboard to obtain user input, usually causing the keyboard to forget the former pairing. The user's laptop has to re-enter the pairing process with the keyboard to establish connections. The attacker uses a malicious device impersonating the keyboard to respond to the pairing request and enter the passkey gained from the user input for pairing authentication. In this way, the attacker can eavesdrop on the user input and inject arbitrary keystrokes. Shi *et al.* [23] introduced a comprehensive formal analysis model of the BLE SC pairing protocol, including all association models in the Bluetooth Specification, to uncover vulnerabilities caused by the interplay between different association models. Their formal analysis confirms two existing attacks (Method Confusion Attack [19] and Key Negotiation Downgrade Attack [27]) and discovers a new attack called Keysize Confusion Attack, which causes invalid bonding by generating a key size conflict.

Some researchers extend the security model to discover new vulnerabilities in the Bluetooth pairing protocols. Troncoso and Hale [40] proposed the CYBORG model to formally analyze the security of human-device interaction during PE model pairing and argued that all versions of PE do not meet any level of CYBORG security. Yin [41] formalized the threat models of Bluetooth to clarify and bridge the gaps in protocol design and practical requirements of the original CYBORG model. He designed Attackers of Confidential- and Authenticated-extra channels to handle the settings in threats and protocol requirements. The analysis results indicate weaknesses and insecurities in SSP protocols.

B Vulnerabilities and Attacks on Authentication Protocols

Two paired Bluetooth devices utilize the shared key (e.g., the Link Key for BC) to authenticate each other when reconnecting. Note that the authentication protocols for BC are specified in the Bluetooth Core Specification and implemented in the Link Layer. In contrast, no dedicated authentication protocol for BLE exists in the Bluetooth Core Specification, allowing manufacturers to design application-level authentication protocols. Antonioli *et al.* [24] found that the Legacy Authentication protocol only provides unilateral authentication, with the device in the Master role always authenticating the one in the Slave role without being authenticated in return. Furthermore, the Master and Slave roles can be switched at any time after the paging process. Although the Secure Authentication protocol provides mutual authentication, it is only used if both devices support Secure Connections (SC). This means that authentication will be downgraded to Legacy Authentication if one of the devices does not support SC, even if they had previously paired using SC. They also proposed the BIAS attack, exploiting these vulnerabilities. An attacker can impersonate one of the paired devices and bypass authentication

from the other device without knowing the Link Key by switching their role to Master and declaring no support for SC, thus downgrading the authentication protocol version. However, the BIAS attacker cannot complete the connection procedure since communications after key length negotiation are encrypted with an encryption key derived from the Link Key. Wu *et al.* [25] conducted a formal analysis of the BLE reconnect process using ProVerif and uncovered two vulnerabilities. The first is that link-layer authentication/encryption is optional, and the second is that authentication can be circumvented in the reactive authentication model. They also discovered flaws in the authentication processes of iOS and Android, where communications are downgraded to plaintext rather than being aborted if the peripheral declares no Long Term Key (LTK). They proposed the BLES attack to impersonate the peripheral and provide spoofed data to the central device.

C Vulnerabilities and Attacks on Key Negotiation and Generation Protocols

Considering the diverse computing capacities of different Bluetooth devices, the Bluetooth standard specifies an encryption key length negotiation process to accommodate the encryption needs of devices with limited capabilities. Antonioli *et al.* [26,27] found that key length negotiation exchanges are neither confidentiality nor integrity protected, and the minimum key length is 1 byte for BC and 7 bytes for BLE. They proposed the KNOB attack, which involves downgrading the encryption key length by modifying the key length suggestions to the minimum value during the pairing/connecting process of two devices. This allows an attacker to brute-force the encryption key in a reasonable amount of time and eavesdrop on Bluetooth communications. Antonioli [28] also discovered that an attacker could reproduce the encryption key by reusing the authentication challenge's random number, leading to issues with forward and backward secrecy. He proposed the BLUFFS attacks, which exploit the flaws in encryption key reuse to compromise the forward and future secrecy of Bluetooth sessions. A BLUFFS attacker can impersonate either the Master or Slave device to spoof a target BC device or launch a MITM attack.

The key generation protocols have also drawn researchers' interest. Ryan [29] noticed that the generation of the Long Term Key (LTK) in LE Legacy Pairing is related to the Temporary Key (TK). However, the generation process of TK is insecure since TK is always 0 in the JW model and is a 6-digit number in the PE model. An attacker can easily crack TK and further compute LTK, with which they can launch MITM attacks. Landrock and Kjaersgaard [30] mentioned an invalid curve attack against SSP in a patent, where an attacker attempts to recover a device's ECDH private key by sending invalid coordinates and computing the key based on the victim's responses. The attacker can obtain the private key if it is not changed in time. The Bluetooth standard provides some methods to mitigate this vulnerability, and most implementations choose to update the private key after three failed pairing attempts. Biham and Neumann [31] found two vulnerabilities in the Bluetooth ECDH protocol: the practice of sending both x-coordinate and y-coordinate during the ECDH key exchange

and only authenticating the x-coordinate. They proposed the Fixed Coordinate Invalid Curve Attack, where an attacker can obtain the ECDH shared key and perform a MITM attack by forcing the shared key to a predictable value (e.g., ∞) by replacing the y-coordinate of the public key with 0. The attacker has a 50% probability of compromising the encryption key and another 50% probability of causing the pairing to terminate. Antonoli *et al.* [32] discovered vulnerabilities in Cross-Transport Key Derivation (CTKD). Bluetooth 4.2 and later versions allow dual-mode Bluetooth devices to derive the BLE Long Term Key from the BC Link Key and vice versa to avoid a second pairing process. However, two devices can execute the pairing process on the transport not currently in use, even if they are already paired on the other transport. Moreover, the key generated by the later pairing process will overwrite the existing shared key regardless of the security level/mode of the pairing. They proposed the BLUR attack by exploiting the CTKD vulnerabilities. An attacker impersonates one of the paired devices and attempts to pair with the other using the JW model on the transport that is not in use. After the pairing is completed, the shared key will be replaced by the attacker's key, and the attacker can establish malicious Bluetooth connections with the victim device.

D Vulnerabilities and Attacks in Miscellaneous Protocols

Besides vulnerabilities in Bluetooth security protocols, design flaws in other protocols can also impact the security of Bluetooth communications. Ai *et al.* [34] identified design flaws in the role-switch mechanism and profile management. They developed the Blacktooth Attack, as shown in Fig. 6, to bypass the authentication and authorization mechanisms. This attack allows the establishment of a malicious connection to control the victim's device in an entirely stealthy manner.

In particular, they identified a flaw in the role design of BC, termed Unfixed Master-Slave Roles (Vulnerability #1). According to the Bluetooth Core Specification, the device initiating the connection assumes the Master role, and the roles of Master and Slave can be switched at any time. This aspect of the protocol design implies that the roles are symmetric in link-layer communication. However, from the perspectives of security protocols and application functionalities, the devices are asymmetric. This inconsistency in protocol design could be exploited, potentially amplifying other threats. An attacker

can proactively send a connection request to assume the Master role and bypass user interaction. This vulnerability, combined with Unilateral Legacy Authentication [24] (Vulnerability #2) and Potential Low Encryption Key Entropy [26] (Vulnerability #3), could enable an attacker to establish a malicious connection with a victim device without user involvement. They also noted that the Bluetooth standard does not provide actionable suggestions regarding profile authentication and authorization. In practice, operating systems (OSes) allow profile connections that are inconsistent with the device's declared function without additional authentication (Vulnerability #4) and authorize new profile functionalities by default (Vulnerability #5). For instance, a headset could establish an HID connection typically used by keyboards, and users would not receive any notification or confirmation. By combining these vulnerabilities, they proposed the Blacktooth Attack. In this attack, an impersonator of a BC device previously paired with the victim device proactively sends a connection request to stealthily initiate the connection and assume the Master role. The attacker then bypasses victim authentication, shortens the key length, and adds sensitive profiles to escalate privileges. Consequently, the attacker can steal sensitive information and inject arbitrary commands into the victim's device.

Case study. Blacktooth Attack is effective against Bluetooth devices manufactured by the major vendors, including Apple, Google, Samsung, HUAWEI, HONOR, OnePlus, OPPO, Xiaomi, etc., running various operating systems, including Android, iOS, iPadOS, macOS, and HarmonyOS. A Blacktooth attacker can create malicious Bluetooth connections with the victim's device and steal private data and inject commands.

E Summary

The security analysis of Bluetooth protocols is a popular topic, with pairing protocol security analysis attracting the most attention. Generally speaking, the discovery of vulnerabilities and the updating of protocols exhibit a trend of alternating progression. On the one hand, the identified vulnerabilities are almost addressed in subsequent security protocol upgrades. On the other hand, vulnerabilities are still discovered in new version protocols.

Meanwhile, with the extensive adoption of automated protocol analysis tools, the efficiency and comprehensiveness of vulnerability discovery in Bluetooth protocols improve significantly. Automated protocol analysis is gradually replacing traditional manual protocol analysis as the primary research method for Bluetooth protocol security analysis now and in the future.

Of course, as protocol security analysis becomes more in-depth and the design of new protocol versions becomes more standardized, the number of potential Bluetooth protocol vulnerabilities will gradually decrease, and the popularity of related research is also likely to decline.

VI Implementation-level Vulnerabilities and Attacks

Implementation refers to the software and hardware products

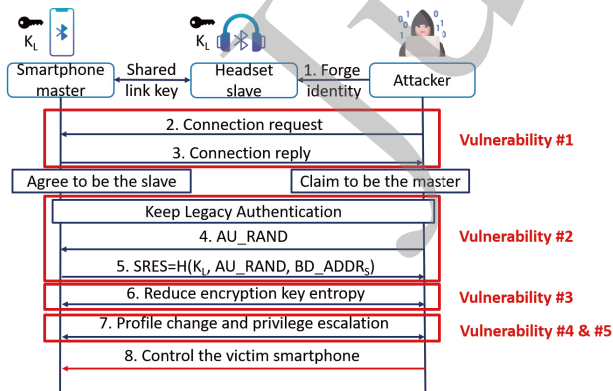


Fig. 6. The process of the Blacktooth attack

that satisfy Bluetooth standards. Since implementations vary among vendors and Bluetooth standard versions, implementation-level vulnerabilities typically exist in specific Bluetooth chips or within manufacturers' stacks/OSes. Furthermore, manufacturers, being familiar with their own implementations, can patch vulnerabilities in a timely manner without waiting for revisions to the Bluetooth standard. Therefore, the impact of implementation-level vulnerabilities and attacks is generally less significant than those at the protocol and algorithm levels. Implementation-level vulnerabilities can be further classified into logic-level flaws and binary-level flaws. Logic-level flaws refer to implementation errors that conflict with Bluetooth specifications or to unspecified implementations with design flaws. Binary-level flaws, on the other hand, pertain to code vulnerabilities in vendors' Bluetooth stacks, which may lead to issues such as buffer overflow, deadlock, etc. We have classified the implementation-level vulnerabilities and attacks as shown in Fig. 7 and summarized them in Table 5.

A Logic-level Vulnerabilities and Attacks

As soon as Bluetooth technology came into practical use, attacks against Bluetooth devices began emerging one after another. The first known Bluetooth attack, Bluejacking, was proposed by a Malaysian IT consultant between 2001 and 2003 [42]. A Bluejacking attacker sends unsolicited messages to Bluetooth-enabled devices such as mobile phones. This is

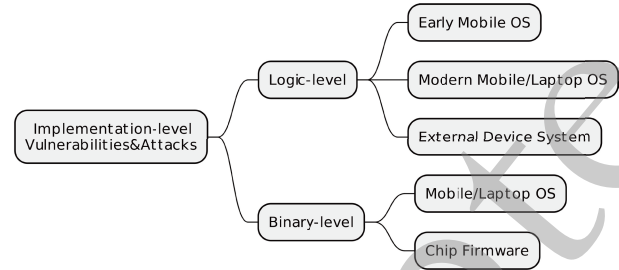


Fig. 7. Classification of Bluetooth implementation-level vulnerabilities and attacks

Table 5. Implementation-level vulnerabilities and attacks summary

Level	Attack Type	Vulnerability&Attack	Effect	Flaw Location	Affected Stack	Affected System/Firmware	Patched
Logic	Key negotiation and generation protocol vulnerability	Bluejacking [42]	Send Unsolicited Message	Communication Authorization	BC	Symbian	✓
		Bluesnarfing [43]	Illegal OPP Connection	OPP Authorization	BC	Symbian	✓
		Bluesnarfing++ [44]	Illegal OPP Connection	OPP Authorization	BC	Symbian	✓
		Bluebugging [45]	Send Arbitrary AT Command	AT Command Interface	BC	Symbian	✓
		HeloMoto [46]	Illegal HSP Connection	HSP Authentication	BC	Symbian	✓
		Device Mis-Bonding [37]	Unauthorized Device Access	App Bluetooth Permission	BC	Android	✓
		Co-located App [38]	Unauthorized Device Access	App Bluetooth Permission	BLE	Android	✓
		BadBluetooth [39]	Profile Privilege Escalation	Profile Authorization	BC	Android	×
		Blurepli [49]	Illegal PBAP/MAP Connection	PBAP/MAP Authorization	BC	Android	×
		SCO Downgrade [50]	Downgrade to Plaintext	SCO Mode Implementation	BLE	Android, Windows, macOS, iOS, Linux	✓
	Key Negotiation and Generation	BlueDoor [51]	Downgrade to Plaintext	Authentication&Encryption	BLE	IoT OSes	✓
		BLEDiff [52]	Security Bypass	Pairing Implementation	BLE	Android, IoT OSes	✓
		Bluesmack [47]	Ping of Death	L2CAP Ping Mechanism	BC	Symbian	✓
		BlueChop [48]	Disrupt Piconet	Piconet Connection	BC	Symbian	✓
		Deauthentication [53]	Reject Legal Connection	Connection Handling Process	BC&BLE	Android, Windows, IoT OSes	×
		BLEDiff [52]	Process Abort/Crash	Pairing Implementation	BLE	Android, IoT OSes	✓
		BlueBorne [54]	Data Leakage, RCE, MITM	8 Unique Implementation Flaws	BC	Android, iOS, Windows, Linux	✓
		BLEEDINGBIT [55]	RCE	Advertising Packet Handling	BLE	TI Chip Firmware	✓
Binary	Unauthorized Service Access	BleedingTooth [56]	RCE	Advertising Packet Handling	BLE	Linux	✓
		Sweyntooth [57]	Deadlock, Crash, RCE	18 Unique Implementation Flaws	BLE	12 Chip Firmwares	✓
		BrakTooth [58]	Deadlock, Crash, RCE	15 Unique Implementation Flaws	BC&BLE	16 Chip Firmwares	✓
		BlueFrag [59]	RCE	HCI Packet Reassembly	BC&BLE	Android	✓
		BTFuzzer [60]	Information Leakage	Boundary Check	BC	Android	✓
		VirtFuzz [61]	Crash	Multiple Connection Complete Events	BC&BLE	Linux	✓

possible because there is no permission control for sending and receiving messages; a device can receive messages via Bluetooth from any other Bluetooth device. Although Bluejacking messages themselves do not harm Bluetooth devices, they can be used for phishing attacks with forged messages or to annoy users with harmful intent. In 2003, Holtmann proposed the Bluesnarfing attack [43], which gains unauthorized access to Bluetooth devices by exploiting a firmware flaw in the OBEX Push Profile (OPP). Through this attack, an attacker can force an OPP connection to the victim's device and silently steal data, including contact lists, text messages, and files, as well as the international mobile equipment identity (IMEI). An advanced version of this, known as Bluesnarfing++ [44], enables an attacker to gain full read/write access to the filesystem of victim devices. In 2004, Laurie identified a firmware flaw in the AT command interface [45], enabling attackers to issue AT commands to a vulnerable device without the device owner's knowledge. This Bluebugging attack allows for reading/writing access to phonebook and SMS, making phone calls, and controlling other Bluetooth device features or services. Laurie also combined Bluesnarfing and Bluebugging to create a new attack called HeloMoto [46], which allows attackers to control victim devices via the Headset Profile (HSP) without authentication. During this period, various DoS attacks were also discovered. Bluesmack [47] is a "Ping of Death" attack that involves sending excessive L2CAP ping packets, while the BlueChop attack [48] disrupts an established piconet by spoofing and forcing a random Slave out of the piconet.

With the popularity of smartphones and laptops, the security of Bluetooth in operating systems, such as Android, iOS, Windows, and Linux, has attracted significant research interest. Some researchers uncover vulnerabilities in external device access control. Naveed *et al.* [37] discovered a vulnerability in the access control of Bluetooth external devices in Android. Specifically, this vulnerability lies in the Android security model, where access authorization to Bluetooth communication is at the channel level rather than the device level. This means one channel might contain several independent Bluetooth device connections. If there is no additional authentication between a Bluetooth external device and its official application, any app with permission for the Bluetooth channel can access all devices connected through this channel, even those it is not authorized to access. Exploiting this vulnerability, they proposed the external Device Mis-Bonding (DMB) attack, where a malicious app establishes a Bluetooth connection with a paired external device belonging to another app and steals sensitive data without authorization. This only requires BLUETOOTH and BLUETOOTH_ADMIN permissions. In the DMB attack, a cloned device replaces the real one by deleting the Link Key, forcing the smartphone to re-pair with the clone. When the official app tries to communicate with its device, it connects to the spoofing device instead and receives fake data. Since application-level authentication can defeat the DMB attack, they analyzed 68 Bluetooth-enabled apps and found that none implemented app-device authentication. They also developed a mechanism called Dabinder to mitigate this vulnerability. Sivakumaran and Blasco [38] introduced the Co-located App attack against BLE

devices, based on the DMB attack. This attack enables a malicious app to steal sensitive data from a connected BLE device with only BLUETOOTH permission. They developed a static analysis tool, BLECrypter, to examine over 18,900 BLE-enabled Android apps, discovering that more than 45% lacked BLE data protection implementation.

Some works discover permission management flaws in Bluetooth profiles within Android. Xu *et al.* [39] identified vulnerabilities in the authentication and authorization mechanisms of Bluetooth profiles in Android. They noted that during the pairing process, a user cannot see the profile list, and after pairing, only some profile service permissions require manual user confirmation while others are granted automatically. Furthermore, Bluetooth devices can arbitrarily change or add profiles without user permission, gaining default authorization. They also highlighted that the Code of Device (CoD) of Bluetooth devices, which identifies device types and presents corresponding icons in the UI, can be modified. Exploiting these vulnerabilities, they proposed the BadBluetooth attack. In this attack, a preinstalled malicious app pairs with a malicious Bluetooth device in Just Works mode to avoid user consent. The malicious device then adds attack-related profiles (e.g., HID, PAN, etc.) and reconnects to the victim's smartphone for privilege escalation. They also developed a protection mechanism to mitigate the BadBluetooth attack. Xu and Xin [49] discovered that the Bluetooth device address lists for PBAP and MAP permissions are correlated in Android, meaning a device with PBAP permission automatically obtains MAP permission, and vice versa. They introduced the BlueRepli attack, where a malicious device can silently read/send text messages, requiring only PBAP permission.

Implementation flaws in OSes can also lead to a downgrade in Bluetooth security. Zhang *et al.* [50] analyzed the security of the Bluetooth Secure Connections Only (SCO) mode implementation in various operating systems, including Android. According to the Bluetooth Core Specification, the SCO mode only requires the BLE peripheral to authenticate the BLE central (typically a smartphone and the connection initiator), but reverse authentication is not mandatory, nor is the programming framework for the BLE SCO mode specified. They found that if the BLE SCO mode implementation of the initiator cannot properly handle SCO initiation, status management, error handling, and bond management, attackers can downgrade the BLE pairing protocols to an insecure mode without users being aware. They exploited these implementation flaws in Android to launch eight types of SCO downgrade attacks, including data injection, information stealing, DoS, MITM, etc. They evaluated these attacks on five Android smartphones and other major OS implementations, including Linux, Windows, macOS, and iOS, finding all vulnerable. They also provided a defense prototype based on the Android Open Source Project (AOSP). Classen and Hollick [62] noted that current user warnings in major OSes about Bluetooth authentication and encryption failure are unclear, leading to users being unaware of potential attacks. Lounis [53] observed that some Bluetooth devices disconnect from existing connections when receiving new connection requests, leading to deauthentication. An adversary can exploit this by impersonating a Bluetooth device and attempting to

connect to the victim device, causing the disconnection from the device being impersonated. Repeated attacks could make this deauthentication permanent. For other devices that reject new connections from a connected device, an attacker's device can establish a Bluetooth connection before the legitimate device, causing the victim device to reject the legitimate connection request.

Some researchers focus on the security of the Bluetooth stack in external devices. Wang *et al.* [51] discovered that BLE device authentication is implemented in the application layer and is decoupled from link-layer encryption. This means that the authentication process can still function even if the connection is downgraded to plaintext. They also observed that BLE central devices generally disable encryption rather than abort the connection when they receive an encryption rejection from a BLE peripheral. Exploiting these vulnerabilities, they proposed the BlueDoor attack. In this attack, the perpetrator impersonates the central device to pair with the peripheral in 'Just Works' mode and simultaneously impersonates the peripheral to connect with the central device and disable encryption. The attacker gains the peripheral's trust by forwarding authentication messages from the central device and then steals sensitive data from the peripheral. In a separate study, Wen *et al.* [63] developed an analysis tool called FirmXRay, designed to detect link-layer vulnerabilities in BLE SoCs produced by Nordic and Texas Instruments (TI). FirmXRay targets three types of vulnerabilities: static addresses/UUIDs (leading to identity tracing), "Just Works" pairing (enabling active MITM attacks), and LE Legacy Pairing (allowing passive MITM attacks). Testing 793 unique firmware samples with FirmXRay, they found that 98.1% of the devices used static Bluetooth addresses, 71.5% used "Just Works" pairing, and 98.5% employed LE Legacy Pairing.

An automatic implementation analysis tool is also developed to identify implementation flaws that conflict with the Bluetooth Specification. Karim *et al.* [52] developed BLEDiff, an automated, scalable, property-agnostic, and black-box protocol noncompliance checking framework that can analyze and uncover flaws in BLE protocol implementations. BLEDiff can extract the protocol FSM of BLE implementations automatically using the active automata learning method with a divide-and-conquer approach. They evaluated BLEDiff with 25 different off-the-shelf BLE devices, revealing 13 different deviant behaviors with 10 exploitable attacks.

B Binary-level Vulnerabilities and Attacks

Armis Labs published an attack vector called BlueBorne [54], exploiting 8 unique vulnerabilities in the Bluetooth Classic stack implementation of major OSes, including Android, Linux, iOS, and Windows. BlueBorne can cause information leakages, remote code execution (RCE), stack overflows, and MITM attacks. They further revealed the BLEEDINGBIT [55], involving 2 Remote Code Execution (RCE) flaws in the BLE firmware of TI SoCs. Nguyen [56] disclosed several zero-click vulnerabilities in the BlueZ BLE implementation under the name of BleedingTooth, causing RCE via BLE advertising packets.

Instead of manual analysis, fuzzing is introduced to Bluetooth stack implementation analysis for automatic vul-

nerability hunting [61, 64]. Kovah [65] identified 4 implementation flaws, leading to RCE, in the BLE stack of chips produced by Silicon Labs and TI using a homemade fuzzer. Karim *et al.* [66] published a dynamic analysis framework called ATFuzzer for automatic analysis of the AT interface in Android, and 4 invalid Bluetooth AT grammars are identified with the evaluation on 10 Android smartphones. Garbelini *et al.* [57] revealed a collection under the name of Sweyntooth consisting of 18 various vulnerabilities in multiple BLE SoCs produced by different manufacturers. These vulnerabilities may cause crashes, deadlocks, and buffer overflows with potential RCE, or even completely bypass security mechanisms. Pairing is not required to trigger the majority of the flaws. They also developed an over-the-air fuzzing framework for automatic vulnerability discovery in arbitrary BC stacks and found 18 new implementation flaws called BrakTooth [58] by fuzzing 13 BC devices, resulting in deadlocks, crashes, and RCE. Furthermore, they extended their fuzzer to analyze BLE stacks and discovered 2 implementation flaws. Ruge *et al.* [59] developed a fuzzing framework based on advanced firmware emulation named Frankenstein and uncovered 2 zero-click RCE vulnerabilities in BC stacks and 1 in BLE stacks. They also disclosed BlueFrag, which is an RCE in Android HCI implementation. Jang *et al.* [60] utilized BTFuzzer to discover 2 new implementation flaws in Android.

Case study. BlueBorne potentially affects billions of smartphones, laptops, and IoT devices. A BlueBorne attacker can implant malicious code into the victim's device via a Bluetooth connection without the user's interaction, enabling remote control of the device.

C Summary

For implementation-level security issues, the most attention from researchers is drawn to privilege escalation and remote execution vulnerabilities. These types of vulnerabilities pose high risks and significant threats to victims. Similar to the development trends in protocol security analysis, automated analysis tools such as fuzzing have gradually been employed for vulnerability mining in Bluetooth implementations, greatly enhancing the efficiency of vulnerability discovery.

However, due to the complexity of Bluetooth implementations and the variability among devices, current fuzzing tools only cover the implementation of some core protocols. The security of implementations related to common profiles at the host layer, among others, has not been sufficiently investigated. Additionally, most flaws discovered through fuzzing tend to lead to system crashes, while the discovery rate of high-value vulnerabilities, such as those enabling privilege escalation or remote execution, remains relatively low.

VII Application-level Vulnerabilities and Attacks

Application refers to the use of Bluetooth technology in commercial products. Even in the absence of vulnerabilities in Bluetooth protocols and implementations, inappropriate methods of applying Bluetooth can also lead to security issues in Bluetooth-enabled devices.

Some of the vulnerabilities lead to sensitive information

leakage. The Bluetooth connections of Logitech mice are not encrypted, since it is believed that mouse movements do not leak useful information. Pan *et al.* [67] argued that sensitive information (e.g., passwords) could be recovered from mouse movement data contained in unencrypted Bluetooth packets during software keyboard clicks. Jung *et al.* [68] found vulnerabilities in Bluetooth-based proximity authentication implementations in Windows Dynamic Lock and Android Smart Lock. An attacker can prevent a Windows computer or Android smartphone from locking by establishing an insecure connection between the device and a forged trusted device and illegally use the device while the user is away. Casagrande *et al.* [69] analyzed various fitness trackers produced by Xiaomi and Fitbit and identified several vulnerabilities due to misuse of BLE security mechanisms (e.g., no/partial communication encryption). They published an automatic tool named Breakmi to launch impersonating/eavesdropping/MITM attacks on fitness trackers, allowing eavesdropping and data manipulation. Liu *et al.* [70] utilized GATTacker [71] to launch an MITM attack to intercept Bluetooth traffic from wearable health devices and obtain sensitive data. Zhang *et al.* [72] pointed out that WiFi credentials can be extracted from BLE communications, resulting in a malicious remote connection between the smart TV and the attacker's controller.

Other vulnerabilities can pose a threat to the property security. Xie *et al.* [73] analyzed the keyless entry system of Tesla and found that BLE link-layer encryption is disabled, enabling attackers to launch GATT-based BLE relay attacks to open Tesla doors. NCC Group [74] also developed a tool for Tesla key relay attack.

Design and implementation flaws in device-finding applications also threaten Bluetooth security and privacy. Yu *et al.* [75] analyzed Samsung's Offline Finding (OF) protocol and identified several flaws, leading to unwanted device tracking and location privacy leakage. Chen *et al.* [76] presented an attack called nRootTag to transform a computer into a trackable "AirTags" without root privileges, which enables an attacker to trace the victim's device infected with Torjan.

An automation tool is also used to discover application-level vulnerabilities. Sivakumaran *et al.* [77] proposed a framework, called Ble-Guude, to identify mobile app-centric security issues. They evaluated Ble-Guude with 17,243 free BLE-enabled Android APKs from the Google Play Store and uncovered that more than 70% of these apps are vulnerable.

Summary. As the number of applications utilizing Bluetooth continues to increase, application-level security analysis is increasingly attracting researchers' attention. However, due to the significant functional diversities among various applications, security research tends to be fragmented, and attacks often lack general applicability.

Overall, the root cause of most application-level security issues is that developers do not follow the recommendations of the Bluetooth Core Specification, leading to insecure usage of Bluetooth functionalities. Considering that more applications will integrate Bluetooth features, efficiently and comprehensively identifying application-level security issues is a valuable research direction.

VIII Privacy-related Vulnerabilities and Attacks

Although privacy protection is considered in Bluetooth specifications and implementations, adversaries can still obtain personal information by exploiting flaws in privacy mechanisms or launching side-channel attacks.

Bluetooth communication sniffing is a significant step for the majority of attacks since the information required to launch attacks (e.g., Bluetooth device address) is contained in the packets. However, because of frequency hopping and data whitening mechanisms in Bluetooth, it is difficult to eavesdrop on Bluetooth communications. Spill [78] presented a sniffing method on Bluetooth Classic communications called BlueSniff to overcome those hurdles. The author also provided a mechanism to determine the Master device address in the undiscoverable state, which can be utilized in device tracing. Albazraq *et al.* [79] developed BlueEar, a Bluetooth Classic traffic sniffer, by predicting adaptive hopping behavior, reaching a packet capture rate of more than 90%.

To avoid device tracking, device address anonymization and randomization mechanisms are developed in Bluetooth technology. However, Bluetooth devices can still be traced under side-channel attacks. Some attacks uncover the Bluetooth device address to trace the victim device. Cominelli *et al.* [80] published a robust de-anonymization technique that hinges on the apparently benign information leaking from frame encoding to infer a piconet's clock, hopping sequence, and ultimately the Upper Address Part (UAP) of the Master device's physical address, which is never exchanged in clear. Building upon the findings of [78] and [80], Tucker *et al.* [81] proposed the Blue's Clues attack, providing a direct, non-disruptive method to fully extract the unique and permanent Bluetooth addresses from target devices in the undiscoverable state, utilizing the Bluetooth Link Manager Protocol (LMP) messages to extract identifiable information. Attackers using the Blue's Clues method can uniquely identify Bluetooth Classic devices in less than 10 seconds on average. Ludant *et al.* [82] proposed a method for linking BLE advertising packets with BC frames, which can be utilized to uncover the Bluetooth device address. This allows attackers to identify specific BLE devices using the uncovered device addresses. Zhang *et al.* [83] argued that the BLE allowlist mechanism provides a side channel for device tracking, as the response behavior differs between devices that are on the allowlist and those that are not. They further found flaws in the current Bluetooth device address randomization mechanism specified in the Bluetooth protocol. This flaw allows a replay attack, where an adversary can probe whether a targeted device responds based on its allowlist by replaying a sniffed Bluetooth device address. Locatelli *et al.* [84] presented a Bluetooth Low Energy Nodes Detect, Enquire, (and) Recognition (BLENDER) framework to enumerate and fingerprint BLE devices for crowd monitoring and recognition purposes, enabling the association of BLE randomized addresses with actual devices. Wu *et al.* [85] created a formal model for BLE untraceability and identified 4 previously unknown issues, which enables 8 active or passive tracking attacks.

Physical-layer fingerprints are also utilized to obtain the device track. Givvehchian *et al.* [86] conducted an evaluation of BLE device identification performance using physical-layer fingerprints in real-world scenarios, finding that even though physical-layer fingerprints can be utilized to track BLE devices, similar fingerprints and temperature variations can significantly reduce identification accuracy. Li *et al.* [87] introduced a physical-layer fingerprint called Transient Dynamic Fingerprint (TDF), which remains stable under various thermal conditions. They further proposed BTrack based on TDF to trace Bluetooth devices.

Summary. Regarding privacy-related security issues, existing research primarily focuses on Bluetooth device identification and tracking. Bluetooth packet sniffing serves as an essential technical foundation for device tracking, while device address de-anonymization and physical-layer fingerprints are currently the two main identification and tracking methods.

Compared to device addresses, the physical characteristics of Bluetooth devices are more difficult to alter. This means that even if a perfectly secure address anonymization mechanism exists, attackers may still be able to identify devices through their physical-layer features. Clearly, research on device identification and tracking will continue.

IX Future Directions

Through our systematic review, we observed the evolution of research on Bluetooth vulnerabilities and attacks. Initially, investigations relied on manual analysis of protocols and implementations, guided by researchers' expertise. Over time, the field has progressed to a stage where formal analysis and fuzz testing are employed for batch vulnerability discovery. This represents a significant improvement in the efficiency and comprehensiveness of Bluetooth security research. Despite these advancements, there is still a considerable gap in Bluetooth security research. In this context, we identify potential future research directions for the analysis of Bluetooth vulnerabilities and attacks.

Profile security evaluation. Existing Bluetooth security analysis mainly concentrates on security issues of the Bluetooth Core Specification, including protocol design and stack implementation. However, the Bluetooth research community still pays insufficient attention to profile security. Note that Bluetooth SIG has approved nearly a hundred profiles, involving sensitive information and permissions. One research direction is to construct a formal analysis framework for profiles, which can utilize formal analysis tools like ProVerif. The other research direction is to develop fuzzing tools for profile implementation security analysis.

Security vacuum identification between protocol design and stack implementation. The current Bluetooth standard does not clearly define the security responsibilities of Bluetooth SIG and device manufacturers. Specifically, it does not provide explicit guidance on which parts of Bluetooth security should be specified by the Bluetooth Specification and which parts should be implemented by the manufacturers. The ambiguity leads to security vacuums in the Bluetooth ecosystem. For example, regarding profile authentication and authorization, the Bluetooth Core Specification only briefly

mentions that authentication and authorization are required during profile configuration, without detailed protocol designs. We have communicated with Bluetooth SIG and various manufacturers about this issue. Bluetooth SIG indicates that the functions for fine-grained profile authentication and authorization are generally considered vendor-specific extensions, while manufacturers claim that their implementations fully comply with the requirements of the Bluetooth standard. Since the Bluetooth Specification does not provide specific protocol requirements for profile authentication and authorization, manufacturers do not implement these features either. We believe that such security vacuums are not limited to the scenario described above. Therefore, more exploration of the security vacuum between protocol design and stack implementation is necessary.

Better Bluetooth UI design for security and its evaluation. Apart from vulnerabilities in Bluetooth design and implementation, user's mistake is also a significant factor leading to successful attacks. On the one hand, these mistakes are derived from the general lack of understanding among users regarding Bluetooth, leading to uninformed actions. On the other hand, incomplete or misleading information displayed in the UI can further contribute to misoperations. To address this issue, one possible solution is to design more user-friendly interfaces that help users avoid mistakes. However, the key challenge lies in how to effectively evaluate the security enhancement brought by UI design. How to conduct systematic user studies properly is worth exploring.

X Conclusions

We conducted a comprehensive survey of security vulnerabilities and attacks on Bluetooth technology, including both Bluetooth Classic and Bluetooth Low Energy. We provided an overview of the development of Bluetooth technology and offered an introduction to its architecture and security mechanisms. Subsequently, we systematically classified and described the vulnerabilities and attacks on Bluetooth according to our taxonomy. Our survey can serve as a valuable resource for researchers and vendors. It enables them to systematically comprehend the various flaws present across different levels of Bluetooth functionality and learn about the latest methods and tools used in the vulnerability analysis of Bluetooth systems. By discovering potential vulnerabilities and revising Bluetooth standards and implementations accordingly, Bluetooth SIG and device vendors can help improve the overall security of Bluetooth technology. We also provided promising research directions for Bluetooth security. To this end, we recommend that the Bluetooth SIG and Bluetooth-related manufacturers enhance their cooperation in order to avoid security management gaps.

Acknowledgments

This work is supported in part by the National Natural Science Foundation of China under grants No. 62372425, No. 61972371, and No. U19B2023, and Youth Innovation Promotion Association of the Chinese Academy of Sciences (CAS) under grant No. Y202093.

Biographies

Mingrui Ai received his bachelor's degree from the School of Cyber Science and Technology, University of Science and Technology of China in 2020. He is currently a Ph.D. student in the School of Cyber Science and Technology. His research interests include Bluetooth security and blockchain technology.

Kaiping Xue (M'09-SM'15) received the bachelor's degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003 and received the Ph.D degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. From May 2012 to May 2013, he was a postdoctoral researcher with the Department of Electrical and Computer Engineering, University of Florida. Currently, he is a Professor in the School of Cyber Science and Technology, USTC. He is also the director of Network and Information Center, USTC. His research interests include next-generation Internet architecture design, transmission optimization and network security. His work won best paper awards in IEEE MSN 2017 and IEEE HotICN 2019, IEEE ICC 2025, the Best Paper Honorable Mention in ACM CCS 2022, the Best Paper Runner-Up Award in IEEE MASS 2018, and the best track paper in MSN 2020. He serves on the Editorial Board of several journals, including the IEEE Transactions on Information Forensics and Security, the IEEE Transactions on Dependable and Secure Computing, the IEEE Transactions on Wireless Communications, and the IEEE Transactions on Network and Service Management. He has served as a Guest Editor for many reputed journals/magazines, and he has also served as a track/symposium chair for some reputed conferences. He is an IET Fellow and an IEEE Senior Member.

Lutong Chen received the bachelor's degree and the Ph.D degree from the School of Cyber Science and Technology, University of Science and Technology of China in 2020 and 2024, respectively. He is currently a Engineer in the Network and Information Center, USTC. His research interests include quantum networking and network security.

Xuanbo Huang received the bachelor's degree in information security from the School of Cyber Science and Technology, University of Science and Technology of China (USTC), Hefei, in 2020. He is currently working toward the Ph.D. degree in information security from the University of Science and Technology of China. His research interests include security issues in Software-Defined Networking and Distributed Denial of Service attacks.

David S.L. Wei (SM'07) received the Ph.D. degree in Computer and Information Science from the University of Pennsylvania in 1991. From May 1993 to August 1997 he was on the Faculty of Computer Science and Engineering at the University of Aizu, Japan (as an Associate Professor and then a Professor). He has authored and co-authored more than 140 technical papers in various archival journals and conference proceedings. He is currently a Professor with the Computer and Information Science Department at Fordham University. He was a lead guest editor or a guest editor for several special issues in the IEEE Journal on Selected Areas in Communications, the IEEE Transactions on Cloud Computing, and the IEEE Transactions on Big Data. He also served as an Associate Editor of IEEE Transactions on Cloud Computing, 2014-2018, an editor of IEEE J-SAC for the Series on Network Softwarization & Enablers, 2018-2020, and an Associate Editor of Journal of Circuits, Systems and Computers, 2013-2018. Dr. Wei is the recipient of IEEE Region 1 Technological Innovation Award (Academic), 2020, for contributions to information security in wireless and satellite communications and cyber-physical systems. He is a member of ACM and AAAS, and is a life senior member of IEEE, IEEE Computer Society, and IEEE Communications Society. Currently, Dr. Wei focuses his research efforts on cloud and edge computing, cybersecurity, and quantum computing and communications.

References

- [1] Bluetooth SIG. Bluetooth core specification v6.0. **2024**. <https://www.bluetooth.com/zh-cn/specifications/specs/core60-html/>, Accessed April 01, 2025.
- [2] Kissel R. Glossary of key information security terms. **2013**. <https://nvlpubs.nist.gov/nistpubs/ir/2013/NIST.IR.7298r2.pdf>. Accessed April 01, 2025.
- [3] The Hacker News. <https://thehackernews.com/>. Accessed April 01, 2025.
- [4] Padgett J, Scarfone K, Chen L, et al. Guide to Bluetooth Security, Special Publication (NIST SP). Gaithersburg, MD, USA: National Institute of Standards and Technology, **2017**, 800: 121.
- [5] Hermelin M, Nyberg K. Correlation properties of the bluetooth combiner. In: Song J, editor. Information Security and Cryptology-ICISC'99. Berlin, Heidelberg: Springer, **2000**: 17-29.
- [6] Fluhrer S, Lucks S. Analysis of the E_0 encryption system. In: Vaudenay S, Youssef A M, editors. Selected Areas in Cryptography-SAC 2001. Berlin, Heidelberg: Springer, **2001**: 38-48.
- [7] Krause M. Bdd-based cryptanalysis of keystream generators. In: Knudsen L R, editor. Advances in Cryptology — EUROCRYPT 2002. Berlin, Heidelberg: Springer, 2002: 222-237.
- [8] Lu Y, Vaudenay S. Cryptanalysis of bluetooth keystream generator two-level E_0 . In: Lee P J, editor. Advances in Cryptology-ASIACRYPT 2004. Berlin, Heidelberg: Springer, **2004**: 483-499.
- [9] Lu Y, Vaudenay S. Faster correlation attack on bluetooth keystream generator E_0 . In: Franklin M, editor. Advances in Cryptology — CRYPTO 2004. Berlin, Heidelberg: Springer, **2004**: 407-425.
- [10] Lu Y, Meier W, Vaudenay S. The conditional correlation attack: A practical attack on bluetooth encryption. In: Shoup V, editor. Advances in Cryptology-CRYPTO 2005. Berlin, Heidelberg: Springer, **2005**: 97-117.
- [11] Zhang B, Xu C, Feng D. Real time cryptanalysis of bluetooth encryption with condition masking. In: Canetti R, Garay J A, editors. Advances in Cryptology-CRYPTO 2013. Berlin, Heidelberg: Springer, **2013**: 165-182.
- [12] Zhang B, Xu C, Feng D. Practical cryptanalysis of bluetooth encryption with condition masking. *Journal of Cryptology*, **2018**, 31: 394-433.
- [13] Shaked Y, Wool A. Cracking the bluetooth PIN. In: Proceedings of the 3rd international conference on mobile systems, applications, and services. New York: ACM, **2005**: 39-50.
- [14] Hypponen K, Haataja K M J. "Nino" man-in-the-middle attack on bluetooth secure simple pairing. In: 2007 3rd IEEE/IFIP international conference in central Asia on Internet. Tashkent, Uzbekistan: IEEE, **2007**: 1-5.
- [15] Haataja K, Toivanen P. Two practical man-in-the-middle attacks on Bluetooth secure simple pairing and countermeasures. *IEEE Transactions on Wireless Communications*, **2010**, 9: 384-392.
- [16] Jangid M K, Zhang Y, Lin Z. Extrapolating formal analysis to uncover attacks in bluetooth passkey entry pairing. In: Proceedings 2023 Network and Distributed System Security Symposium, San Diego, CA, USA: ISOC, **2023**: 1-18.
- [17] Sun D Z, Mu Y, Susilo W. Man-in-the-middle attacks on Secure Simple Pairing in Bluetooth standard V5.0 and its countermeasure. *Personal and Ubiquitous Computing*, **2018**, 22: 55-67.
- [18] Barnickel J, Wang J, Meyer U. Implementing an attack on bluetooth 2.1+ secure simple pairing in passkey entry mode. In: 2012 IEEE 11th international conference on trust, security and privacy in computing and communications. Liverpool, UK: IEEE, **2012**: 17-24.
- [19] von Tschirschnitz M, Peuckert L, Franzen F, et al. Method confusion attack on bluetooth pairing. In: 2021 IEEE symposium on security and privacy (SP). San Francisco, CA, USA: IEEE, **2021**: 1332-1347.
- [20] Claverie T, Avoine G, Delaune S, et al. Tamarin-based analysis of Bluetooth uncovers two practical pairing confusion attacks. In:

- Tsudik G, Conti M, Liang K, editors. Computer security—ESORICS 2023. Cham, Switzerland: Springer, **2024**: 100–119.
- [21] Claverie T, Esteves J L. BlueMirror: Reflections on bluetooth pairing and provisioning protocols. In: 2021 IEEE security and privacy workshops (SPW). San Francisco, CA, USA: IEEE, **2021**: 339–351.
- [22] Sethi M, Peltonen A, Aura T. Misbinding attacks on secure device pairing and bootstrapping. In: Proceedings of the 2019 ACM Asia conference on computer and communications security. New York, NY, USA: ACM, **2019**: 453–464.
- [23] Shi M, Chen J, He K, et al. Formal analysis and patching of BLE-SC pairing. In: Proceedings of the 32nd USENIX Conference on Security Symposium, Anaheim, CA, USA: USENIX Association, **2023**: 37–52.
- [24] Antonioli D, Tippenhauer N O, Rasmussen K. BIAS: Bluetooth impersonation AttackS. In: 2020 IEEE symposium on security and privacy (SP). San Francisco, CA, USA: IEEE, **2020**: 549–562.
- [25] Wu J, Nan Y, Kumar V, et al. BLESAs: Spoofing attacks against reconnections in bluetooth low energy. In: Proceedings of the 14th USENIX Conference on Offensive Technologies, Anaheim, CA, USA: USENIX Association, **2020**: 8.
- [26] Antonioli D, Tippenhauer N O, Rasmussen K B. The KNOB is broken: Exploiting low entropy in the encryption key negotiation of bluetooth BR/EDR. In: Proceedings of the 28th USENIX Conference on Security Symposium. Santa Clara, CA, USA: USENIX Association, **2019**: 1047–1061.
- [27] Antonioli D, Tippenhauer N O, Rasmussen K. Key negotiation downgrade attacks on bluetooth and bluetooth low energy. *ACM Transactions on Privacy and Security*, **2020**, 23: 14.
- [28] Antonioli D. BLUFFS: Bluetooth forward and future secrecy attacks and defenses. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: ACM, **2023**: 636–650.
- [29] Ryan M. Bluetooth: With low energy comes low security. In: Proceedings of the 7th USENIX Workshop on Offensive Technologies (WOOT '13). Washington, D. C. , USA: USENIX Association, **2013**: 1–7.
- [30] Landrock P, Kjaersgaard J U. Protecting against security attack (U. S. Patent No. 8, 077, 866). **2011**.
- [31] Biham E, Neumann L. Breaking the bluetooth pairing—the fixed coordinate invalid curve attack. In: Paterson K, Stebila D, editors. Selected Areas in Cryptography—SAC 2019. Cham: Springer, **2020**: 250–273.
- [32] Antonioli D, Tippenhauer N O, Rasmussen K, et al. BLURtooth: Exploiting cross-transport key derivation in bluetooth classic and bluetooth low energy. In: Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security. New York, NY, USA: ACM, **2022**: 196–207.
- [33] Wu J, Wu R, Xu D, et al. Formal model-driven discovery of bluetooth protocol design vulnerabilities. In: 2022 IEEE symposium on security and privacy (SP). San Francisco, CA, USA: IEEE, **2022**: 2285–2303.
- [34] Ai M, Xue K, Luo B, et al. Blacktooth: Breaking through the defense of bluetooth in silence. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. New York, NY, USA: ACM, **2022**: 55–68.
- [35] von Tschirschnitz M, Peuckert L, Buhl M, et al. Rediscovering method confusion in proposed security fixes for bluetooth. In: Proceedings 2025 Network and Distributed System Security Symposium. San Diego, CA, USA: Internet Society, **2025**: 1–20.
- [36] Chang R, Shmatikov V. Formal analysis of authentication in bluetooth device pairing. In: Proceedings of the LICS/ICALP Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis. Wroclaw, Poland: ICALP, **2007**: 45–62.
- [37] Naveed M, Zhou X, Demetriou S, et al. Inside job: Understanding and mitigating the threat of external device mis-binding on Android. In: Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS). San Diego, CA, USA: Internet Society (ISOC), **2014**.
- [38] Sivakumaran P, Blasco J. A study of the feasibility of co-located app attacks against BLE and a large-scale analysis of the current application-layer security landscape. In: Proceedings of the 28th USENIX Security Symposium. Santa Clara, CA, USA: USENIX Association, **2019**: 18.
- [39] Xu F, Diao W, Li Z, et al. BadBluetooth: Breaking Android security mechanisms via malicious bluetooth peripherals. In: Proceedings 2019 Network and Distributed System Security Symposium. San Diego, CA, USA: Internet Society, **2019**.
- [40] Troncoso M, Hale B. The bluetooth CYBORG: Analysis of the full human-machine passkey entry AKE protocol. In: Proceedings 2021 Network and Distributed System Security Symposium. Online: Internet Society, **2021**.
- [41] Yin H. Security analysis of bluetooth secure simple pairing protocols with extended threat model. *Journal of Information Security and Applications*, **2023**, 72: 103385.
- [42] Bjerkebanen. Bluejacking. **2002**. <https://www.esato.com/board/viewtopic.php?topic=8483&forum=2>. Accessed: April 01, 2025.
- [43] Laurie A, Holtmann M. BlueSnarfing. **2003**. <https://trifinite.org/stuff/bluesnarf/>. Accessed: April 01, 2025.
- [44] Laurie A, Holtmann M, Herfurt M. BlueSnarfing++. **20025**. <https://trifinite.org/stuff/bluesnarfpp/>. Accessed: April 01, 2025.
- [45] Herfurt M. BlueBugging. **2004**. <https://trifinite.org/stuff/bluebug/>. Accessed: April 01, 2025.
- [46] Laurie A. HeloMoto. **2005**. <https://trifinite.org/stuff/helomoto/>. Accessed: April 01, 2025.
- [47] Laurie A, Holtmann M, Herfurt M. BlueSmack. **2004**. <https://trifinite.org/stuff/bluesmack/>. Accessed: April 01, 2025.
- [48] Laurie A, Holtmann M, Herfurt M. BlueChop. **2006**. <https://trifinite.org/stuff/bluechop/>. Accessed: April 01, 2025.
- [49] Xu S, Xin X. Stealthily access your Android phones: Bypass the bluetooth authentication. **2020**. <https://www.blackhat.com/us-20/briefings/schedule/index.html>. Accessed: April 01, 2025.
- [50] Zhang Y, Weng J, Dey R, et al. Breaking secure pairing of bluetooth low energy using downgrade attacks. In: Proceedings of the 29th USENIX conference on security symposium. Online: USENIX Association, **2020**: 37–54.
- [51] Wang J, Hu F, Zhou Y, et al. BlueDoor: Breaking the secure information flow via BLE vulnerability. In: Proceedings of the 18th international conference on mobile systems, applications, and services. New York, NY, USA: ACM, **2020**: 286–298.
- [52] Karim I, Al Ishtiaq A, Hussain S R, et al. BLEDiff: Scalable and property-agnostic noncompliance checking for BLE implementations. In: 2023 IEEE Symposium on Security and Privacy (SP). San Francisco, CA, USA: IEEE, **2023**: 3209–3227.
- [53] Lounis K. Cut it: Deauthentication attack on bluetooth. In: 2021 14th International Conference on Security of Information and Networks (SIN). Edinburgh, United Kingdom: IEEE, **2021**: 1–8.
- [54] Armis. Blueborne. **2017**. <https://armis.com/research/blueborne/>. Accessed: April 01, 2025.
- [55] Armis. Bleedingbit. **2019**. <https://armis.com/research/bleedingbit/>. Accessed: April 01, 2025.
- [56] Nguyen A. BleedingTooth: linux bluetooth zero-click remote code execution. Google Security Research, **2020**.
- [57] Garbelini M E, Wang C, Chattopadhyay S, et al. SweynTooth: Unleashing mayhem over bluetooth low energy. In: 2020 USENIX Annual Technical Conference. San Diego, CA, USA: USENIX Association, **2020**: 911–925.
- [58] Garbelini M E, Bedi V, Chattopadhyay S, et al. BrakTooth: Causing havoc on bluetooth link manager via directed fuzzing. In: Proceedings of the 31st USENIX Security Symposium (USENIX security 22). Boston, MA, USA: USENIX Association, **2022**: 1025–

- 1042.
- [59] Ruge J, Classen J, Gringoli F, et al. Frankenstein: Advanced wireless fuzzing to exploit new bluetooth escalation targets. In: Proceedings of the 29th USENIX security symposium. New York, NY, USA: ACM, **2020**: 19–36.
- [60] Jang M, Hwang Y, Kwon Y, et al. BTFuzzer: A profile-based fuzzing framework for bluetooth protocols. In: Seo H, Kim S, editors. Information Security and Cryptology–ICISC 2023. Singapore: Springer, **2024**: 20–38.
- [61] Huster S, Hollick M, Classen J. To boldly go where No fuzzer has gone before: Finding bugs in Linux’ wireless stacks through VirtIO devices. In: 2024 IEEE symposium on security and privacy (SP). San Francisco, CA, USA: IEEE, **2024**: 4629–4645.
- [62] Classen J, Hollick M. Happy MitM: Fun and toys in every bluetooth device. In: Proceedings of the 14th ACM Conference on Security and Privacy in Wireless and Mobile Networks. New York, NY, USA: ACM, **2021**: 72–77.
- [63] Wen H, Lin Z, Zhang Y. FirmXRy: Detecting bluetooth link layer vulnerabilities from bare-metal firmware. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications security. New York, NY, USA: ACM, **2020**: 167–180.
- [64] Ahn P, Jang Y, Woo S, et al. BloomFuzz: unveiling bluetooth L2CAP vulnerabilities via State cluster fuzzing with Target-oriented state machines. In: Garcia-Alfaro J, Kozik R, Choraś M, editors. Computer security–ESORICS 2024. Switzerland, Cham: Springer, **2024**: 110–129.
- [65] Kovah V. Finding new bluetooth low energy exploits via reverse engineering multiple vendors’ firmwares. **2020**. <https://i.blackhat.com/USA-20/Wednesday/us-20-Kovah-Finding-New-Bluetooth-Low-EnergyExploits-Via-Reverse-Engineering-Multiple-Vendors-Firmwares.pdf>. Accessed: April 01, 2025.(We can’t open the link)
- [66] Karim I, Cicala F, Hussain S R, et al. Opening Pandora’s box through ATFuzzer: dynamic analysis of AT interface for Android smartphones. In: Proceedings of the 35th annual computer security applications conference. New York, NY, USA: ACM, **2019**: 529–543.
- [67] Pan X, Ling Z, Pingley A, et al. Password extraction via reconstructed wireless mouse trajectory. *IEEE Transactions on Dependable and Secure Computing*, **2016**, *13*: 461–473.
- [68] Jung Y, Shin J, Jang Y. Bluemaster: Bypassing and fixing bluetooth-based proximity authentication. **2019**. <https://i.blackhat.com/eu-19/Thursday/eu-19-Jung-BlueMaster-Bypassing-And-Fixing-BluetoothBased-Proximity-Authentication.pdf>. Accessed: April 01, 2025.(We can’t open the link)
- [69] Casagrande M, Losiouk E, Conti M, et al. BreakMi: Reversing, exploiting and fixing xiaomi fitness tracking ecosystem. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, **2022**: 330–366. DOI: 10.46586/tches.v2022.i3.330-366
- [70] Liu Q, Zhang Y, Sun Y. Intercepting bluetooth traffic from wearable health devices. In: 2024 IEEE security and privacy workshops (SPW). San Francisco, CA, USA: IEEE, **2024**: 267–273.
- [71] Jasek S. Gattacking bluetooth smart devices. **2016**. <https://www.blackhat.com/docs/us-16/materials/us-16-JasekGATTacking-Bluetooth-Smart-Devices-Introducing-a-New-BLE-Proxy>. Tool-wp. Accessed: April 01, 2025.(We can’t open the link)
- [72] Zhang Y, Ma S, Chen T, et al. EvilScreen attack: Smart TV hijacking via multi-channel remote control mimicry. *IEEE Transactions on Dependable and Secure Computing*, **2024**, *21*: 1544–1556.
- [73] Xie X, Jiang K, Dai R, et al. Access your Tesla without your awareness: Compromising keyless entry system of model 3. In: Proceedings 2023 Network and Distributed System Security Symposium. San Diego, CA, USA: Internet Society, **2023**.
- [74] Boone J. Technical advisory - tesla ble phone-as-a-key passive entry vulnerable to relay attacks. **2022**. <https://www.nccgroup.com/us/researchblog/technical-advisory-tesla-ble-phone-as-a-key-passive-entryvulnerable-to-relay-attacks>. Accessed: April 01, 2025.(We can’t open the link)
- [75] Yu T, Henderson J, Tiu A, et al. Security and privacy analysis of samsung’s crowd-sourced bluetooth location tracking system. In: Proceedings of the 33rd USENIX Security Symposium. Philadelphia, PA, USA: USENIX Association, **2024**: 5449–5466.
- [76] Chen J, Ma X, Luo L, et al. Tracking you from a thousand miles away! turning a bluetooth device into an apple AirTag without root privileges. In: Proceedings of the 34th USENIX Security Symposium. Seattle, WA, USA: USENIX Association, **2025**: 4345–4362.
- [77] Sivakumaran P, Zuo C, Lin Z, et al. Uncovering vulnerabilities of bluetooth low energy IoT from companion mobile apps with ble-guidie. In: Proceedings of the ACM Asia Conference on Computer and Communications Security. New York, NY, USA: ACM, **2023**: 1004–1015.
- [78] Spill D, Bittau A. BlueSniff: Eve meets Alice and bluetooth. In: Proceedings of the first USENIX workshop on Offensive Technologies. Boston, MA, USA: USENIX Association, **2007**: 1–10.
- [79] Albazraqoe W, Huang J, Xing G. A practical bluetooth traffic sniffing system: Design, implementation, and countermeasure. *IEEE/ACM Transactions on Networking*, **2019**, *27*: 71–84.
- [80] Cominelli M, Gringoli F, Patras P, et al. Even black cats cannot stay hidden in the dark: Full-band de-anonymization of bluetooth classic devices. In: 2020 IEEE Symposium on Security and Privacy (SP). San Francisco, CA, USA: IEEE, **2020**: 534–548.
- [81] Tucker T, Searle H, Butler K, et al. Blue’s clues: Practical discovery of non-discoverable bluetooth devices. In: 2023 IEEE symposium on security and privacy (SP). San Francisco, CA, USA: IEEE, **2023**: 3098–3112.
- [82] Ludant N, Vo-Huu T D, Narain S, et al. Linking bluetooth LE & classic and implications for privacy-preserving bluetooth-based protocols. In: 2021 IEEE symposium on security and privacy (SP). San Francisco, CA, USA: IEEE, **2021**: 1318–1331.
- [83] Zhang Y, Lin Z. When good becomes evil: Tracking bluetooth low energy devices via allowlist-based side channel and its countermeasure. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Ssecurity. New York, NY, USA: ACM, **2022**: 3181–3194.
- [84] Locatelli P, Perri M, Jimenez Gutierrez D M, et al. Device discovery and tracing in the Bluetooth Low Energy domain. *Computer Communications*, **2023**, *202*: 42–56.
- [85] Wu J, Traynor P, Xu D, et al. Finding traceability attacks in the bluetooth low energy specification and its implementations. In: Proceedings of the 33rd USENIX Conference on Security Symposium. Philadelphia, PA, USA: USENIX Association, **2024**: 4499–4516.
- [86] Givehchian H, Bhaskar N, Herrera E R, et al. Evaluating physical-layer BLE location tracking attacks on mobile devices. In: 2022 IEEE symposium on security and privacy (SP). San Francisco, CA, USA: IEEE, **2022**: 1690–1704.
- [87] Li Y, Lin H, Lv J, et al. BLE location tracking attacks by exploiting frequency synthesizer imperfection. In: IEEE INFOCOM 2024 - IEEE Conference on Computer Communications. Vancouver, BC, Canada: IEEE, **2024**: 1860–1869.