Open Access Original Article

Bcrypt-Based Optimized Password Hashing Against Targeted Internet of Things Attacks

Jason A. Obiri-Tetteh [1] , Regina E. Turkson [1] , Enoch A. Frimpong [2] , Alimatu-Saadia Yussif [1] , Daniel A. Asante [3] , Elliot Attipoe [1]

1. Department of Computer Science and Information Technology, University of Cape Coast, Cape Coast, GHA 2. Department of Computer Science and Technology, Cape Coast Technical University, Cape Coast, GHA 3. Department of Computer Science, Iowa State University, Ames, USA

Corresponding authors: Jason A. Obiri-Tetteh, jason.obiri-tetteh@stu.ucc.edu.gh, Regina E. Turkson, rturkson@ucc.edu.gh

Received 04/30/2025
Review began 05/07/2025
Review ended 09/06/2025
Published 09/15/2025

DOI:
https://doi.org/10.7759/s44389-025-05167-y

Abstract

The proliferation of Internet of Things (IoT) devices across various industries has revolutionized data collection, processing, and utilization, but it has also introduced significant security challenges. This study focuses on optimizing password hashing algorithms to fortify IoT device security, particularly in device authentication, data integrity, and resilience against targeted attacks. Traditional password mechanisms often fall short in the IoT context due to their limited capability to address the unique constraints and vulnerabilities of these devices. Our research proposes the use of Bcrypt, a lightweight and resource-efficient hashing algorithm, enhanced with an adaptive timer that changes the hash value at regular intervals. This approach not only secures passwords against brute force and other sophisticated attacks but also aligns with the resource limitations inherent in IoT environments. The findings underscore the importance of refining password-hashing techniques to safeguard against a wide array of IoT-specific threats while operating efficiently within the constraints of these devices.

Categories: Security and Privacy, IoT Applications, IoT Security and Privacy
Keywords: bcrypt, cyberattack, iot devices, password hashing, security

Introduction

The rapid expansion of the Internet of Things (IoT) has revolutionized connectivity but also introduced significant security challenges. IoT devices such as smart wearables and sensors transmit vast amounts of data, requiring robust protection against threats like unauthorized access and data breaches [1,2]. Traditional password-based mechanisms, though common, are increasingly inadequate due to vulnerabilities to brute-force, phishing, and dictionary attacks [3]. Poor password management further increases the risk of credential compromise.

This paper focuses on enhancing password-based authentication in IoT through optimized hashing. It evaluates existing schemes and identifies Bcrypt as the preferred algorithm for its strong security, efficiency, and adaptive timer that periodically updates hashes. This method improves device authentication, ensures data integrity, and increases resistance to attacks, while supporting future IoT security advancements.

Related works

Optimizing password hashing is crucial for secure authentication, but IoT devices' resource constraints require efficient algorithms that provide robust security without overloading system capacities. Research indicates that unconscious use, not changing passwords, and the lack of device updates have increased cybersecurity risks and access to malicious applications to the IoT systems' sensitive data [4]. Hence, Yu and Huang [5] suggest that users' passwords should be frequently encrypted or hashed. Sriramya and Karthika [6] emphasize the superiority of hashing over encryption for password protection due to its one-way function, which makes it impossible to reverse-engineer the plain text from the hash value. Nevertheless, inadequate security procedures raise the possibility of a data leak and other dangers [3]. Most security experts suggest that IoT is a prime target for cyberattacks due to weak security policies and procedures. Recent studies suggest that even developers do not use appropriate hash functions to protect passwords, since they may not have adequate security expertise [7].

Authentication is a critical component in securing IoT devices, which are often vulnerable due to their mass production and lack of robust security measures. A novel mode-based hash chain for secure mutual authentication has been proposed to address these vulnerabilities. Landge and Satopay [8] propose a system that uses MD5 for secure communication between IoT devices by combining a timestamp, password, and control data to generate a hash that changes every 60 seconds, thus mitigating the risk of collision attacks and unauthorized access. The system's use of a changing hash and password ensures authentication, integrity, and protection against unauthorized access. As MD5 does not incorporate a salt function, Strahs [9] designed a unique system approach that involves the use of a large random salt, ensuring that each login receives a distinct, unpredictable password. This not only thwarts offline attacks but also provides an additional layer of defense against phishing and shoulder surfing. Password Agent's design is user-centric, maintaining the familiar password entry process while enhancing security and usability. Sriramya and Karthika [6] also advocate for the use of salting, thus making each hash unique and resistant to pre-calculated attacks.

Álvarez et al. [10] suggested that performance can be significantly enhanced by optimizing password hashing algorithms using hardware-accelerated symmetric encryption. New password-based key derivation functions (PBKDFs) were created by utilizing hardware acceleration and the Advanced Encryption Standard as a pseudo-random generator. It is demonstrated that these functions outperform current algorithms like Scrypt and Argon2. However, a significant number of Content Management System use outdated MD5 hash functions, which are highly parallelizable and thus susceptible to GPU-based password-guessing attacks. Some Content Management System utilize SHA1, SHA256/SHA512, or PBKDF2, which are also parallelizable to some extent, affecting the security against password guessing attacks. The study found that 48.94% of web application frameworks use BCRYPT, which is a positive trend towards more secure hashing schemes [7]. Sriramya and Karthika's [6] Bcrypt algorithm is introduced as a robust solution for password hashing, capable of encrypting data up to 512 bits and providing a longer encryption key. Bcrypt is also adaptive, allowing for an increase in computational cost to stay ahead of hardware improvements and remain secure against brute-force attacks.

The reviewed studies highlight both progress and limitations in password hashing for securing IoT and web-based systems. While hardware-accelerated PBKDFs show strong performance advantages, widespread use of weaker legacy algorithms continues to expose systems to significant risks. Bcrypt adoption is growing, but a gap remains in optimizing it for resource-constrained IoT environments, where balancing strong security with lightweight performance is essential. This research addresses that gap by proposing an adaptive Bcrypt-based framework with a timer mechanism, designed to enhance device authentication, data integrity, and resilience against targeted attacks while remaining deployable in constrained IoT systems.

Background study

This section provides a brief background of the IoT and password.

Internet of Things

Kevin Ashton introduced the term "Internet of Things" in 1999 while proposing RFID chips for tracking products at Procter & Gamble [4]. Although the idea of integrating sensors into physical objects dates back to the 1960s, its adoption was slow due to technological limitations. Initially known as embedded Internet or pervasive computing, IoT has evolved into a network of interconnected devices that communicate over the Internet. The rapid growth of connected devices has led to significant data traffic, highlighting security concerns. In 2020, approximately 26.66 billion IoT devices were in use, and by 2023, this number was expected to reach 43 billion, nearly tripling since 2018 [4,11]. By 2025, the global installed base of IoT devices is projected to reach 30.9 billion units, reflecting a substantial rise from 13.8 billion in 2021, as shown in Figure *1*. This exponential growth underscores the increasing need for robust security measures to protect the expanding digital ecosystem.

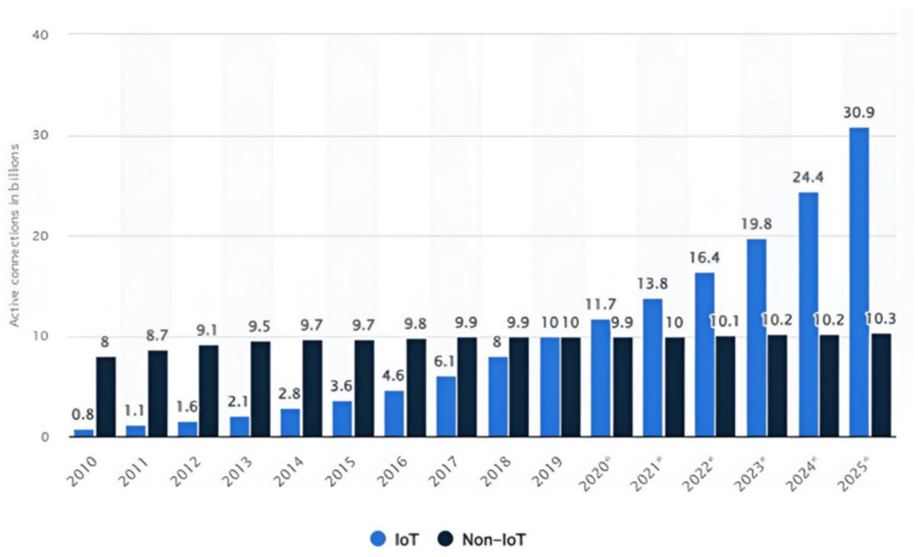

FIGURE 1: IoT and non-IoT connections worldwide 2010-2025

Source: [12]

IoT, Internet of Things

Password

Passwords are the most common form of authentication for securing access to files, computers, and network servers, with their use dating back to the UNIX password scheme [3]. They consist of characters known only to the user, ensuring authorized access. Passwords can include numbers, alphabets, symbols, or combinations, such as PINs, passphrases, alphanumeric, biometric, and pattern-based passwords [13]. To enhance security, modern applications require uppercase letters, numbers, and symbols to resist cracking and guessing attempts [14].

However, passwords remain vulnerable to attacks, particularly when reused across multiple accounts or if they are common and easily guessed. This increases the risk of brute force attacks, where attackers systematically try combinations until successful. To improve security, it is essential to enhance password strength, use multi-factor authentication, and encourage regular updates. Despite these measures, the need for further optimization persists due to evolving security threats. Table *1* shows some commonly used passwords.

Some common passwords		
123456	P@ssw0rd	password
1234567890	abc123	1234
hello	iloveyou	Pa$$w0rd
aaaaa	hello123	abcd1234
999999999	12345678	111111
asdasd123	asdfghjkl	Qwerty123
P@55w0rd	admin123 654321	username123
112233	password123	azerty123
qwertyuiop	password1	Admin@123
qwerty	Aa123456	QWERTY123

TABLE 1: Common passwords

Source: [15]

IoT, Passwords, and Hashed Password

While users are responsible for creating secure passwords, IoT devices share the duty of safeguarding user credentials. Storing plaintext passwords is highly insecure, as it exposes sensitive data in the event of a database breach. Even weak or reused passwords can lead to compromise if not protected with proper security mechanisms. Notably, 81% of data breaches in 2020 were due to compromised credentials [16]. Given the declining reliability of traditional passwords due to predictable structures, implementing secure verification methods like password hashing is essential.

Password hashing transforms plaintext passwords into fixed-length strings using cryptographic algorithms, providing stronger protection than storing passwords in plaintext. Even if a hash is compromised, the original password remains concealed. However, the rise in high-performance computing tools, such as GPUs, enables faster password cracking, increasing security threats [7]. The hashing process is depicted in Figure *2*.

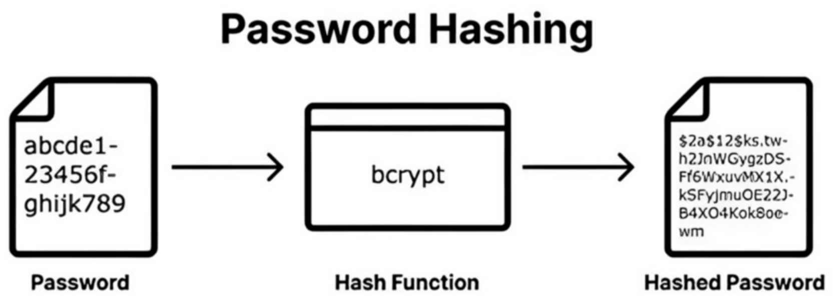

FIGURE 2: Password hashing

Source: [17]

Password Cracking

Password cracking refers to the process of retrieving a password from its hashed or encrypted form to gain unauthorized access [5]. Attackers typically exploit weak passwords, flaws in algorithms, or reused credentials. There are two main types of attacks: online and offline. Online attacks involve direct interaction with systems, typically using commonly guessed passwords [3,5], but can be mitigated with security measures such as login attempt limits or IP blocking [7]. In contrast, offline attacks allow unlimited password attempts using stolen hashes without detection, relying heavily on time and computing power [3,7], with common strategies including brute-force, dictionary, and rainbow table attacks [5].

Brute-force cracking: Brute-force attacks involve systematically guessing every possible password combination, sometimes using personal information to increase success rates [18]. Known as cryptanalysis, this technique aims to uncover plaintext or encryption keys [19]. While early methods were manual, modern brute-force attacks use automated bots.

Dictionary cracking: Dictionary attacks rely on pre-compiled lists of likely passwords rather than all combinations, offering faster execution than brute-force attacks [5]. These lists often include leaked passwords and variations (e.g., "p@ssw0rd") to improve success rates, exploiting users' tendencies to use predictable credentials [20].

Rainbow table cracking: Rainbow table attacks utilize precomputed tables of hashes and their corresponding plaintext values to quickly reverse hash functions. Attackers only need a matching hash, not the exact password. This efficient technique can break encrypted files and is often facilitated by tools like John the Ripper [5].

Password Hashing Schemes

Password hashing converts a plain-text password into a fixed-size cryptographic string, enhancing password storage security and mitigating unauthorized access even if databases are compromised. Widely used algorithms include MD5, SHA3, Bcrypt, Scrypt, Catena, Lyra2, and Argon2, each with varying levels of protection and resistance to attacks [21]. While MD5 is efficient, it is highly vulnerable to brute force, rainbow table, and collision attacks. SHA3 improves upon MD5 in collision resistance but still poses risks if poorly implemented [7]. To address the constraints of IoT systems, more secure alternatives like Bcrypt, Scrypt, and Argon2 integrate salting to generate unique hashes even for identical passwords, mitigating rainbow table threats.

Among these, Bcrypt stands out for its strong balance of security and efficiency, making it ideal for resource-constrained IoT devices, as illustrated in Figure *3*. Its lightweight footprint - requiring under 18 KB RAM and 36 KB ROM - and support for adaptive work factors provide robust defense against brute-force attacks [21]. Compared to MD5 and SHA3, Bcrypt's features offer superior resilience against modern threats, positioning it as the preferred solution for safeguarding IoT environments. Figure *3* shows a Code size of Bcrypt as against other encryption algorithms.

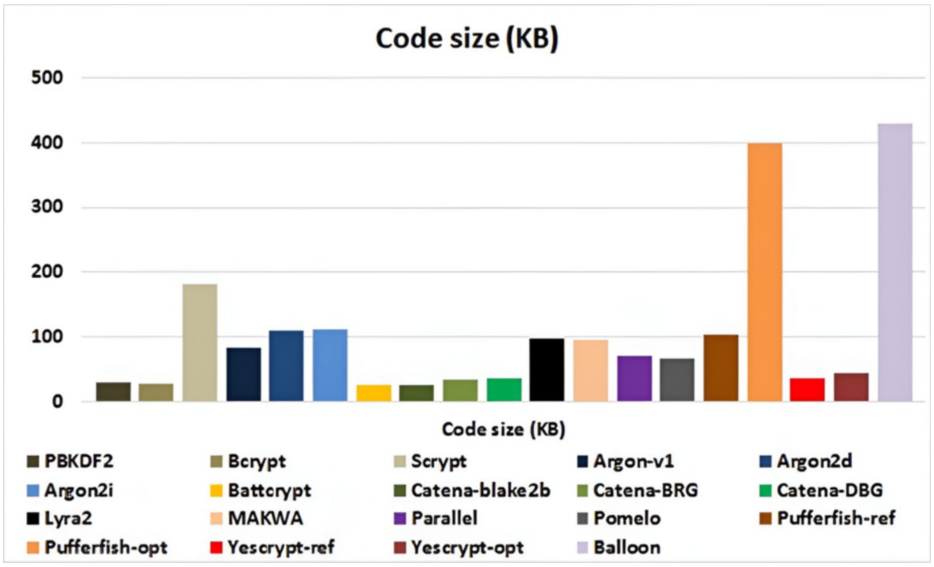

FIGURE 3: The code size of PBKDF2, Bcrypt, Scrypt, the nine PHC finalists, and Balloon

Source: [21]

PBKDF2, Password-Based Key Derivation Function 2; PHC, Password Hashing Competition

Materials And Methods

Optimized password hashing with Bcrypt

Bcrypt is a password hashing function based on the Blowfish encryption algorithm. Developed by Niels Provos and David Mazières in 1999 [22], bcrypt is known for its "CPU-hardened" nature, requiring significant CPU cycles to compute a hash. This makes bcrypt resistant to brute-force attacks. The key design features of Bcrypt include [6,23]:

EksBlowfish Key Setup: Bcrypt employs an expensive key schedule (EksBlowfish) that repeatedly mixes the password and a cryptographic salt into the Blowfish state. This ensures that even minor password changes result in drastically different hashes [24].

Adaptive Cost Parameter: A configurable cost factor determines the number of iterations applied during hashing. This allows Bcrypt to be “future-proof,” since the computational difficulty can be increased as hardware capabilities improve.

Salt Integration: Bcrypt uses a 128-bit salt for each password. This ensures that even identical passwords produce different hash outputs, preventing the use of precomputed rainbow tables.

Output Format: Bcrypt outputs hashes in the Modular Crypt Format, which stores the algorithm identifier, cost factor, salt, and hash value in a single string. This makes verification straightforward and portable.

The function allows for adjusting a work factor, which controls the number of iterations involved in the hashing process. As computational power increases, the work factor can be raised to maintain the security of the hashed passwords. Bcrypt is essential because it includes a salting function, which ensures that even identical passwords generate distinct hash values. Importantly, the value generated by Bcrypt does not remain constant within a particular time frame; each hashing operation produces a unique output due to the random salt, even when the same password is hashed repeatedly. This property distinguishes Bcrypt from weaker algorithms and enhances its resistance against rainbow table and replay attacks.

As shown in the github repository *https://github.com/GodlyJay/Passwordhashing.git*, implementing Bcrypt in Python demonstrates how salting results in different hash values each time a password is hashed. With the same password, “supersecretpassword”, when executed twice results in different hash values as indicated in Tables *2* and 3. Figure *4* shows the flow of how the algorithm works. This variability complicates the process for attackers attempting to crack the password. To further strengthen authenticity, integrity, and resistance to targeted attacks, security can be enhanced by incorporating a timer that periodically changes the hashed password within a specified time frame. This dynamic approach makes it significantly more challenging for attackers to crack the password, as the hash value continuously updates, as shown in Table 4, increasing the complexity of any potential cracking attempt.

Salting results	
Original password	supersecretpassword
Hashed password	b'$2b$12$R/nMg5Rg8d5GS6h IaxIzL.NPDz48lY3FEih4oi 0S7FPwwjrosgJ0S'
Password match	True

TABLE 2: Output 1

Salting results	
Original password	supersecretpassword
Hashed password	b'$2b$12$fC9wxZ0WTmrcmm dhQZNmL.C7K7g08U46Z 3CsAihjqMzCZxYLGdpEi'
Password match	True

TABLE 3: Output 2

Salting results	
Original password	supersecretpassword
Hashed password	b'$2b$12$XIuh7k1lGl txLMTlftjoBe1QpdTVB aMj5nzNowWPsdVh taL6IMR0i'
Hashed password updated	b'$2b$12$xxEpd4CYy CwP9HSootZMCuwlrt CRy8bbUC1WJSj0IZH 1klAGXF7W2'
Hashed password updated	b'$2b$12$lAmlpq3AAj GMec7g333FZelQv lPf0XZXpXuovliSH yVjM1mVSbQH2'
Hashed password updated:	b'$2b$12$9y/l98zf9fob J16LQO0cguvzstuu trvUlQv1nr81SkYM4 h0VQEJ06'
Original password	HelloWorld!
Hashed password	b'$2b$12$eE8CogBuG.jh/TJ2EyyPwevt1pJ3STfVISP6i1OCKKtk6G4WyZ2Xq'
Hashed password updated	b'$2b$12$ftt3a8aIATqqmWwwMGLdre.p/.wNpNNUazR1YUfiJ7sSb5iwTPVLO'
Original Password	Stay_Away
Hashed Password	b'$2b$12$zEg2vn6dMkIsqTEbYt5DSePxhWXVmxH8o7nC.uYq8RAEpOjMgIrzi'
Hashed Password updated	b'$2b$12$UGbFzYIo3tidZU0ZwUACpuhVKnVZDMSaEbx6o0T8YtXEryI7jsN2y'

TABLE 4: Adaptive timer output

Github repository for Optimised Password Hashing: https://github.com/GodlyJay/Passwordhashing.git

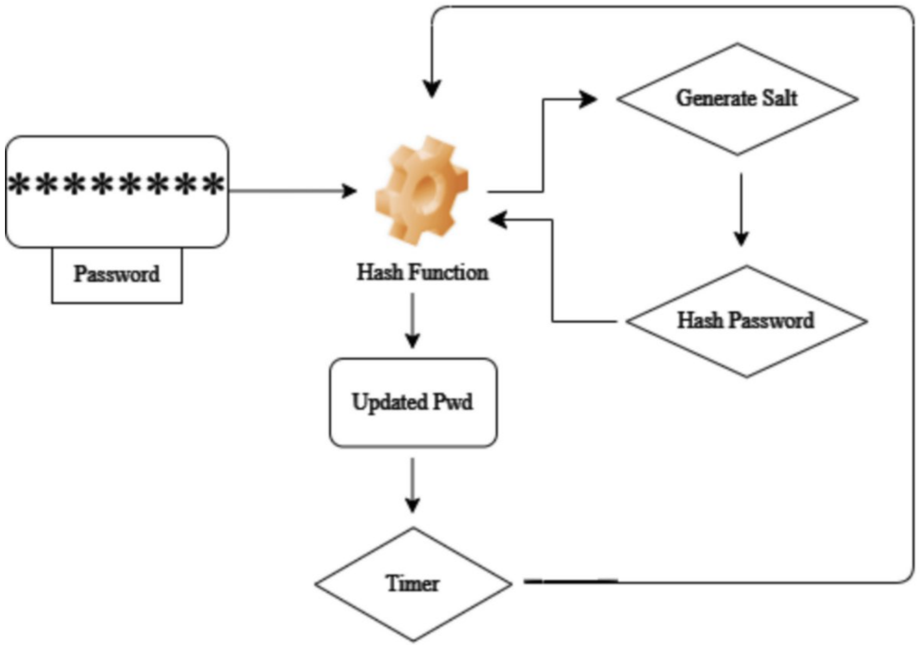

FIGURE 4: Framework

Mathematical framework

This section outlines the techniques and mathematical foundations employed to develop and evaluate the proposed framework for optimizing password hashing in IoT environments.

1) Password Hashing Process Using Bcrypt: The Bcrypt password hashing algorithm is selected due to its security features, including salting, an adjustable work factor, and adaptive timing. The password hashing process can be formalized as:

$$H(P, S) = Bcrypt(P, S, w) \tag{1}$$

where:

H(*P*, *S*) is the generated password hash.

P represents the user's plaintext password.

S is a random salt value generated for each password.

w is the work factor controlling the computational cost of hashing (typically expressed as 2w iterations).

The inclusion of salt (*S*) ensures that:

$$H(P1, S1) = H(P1, S2) \text{ for any } S1 = S2 \tag{2}$$

Even if two users have the same password, the resulting hashes will differ due to the unique salt values.

2) Adaptive Timer for Hash Updates: To enhance security, an adaptive timer is introduced to periodically refresh the password hash values at time intervals t. The timer-based hash update can be represented as:

$$H_t = Bcrypt(P, S, w + f(t)) \tag{3}$$

where:

H_t is the updated hash at time *t*.

f(*t*) is a time-dependent function that increases the computational cost as time progresses, mitigating brute-force attacks.

For example, a linear time increment function can be defined as:

$$f(t) = k \cdot t \tag{4}$$

where *k* is a constant factor controlling the rate of hash complexity increment.

3) Entropy of Password Space: The entropy *E* of a password space, which measures its resistance to brute-force attacks, is defined as:

$$E = \log_2(C^L) \tag{5}$$

where:

C is the number of possible characters in the password set (e.g., 62 for alphanumeric characters).

L is the password length.

A higher entropy value indicates greater resistance to brute-force attacks. The proposed system encourages higher entropy by recommending complex passwords and using adaptive hash functions.

4) Computational Cost of Bcrypt Hashing: The computational complexity of Bcrypt hashing is influenced by its work factor w, which determines the number of iterations:

$$C_{Bcrypt} \propto 2^w \tag{6}$$

Increasing *w* exponentially raises the computational effort required, making brute-force attacks infeasible for higher values of w. The framework dynamically adjusts w based on system requirements and time.

5) Security Against Rainbow Table Attacks: The probability of a successful rainbow table attack ($P_{success}$) is reduced by incorporating salt into the hashing process:

$$P_{success} \propto \frac{1}{|S|} \tag{7}$$

where $|S|$ is the size of the salt space. A larger salt space makes precomputed attacks impractical.

By integrating these mathematical foundations into the proposed framework, the paper demonstrates how adaptive timing, salting, and computational complexity effectively enhance IoT device security. These formulas formalize the system's mechanisms, providing a robust foundation for evaluating its resilience against targeted attacks.

Enhancing authentication security

Bcrypt is chosen for its ability to provide strong password hashing, which is particularly crucial for IoT devices that often have limited computational resources. Bcrypt is resistant to brute-force attacks because of its adaptive features, which include the use of a variable work factor. This strengthens the authentication process by ensuring that even if an attacker manages to obtain the hashed passwords, it will take a lot more time and effort to crack them. The hashed password is changed regularly by the adaptive timer included in the framework. This feature narrows the window of opportunity for successful attacks, making it harder for attackers to compromise the system. The dynamic properties of the hash values further secure the device by making sure that even if an attacker manages to obtain a hash, it will eventually become outdated.

Enhancing data integrity

The framework ensures that the hashed passwords are distinct and reliable even if two passwords with the same hash are used by utilizing Bcrypt, which has properties like salting and a high processing cost. As a result, the integrity of the data is almost completely preserved, and attackers are unable to reverse-engineer passwords using precomputed tables (such as rainbow tables).

Strong, regular password hashing ensures that any illegal alterations to the data may be identified, which helps prevent tampering. The hash value could change if an attacker attempted to modify the data, indicating a possible compromise of data integrity.

Resilience against targeted attacks

The framework is designed to withstand a variety of targeted attacks, including phishing, password guessing, and dictionary attacks. Combined with the adaptive timing, Bcrypt's strong hashing ensures that even highly skilled attackers will have a difficult time breaching IoT devices.

The framework optimizes the trade-off between security and performance, considering the resource limitations common to IoT devices. The framework offers robust protection without overtaxing the limited processing power of IoT devices by carefully adjusting Bcrypt's work factor and adding adaptive timing, preserving operational efficiency while guaranteeing excellent security.

Results

The study emphasizes the critical need for optimized password hashing techniques to enhance the security of IoT devices.

1) Inadequacy of Traditional Password Mechanisms: Traditional password mechanisms, such as simple password-based systems, fail to provide sufficient protection against modern threats, including brute-force attacks, phishing, and password guessing. The study underscores the necessity of developing enhanced security protocols to address these vulnerabilities and ensure robust protection for IoT environments.

2) Importance of Password Hashing: Password hashing is identified as a fundamental element of secure authentication processes. Optimized hashing algorithms are essential for delivering robust security while remaining efficient enough to operate within the resource constraints of IoT devices.

3) Selection of Bcrypt: After evaluating various password hashing schemes, the study identifies Bcrypt as the most suitable algorithm for IoT environments due to its lightweight implementation, resource efficiency, and robust security. Bcrypt incorporates essential features, such as salting, which ensures that even identical passwords generate unique hash values, complicating attackers' use of precomputed hash tables (e.g., rainbow tables), and adaptive timer, whic increases the computational cost of hashing to resist brute-force attacks. The inclusion of an adaptive timer in Bcrypt, which periodically updates hashed passwords, further strengthens the security framework by limiting the attack window for potential breaches.

4) Improvements in Security: The implementation of Bcrypt demonstrates significant improvements in Device Authentication (ensuring only legitimate users can access the system) and Data Integrity (protecting data from tampering by generating unique and reliable hash values). The use of Bcrypt reinforces the overall security of IoT ecosystems by mitigating targeted attacks.

5) Resilience Against Targeted Attacks: The framework effectively defends against a variety of modern threats, including phishing, dictionary attacks, and password guessing. By combining Bcrypt's strong hashing capabilities with adaptive timing, attackers face increased difficulty in breaching IoT systems.

6) Scalability and Efficiency: The optimized hashing approach balances security and performance, making it feasible for IoT devices with constrained computational resources. The framework's lightweight design ensures scalability, enabling its adoption across a wide range of IoT devices and applications without overburdening system resources.

Discussion

This research highlights the limitations of traditional password mechanisms and the pressing need for optimized password hashing techniques in IoT environments. By selecting Bcrypt, a hashing algorithm that balances security and resource efficiency, and incorporating adaptive timing and salting mechanisms, the framework significantly enhances device authentication, data integrity, and resilience against targeted attacks. These findings reinforce the importance of adopting modern password hashing strategies to secure IoT devices while addressing their unique resource constraints.

Implementing the proposed framework for strengthening IoT device authentication, data integrity, and resilience against targeted attacks in real-world scenarios could present several challenges such as:

Memory and Storage Limitations: The adaptive features of Bcrypt, including the storage of salts and adaptive timer data, require additional memory and storage, which might not be available in some low-cost IoT devices. Bcrypt by itself is lightweight and resource-efficient, which makes it suitable for IoT environments, but additional overhead in our proposed framework arises from the adaptive timer, which triggers periodic rehashing to reduce the attack window.

Scalability Issues: The framework's periodic re-hashing of passwords and the communication of new hash values across the network could introduce additional network traffic, which may be problematic in largescale IoT deployments with limited bandwidth.

Increased Development and Deployment Costs: The complexity of implementing and maintaining this framework may lead to higher development and deployment costs, particularly for organizations with many IoT devices.

Adoption Barriers: Organizations may be resistant to adopting new security frameworks due to concerns about cost, complexity, or potential disruptions to existing operations. Convincing stakeholders of the framework's benefits and securing buy-in could be a significant challenge.

Scheme	Code size (KB)	Memory usage	Security level	Latency (ms)	CPU overhead
PBKDF2	30	Low (<10 KB)	Vulnerable	Low (<5 ms)	Low
Bcrypt	27	Low (≈18 KB RAM, 36 KB ROM)	Moderate	Moderate (10-50 ms)	Moderate
Argon2	110	High (0.5–2 MB)	High	High (>80 ms)	High
Scrypt	182	Very High (>1 MB)	High	High (>100 ms)	Very High
Balloon	430	Configurable (100 KB–several MB)	Very High	Moderate-High	Moderate-High

TABLE 5: Benchmark comparison of password hashing algorithms in IoT contexts

PBKDF2, Password-Based Key Derivation Function 2

Table *5* compares Bcrypt with other state-of-the-art password hashing algorithms in terms of memory usage, latency, and CPU overhead. PBKDF2 is lightweight and fast but offers weaker resistance against hardware-accelerated attacks, making it less robust for modern IoT threats. Scrypt and Argon2 provide stronger security but require significantly higher memory and CPU resources, which limits their applicability in constrained IoT devices. Balloon hashing offers tunable parameters but remains less widely adopted and tested. Bcrypt, by contrast, achieves a favorable balance between security and computational efficiency, with modest resource requirements (≈18 KB RAM, 36 KB ROM) and adaptive complexity, making it well suited for typical IoT environments.

In summary, the suggested design provides strong security improvements for IoT devices; nevertheless, resource limitations, scalability, interoperability, complexity, usability, cost, regulatory compliance, and acceptance hurdles may pose difficulties for its practical application. For the framework to be successfully deployed in various IoT contexts, these issues must be resolved.

Conclusions

This paper has demonstrated that traditional password mechanisms are insufficient to protect IoT devices from modern threats. The use of optimized hashing algorithms like Bcrypt, with enhancements such as adaptive timers, offers a significant improvement in securing IoT environments. The findings underscore the necessity of refining password hashing techniques to ensure the protection of user data and the integrity of IoT systems.

Future research should focus on advancing password hashing algorithms tailored for IoT devices by balancing strong security with lightweight performance. This includes developing automated benchmarking tools to determine optimal Bcrypt cost parameters for specific hardware configurations, while evaluating energy consumption trade-offs between security levels and battery life. Lightweight multi-factor authentication methods, such as device fingerprinting, behavioral analysis, or proximity-based verification using IoT sensors, should be explored to complement Bcrypt. Systematic benchmarking against GPU-based and real-world attacks, alongside field studies across diverse IoT deployments, will help validate its effectiveness. Further directions include investigating post-quantum resistance, integrating password-authenticated key exchange protocols to reduce reliance on stored hashes, and strengthening side-channel resistance against timing and power analysis. Practical implementation strategies, such as industry-specific guidelines for IoT password hashing and secure fleet-wide update mechanisms, are also critical. Finally, adapting memory-hard functions like Argon2 to highly constrained IoT environments and leveraging AI-assisted attack detection based on hashing patterns could significantly enhance resilience. Together, these directions address current gaps while ensuring scalable and future-proof IoT security solutions.

Additional Information

Author Contributions

All authors have reviewed the final version to be published and agreed to be accountable for all aspects of the work.

Concept and design: Jason A. Obiri-Tetteh, Regina E. Turkson, Enoch A. Frimpong, Daniel A. Asante, Elliot Attipoe, Alimatu-Saadia Yussif

Acquisition, analysis, or interpretation of data: Jason A. Obiri-Tetteh

Drafting of the manuscript: Jason A. Obiri-Tetteh, Daniel A. Asante, Alimatu-Saadia Yussif

Critical review of the manuscript for important intellectual content: Regina E. Turkson, Enoch A. Frimpong, Elliot Attipoe

Supervision: Regina E. Turkson

Disclosures

Human subjects: All authors have confirmed that this study did not involve human participants or tissue. **Animal subjects:** All authors have confirmed that this study did not involve animal subjects or tissue. **Conflicts of interest:** In compliance with the ICMJE uniform disclosure form, all authors declare the following: **Payment/services info:** All authors have declared that no financial support was received from any organization for the submitted work. **Financial relationships:** All authors have declared that they have no financial relationships at present or within the previous three years with any organizations that might have an interest in the submitted work. **Other relationships:** All authors have declared that there are no other relationships or activities that could appear to have influenced the submitted work.

References

1. IoT devices (internet of things devices). (2025). Accessed: September 12, 2025: https://www.techtarget.com/iotagenda/definition/IoT-device.
2. Xia F, Yang L, Wang L, Vinel A: Internet of things. International Journal of Communication Systems. 2012, 25:1101-1102. 10.1002/dac.2417
3. Yazdi SH: Analyzing password strength & efficient password cracking [thesis]. Florida State University, Florida; 2011.
4. Tawalbeh L, Muheidat F, Tawalbeh M, Quwaider M: IoT privacy and security: Challenges and solutions. Applied Sciences. 2020, 10:4102. 10.3390/app10124102
5. Yu F, Huang Y: An overview of study of password cracking. 2015 International Conference on Computer Science and Mechanical Automation. 2015, 25-29. 10.1109/CSMA.2015.12
6. Sriramya P, Karthika RA: Providing password security by salted password hashing using Bcrypt algorithm. ARPN Journal of Engineering and Applied Sciences. 2015, 10:5551-5556.
7. Ntantogian C, Malliaros S, Xenakis C: Evaluation of password hashing schemes in open source web platforms. Computers & Security. 2019, 84:206-224. 10.1016/j.cose.2019.03.011
8. Landge IA, Satopay H: Secured IoT through hashing using MD5. IEEE International Conference on Advances in Electronics, Electrical & Computational Intelligence (AEEICB). 2018, 1-5. 10.1109/AEEICB.2018.8481007
9. Strahs B: Secure passwords through enhanced hashing. Bachelor's thesis, College of William and Mary. (2009). Accessed: September 15, 2025: https://scholarworks.wm.edu/items/b6089f3e-c7a4-4675-9919-139c112d0e7c.
10. Álvarez R, Andrade A, Zamora A: Optimizing a password hashing function with hardware-accelerated symmetric encryption. Symmetry. 2018, 10:705. 10.3390/sym10120705
11. Growing opportunities in the internet of things. (2019). Accessed: September 15, 2025: https://www.mckinsey.com/industries/private-capital/ourinsights/growing-opportunities-in-the-internet-of-things.
12. Internet of Things (IoT) and non-IoT active device connections worldwide from 2010 to 2025. (2025). Accessed: September 15, 2025: https://www.statista.com/statistics/1101442/iot-number-of-connecteddevices-worldwide/.
13. Types of password. (2025). Accessed: September 15, 2025: https://www.geeksforgeeks.org/types-of-password/.
14. Random password generator. Accessed: September 15, 2025: https://www.avast.com/random-password-generator#pc.
15. Top 200 most common passwords. Accessed: September 15, 2025: https://nordpass.com/most-common-passwords-list/.
16. Six types of password attacks and how to stop them. Accessed: September 15, 2025: https://www.onelogin.com/learn/6-types-password-attacks.
17. Password Hashing & Salting (2025): Argon2id vs bcrypt, PBKDF2 Explained. (2025). Accessed: September 15, 2025: https://www.authgear.com/post/password-hashing-salting.
18. Dave KT: Brute-force attack "seeking but distressing". International Journal of Innovations in Engineering and Technology. 2013, 2:75-78.
19. Stallings W: Network Security Essentials: Applications and Standards. 4th Edition. Prentice Hall, New Jersey; 2010.
20. Bošnjak L, Sreš J, Brumen B: Brute-force attack and dictionary attack on hashed real-world passwords. 2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). 2018, 1161-1166. 10.23919/MIPRO.2018.8400211

21. Hatzivasilis G: Password-hashing status. Cryptography. 2017, 1:10. 10.3390/cryptography1020010
22. Batubara TP, Efendi S, Nababan EB: Analysis performance bcrypt algorithm to improve password security from brute force. Journal of Physics: Conference Series. 2021, 1811:012129. 10.1088/1742-6596/1811/1/012129
23. Ertaul L, Kaur M, Gudise VAKR: Implementation and performance analysis of pbkdf2, bcrypt, scrypt algorithms. Proceedings of the International Conference on Wireless Networks (ICWN). 2016, 1-7.
24. Bcrypt. (2025). Accessed: September 6, 2025: https://en.wikipedia.org/w/index.php?title=Bcrypt&oldid=1298892562.