IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Android Pattern Bypassing Using A Microcontroller

DABBETA RAKESH

(PG Diploma in Cybersecurity and Digital Forensics)

Abstract: The rapid expansion of Android devices in the consumer and business markets highlights how important it is to have strong security standards in place to guard against changing online threats. Inherent weaknesses still exist despite improvements in security protocols, providing opportunities for malevolent actors to take advantage of. The strategic integration of an ATTiny85 microcontroller via the USB interface to take advantage of potential weaknesses is the main emphasis of this project, which explores the complex world of Android device security. Additionally, by highlighting potential flaws and associated exploitation methods, this initiative adds to the continuing discussion about Android device security. It seeks to clarify the complexities of contemporary security threats and promote a proactive strategy for protecting Android devices against malevolent intrusions through empirical research and testing.

Keywords:

Project Scope and Limitations, Technical Restrictions, Problem with Compatibility, Legal and Moral Aspects to consider, Risk to Security, Objective of the Proposed Model, Attity 85 USB, Injecting Brute Force Code, Plug Android to Device, Attempt to unlock device, Algorithm Code, System Requirements, Future scope and Limitations, Conclusion

Introduction:

Understanding and strengthening Android devices' security features has become more important as a result of their broad usage, particularly in authentication procedures like pattern locks. However, forgetting device passwords is not unusual, and it frequently results in important data being inaccessible. This project uses the ATtiny85 microcontroller, which is well-known for its adaptability and versatility in a variety of applications, to suggest a fresh method of getting around device passwords.

The AVR family includes the ATtiny85, a small 8-bit microcontroller that is well-known for its versatility and efficiency. It can easily interface with external USB-to-Serial converters, allowing communication with USB devices, even if it lacks native USB capabilities. This project takes advantage of this feature to apply USB-based brute force attacks, made possible by the rubber duck technique, against Android devices that have lost their passwords.

In addition to its success with microcontrollers, the ATtiny85 is useful in a variety of embedded systems, from control systems to sensor interfaces. Because of its low power consumption, it is perfect for energy-efficient and battery-powered devices, such as wearable technology and remote sensors. Additionally, its use in IoT devices makes it easier to link sensors and actuators, allowing for remote control and data gathering features. This project highlights the adaptability of microcontroller-

based techniques in cybersecurity scenarios while providing a workable way to get around forgotten device passwords by utilising ATtiny85's capabilities. It clarifies the viability of using affordable and easily accessible components to address security issues in contemporary computing settings through practical trial and analysis.

Objective:

The main goal of this project is to look into and take advantage of any security flaws in Android devices, with an emphasis on authentication methods like pattern locks. In order to fully comprehend the underlying security methods used by Android devices, our approach entails carrying out a thorough study and analysis step. After that, we plan to use the ATtiny85 microcontroller's capabilities along with USB-to-Serial converters to create an advanced software program that can carry out USB-based brute force attacks on Android smartphones. In an effort to get beyond device authentication, this software will methodically repeat through different passcode combinations, exposing any flaws in the security architecture.

We will uphold strict adherence to ethical standards throughout the project lifespan, making sure that all operations are carried out in compliance with legal and ethical requirements and steadfastly protecting user privacy and data integrity. Furthermore, in order to enable thorough distribution of our findings to relevant stakeholders, painstaking documentation will be done to precisely capture the development process, experimental approaches, and ensuing insights. Additionally, we will try to find potential directions for further research and development activities, such as investigating cutting-edge methods like machine learning algorithms to maximise attack effectiveness or creating preventative measures to lessen the risks connected with brute force attacks.

Project Scope and Limitations:

Utilising an ATTiny85 microcontroller to implement Android pattern bypass offers both exciting potential and inherent constraints and restrictions. Because of the ATTiny85's capabilities, pattern bypass is possible without the need for time intervals, enabling the use of complex algorithms to unlock Android devices quickly. Furthermore, the microcontroller can be configured to record information from the intended device, possibly obtaining private data like login credentials or keystrokes. This function offers insights into device usage patterns and potential vulnerabilities, making it useful for forensic analysis or security testing. Furthermore, the ATTiny85's capacity to insert malware or viruses into the device raises serious security issues, highlighting the significance of putting strong security measures in place to thwart unwanted access and reduce possible hazards.

The development of firmware for the ATTiny85 microcontroller includes coding algorithms for data extraction, pattern recognition, and perhaps virus injection. It is necessary to handle the microcontroller's constraints, such as its limited memory and computing capability. These limitations could limit the amount of data that can be injected or recorded, as well as the complexity of algorithms. It is essential to make sure that the ATTiny85 hardware and Android devices are compatible, taking into account factors like firmware compatibility, power requirements, and communication protocols.

Additionally, the project needs to follow the ethical and legal rules that regulate penetration testing and security research. Malicious activity or unauthorised access may result in severe legal repercussions. As a result, user education and awareness are crucial project components. Security concerns can be reduced and overall device security can be improved by informing users about the possible dangers of pattern bypass attacks and encouraging recommended practices, such as creating strong passwords and turning on extra security features.

In conclusion, the ATTiny85 microcontroller presents considerable difficulties and restrictions, even though it has interesting potential for developing Android pattern bypass. To guarantee acceptable and legal use of this technology, careful evaluation of the project's scope, technical limitations, and ethical considerations is required.

Technical Restrictions:

The ATTiny85 microcontroller's limited memory and processing capacity may limit the intricacy of algorithms. constraints imposed by hardware on the volume of data that can be injected or collected.

Problems with Compatibility:

ensuring that Android devices and the ATiny85 hardware are compatible, including taking care of power needs and communication standards. Firmware updates and version mismatches between the microcontroller and Android devices present compatibility issues.

Legal and Moral Aspects to Consider:

compliance with legal requirements for penetration testing and security research in order to prevent illegal access and any legal repercussions. Ethical issues pertaining to the prudent use of technology and its possible effects on user security and privacy.

Risks to Security:

potential for using the ATTiny85's virus injection capability to infect Android devices with malware or security flaws.

Risks associated with unauthorized data capture and potential misuse of captured information, highlighting the importance of robust security measures.

User Awareness and Education:

Spreading best practices for improving device security and informing users about the dangers of lock bypass attacks is crucial. To lesson any risks, users should be encouraged to be vigilant and to implement security precautions like creating strong passwords.

Objective of the Proposed Model:

In addition to providing an alternate approach for accessing Android devices with forgotten pattern locks, the suggested model seeks to address a number of crucial aspects in order to improve its effectiveness and usability. First, the approach aims to prioritise user privacy and data security by ensuring that the bypass process does not jeopardise the device's integrity or reveal important information to unauthorised parties. Using a microcontroller-based approach, the model seeks to accomplish the bypass operation in a secure and non-invasive manner, reducing the danger of data loss or security breaches that could occur with more intrusive approaches like rooting or software exploits.

Furthermore, the suggested approach seeks to provide an affordable solution that is available to a broad spectrum of consumers, irrespective of their level of technical proficiency or available resources. The model aims to reduce the obstacles to entry for users who want to incorporate the bypass mechanism into their devices by utilising open-source software libraries and widely accessible microcontroller hardware. For people who might not have access to specialised tools or technological help, this accessibility is especially crucial.By giving users precise instructions and documentation on how to utilise the bypass mechanism in a safe and responsible manner, the concept also seeks to encourage responsibility and openness. To help users make wise decisions, this contains recommendations for best practices in risk mitigation, data backup, and device security.

The overall goal of the suggested architecture for utilising a microcontroller to get around Android pattern locks is to prioritise user privacy, accessibility, and transparency in addition to offering a workable way to access protected devices. By taking these factors into account, the model seeks to provide a thorough and user-focused method for resolving pattern lock-related issues on Android devices. The ATTiny 85 microcontroller is used to implement pattern cracking for Android Pattern Bypassing.

ATTINY 85 USB:

As the central processing unit in charge of coordinating the entire procedure, the ATTiny85 microcontroller plays a crucial part in the workflow for getting around Android pattern locks. Its duties include a number of vital functions that are essential to the bypass mechanism's effective operation. Initially, the ATTiny85 establishes communication through its USB interface and serves as the receiver of commands

from the computer system. Usually, these commands contain directions for adding pattern combinations to the lock screen interface of an Android device. In order to interpret the intended activities to be performed, the microcontroller parses the data and extracts pertinent information after receiving the orders.

After processing the commands, the ATTiny85 microcontroller moves on to the execution phase, where it follows the instructions to implement the bypass mechanism. In order to mimic user interactions with the Android device's touchscreen interface, signals and control sequences must be generated. The microcontroller uses a brute force method to get over the security lock by methodically injecting pattern combinations into the device's lock screen. The ATTiny85 works closely with the Android handset through its USB connection during the bypass procedure. It transmits the simulated user inputs to the device's lock screen interface, including touch motions that match pattern combinations. The microcontroller keeps an eye on the device's reactions in real time and modifies its behaviour in response to input. Until the right unlock pattern is found, this iterative process keeps going.

Injecting Brute Force Code:

One of the most important steps in the process of getting around pattern locks on Android devices is introducing brute force code. This part involves the methodical execution of specially written code designed to mimic user input on the touchscreen interface of the device. The ATTiny85 microprocessor, which powers the injection procedure, is at the centre of this activity. Custom code created especially for the brute force attack is programmed into the ATTiny85 microcontroller. The microcontroller is instructed by this code to simulate user inputs on the Android device's lock screen, including touch movements and pattern entries. Sending commands and interacting with the device is made possible by the microcontroller's access to the touchscreen interface through a USB interface.

The ATTiny85 microcontroller begins carefully attempting different pattern combinations one after the other when it is told to initiate the brute force attack. The injected code sets the pattern entry sequence, which can range from simple to complex combinations, as the microcontroller cycles through multiple permutations in a brute force manner. Throughout the injection operation, the microcontroller closely monitors the device's response to each pattern entry. It evaluates the device's feedback, including whether the entered pattern is accepted or refused, to determine the next course of action. If the pattern entered is wrong, the microcontroller adjusts its strategy and advances to the next combination in the sequence.

Until the right unlock pattern is found or a preset threshold is achieved, the brute force injection process keeps going iteratively. In the former case, access to the Android device is made possible by the microcontroller successfully getting beyond the pattern lock. On the other hand, the injection procedure might be stopped, and other approaches might need to be taken into consideration if the threshold is reached unsuccessfully. In general, introducing brute force code into the Android smartphone is a methodical and regulated way to get around pattern locks. This component uses the ATTiny85 microcontroller's programmable features to run custom code that simulates user interactions. It repeatedly tries various pattern combinations until it gains successful access.

Plug Android to Devices:

An essential component of the bypassing procedure is the Android device's USB connection, which allows the ATTiny85 microcontroller and the target device to communicate seamlessly. The microcontroller can communicate with the Android device's lock screen interface thanks to this physical connection, which acts as a conduit for data transfer and command execution. A USB cable, which creates a direct connection between the ATTiny85 and the Android device, is the primary component of this connection. Standard USB connectors are usually found on both ends of the USB cable, making it compatible with a variety of devices. The microcontroller can successfully communicate with the Android device thanks to the USB cable's ability to supply power and data communication once it is connected.

The microcontroller and the Android device exchange data in both directions via the USB connection. This implies that the device can

receive feedback and replies from the microcontroller in addition to orders and instructions. Because it allows the microcontroller to track the device's responses and modify its operations accordingly, this bidirectional connection is crucial for carrying out the bypass mechanism. The USB connection allows the Android device to power the microcontroller in addition to transferring data. This guarantees that the microcontroller stays switched on and functional during the bypassing procedure, enabling the bypass mechanism to be executed continuously. As a result, the USB connection has two functions: it supplies power to enable communication and offers data connectivity.

All things considered, the Android device's USB port serves as a vital component of the circumvention process, facilitating smooth data transfer and communication between the target device and the ATTiny85 microcontroller. This connection gives the microcontroller access to the lock screen interface of the device, enabling the bypass mechanism to be executed and, eventually, offering a technique to unlock locked Android smartphones.

Attempt to Unlock Device:

After the ATTiny85 microcontroller injects brute force code, the attempt to unlock the Android smartphone is a crucial step in the bypassing procedure. The system enters the unlocking phase after the custom code has been entered into the lock screen interface of the device.

The ATTiny85 microcontroller acts as the main executor during this stage, simulating user inputs by delivering signals to the Android device that correspond to different pattern combinations. The microcontroller uses its programmable capabilities to carefully plan the pattern entry sequence, cycling through several permutations in a methodical way. By simulating a user trying to manually unlock the device, each pattern entry is sent to the device through the USB connection. The algorithm keeps a careful eye on how the device reacts to each pattern entry as it moves forward with the unlocking attempts. To decide what to do next, the microcontroller decodes the device's feedback, which includes whether the entered pattern was accepted or rejected.

Until the right unlock pattern is found or a preset threshold is achieved, the unlocking procedure iteratively continues. In the former case, the Android device is accessed after the system successfully gets around the pattern lock. On the other hand, the system could need to reconsider its strategy and look into different ways to unlock the device if the barrier is approached without success. Overall, when brute force code was injected, the attempt to unlock the Android device shows a methodical and controlled method of getting over pattern locks. The technique provides a workable way to unlock locked Android smartphones by iteratively testing various pattern combinations until successful access is obtained through careful coordination and monitoring by the ATTiny85 microcontroller.

Algorithm Code:

1. Initialisation (setup):

The ATTiny85 microcontroller and other required parts are initialised by the setup function. Updating the Digi Keyboard library, which offers features for simulating keyboard input, is part of this.

2. Main loop:

The ATTiny85 microcontroller coordinates the brute force attempt to unlock the Android device in the main loop (loop ()).

3. Counting Attempts:

The microcontroller records how many times the user tries to open the

gadget that makes use of the 'count' variable. This enables the system to manage situations in which the maximum number of attempts is reached and to restrict the number of attempts.

4. Managing Maximum Attempts:

The microcontroller enters a block of code to deal with this scenario when the number of attempts hits 5, signifying that the maximum number of attempts has been achieved. In order to avoid detection and keep the device from locking out

subsequent attempts, it starts a 31-second pause before trying again.

5.Brute Force Keystrokes:

The ATTiny85 microcontroller uses the Digi Keyboard library to transmit keystrokes to the Android device in a methodical way. It tries to unlock the device's pattern lock by sending keystrokes that correspond to various digit combinations.

6. Incrementing Digits:

The microcontroller increases the digits in accordance with each set of keystrokes. Variables (a, b, c, d, e, f, g, h) that correspond to each pattern lock digit govern this operation. A digit is incremented to the next one and resets to 0 if it reaches 9.

7. Resetting Digits:

The microcontroller resets the attempt count and begins anew if the first digit (a) hits 9, signifying that every possible combination of eight digits has been tried. This guarantees that, if required, the system can keep trying to unlock the device.

8. End of Loop:

Until successful access is obtained or the maximum number of attempts is reached, the main loop keeps going endlessly, methodically trying various digit combinations.

System Requirements:

1.Hardware:

The ATtiny85 is a microcontroller from Atmel (now part of Microchip Technology) of the AVR family. When you state "Attiny 85 USB," you are most likely referring to a specific use case in which the ATtiny85 microcontroller is designed to serve as a USB device. A TTL (transistor-transistor logic) converter is a device that converts signals between TTL voltage levels and other voltage standards, which are commonly used in digital electronic circuits. It ensures compatibility across systems with differing voltage requirements. A TTL to RS-232 converter, for example, allows TTL-level microcontrollers to communicate with devices that use RS-232 voltage levels. These converters serve an important role in integrating components that use different voltage standards, allowing for seamless communication and interaction within electronic systems.

An Android device is fundamentally a smartphone that operates on Google's Android platform. Android devices are popular due to their operating system being open source, having a vast number of available applications, and their ability to be modified. These devices typically feature touch screens, support for different systems, and connectivity options such as Wi-Fi and mobile networks. Android smartphones are widely used across the globe, and variety of companies engineer them with specific features and designs. Apps, widgets and mobile settings can be changed by users to personalize their experience with their Android device.

2.Software:

An Integrated Development Environment (Ide) is an environment for coding software which is uploaded to microcontroller, here Arduino Microcontroller. Creating programs, their modification, uploading to Arduino boards, in brief, everything that concerns programming is available for users in the Arduino Ide. Essential features are: text pane for code composition, message pane to display useful feedback and some buttons for common functions. The programming language of all IDE variations, however, is C/C++. Any of the several USB ports available allows the user to send code to the microcontroller onto Arduinos readily. The Arduino Ide makes the tasks of developing projects much easier. The Arduino Ide is interface-free and shapes a design space through standardizing the projects' components, as well as the way in which they are connected to each other.

3.Arduino IDE:

The Integrated Development Environment (IDE), also referred to as the Arduino IDE, is a software platform that was developed specifically for interfacing with an Arduino microcontroller. With the help of this IDE, users are able to utilize a range of tools intended to improve the ease of coding and development. The core of the IDE is the integrated

text editor, which in fact implements a separate space for writing and modifying the Arduino code. This editor is accompanied with a message section which compliments this editor with helpful comments on what went wrong during the code compilation and uploading stages. In addition, there is a toolbar within the IDE which houses buttons/ icons to easily access commonly used functions which helps in effective coding. One notable characteristic of the Arduino IDE is that it features the Arduino programming language, a dialect derived from C/C++.

Furthermore, the Arduino IDE streamlines the integration of third-party libraries, allowing users to simply expand the capabilities of their projects with new functionality. This versatility is essential for accessing the enormous ecosystem of Arduino-compatible sensors, modules, and shields on the market, allowing developers to tackle a wide range of projects with relative simplicity. Furthermore, the Arduino IDE integrates seamlessly with Arduino boards, automatically identifying connected devices and configuring the necessary settings for easy programming and debugging. This level of automation lowers the barrier to entry for newbies while speeding up the development cycle for experienced users.

4.Digi Spark:

The Digi Spark is a compact and flexible microcontroller board distinguished by its small size and USB connectivity. This small board is especially popular since it is compatible with the Arduino IDE, allowing users to take advantage of the ease of Arduino programming for a variety of projects. The Digi Spark contains a microcontroller that can be programmed using the Arduino programming language. This makes it accessible to users with diverse degrees of programming experience, which is consistent with the user-friendly attitude of Arduino development. One distinctive feature of the Digi Spark is its USB connectivity, which not only makes programming easier but also opens up possibilities for USB-based projects. Users can use the Digi Spark to simulate Human Interface Devices (HID) or construct custom USB functionality, expanding the range of possible applications.

This step is critical to a seamless programming and uploading experience. Given its small size and versatility, the Digi Spark is useful in a variety of projects, particularly those with limited space or that require USB connectivity. The active Digi Spark community helps to the platform's evolution by sharing projects, tutorials, and support, which improves the overall development experience for users exploring the potential of this small yet powerful microcontroller.

5. programming Language:

In the world of Android pattern bypass using an Attiny85 microcontroller, the importance of the C language is demonstrated by its close interface with hardware and efficiency in managing low-level tasks. The technique of circumventing the Android pattern lock requires complicated communication between the Attiny85 microcontroller and the Android device's touchscreen, as well as precise control over hardware components. The C programming language allows this connection by allowing developers to directly modify the Attiny85's GPIO pins, allowing signals to be transmitted to the touchscreen controller in order to properly replicate human input events.

Efficient code execution is critical in situations where resources are restricted, such as with the Attiny85 microcontroller. C's ability to generate optimised code ensures that the microcontroller can do the essential activities quickly and with little resource consumption. This efficiency is critical for enabling smooth pattern bypass without significantly exhausting the microcontroller's battery, hence increasing the device's overall usability. Furthermore, C may manage peripheral devices attached to the microcontroller, such as LEDs or buzzers, which are used to provide feedback throughout the pattern bypass process. By interacting with these peripherals via C code, developers can build sensible user feedback systems, improving the device's user experience and usability.

C's portability and interoperability contribute to its popularity in this environment. C code for the Attiny85 microcontroller can be readily converted to work on other hardware platforms if necessary. This flexibility not only speeds up the

development process, but it also allows for smooth integration with a wide range of microcontroller architectures, giving you more hardware options for different projects or applications. Furthermore, C's capability for interrupt management is useful in circumstances where the Attiny85 must respond quickly to external events or timers. Developers can efficiently manage these events by using interrupt service routines (ISRs) written in C, allowing the microcontroller to execute duties such as timing operations and responding to user input in a timely way.

Future Scope and Limitations:

1. Attiny85 enhances pattern bypass capabilities in brute force attacks, improving security bypass methods.
2. Its small size and low power consumption enable discreet data capture from devices, facilitating forensic analysis and security testing.
3. Ethical considerations and limitations should be acknowledged when using Attiny85.
4. While powerful, Attiny85's potential to inject viruses or malicious code into devices poses significant risks.
5. Ongoing research and vigilance are essential as technology evolves to counteract emerging security threats in cybersecurity.

Conclusion:

In conclusion, a viable solution to the security issues encountered by users who can forget their pattern locks is the creation of an Android pattern bypass using a microcontroller. We have effectively shown through this research that it is possible to combine software and hardware elements to produce a dependable and effective workaround. We have given consumers a practical way to get back into their gadgets without sacrificing security by utilising a microcontroller's capabilities. This project can be improved in a number of ways going forward to increase its usefulness and functionality. Adding further authentication layers to make sure that only authorised users can access the bypass capability is one possible improvement. This can entail using a fingerprint or other biometric authentication.

Furthermore, improving the bypass mechanism's user interface and experience has the potential to dramatically increase its utility. This could entail creating a separate mobile app that easily connects with the bypass functionality, giving users a more natural and streamlined experience. Additionally, investigating the incorporation of cloud-based backup and synchronisation functions may improve the project's resilience and accessibility. This would allow users to securely store their pattern lock data and bypass credentials, allowing them to regain access to their devices from anywhere with an internet connection. Overall, by iterating and improving on the current solution, we can ensure that the Android pattern bypass using a microcontroller project remains relevant and effective in meeting the changing security needs of customers.

References:

1. K. Barmpatsalou, D. Damopoulos, G. Kambourakis, and V. Katos, "A critical review of 7 times of mobile phone exploration," Digital Investigation, vol. 10, no. 4, pp. 323-349, 2013.

2. CCL Group Ltd, the UK's leading exploration and consultancy firm in 2016.

3. E. Onyejegbu, A. Dorzhigulov, and A.P. James, "Biometric Pixel Fusion Crossbar," 2021 IEEE International Symposium on Circuits and Systems (ISCAS), pages 1–5, 2020.

4. Yeun Ku, Leo Hyun Park, Soyeon Shin, and Tekyoung Kwon, "As reported behavioural enrolment for mobile stoner authentication," IEEE Access, Volume 7, runners 69363–69378, 2019.

5. Puninder Kaur, Geeta, and Vidhu Kiran Sharma, "Analysis of Secure Locking Techniques on Smart Phones," 2022 5th International Conference on Contemporary Computing and Informatics (IC3I), pp. 1807-1811.

6. Altaf Khan, Alexander G. Chefranov, “Captcha-based graphical encryption with strong password space and usability study,” 2020 International Conference on Electrical, Communications and Computer Engineering (ICECCE), p. 1-6, 2020.

7.P. Dibb and M. Hammoudeh, "Forensic Data Recovery from Android OS Devices: An Open Source

Toolkit", 2013 European Intelligence and Security Informatics Conference, 2013.

8. Jeff Lessard and Gary Kessler, "Android Forensics: Simpliying Cell Phone Examinations", Small Scale Digital Device Forensics Journal, vol. 4, no. 1, pp. 1941-6164, 2010.