One-Time Passwords: Security Against Eavesdropping

In Partial Fulfillment of the Requirements
for Course Project

CS 409 Introduction to Cryptography

Supervised By: Prof. Sruthi Sekar

Written By: Parth Nawar and Rishabh Ravi

Date: November 24, 2024

Abstract

One-Time Passwords: Security Against Eavesdropping

This report provides an analysis of One-Time Password (OTP) security protocols—HOTP, TOTP, and S/Key—against eavesdropping. HOTP and TOTP are widely used for generating OTPs in authentication processes, with HOTP utilizing a counter and TOTP synchronizing OTPs with time. Both protocols offer secure mechanisms for password generation, but they are vulnerable to issues like synchronization errors and infrequent usage. The report then examines the S/Key protocol, an earlier OTP system that employs a cryptographic hash chain for secure authentication. While S/Key's reliance on pre-image and second pre-image resistance provided enhanced security compared to previous methods, vulnerabilities such as outdated hash functions and susceptibility to man-in-the-middle (MITM) attacks remain significant concerns. The analysis highlights how each OTP protocol addresses eavesdropping risks through strong cryptographic properties, but also reveals their respective limitations in real-world scenarios. Ultimately, the report illustrates the evolution of OTP systems from S/Key to modern protocols like HOTP and TOTP, emphasizing their security mechanisms against eavesdropping.

Acknowledgements

We want to express our sincere gratitude towards Prof. Sruthi Sekar for allowing us to dive deeper into the cryptographic techniques of One-Time Passwords.

Contents

List of Figures

Chapter 1

Introduction

Security against eavesdropping focuses on protecting communications from unauthorized interception. It involves encrypting data, using secure networks, and adopting strong authentication methods to safeguard sensitive information. Aiming to develop ID protocols secure against eavesdropping, we come across OTPs and S/Keys.

HOTP (Hash-based One-Time Password) and TOTP (Time-based One-Time Password) are secure authentication methods generating unique, temporary codes. HOTPs rely on counters, while TOTPs are time-synchronized. Both enhance security by reducing reliance on static passwords.

S/Key (Secure Key) is another pioneering one-time password (OTP) system designed to strengthen authentication against eavesdropping. Unlike HOTP and TOTP, which generate OTPs dynamically using counters or timestamps, S/Key relies on a cryptographic hash chain. It uses a secret seed to compute a series of hash outputs, each serving as a unique password. The server verifies the OTP by applying the same hash function, ensuring that even if an OTP is intercepted, it cannot be reused or reverse-engineered. Although largely replaced by more modern methods, S/Key was instrumental in demonstrating the practical application of cryptographic principles in early secure authentication systems.

Chapter 2

Background

2.1 Attacker Model

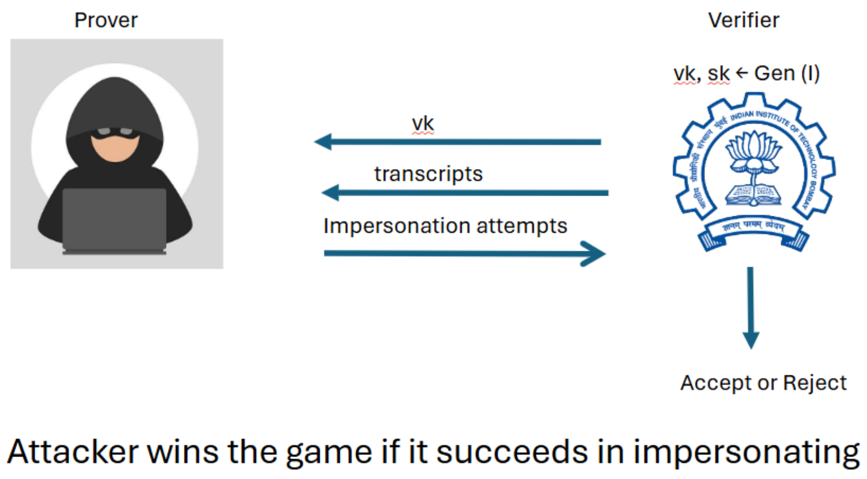

Figure 2.1: Diagram depicting Attack Model

The model comprises of three phases

- *Key generation phase* The verifier generates the key
- *Eavesdropping phase* Where an adversary eavesdrops an interaction between a prover and verifier
- *Impersonation attempt* The adversary tries to impersonate the prover to get unauthorized access

We say that the adversary wins the game if the Verifier outputs accept at the end of the interaction. We take the practical case of an IITB student logging into sso.iitb.ac.in as the interaction between a prover and verifier. The computer center verifies, ensuring that only authorized personnel access the site. The student is the prover, trying to prove its authorization to access the site.

2.2 Defining Security

Define adversary A's advantage with respect to protocol I, denoted ID2adv[A, I], as the probability that A wins the game.

Definition 1. *We say that an identification protocol I is secure against eavesdropping attacks if, for all efficient adversaries A, the quantity ID2adv[A, I] is negligible*

Keeping *vk* secret is the first protocol against eavesdropping, but it motivates a weaker attack. We must now allow the adversary to make multiple impersonation attempts. The impersonator wins if at least one of its impersonation attempts is accepted by the verifier. The reason is that some information about *vk* could leak through the interactions between the prover and verifier. Let wID2adv[A, I] denote the adversary's advantage in winning this weaker version of attack, which gives us a definition

Definition 2. *We say that an identification protocol I is weakly secure against eavesdropping attacks if, for all efficient adversaries A, the quantity wID2adv[A, I] is negligible.*

This requires the protocol to be stateful. In a stateful protocol, after each invocation of the protocol, the pair (*vk, sk*) changes: the prover P updates *sk*, and the verifier V updates *vk*. However, we shall assume that V only updates *vk* if it outputs accept.

With the definitions made clear and the attack model set, we proceed to look at protocols.

Chapter 3

PRF Based One Time Passwords

The simplest ID protocols secure against eavesdropping attacks are called one-time password protocols.

HOTPs are a weakly secure ID protocol that stands for Hash-based One-time-password

3.1 HOTPs

3.1.1 Algorithm

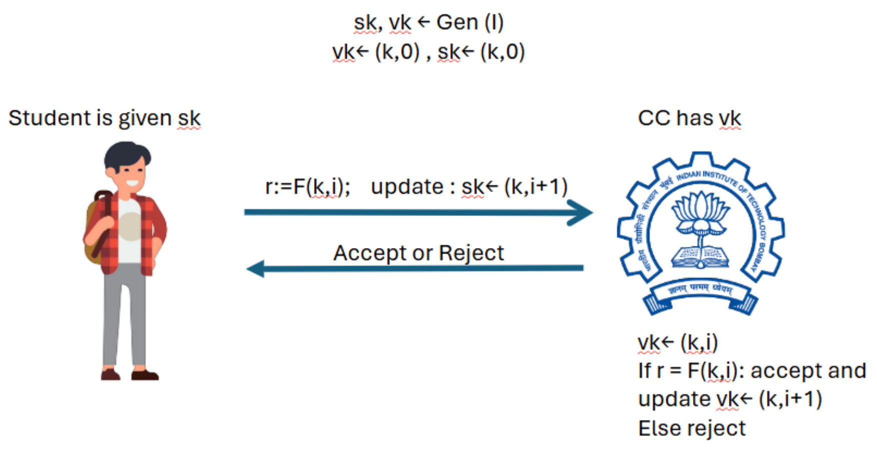

Figure 3.1: Working of HOTPs

Let F be a PRF defined over $(\kappa, \mathcal{Z}_N, Y)$ for some large integer N, say N $=$ 2^{128}. This F is used to update the password after every successful invocation. The HOTP protocol HOTP $= (G, P, V\ )$ works as follows:

- G: choose a random κ and output $vk = (k, 0)$ and $sk = (k, 0)$
- The prover (student) given sk and the verifier (CC) given vk, the prover in the i^{th} step sends the message $r = F(k, i)$ to the verifier (CC) and updates the key sk as $sk = (k, i + 1)$.
- The verifier (CC) checks if the received r satisfies $r = F(k, i)$ and accordingly outputs accept or reject. If it does accept it updates vk as $vk = (k, i + 1)$

Here, both vk and sk must be kept secret, and therefore, HOTP is only weakly secure against eavesdropping.

Definition 3. *If the PRF F is defined over $(\kappa, \mathcal{Z}_N, Y)$, for a super-poly N and $|Y|$, HOTPs are weakly secure.*

With the following properties of the PRF, the attacker cannot differentiate between a completely random function F and the PRF. In such a case, the impersonation probability falls to $1/|Y|$. As $|Y|$ is super-poly, this approaches zero. The requirement of N being super-poly is to ensure the values don't wrap around too often.

3.1.2 Usage

HOTP can be used in a car key fob system to wirelessly unlock a car, VPN Access - authentication for remote workers, Account recovery codes in email services, access to admin dashboards in Cloud Services, and in Banking apps.

In all applications, the functionality remains the same

- The secret PRF key k is stored on the verifier and at the prover's side
- Every time the user uses the key, the internal counter by one and sends the derived one-time password to the verifier, along with the previous counter

3.1.3 Drawbacks

Firstly, in HOTPs, the counter value is needed to synchronize the token and the remote server in case they go out of sync. This increases the message size.

Secondly, HOTPs are updated only after every use. If the user authenticates infrequently, the OTP would be unchanged for an entire month, providing adequate time for a forgery attack.

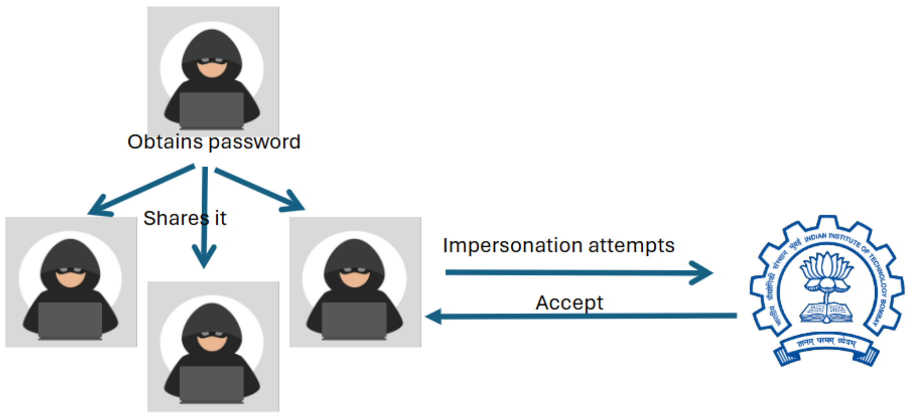

Figure 3.2: Attacker selling the recovered OTP

With the following drawbacks, we proceed to the improved ID protocol Time-based One Time Passwords (TOTPs)

3.2 TOTPs

3.2.1 Algorithm

In TOTP, the counter is incremented by one every 30 seconds, whether the user authenticates or not. Apart from this, the algorithm remains the same as HOTPs. This means that every one-time password is only valid for a short time.

Whenever the user authenticates to the remote server, the server uses the current time to determine the value of the counter. To account for clock skew between the server and the token, the server will accept any of $F(k,(i-c)),...,F(k,(i+c))$ as valid passwords for a small value of c

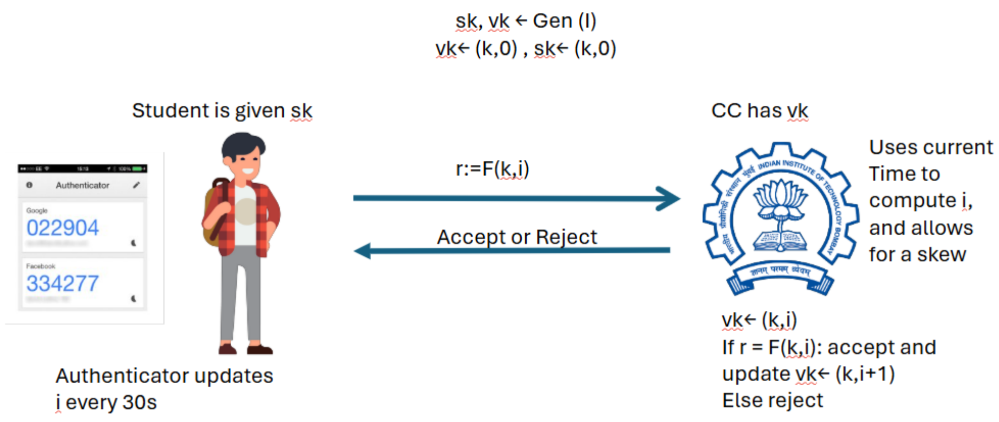
sk, vk ← Gen (l)
vk← (k,0) , sk← (k,0)
Student is given sk
CC has vk
022904
334277
r:=F(k,i)
Accept or Reject
Uses current Time to compute i, and allows for a skew
Authenticator updates i every 30s
vk← (k,i)
If r = F(k,i): accept and update vk← (k,i+1)
Else reject

Figure 3.3: Working of TOTPs

3.2.2 Usage

TOTPs are used in 2FA (2-factor authentication) for apps and accounts using Google Authenticator and Microsoft Authenticator, Time-based transaction approvals in online banking, secure wallet access in cryptocurrency wallets, and time-limited access to sensitive data.

3.2.3 Drawbacks

Some of the drawbacks of TOTPs are

- Requires accurate clock synchronization
- Limited Usability Window
- Cannot be revoked once generated
- Loss of device means loss of access
- Implementation Complexity involving secure storage of shared secrets

Additionally, in a number of well-publicized cases, an adversary steals *vk* without being detected, and then all security is lost. This motivates us to look for a better ID protocol that overcomes these drawbacks

Chapter 4

S/Key

We will discuss how S/Key works, the cryptographic properties of S/key, security, and potential attack,s and limitation

4.1 How S/Key Works

S/Key, as an early one-time password (OTP) system, relies on hash chains to provide secure authentication over an untrusted network. Here is a detailed step-by-step explanation of the S/Key protocol:

4.1.1 Initialization

The user (Prover) and the server (Verifier) agree on the following during initialization. We can consider this to be equivalent to registration/account creation.

- A secret key S known only to user
- A secure cryptographic Hash function, $H(x)$
- The desired number of OTPs generation, N

The user computes the initial hash chain starting from the secret S as follows:

$$K_N = H^N(S), \quad \text{where } H^i(x) = H(H^{i-1}(x)) \text{ and } H^0(x) = x.$$

This means $K_N = H(H(\ldots H(S)))$ is the hash applied N times.
The final value K_N is sent to the server during the registration phase.

4.1.2 Authentication

To authenticate for the i-th session, the user sends K_{N-i} as the OTP to the server. The server verifies the OTP by applying the hash function H to the received value and comparing it to the previously stored OTP:

$$H(K_{N-i}) \stackrel{?}{=} K_{N-i+1}.$$

If the comparison is successful, the server updates its stored OTP to K_{N-i} and grants access. Otherwise, authentication fails.

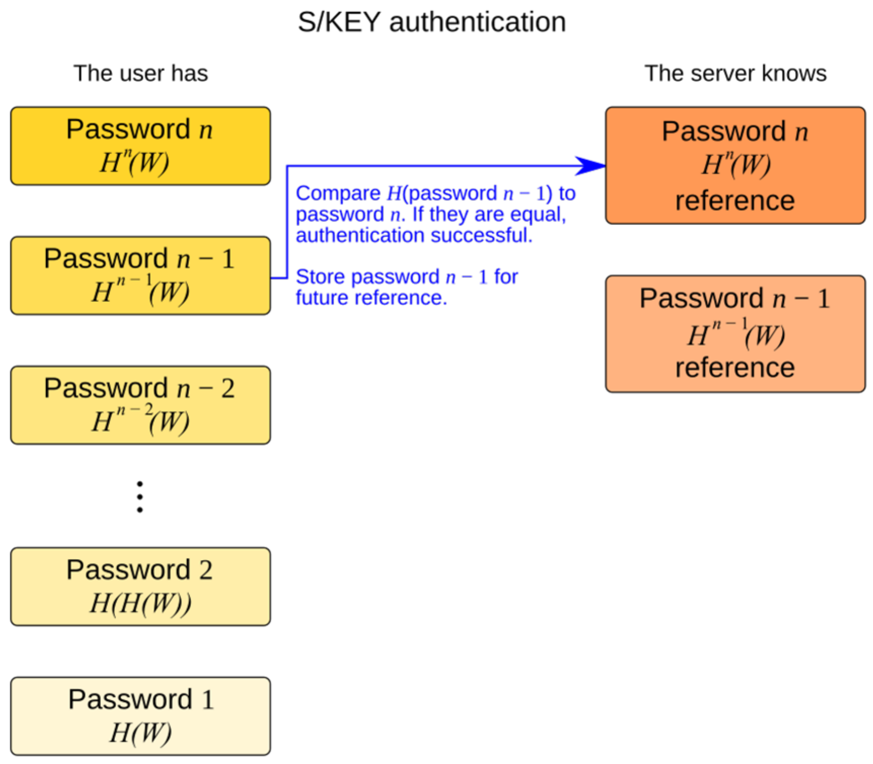
S/KEY authentication
The user has
The server knows
Password n $H^n(W)$
Password $n-1$ $H^{n-1}(W)$
Password $n-2$ $H^{n-2}(W)$
Password 2 $H(H(W))$
Password 1 $H(W)$
Compare H(password $n-1$) to password n. If they are equal, authentication successful.
Store password $n-1$ for future reference.
Password n $H^n(W)$ reference
Password $n-1$ $H^{n-1}(W)$ reference

Figure 4.1: S/key authentication

4.2 Cryptographic Properties of S/Key

4.2.1 Pre-image Resistance

Pre-image resistance ensures that, given a hash output $y = H(x)$, it is computationally infeasible to determine the original input x. Formally, the probability of an adversary successfully finding x is negligible:

$$\Pr[\text{Adversary finds } x : H(x) = y] \leq \epsilon(n),$$

where $\epsilon(n)$ is a negligible function of the computational power n.
In the S/Key system, pre-image resistance guarantees that even if an attacker intercepts an OTP $H^n(S)$, they cannot compute the secret S or any earlier OTPs, ensuring forward security.

4.2.2 Second Pre-image Resistance

Second pre-image resistance ensures that, given a specific input x, it is computationally infeasible to find another input $x' \neq x$ such that $H(x) = H(x')$. Formally:

$$\Pr[\text{Adversary finds } x' : H(x') = H(x)] \leq \epsilon(n).$$

In S/Key, this property prevents an attacker from forging another valid OTP, even if they intercept a legitimate one.

4.3 Resistance to Brute Force

Brute force attacks aim to invert the hash function $H(x)$ or exhaustively guess the original seed S. The S/Key system resists brute force attacks due to the computational infeasibility of reversing $H(x)$. For a secure hash function, the expected time to find a pre-image is 2^λ, where λ is the output length of the hash function.
Formal Argument: Let $P_{\text{brute-force}}$ denote the success probability of deriving S or $K_{N-(i+1)}$ within t attempts:

$$P_{\text{brute-force}}(t) = \frac{t}{2^\lambda}.$$

For practical security, $t \ll 2^\lambda$, ensuring:

$$P_{\text{brute-force}}(t) \leq \text{negl}(\lambda).$$

4.4 Security Analysis under Eavesdropping

4.4.1 Protocol Flow and Security Analysis

1. Initial Authentication Request

$$A \xrightarrow{\text{OTP}_{N-i}} B$$

Description: Alice sends OTP_{N-i}, the i-th OTP in the chain, to Bob for verification.
Relevant Cryptographic Properties:

- *Pre-image resistance*: Ensures that even if the adversary $\mathcal{A}$ intercepts OTP_{N-i}, they cannot compute the previous OTP, $\mathrm{OTP}_{N-(i+1)}$, or the seed S.
- *Collision resistance*: Prevents the adversary from finding a different input that produces the same hash, mitigating forgery attempts.

Attack Analysis:

- The intercepted OTP OTP_{N-i} becomes invalid immediately after its use, nullifying replay attacks.
- The probability of inverting $H(x)$ to compute earlier values is negligible:
$$P_{\text{prev}} \leq \mathrm{negl}(\lambda).$$

2. Verification and Response

$$A \xleftarrow{\text{ACK/REJECT}} B$$

Description: Bob verifies OTP_{N-i} by applying the hash function H to his stored $\mathrm{OTP}_{N-(i+1)}$. If $H(\mathrm{OTP}_{N-i}) = \mathrm{OTP}_{N-(i+1)}$, Bob sends an acknowledgment (ACK); otherwise, he rejects.
Relevant Cryptographic Properties:

- *Second pre-image resistance*: Ensures that even if $\mathcal{A}$ has $\mathrm{OTP}_{N-(i+1)}$, they cannot compute a different OTP'_{N-i} such that $H(\mathrm{OTP}'_{N-i}) = \mathrm{OTP}_{N-(i+1)}$.
- *Replay resistance*: Inherent due to the hash chain, as each OTP is unique and invalid after a single use.

Attack Analysis:

- Observing ACK/REJECT provides no information about $\mathrm{OTP}_{N-(i-1)}$ due to forward security.
- The probability of forging a valid OTP remains negligible.

3. Subsequent Authentication

$$A \xrightarrow{\text{OTP}_{N-(i-1)}} B$$

Description: Alice uses the next OTP, $\text{OTP}_{N-(i-1)}$, in her chain for the next authentication round.
Relevant Cryptographic Properties:

- *Pre-image resistance*: Ensures $\mathcal{A}$ cannot predict $\text{OTP}_{N-(i-1)}$ from OTP_{N-i}.
- *Forward security*: Guarantees that knowledge of OTP_{N-i} does not compromise any subsequent OTPs.

Attack Analysis:

- Even with unlimited computational resources, $\mathcal{A}$ has only a $\frac{1}{2^\lambda}$ probability of guessing $\text{OTP}_{N-(i-1)}$, where λ is the output length of the hash function:
$$P_{\text{future}} = \frac{1}{2^\lambda}.$$

4.4.2 Key Takeaways on Eavesdropping

- **Intercepting OTPs**: The cryptographic properties of the hash function—pre-image resistance, second pre-image resistance, and forward security—ensure that an intercepted OTP OTP_{N-i} provides no advantage for computing past or future OTPs.
- **Replay Protection**: Each OTP is valid for one-time use only, rendering intercepted OTPs unusable.
- **Strength of Security**: Using strong hash functions (e.g., SHA-256) ensures that brute-force attacks or attempts to reverse-engineer the hash chain are computationally infeasible.

4.5 Potential Attacks and Vulnerabilities

4.5.1 Birthday Attack

Description: A birthday attack targets the collision resistance of the hash function used in S/Key. It exploits the mathematical probability that, in a set of n randomly generated hash values, a collision can be found with a probability higher than intuition suggests. The Birthday Paradox governs this.

Applicability to S/Key: While S/Key does not rely heavily on collision resistance, hash functions like MD5 (commonly used in early implementations) are vulnerable to this type of attack. If an attacker finds a collision within the hash chain, they might create a duplicate OTP sequence.

Mitigation:

- Use hash functions with a large output size (e.g., SHA-256).
- Avoid outdated hash functions such as MD5 or SHA-1.
- Frequently reinitialize the hash chain with fresh seeds to reduce the risk of collisions.

4.5.2 Man-in-the-Middle (MITM) Attack

Description: In an MITM attack, the adversary intercepts and possibly alters the communication between Alice (the client) and Bob (the server). If the OTP is not transmitted over a secure channel, an attacker might capture the OTP and replay it.

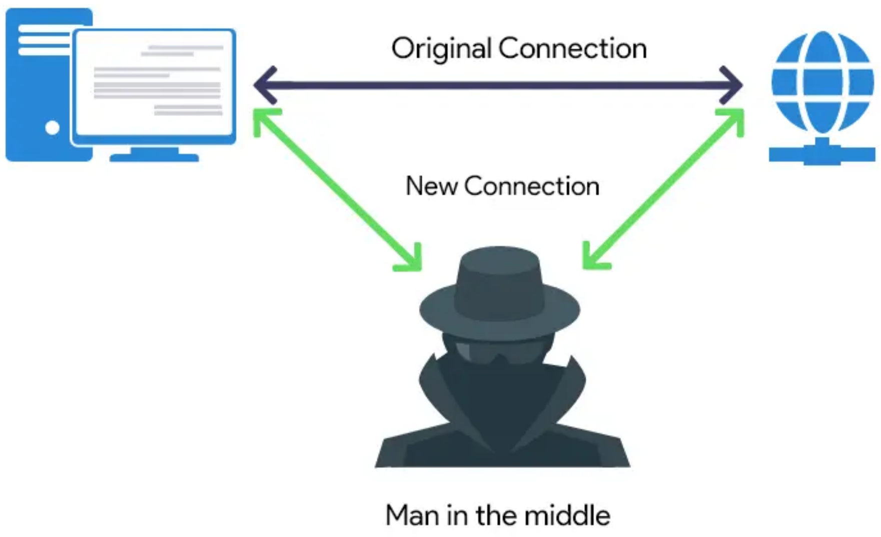

Figure 4.2: Man-in-the-middle attack

Applicability to S/Key: The S/Key system resists replay attacks because an OTP is valid only once. However, if an attacker intercepts OTP_{N-i}, they could disrupt the session

Mitigation:

- Use encrypted communication channels such as TLS for transmitting OTPs.
- Implement additional security layers, such as timestamps or challenge-response mechanisms, to detect and thwart replay or tampering attempts.

4.6 Conclusion

The S/Key system, introduced in the early 1990s, represented a significant advancement in secure authentication by utilizing a one-time password (OTP) mechanism based on a cryptographic hash chain. Its strength lies in its ability to provide secure, ephemeral authentication credentials that reduce the risks associated with static passwords and mitigate issues like password reuse and interception. Historically, S/Key served as a foundation for many subsequent OTP-based systems and helped pave the way for modern alternatives like HOTP and TOTP.
Despite its early success, S/Key has certain limitations, particularly in its reliance on outdated hash functions like MD5, which are susceptible to vulnerabilities such as collision attacks. The lack of inherent protection against man-in-the-middle (MITM) attacks also highlights the need for additional layers of security, such as encrypted communication channels. As the security landscape evolved, modern systems have built upon the principles of S/Key, enhancing them with improved hash functions, better synchronization mechanisms, and more robust attack resistance strategies. Today, systems like TOTP and HOTP, which incorporate these improvements, are more widely used and better suited to the demands of contemporary digital security.
In conclusion, while S/Key remains an important historical reference, modern authentication methods have improved upon its design to address its vulnerabilities and offer enhanced protection for users in an increasingly connected world.

Bibliography

[1] Boneh, Dan, and Victor Shoup. "A graduate course in applied cryptography." Draft 0.5 (2020).

[2] N. M. Haller, "The S/Key one-time password system," Bellcore, Morristown, New Jersey, 1994. Available: `https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=a1836810bddf25f8086d64c15c11369cf3fb7979`.

[3] Wikipedia contributors, "S/KEY," Wikipedia, The Free Encyclopedia, 2024. [Online]. Available: `https://en.wikipedia.org/w/index.php?title=S/KEY&oldid=1215624032`. Accessed: 2024-11-24.

[4] M. Abell, S. Agarwal, et al., "HOTP: An HMAC-based one-time password algorithm," *IETF RFC 4226*, 2005. Available: `https://tools.ietf.org/html/rfc4226`.

[5] M. D. N. Swamy, A. R. D. Thorpe, "The Time-based One-Time Password Algorithm (TOTP) for Two-Factor Authentication," *IETF RFC 6238*, 2009. Available: `https://tools.ietf.org/html/rfc6238`