ELE2742 / ENG1467 - PUC-Rio, 2019.1

Lecture 4: Authentication

DATE

Lecturer: Yi-Fan Tseng *Scribe: Yi-Fan Tseng*

In cryptography, an authentication protocol is a two-party protocol. It allows the receiving entity to authenticate the connecting entity (e.g. Client connecting to a Server) as well as authenticate itself to the connecting entity (Server to a client) by declaring the type of information needed for authentication as well as syntax. There are various types of authentication in cryptography, e.g. nonce-based authentication, password-based authentication, biometrics-based authentication, two-factor authentication, three-factor authentication, etc.

Usually, an authentication protocol consists of two phases:

- **Registration Phase**: In this phase, some secrets are shared between the two parties involved in the authentication protocol.
- **Authentication Phase**: In this phase, the two parties authenticate each other in order to check whether the other party is the one whom the secrets are shared with. Usually, in the end of this phase, a session key will be constructed between the two parties.

There are three common factors in an authentication protocol.

- Something you know (Password)
- Something you have (Smart Card, usually stores a pre-shared secret)
- Something you are (fingerprint, iris)

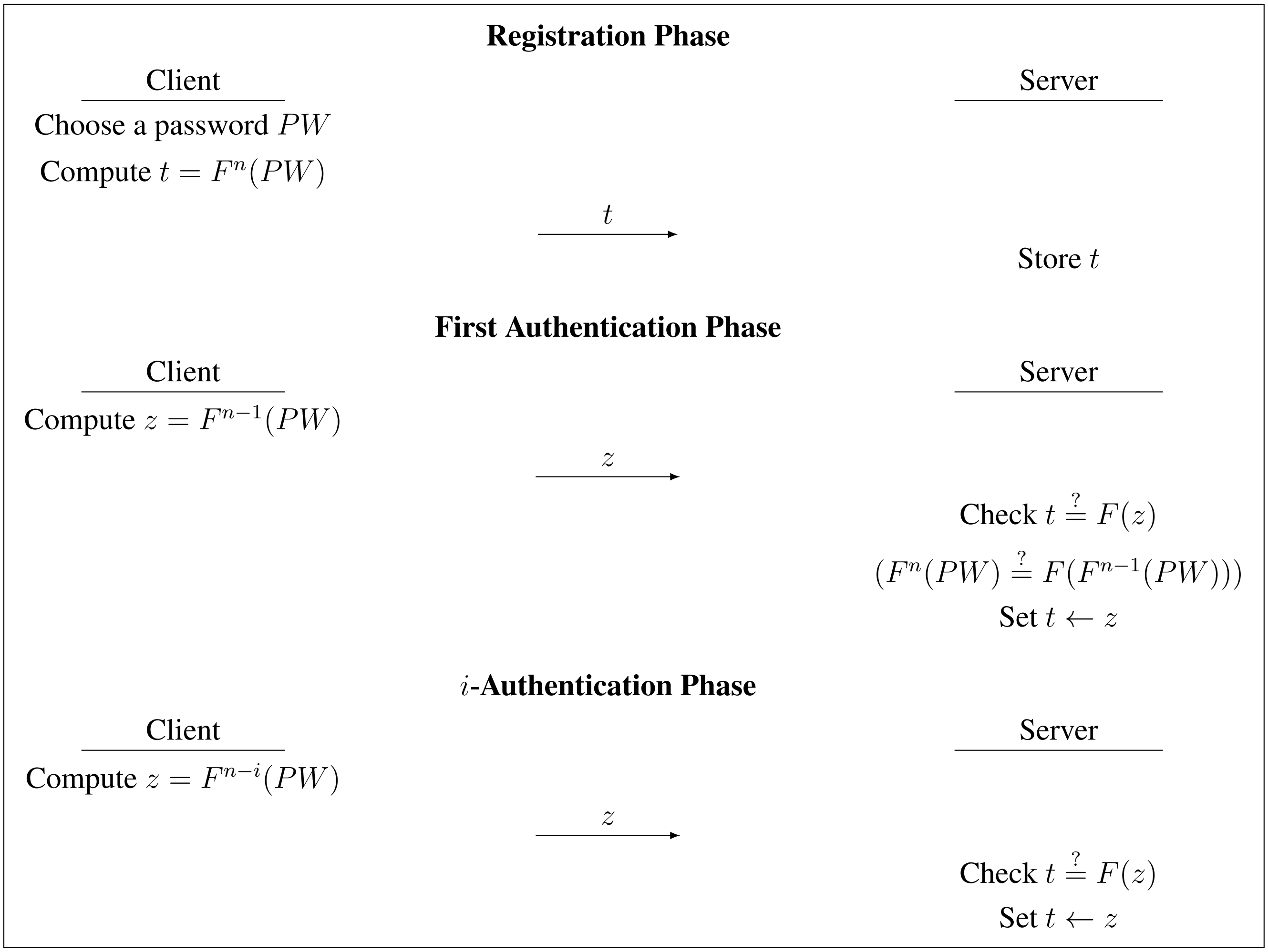
We next discuss the common attacks to an authentication scheme.

- Replay attacks: An attacker intercepts all the packages transmitted between the two parties, and tries to pass the authentication by sending the same communication to one of the parties. It can be prevented by randomizing the communication or padding the message with a time stamp.
- Offline dictionary attacks: The attacker tries to derive the password from the intercepting communications. Such attacks work due to the low entropy of passwords.

In the rest of the this slide, we consider the scenario described below. A client first registers itself to a server in the registration phase. Then in the authentication phase, the client and the server want to authenticate each other. Finally, at the end of the authentication phase, a session key is constructed between the client and the server.

1 Password Authentication

In 1981, Lamport [1] proposed a password authentication protocol using one-way functions, which allows a client to be authenticated with a fix number of times. A password is a low-entropy string, which is able to be remembered by human. Let $F(\cdot)$ be a one-way function, and $F^n(\cdot) = \underbrace{F(F(\dots F(\cdot)))}_{n}$ denote n successive evaluation of F. Figure 1 shows an illustration for Lamport's protocol. The core idea is that, the i-th password is $F^{n-i}(PW)$, where n is the time to login the server. The security of the protocol is based on the hardness to inverse an own-way function. For simplicity, let $n = 1000$. If an adversary knows the z's in first 987 authentications, i.e. $F^{999}(PW), F^{998}(PW), \dots, F^{13}(PW)$. To impersonate the client to pass the next authentication, the adversary needs to compute $F^{12}(PW)$, which is hard to compute from its knowledge due to the one-wayness of F. In practice, F can be replaced by cryptographic hash functions, e.g. SHA3.



Registration Phase

Client | Server

Choose a password PW

Compute $t = F^n(PW)$

$\xrightarrow{t}$

Store t

First Authentication Phase

Client | Server

Compute $z = F^{n-1}(PW)$

$\xrightarrow{z}$

Check $t \overset{?}{=} F(z)$

$(F^n(PW) \overset{?}{=} F(F^{n-1}(PW)))$

Set $t \leftarrow z$

i-Authentication Phase

Client | Server

Compute $z = F^{n-i}(PW)$

$\xrightarrow{z}$

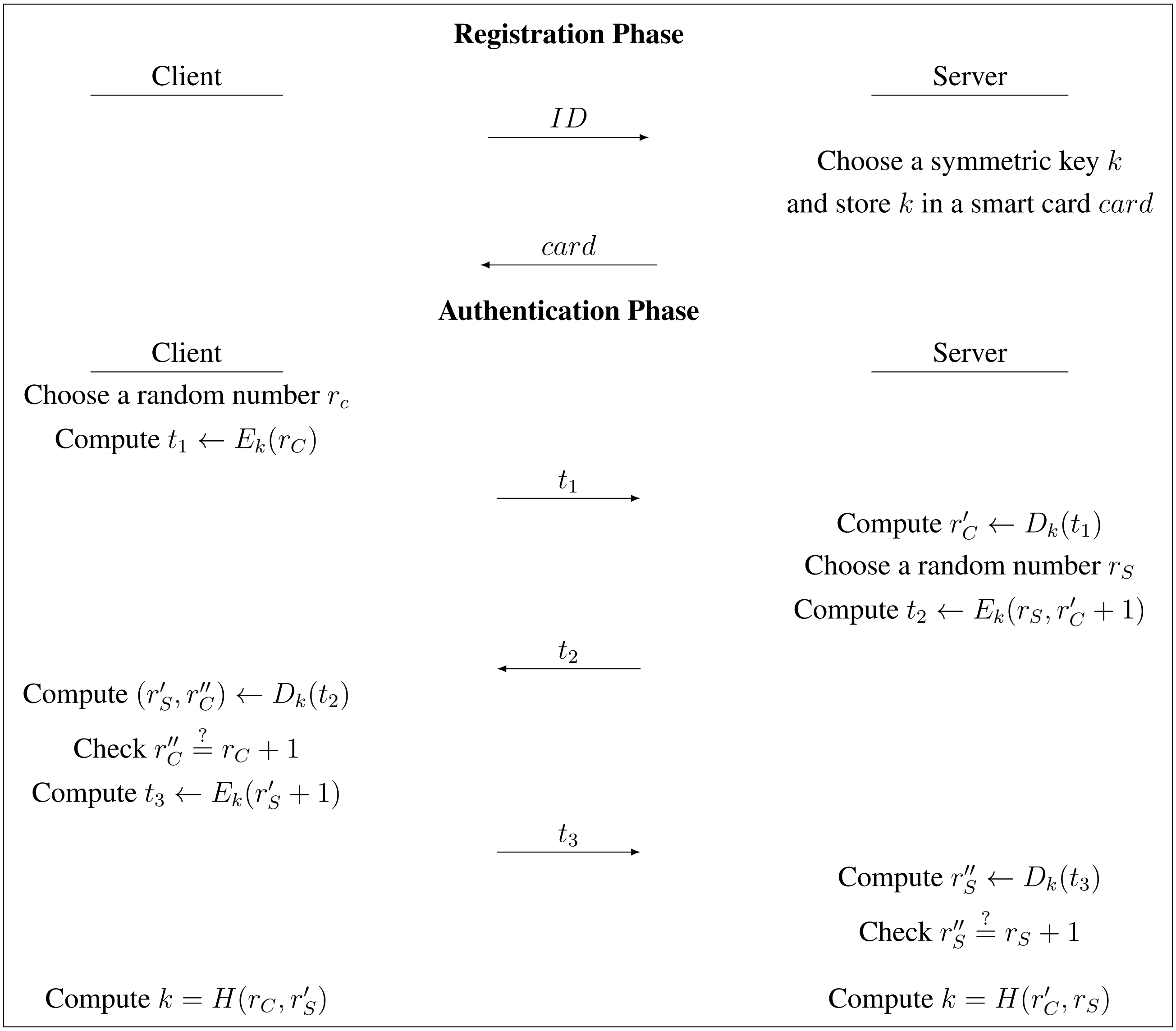
Check $t \overset{?}{=} F(z)$

Set $t \leftarrow z$

Figure 1: Password Authentication

2 Nonce-Based Authentication

Nonce-based authentication, a type of single-factor authentication, is one of the simplest yet most efficient construction among authentication protocols. Here the term “nonce” means random numbers, and thus in such protocols, the two parties sends random numbers encrypted with the pre-shared secret. Let $(E_k(\cdot), D_k(\cdot))$ be a pair of symmetric encryption/decryption algorithm, such as AES, using a symmetric key k, and H be a secure hash function, such as SHA3. Figure 2 shows an example of an nonce-based authentication scheme, which is also known as the “challenge-response paradigm”.



Registration Phase

Client | Server

$\xrightarrow{ID}$

Choose a symmetric key k
and store k in a smart card $card$

$\xleftarrow{card}$

Authentication Phase

Client | Server

Choose a random number r_c
Compute $t_1 \leftarrow E_k(r_C)$

$\xrightarrow{t_1}$

Compute $r'_C \leftarrow D_k(t_1)$
Choose a random number r_S
Compute $t_2 \leftarrow E_k(r_S, r'_C + 1)$

$\xleftarrow{t_2}$

Compute $(r'_S, r''_C) \leftarrow D_k(t_2)$
Check $r''_C \overset{?}{=} r_C + 1$
Compute $t_3 \leftarrow E_k(r'_S + 1)$

$\xrightarrow{t_3}$

Compute $r''_S \leftarrow D_k(t_3)$
Check $r''_S \overset{?}{=} r_S + 1$

Compute $k = H(r_C, r'_S)$ | Compute $k = H(r'_C, r_S)$

Figure 2: Nonce-Based Authentication

The security of the protocol is based on the security of the underlying symmetric encryption scheme. In the first flow, the client sends t_1 to the server as a challenge. Since the server owns the same symmetric key k, it can successfully decrypt t_1 to obtain the randomness r_C chosen from the client. Next the server sends back the responses $(r_C + 1)$ along with a new challenge r_S. After receiving t_2, the client decrypts it to obtain r'_S and r''_C. If the equation

$$r''_C \overset{?}{=} r_C + 1$$

holds, the server is authenticated. Then the client sends back $t_3 = E_k(r'_S + 1)$ as the response to the server. Similarly, the server can test

$$r''_S \overset{?}{=} r_S + 1$$

to authenticate the client. This protocol is a *mutual authentication* protocol, i.e., the two parties in communications will authenticate each other. If one of the parties does not know k, then it cannot make a correct response. At the end of the authentication phase, the client and the server both obtain the same session key k.

3 Insecure Examples

Here we introduce two insecure examples for the better understanding of the aforementioned attacks. Consider first the authentication protocol shown in Figure 3. Let $x \in \{0,1\}^\ell$ be the master secret of the server, and $H : \{0,1\}^* \to \{0,1\}^\ell$ be a secure hash function.

To see why Example 1 is insecure, first observe that the password and the identity can be cancelled out in the term R. After finishing the registration, a client is able to generate a valid card data to impersonate another user. Given $card$ and another identity ID', the client ID performs as follows.

1. Retrieve R from $card$.
2. Recover x by computing $R \oplus H(ID) \oplus H(PW)$.
3. Choose a random password PW' for ID'.
4. Compute $R' = H(ID') \oplus x \oplus H(PW')$.

It is easy to verify that R' is a valid card data for ID'. It is also easy to fix the weakness by XOR ID with x before hashing. The improved version is shown in Figure 4.

Though the protocol Example 2 fix the previous weakness, it still suffers from the *replay attacks*. If an adversary intercepts the package (B, ID) transmitted in the authentication phase, it can use the package to pass the authentication. The reason is that the package sent in the authentication phase is always the same. To fix the problem, we can add a *time stamp* T in the massage.

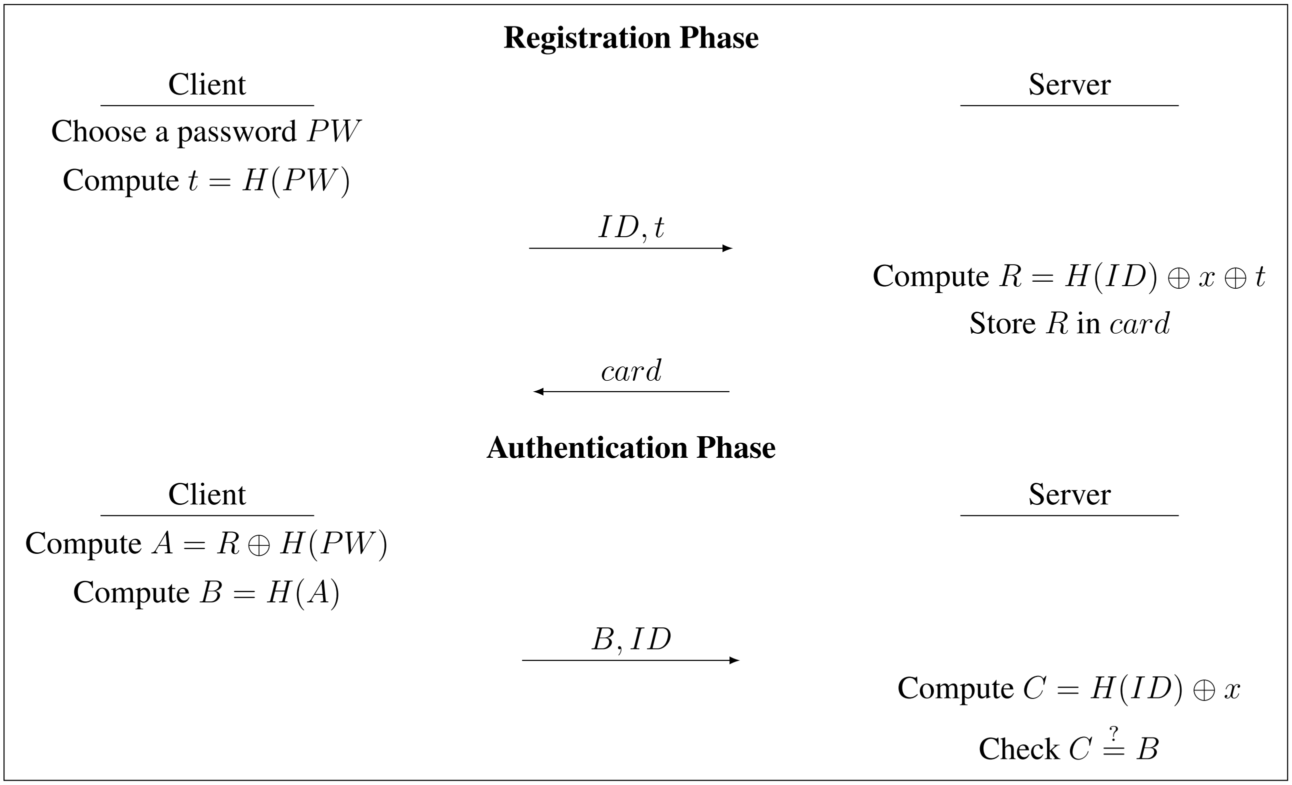


Figure 3: Insecure Example 1

An improved version is shown in Figure 5.

In the protocol Example 3, by check whether the time stamp T is fresh, i.e., check whether the period between T and the current time stamp is lower than a pre-determined, acceptable value. However, the improved protocol is still insecure against *offline dictionary attacks*. The protocol Example 3 is a two-factor authentication protocol, i.e. the password and the smart card. A secure two-factor authentication protocol requires that the two security factors should be independent to each other. That is,

the protocol should remain secure even if an adversary obtain one of the two factors.

In the authentication phase, the adversary is able to eavesdrop the communication and obtain (B, ID, T). We can further assume that the adversary has already got the smart card, and thus obtain the knowledge of R. Observe that

$$B = H(R \oplus H(PW), T). \tag{1}$$

The equation (1) contains only one unknown variable, i.e. PW, and hence the protocol suffers from the offline dictionary attacks.

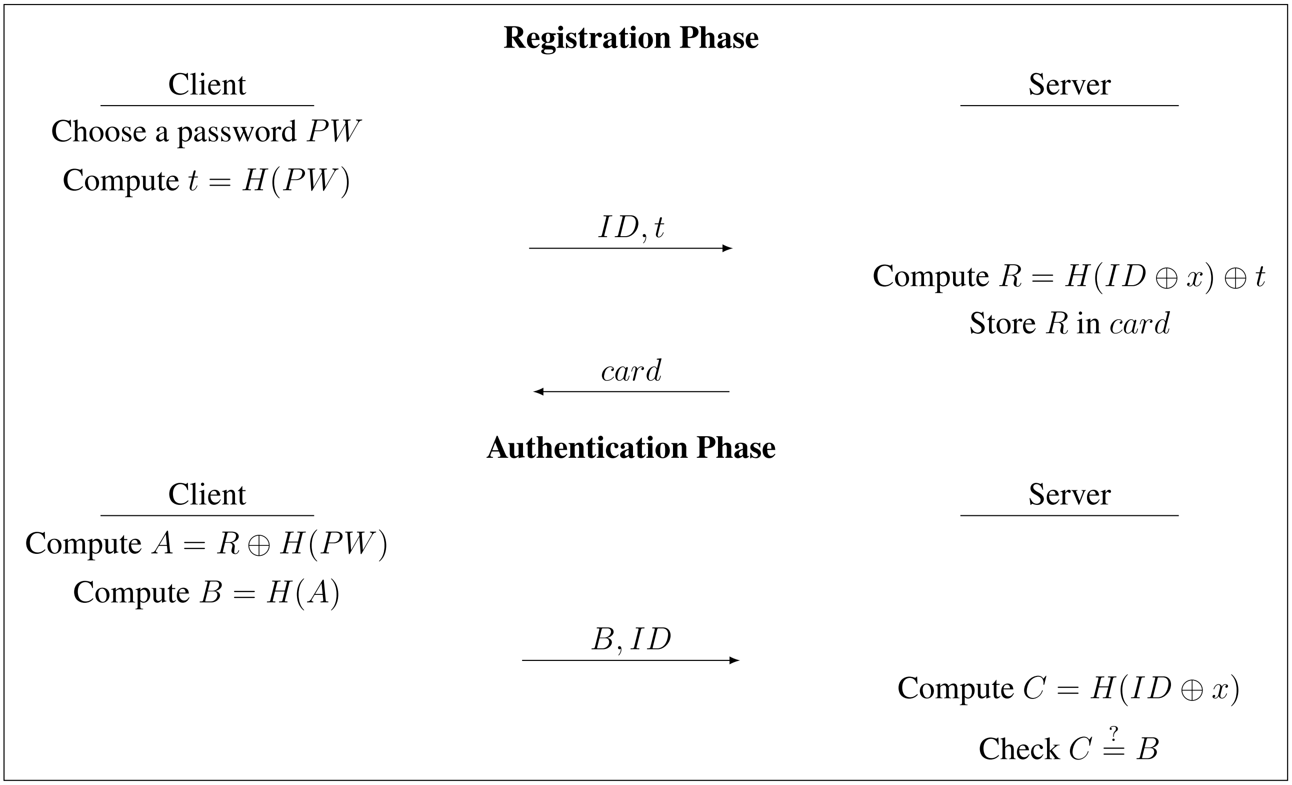


Figure 4: Insecure Example 2

References

[1] L. Lamport. Password authentication with insecure communication. *Commun. ACM*, 24(11):770772, Nov. 1981.

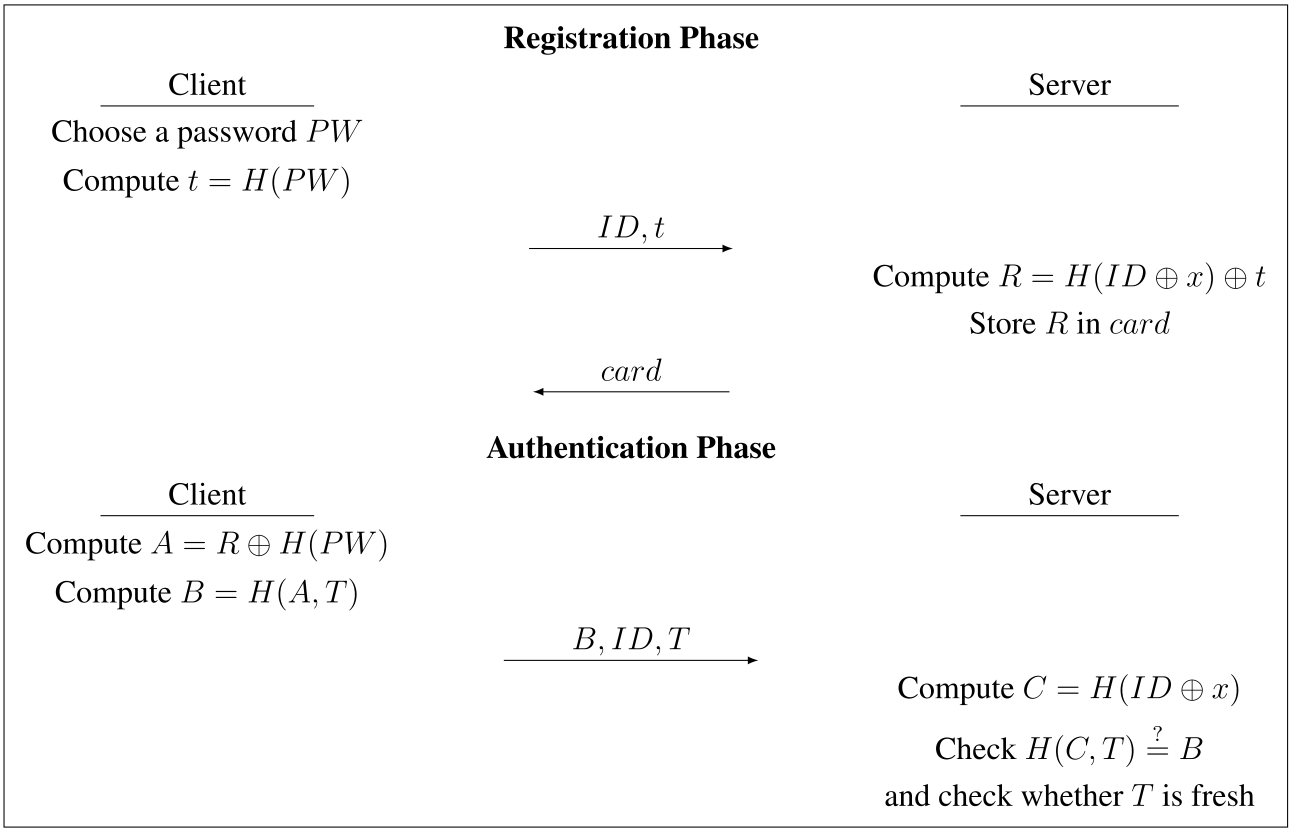


Figure 5: Insecure Example 3