A Model for Security and Privacy in e-Health

Peishun Wang[1] Jeffrey Soar[2]

[1]IBM, Sydney, Australia
patrwan@au1.ibm.com
[2]School of Information Systems, Faculty of Business and Law
University of Southern Queensland, Australia
jeffrey.soar@usq.edu.au

Abstract. Service-Oriented Architecture enables service combination and makes it possible to develop new services in a cost-effective and time-efficient way. As a service combiner, All-in-One e-Health Service, which is described in this paper, empowers e-health organizations and consumers through embracing a holistic paradigm in which consumers, intermediaries, and e-health service providers establish on-demand interactions, to obtain services. In this paper, we investigate the security and privacy issues in the All-in-One e-Health Service model, which include authentication, authorization, identity management, data confidentiality and integrity, and privacy, propose four authentication protocols, and give an analysis comparison on our proposed protocols with two protocols available in the market.

Keywords: e-Health, security, privacy, aggregated service.

1. Introduction

With the wide-spread acceptance of concepts like Service-Oriented Architecture (SOA) and Web services in areas such as e-business, e-health and e-government, the boundaries between outsider (the Internet) and insider (internal systems and applications) and between known applications or - in most cases - network domains, which were more or less possible to distinguish in the past decades, are vanishing, and service combination becomes a cost-effective and time-efficient way to develop new applications and services [4]. It is a trend for a business entity to combine the various services offered by various service providers in different domains along with some of its own services and resell the aggregated service to consumers.

As a service combiner, All-in-One e-Health Service (AHS) empowers e-health organizations and consumers through embracing a holistic paradigm in which consumers and e-health service providers establish on-demand interactions, in real-time if required, to realize useful experiences and to obtain aggregated e-health services. Health service providers include government health departments, general practitioners, private specialists, hospitals, pharmacies, diagnostic service providers, clinics, health insurance companies, and so on. The benefit of AHS originates from the added value generated by the possible interactions and by the large scale rather than by the capabilities of its individual e-health service provider separately. Clearly, this paradigm creates tremendous opportunities in e-health. However, the distributed nature of the system of AHS raises serious challenges in the domains for security and privacy management. Even though the security and privacy assurance of component services are given, new challenges still arise. For example, questions such as how data security can be guaranteed when data elements are dispatched to multiple e-health service providers and how consumer's privacy can be protected when personal identifiable information needs to be shared by multiple service providers, need to be addressed.

Specifically, external services distributed all over the world are exposed to the Internet and therefore to attacks (not to speak about internal attacks against internal services), the complexity of software poses challenges for security management. The flexibility and extensibility of applications make it very hard to know the security impact of new functionality and its potential side-effects on the existing functionality. Therefore, the framework of AHS requires solutions in infrastructure level, application level and business level, which can efficiently facilitate the following for participants and data: preserve privacy, ensure authenticity, provide robust authorization, securely route end-to-end messages, and minimize loss to the AHS system due to attacks or other hostile events.

In addition, AHS would enter into business contracts with the various component e-health service providers, and each service will have its own terms and conditions, quality of service guarantees, and legal constraints and requirements, which could either be predefined or could be negotiated on-demand. For example, service A may have a scope of rights, which permits its usage in AHS, while another service B might not allow usage of the service in AHS. From the perspective of AHS, it is critical to have such knowledge when combining services. Similarly, a service may have an associated usage policy which restricts its participation in AHS for some purposes. When combining services, AHS needs to be aware of these policies and needs to be able to understand and interpret the varying and potentially conflicting terms and conditions.

The remainder of the paper is organized as follows. We briefly describe the system of AHS in Section 2, analyse the security issues existing in AHS in Section 3, identify the privacy issues in AHS in Section 4, and propose four authentication protocols and analyze their efficiency in Section 5. Finally we conclude with suggestions for possible future research directions in Section 6.

2. The System of AHS

There are three types of parties in the system: consumers, AHS and e-health service providers. The e-health service providers provide various public and professional services ranging from simple information retrieval services to more complex transaction oriented services, and AHS works as an agency for e-health service providers to aggregate various services offered by e-health service providers along with some of its own services to serve consumers.

When a consumer would like to access an e-health service, she can send a service request to AHS. AHS will search for all services associated with the request from some e-health service providers, and display public services that are free to consumers and protected services that need the consumer to sign in. To obtain the protected services, the consumer has to authenticate herself to AHS or some e-health service providers. After obtaining the authorization, the consumer can access any services that she wants.

3. Security Issues

Because each component in the system is located in an independent security domain and enforces an aspect of security, there can be multiple identity management, authentication, and authorization mechanisms. Integrating the security mechanisms will be a challenge. The security issues we need to address include authentication, authorization, identity propagation, and data confidentiality and integrity.

3.1 Authentication

Formally, authentication is the process of verifying whether someone or something is, in fact, who or what it is declared to be. According to the types of identity proof, the authentication can be classified into two levels, password based authentication (a name and a password) and credential based authentication (a certificate, a smart card). Without authentication mechanism, any people can access the resources. In some extreme situations, an individual could pose as a willing consumer and accept the services, but repudiate the transaction. With weak authentication, e.g., based on membership numbers, the system is not secure, since they can be easily guessed [2].

The authentication requirement defined for one e-health service provider cannot be applied to another one. It is obvious that for a password based authentication system, a consumer password cannot be applied to a different organisation. Similarly, this issue also impacts on the credential based authentication. A credential for a consumer might not be applicable to multiple e-health service providers.

Each request received by AHS needs to be authenticated. Consumers are able to send requests to AHS for some services by their mobile devices and AHS web. Since AHS only provides the protected services to its members, the information sent in the request must include identity proofing, and AHS needs to authenticate the request -- verifying whether the consumer who is trying to use its services is a legitimate member.

Each request received by an e-health service provider needs to be authenticated. After the AHS received a request, it adds some information or hides some information about the consumer and request to generate a new request, and sends the new request to an e-health service provider. Can the e-health service provider trust the authentication done by one service of AHS and reuse it? Because the service of the e-health service provider is invoked by the service of AHS, not by the consumer directly, how does AHS provide the consumer's identity proofing to the e-health service provider? How do both parties make sure they communicate the results of authentication? If a service of an e-health service provider requested by a consumer needs to invoke a service of another e-health service provider, the first e-health service provider has to check if the service of the second e-health service provider is authenticated one. How does an e-health service provider authenticate a consumer to access a service invoked by a service of another e-health service provider? All those questions are complex, and have no answers that work well.

Each response to a request received by AHS or a consumer needs to be authenticated. Firstly AHS received a response to its request from the e-health service provider, and then it generates a new response with its received response that the e-health service provider sent and replies the consumer. However, AHS has to make sure that the response received are from the e-health service provider that it sent the request to, and the consumer must confirm that the response she received are genuine ones. Without such authentications, any individual could pose as an e-health service provider, and besmirch a provider's good name by failing to deliver goods and billing up credit card bills [1]. So, how to protect against the possibility of someone else capturing the request and replying is a big concern.

3.2 Authorization

To adapt to new requirements and regulations, every information system needs a flexible, customizable infrastructure. It is impossible for an organization to alter an access control policy for every service that is added and deleted dynamically [3]. To process a consumer's request that invokes multiple services across multiple secure domains efficiently, it is necessary to decouple consumer's identity from the services. So, each organization must create an authorization mechanism to suit such a flexible infrastructure of services in multiple domains.

AHS and e-health service providers authorize every consumer to access the functionalities requested. When AHS or an e-health service provider receives a request for some functionality, it will verify whether the authenticated consumer has the authority to access the functionalities she is requesting. Since different consumers, such as common members and golden members, have different access rights to different resources, and a request may invoke multiple actions in constituent services from different domains. They should ideally check the access control rules of all constituent services before initiating an action. If the request invokes a service, which needs to invoke another service that might be in another domain, and the consumer has only authorization to access the first service and not the second one, but the first service has an authorization to access the second one, what is the final result? Therefore, how to resolve such a conflict is a major issue in authorization strategy.

In addition, when a consumer needs to gain access to services in different security domains, multi-level authorization which prevents unauthorized consumers from accessing information at a higher classification than their authorization, is always a problem.

Data sharing requires proper authorization. A consumer can organize other consumers into groups and create rules/permissions to share her own data with other consumers. The permissions have:

- a data object (location, health insurer, medication, immunization, etc) to which they restrict access
- a principal object (single consumer, group, collection thereof) which they grant or deny access
- conditions that must be satisfied
  - time-based: all time, never, time period, before, since
  - location-based: everywhere, nowhere, street, suburb, ZIP, city, state, country, continent

3.3 Identity Propagation

Identity management is related to how consumers are identified and authorized when they access systems. There are a certain number of issues to be addressed in AHS. When consumers interact with the portal of AHS to access multiple related but independent e-health services, the consumer identities need to be propagated throughout the transaction to the back-end systems without the need for multiple sign-on. Therefore, single sign-on and single sign-off are much needed. With single sign-on, a consumer logs in once and gains access to all services without being prompted to log in again at each of them [5]. As single sign-on implies the sharing of consumer credentials across security domains, increased consumer credential protection is needed and strong authentication methods can be used. Similarly, single sign-off allows a single action of signing out terminates access to multiple services. For example, the consumer's service request is past from one e-health service provider to another one. This service request traverses security domains and be able to flow identity context as part of an end-to-end transactional flow. AHS needs to propagate and transform identities across domains in a seamless but secure fashion. The major issue is how to propagate identities across a range of entities to make consumers' identities available to back-end services for authentication and authorization. So, there is a need to propagate the identity throughout the transaction, regardless of the number of domains in the architecture.

3.4 Data Confidentiality and Integrity

It is common that one service provided by an e-health service provider is based on the output of another service provided by another e-health service provider in AHS. The data will flow among e-health service providers. It is hard for the traditional secure transmission to safeguard the data confidentiality and integrity.

Confidentiality. Data exchanging between a consumer and an e-health service provider crosses enterprise boundaries between AHS and the e-health service provider. Messages may also be kept in repositories of AHS or the e-health service provider, such as message queues or databases. Some of the data within the messages are considered to be sensitive in nature, such as identity proofing, medical information. There is a risk that an attacker can gain access to sensitive data, either by eavesdropping on the network or accessing a repository.

If the sensitive data of the consumer was stolen in transit or locations where it is persisted, the disclosure would result in harm. It may cause social embarrassment or prejudice, or affect consumer's insurability, or limit her ability to get and hold a job. In order to avoid such damages, any data that contains sensitive information must be protected from unauthorized people. In particular, the request of a consumer is transferred through AHS to an e-health service provider. Even though every channel used in the system, including the channels between AHS and its consumers and between AHS and the e-health service provider, is secured with HTTPS, it can only secure point-to-point exchange and not end-to-end exchange. Observe that SSL/TLS is not enough to address the data confidentiality concerns here. SSL/TLS can protect the confidentiality of consumer's message when it is passing from the consumer to AHS, but once the message reaches AHS, SSL/TLS's responsibility ends and AHS is free to read and use all the data in the request. That means everybody in AHS can obtain the sensitive data that the consumer would like to send to the e-health service provider.

When a consumer’s request needs to synthesise different sensitive data from different e-health service providers, it is critical not to disclose the sensitive data of one service provider to other e-health service providers.

Integrity. When AHS receives a request, it needs to make sure that the request received is exactly what the consumer sent and not something that is fabricated or tampered with by a man in the middle. Likewise, for a received response, AHS should check if the response is the original one that the e-health service provider sent. In other words, AHS is responsible for verifying integrity of data received over a network. Furthermore, the consumer/e-health service provider should not be able to repudiate or deny having sent the request/response.

As with data confidentiality, the traditional strategy of using SSL/TLS to protect data integrity and non-repudiation is not enough when higher-level services bring together lower-level services from different service providers, because a secure channel provided by SSL/TLS cannot prevent attackers in AHS from alternating the data.

4. Privacy Issues

Combining services implies the sharing of information between services. If the information is personal health data, privacy protection must be considered. E-Health service providers are prohibited by law or by contract from disclosing the personal/private information of consumers/partners to third parties. This prohibition applies as much to applications as it does to humans [8]. In other words, the applications of any e-health service providers must be carefully designed and managed to avoid leakage of consumers’ health information. To protect consumers’ privacy, each e-health service provider employs its privacy policy. When a consumer accesses proprietary services provided by a single e-health service provider, consumer’s privacy is protected by the privacy policy of the provider regarding information collection and disclosure. However, when a consumer’s health information needs to be shared by multiple e-health service providers, privacy protection obviously becomes a more challenging problem.

In order to request protected services, every consumer has to register in AHS to be a legal member. The information for registration includes personal information and credentials, which are stored in AHS. The registration is done with one time transmission of membership proof in plaintext over the encrypted channel. There are some new highly sensitive data that will be collected online, such as current medical condition, allergy history of consumers. There are other kinds of information, such as, non-public personal health data retrieved from the internal system of an e-health service provider. Privacy considerations make gathering all potentially needed credentials from consumers difficult. Furthermore, this may simply be impossible. If different e-health service providers require different credentials to a consumer, how to dissolve such a conflict is an issue in AHS.

Multiple e-health service providers involved might have some inconsistent or even conflicting privacy policies. Therefore, advanced privacy right management should be investigated. This includes the autonomous comparison of privacy policies, detection of inconsistency/conflicts, analysis of risks associated with the detected inconsistency/conflicts, and resolution of inconsistency/conflicts. Consumers need to keep informed about any potential privacy breach, and consumers should have control over what information can be disclosed, and to whom.

The privacy policies of different e-health service providers may be represented with different privacy language since they may be used with different systems. Suppose one e-health service provider's privacy policy is represented with P3P while AHS is represented with XACML, how can AHS consolidate these two privacy policies?

If a consumer does not want the AHS to know some of her personal information past to e-health service providers, as the discussion on data confidentiality in the Section 3.4, there is no existing way to protect the consumer's privacy in such a situation. Additionally, a consumer does not want to disclose her identity to an e-health service provider, but her request needs a service of AHS, which needs to invoke a service of the e-health service provider. Even though AHS holds back real identities (using pseudonyms), it is possible to guess identity based on patterns of usage. Protecting against such leaks is difficult and is not often attempted.

5. Authentication Solutions

Authentication process requires an initial validation of the identity -- identity proofing. There exist various methods of identity proofing, which range from person validation with authority issued identification to anonymous verification remaining claimant anonymous but known to the system if they return. In order to provide an assurance level commensurate with the intended use of the identity within the system, the method used for identity proofing usually asserts an identity together with a factor doing validation [9]. Surely the identity must be unique within its security domain.

There are a lot of mechanisms over which authentication information can be carried, such as HTTP Digest and consumer side certificates. User names and passwords are the most common form of authentication in practice. No matter what evidence consumers present to a system, the system needs to possess a securely stored master copy of the evidence. This means that, the copy should be stored in such a way that the system can use it to do validation, but it should not be able to be used by any adversary who manages to compromise the system. To guard the security of consumers' evidences, there are two methods to take into account. One is to protect evidences from line eavesdropping and altering within the system, and the other is to protect evidences from disclosure outside of the system. The common way in use is the cryptographic scheme. Based on the cryptographic approach adopted, authentication protocols can be categorized by symmetric key protocols and asymmetric (public and private) key protocols. There is another way to categorize authentication protocols. That is, whether one party wishes to convince the second of some matter (unilateral authentication), and whether both parties wish to convince each other of something (mutual authentication). Based on the analysis at Section 3.1, the authentication in AHS is a mutual protocol. Now, we describe the procedure.

5.1 Authentication Procedure in AHS

The authentication procedure in AHS consists of four messages exchanged among consumers, AHS and e-health service providers. The messages flowing procedure in the authentication is indicated in the Fig. 1.

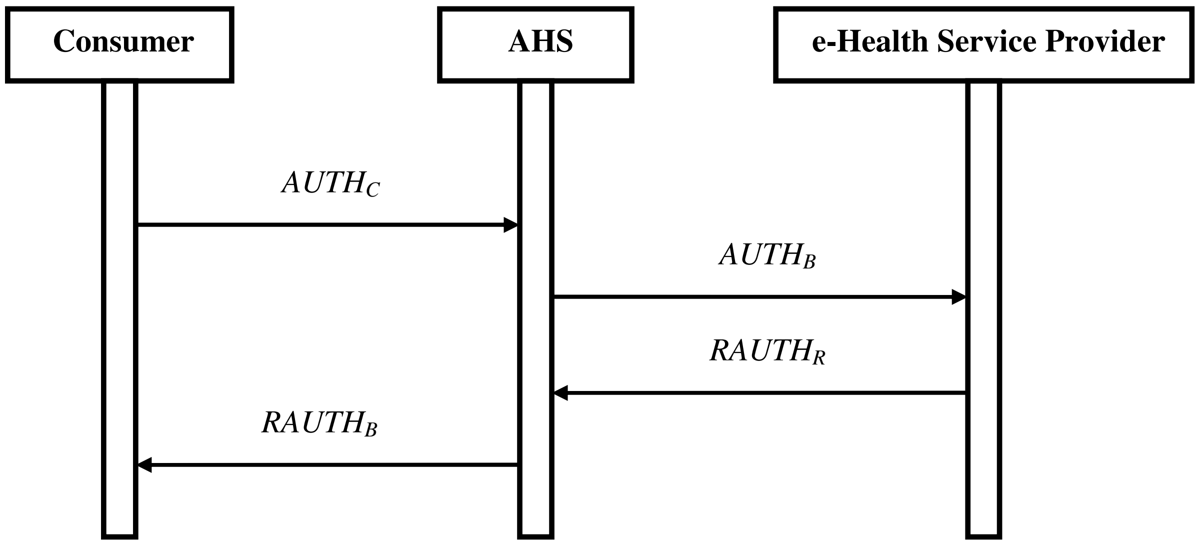


Fig. 1. The authentication procedure

A consumer first sends an authentication request $AUTH_C$ to AHS. Based on the consumer's message, AHS creates a new authentication request $AUTH_B$ and sends it to an e-health service provider. After verifying the identity of the consumer, the e-health service provider returns an authentication response $RAUTH_R$ to AHS. According to the response from the e-health service provider, AHS generates an authentication response $RAUTH_B$ and replies the consumer.

5.2 Proposed Authentication Protocols

Based on the cases in real world, we deal with two different system environments: one is that consumers register in AHS only, and the other is that consumers register in AHS and all e-Health Service providers.

In the first environment, we consider three different security requirements.

Security Requirement 1: The communication channel between AHS and each e-Health Service provider is secure with their secret session key. After the authentication, the communication channel between a consumer and AHS is secure with their secret session key, which is built in the procedure of authentication. The session key between a consumer and AHS is refreshed after a service access is finished.

Security Requirement 2: The communication channel between AHS and each e-Health Service provider is secure with their secret session key, and the communication channel between a consumer and AHS is secure with their secret session key. The session key between AHS and an e-Health Service provider is refreshed after a service access is finished.

Security Requirement 3: The communication channel between AHS and each e-Health Service provider is secure with their asymmetric cryptosystem, and the communication channel between a consumer and AHS is secure with their asymmetric cryptosystem. After the authentication procedure, the communication channels between a consumer and AHS and between AHS and an e-Health Service provider are secured with their secret session keys, which are refreshed after a service access is finished.

In the second environment, the security requirement is as follows. The communication channel between AHS and each e-Health Service provider is secure with their secret session key. After the authentication, the channel between a consumer and AHS is secure with their secret session key, which is built in the procedure of authentication. The session key between a consumer and AHS is refreshed after a service access is finished

Based on these four different system requirements, we proposed four authentication protocols. Due to the limited space, we present their detailed constructions in the Appendix.

In addition, we analyze two authentication protocols available in the market – Oauth [6] and OpenID [7], and find that OpenID cannot be applied in AHS and OAuth can be used in the first system environment only. Therefore, we compare OAuth with the proposed three authentication protocols in a range of functions including mutual authentication, freshness of the agreed session key, confidentiality, integrity, symmetric encryption, asymmetric encryption, digital signature, needing TLS or SSL, risks of redirection (e.g. Phishing attacks), number of secret keys of a consumer, number of rounds, and computation cost. The result of comparison indicates that our proposed protocols are more efficient than OAuth. We omit the comparison report here, which is given in the Appendix.

6. Conclusion

In this paper we analyzed and identified the security and privacy issues existing in AHS. It is clear that finding solutions to these concerns becomes necessary. We proposed our new authentication protocol, and compared with two protocols (OAuth and OpenID) available in the market.

Except for some solutions to authentication issues in AHS, there are a lot of challenging open problems on security and privacy in AHS. For the future research, we will build a logical model for AHS in cloud computing and design schemes to solve the security and privacy issues in the model.

References

1. Blaze, M., Feigenbaum, J., Ioannidis, J., Angelos, D., Keromytis. The Role of Trust Management in Distributed Systems Security. Secure Internet Programming: Issues for mobile and distributed objects, Springer-Verlag, 1999. pp.185-210.
2. Forouzan, B.A., 2008. Introduction to Cryptography and Network Security, McGraw Hill.
3. Jajodia, S., Samarati, P., Sapino, M.L., and Subrahmanian, V.S., Flexible support for multiple access control policies, TODS 26 (2) (2001) 214–260.
4. Kanneganti R., Chodavarapu P. SOA Security. Manning Publications Co. Greenwich, CT, 2008.
5. Kormann DP, Rubin AD. Risks of the passport single signon protocol. Computer Networks: The International Journal of Computer and Telecommunications Networking, v.33 n.1-6, pp.51-58, June 2000.
6. OAuth Core 1.0. http://oauth.net/core/1.0
7. OpenID. http://openid.net/. 2009

8. Rindfleisch, T., C., Privacy, Information Technology, and Health Care, Communications of the ACM, 1997, pp. 93-100.
9. Stallings, W., 2006. Cryptography and Network Security: Principles and Practice, Prentice-Hall.

Appendix

A. Notations

The notations we use in this paper are defined as follows:

Table 1: Nomenclature

Notation	Explanation
C	A consumer
R	A e-Health Service Provider
$Identity_C$	C's identity registered in R
$Passwd_C$	C's password registered in R
$Identity_B$	Identity of AHS
$Identity_R$	Identity of R
puk_C, prk_C	Public/private key pair of C
puk_B, prk_B	Public/private key pair of AHS
puk_R, prk_R	Public/private key pair of R
N_C,	A nonce number generated by C
N_B	A nonce number generated by AHS
$AUTH_C$	Authentication Request Message sent by C to AHS
$AUTH_B$	Authentication Request Message sent by AHS to R
$RAUTH_R$	Authentication Data Response Message sent by R to AHS
$RAUTH_B$	Authentication Data Response Message sent by AHS to C
SK_{CB}	Session key shared by the C and AHS
SK_{BR}	Session key shared by the AHS and R
$\mathcal{E}_K(x)$	The encryption function with the key K over the fields x.
$TStamp_C$	A timestamp generated by C
$\|\|$	The operation of concatenation

B. Proposed password based authentication protocols

B.1 Protocol 1

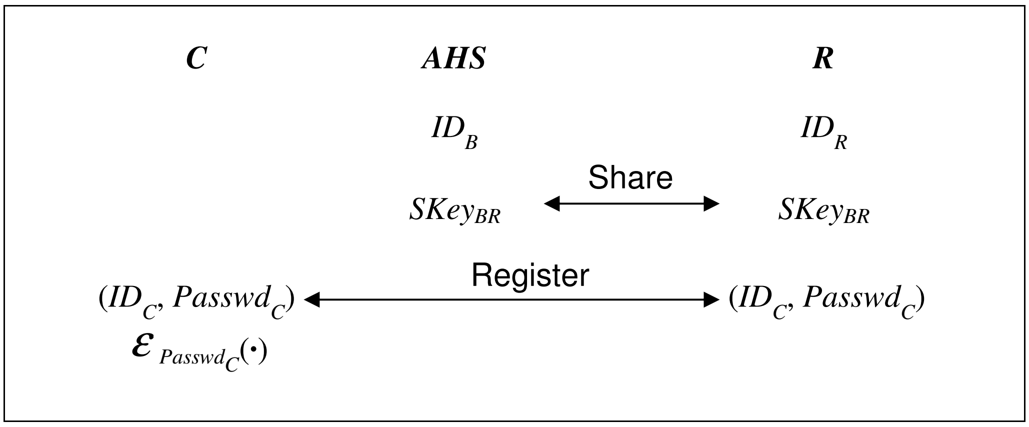
Security requirement:

1. The communication channel between AHS and R is secure with their secret session key $SKey_{BR}$.

2. After the authentication, the channel between C and AHS is secure with their secret session key $SKey_{CB}$ that is built in the procedure of authentication.

3. The session key $SKey_{CB}$ between C and AHS is refreshed after a service access is finished.

B.1.1 Setup



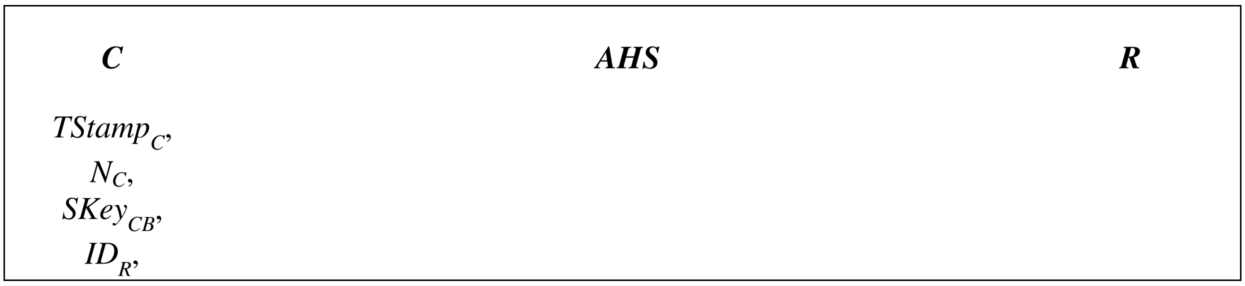
The details of procedure are as follows.

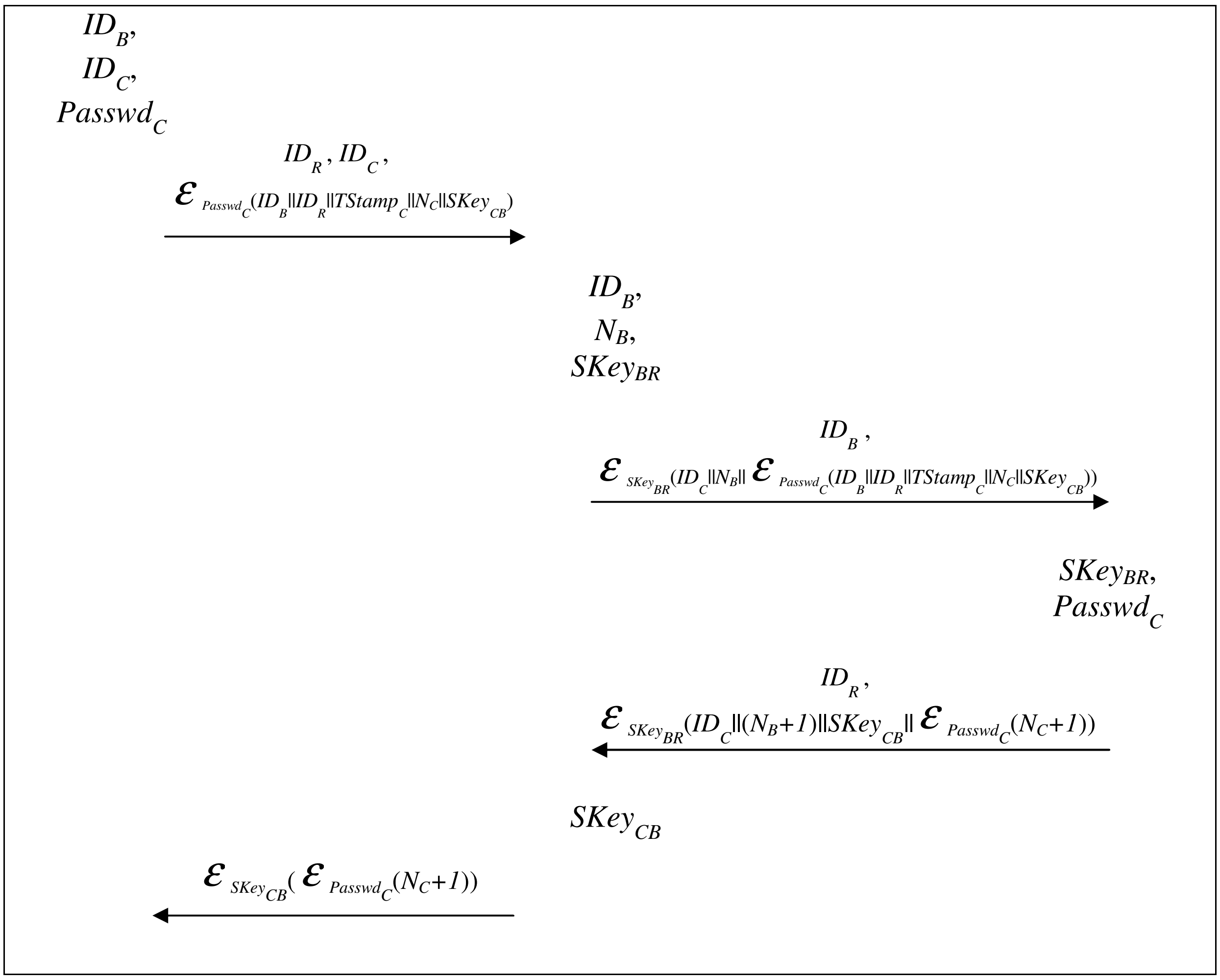
1. AHS has its identity ID_B known to all consumers and R, and shares a secret session key $SKey_{BR}$ with R.

2. R has its identity ID_R, which is known to all consumers and AHS, and shares a secret session key $SKey_{BR}$ with AHS.

3. Every client C registers at R, and obtains the identity and corresponding password (ID_C, $Passwd_C$), where the password can be used as a secret encryption key in a symmetric cryptosystem $\mathcal{E}(\cdot)$ that both AHS and R know. If the password stored at R is a hash code, then the password used for encryption must be a hashed one.

B.1.2 The protocol




ID_B, ID_C, $Passwd_C$
ID_R, ID_C, $\mathcal{E}_{Passwd_C}(ID_B \| ID_R \| TStamp_C \| N_C \| SKey_{CB})$
ID_B, N_B, $SKey_{BR}$
ID_B, $\mathcal{E}_{SKey_{BR}}(ID_C \| N_B \| \mathcal{E}_{Passwd_C}(ID_B \| ID_R \| TStamp_C \| N_C \| SKey_{CB}))$
$SKey_{BR}$, $Passwd_C$
ID_R, $\mathcal{E}_{SKey_{BR}}(ID_C \| (N_B+1) \| SKey_{CB} \| \mathcal{E}_{Passwd_C}(N_C+1))$
$SKey_{CB}$
$\mathcal{E}_{SKey_{CB}}(\mathcal{E}_{Passwd_C}(N_C+1))$

The details of authentication procedure are as follows.

Step 1: Authentication request message from the client C

When C needs to authenticate itself to access services of R, it invokes the distribution of authentication procedure by sending the authentication request message $AUTH_C$ to AHS. C does the following:

1. Generates the following:
    - A timestamp $TStamp_C$ and a nonce number N_C.
    - A session key $SKey_{CB}$ for the following communication
    - $AUTH_C = (ID_R, ID_C, \mathcal{E}_{Passwd_C}(ID_B \| ID_R \| TStamp_C \| N_C \| SKey_{CB}))$.
2. Sends $AUTH_C$ to AHS as the authentication request message.

Step 2: Authentication request message from AHS

When AHS receives the request message $AUTH_C$, it finds the identity of the service provider R, and builds its authentication request message, which will be sent to R. It does the following:

1. Finds the identity of the service provider R, ID_R, and saves the client's identity ID_C for the future use.
2. Generates a nonce number N_B.
3. Generates $AUTH_B = (ID_B, \mathcal{E}_{SKey_{BR}}(ID_C \| N_B \| \mathcal{E}_{Passwd_C}(ID_B \| ID_R \| TStamp_C \| N_C \| SKey_{CB})))$ with the session key $SKey_{BR}$ between AHS and R.
4. Sends $AUTH_B$ to R as its authentication request message.

Step3: Authentication response message from R

Upon receipt of the $AUTH_B$, R decrypts the message with the session key SK_{BR} and the client's password $Passwd_C$, verifies the sender of $AUTH_B$ and the membership of the original requester C, and then builds the authentication response message and sends it back to AHS. R does the following:

1. Finds the session key $SKey_{BR}$ corresponding to the identity ID_B and decrypts the message to find the password $Passwd_C$ of the original requester C identified by ID_C.
2. Decrypts the $\mathcal{E}_{Passwd_C}(ID_B \| ID_R \| TStamp_C \| N_C \| SKey_{CB})$ with the password $Passwd_C$ to get ID_B, ID_R, $TStamp_C$, N_C and $SKey_{CB}$, and verifies the identity ID_B and ID_R to confirm that the message $AUTH_B$ is not modified and the receiver of the message is itself.
3. Checks the validity of membership of C and the correctness of $TStamp_C$.
4. Generates $RAUTH_R = (ID_R, \mathcal{E}_{SKey_{BR}}(ID_C \| (N_B+1) \| SKey_{CB} \| \mathcal{E}_{Passwd_C}(N_C+1)))$.
5. Sends $RAUTH_R$ back to AHS as its authentication response message.

Step 4: Authentication response message from AHS

Upon receipt of the authentication response message $RAUTH_R$ from R, AHS decrypts the message, verifies if the message is received from the intended R, and finds the client C the authentication response message is for. Then, AHS builds the authentication response message and sends it to C. C verifies the response message to confirm that the authentication response message is from the intended senders. This process is achieved as follows:

1. AHS finds the session key $SKey_{BR}$ corresponding to the identity ID_B.

2. AHS decrypts the message to get ID_C, (N_B+1), $SKey_{CB}$ and $\mathcal{E}_{Passwd_C}(N_C+1)$, and checks the correctness of N_B+1 with the aid of N_B that it generated to make sure that the response message is from R and not modified

3. AHS confirms the identity of the client C and saves $SKey_{CB}$ for the following communication between AHS and C.

4. AHS generates $RAUTH_B = (\mathcal{E}_{SKey_{CB}}(\mathcal{E}_{Passwd_C}(N_C+1)))$ and sends it as the authentication response message to C.

5. C decrypts $\mathcal{E}_{SKey_{CB}}(\mathcal{E}_{Passwd_C}(N_C+1))$ with the session key $SKey_{CB}$ that it generated to get $\mathcal{E}_{Passwd_C}(N_C+1)$, and decrypts $\mathcal{E}_{Passwd_C}(N_C+1)$ with its password $Passwd_C$ to get the values of N_C+1, and then verifies that the value of N_C+1 is correct by comparing it with N_C generated by itself.

6. If the result of verification is correct, then the authentication is successful.

B.2 Protocol 2

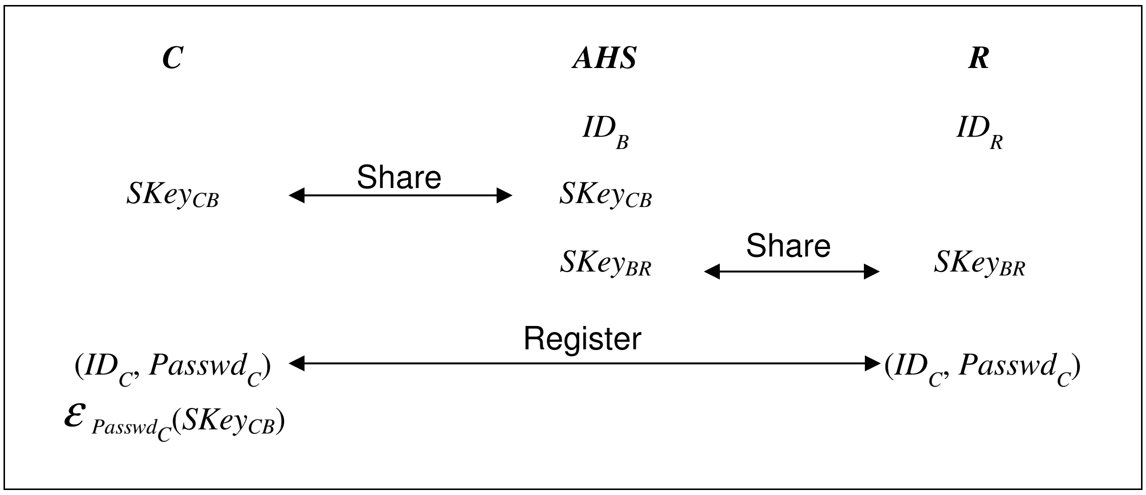
Security requirement:

1. The communication channel between C and AHS is secure with their secret session key $SKey_{CB}$.

2. The communication channel between AHS and R is secure with their secret session key $SKey_{BR}$.

3. The session key $SKey_{CB}$ between C and AHS is refreshed after a service access is finished.

B.2.1 Setup



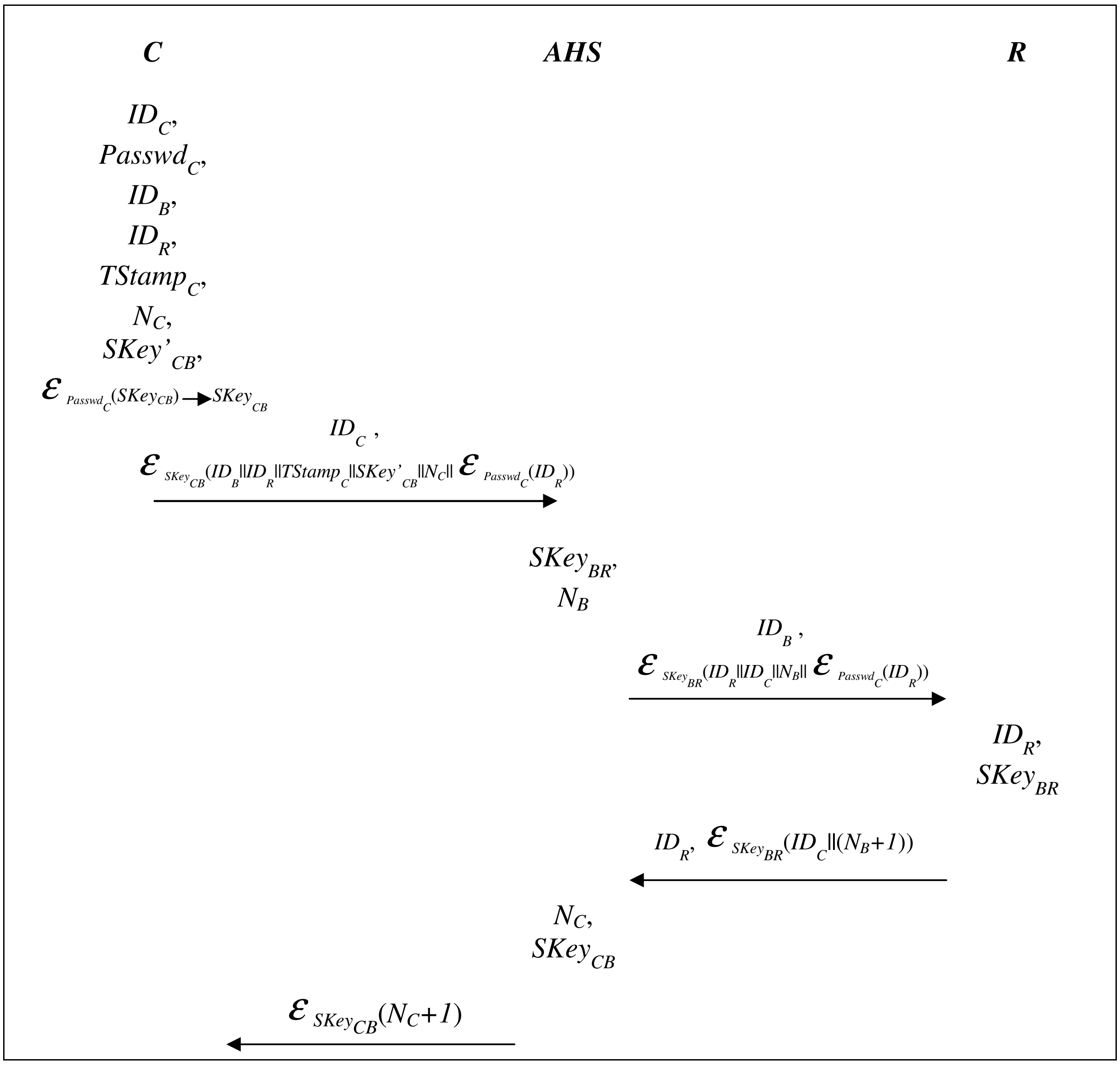
The details of procedure are as follows.

1. AHS has its identity ID_B known to all clients and R, and shares a secret session key $SKey_{CB}$ with each client and a secret session key $SKey_{BR}$ with R.

2. R has its identity ID_R, which is known to all consumers and AHS, and shares a secret session key $SKey_{BR}$ with AHS

3. Every client registers at R, and obtains the identity and corresponding password (IDC, PasswdC). She also shares a secret session key $SKey_{CB}$ with AHS. To protect the session key, she uses her password as the key to encrypt $SKey_{CB}$. If the password stored at R is a hash code, then the password used for encryption must be a hashed one.

B.2.2 The protocol



C
AHS
R
ID_C, $Passwd_C$, ID_B, ID_R, $TStamp_C$, N_C, $SKey'_{CB}$,
$\mathcal{E}_{Passwd_C}(SKey_{CB}) \rightarrow SKey_{CB}$
ID_C, $\mathcal{E}_{SKey_{CB}}(ID_B \| ID_R \| TStamp_C \| SKey'_{CB} \| N_C \| \mathcal{E}_{Passwd_C}(ID_R))$
$SKey_{BR}$, N_B
ID_B, $\mathcal{E}_{SKey_{BR}}(ID_R \| ID_C \| N_B \| \mathcal{E}_{Passwd_C}(ID_R))$
ID_R, $SKey_{BR}$
ID_R, $\mathcal{E}_{SKey_{BR}}(ID_C \| (N_B+1))$
N_C, $SKey_{CB}$
$\mathcal{E}_{SKey_{CB}}(N_C+1)$

The details of authentication procedure are as follows.

Step 1: Authentication request message from C

When C needs to authenticate itself to access services of R, it invokes the distribution of authentication procedure by sending the authentication request messages $AUTH_C$ to AHS. C does the following:

1. C applies its password $Passwd_C$ to decrypt $\mathcal{E}_{Passwd_C}(SKey_{CB})$ to get the session key $SKey_{CB}$.
2. Generates the following:
    - A timestamp $TStamp_C$ and a nonce number N_C.
    - A new session key $SKey'_{CB}$.
    - $AUTH_C = (ID_C, \mathcal{E}_{SKey_{CB}}(ID_B||ID_R||TStamp_C||SKey'_{CB}||N_C||\mathcal{E}_{Passwd_C}(ID_R))$.
3. Sends $AUTH_C$ to AHS as the authentication request message.

Step 2: Authentication request message from AHS

When AHS receives the request message $AUTH_C$, it verifies that the timestamp $TStamp_C$ is acceptable and the request is sent by C to it, and builds its authentication request message, which will be sent to R. It does the following:

1. Decrypts the message with the session key $SKey_{CB}$ corresponding to the identity ID_C and check if the decrypted data ID_B is its identity.

2. Checks the validity of timestamp $TStamp_C$ and saves $SKey'_{CB}$ and N_C for future use.

3. Generates a nonce number N_B.

4. Applies $SKey_{BR}$ corresponding to the identity ID_R to generates its authentication request message $AUTH_B = (ID_B, \mathcal{E}_{SKey_{BR}}(ID_R||ID_C||N_B||\mathcal{E}_{Passwd_C}(ID_R))$.

5. Sends $AUTH_B$ to R that is identified by ID_R.

Step 3: Authentication response message from R

Upon receipt of the $AUTH_B$, R verifies that the timestamp T_C is acceptable and the requests are from AHS and C, and then builds the authentication response message and sends it back to AHS. R does the following:

1. Finds the session key $SKey_{BR}$ corresponding to the identity ID_B and decrypts the message to check if it is the receiver identified ID_R.

2. Verifies the validity of membership of C with (ID_C, $Passwd_C$) by checking if the data decrypted from $\mathcal{E}_{Passwd_C}(ID_R)$ is its identity ID_R.

3. Uses the session key $SKey_{BR}$ to generate $RAUTH_R = (ID_R, \mathcal{E}_{SKey_{BR}}(ID_C \| (N_B+1)))$.

4. Sends $RAUTH_R$ back to AHS as the authentication response message.

Step 4: Authentication response message from AHS

Upon receipt of the authentication response message $RAUTH_R$ from R, AHS verifies if the message is sent by R, and finds the client the authentication response message is for. Then, AHS builds the authentication response message and sends it to C, and C verifies the response message. This process is achieved as follows:

1. AHS finds the session key $SKey_{BR}$ corresponding to the identity ID_R to decrypt the message.

2. Checks the correctness of N_B+1 with the aid of N_B that it generated to make sure that the response message is from R and not modified.

3. AHS finds the session key $SKey_{CB}$ corresponding to the identity ID_C.

4. AHS generates $RAUTH_B = \mathcal{E}_{SKey_{CB}}(N_C+1)$ as the authentication response message with the value N_C that it received and sends the message to C.

5. C decrypts $\mathcal{E}_{SKey_{CB}}(N_C+1)$ with its session key $SKey_{CB}$ to get the value (N_C+1) and verifies if the value is correct by comparing it with N_C generated by itself.

6. If the result of verification is correct, then the authentication is successful.

B.3 Protocol 3

Security requirement:

1. The communication channel between C and AHS is secure with their asymmetric cryptosystems.

2. The communication channel between AHS and R is secure with their asymmetric cryptosystems.

3. After the authentication procedure, the communication channels between C and AHS and between AHS and R are secured with their secret session keys, which are refreshed after a service access is finished.

B.3.1 Setup

$$\begin{tikzcd}
\boldsymbol{C} & \boldsymbol{AHS} & \boldsymbol{R} \\
 & ID_B & ID_R \\
(puk_C, prk_C) & (puk_B, prk_B) & (puk_R, prk_R) \\
(ID_C, Passwd_C) \arrow[rr, leftrightarrow, "\text{Register}"] & & (ID_C, Passwd_C) \\
\mathcal{E}_{Passwd_C}(prk_C) & &
\end{tikzcd}$$

The details of procedure are as follows.

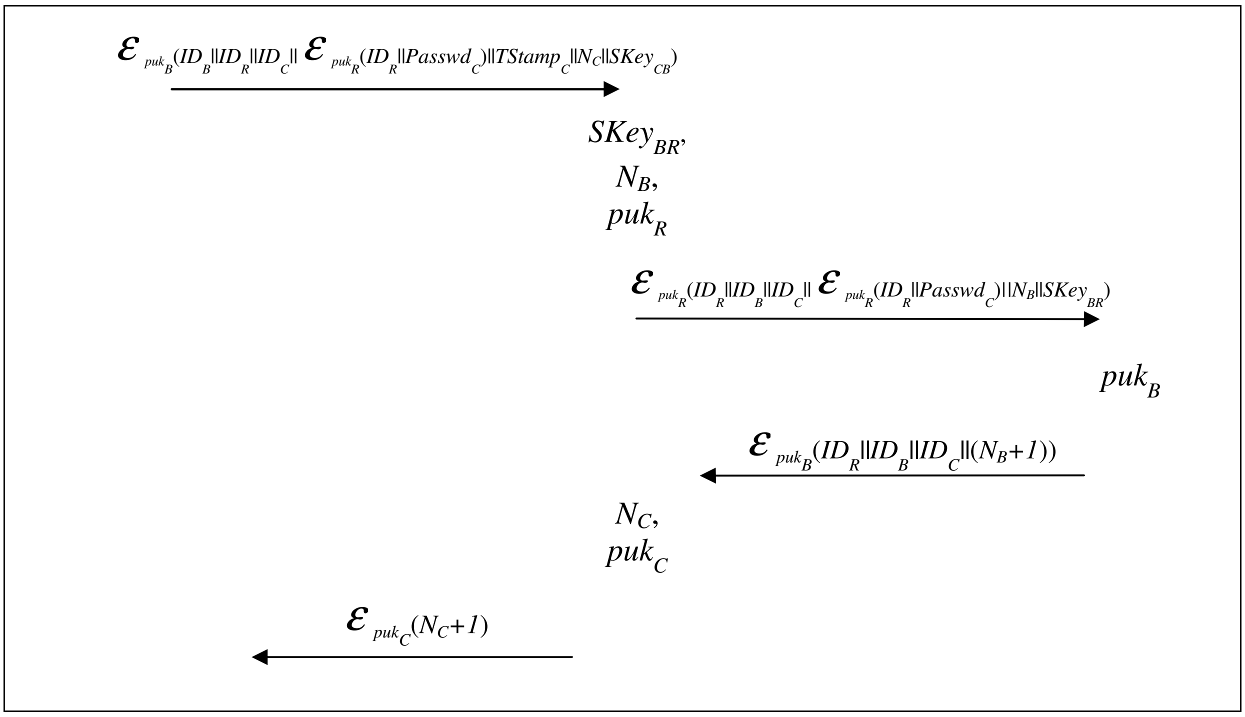
1. AHS has its identity ID_B known to all clients and R, and has an asymmetric cryptosystem with the pair of public key and private key (puk_B, prk_B).

2. R has its identity ID_R, which is known to all consumers and AHS, and has an asymmetric cryptosystem with the pair of public key and private key (puk_R, prk_R).

3. Every client registers in R, and obtains the identity and corresponding password (ID_C, $Passwd_C$). She has an asymmetric cryptosystem with the pair of public key and private key (puk_C, prk_C). To protect the private key, she uses her password as the key to encrypt prk_C.

B.3.2 The protocol

$$\begin{array}{ccc}
\boldsymbol{C} & \boldsymbol{AHS} & \boldsymbol{R} \\
ID_B, & & \\
ID_R, & & \\
ID_C, & & \\
Passwd_C, & & \\
TStamp_C, & & \\
N_C, & & \\
SKey_{CB}, & & \\
puk_B & &
\end{array}$$


$\mathcal{E}_{puk_B}(ID_B||ID_R||ID_C|| \mathcal{E}_{puk_R}(ID_R||Passwd_C)||TStamp_C||N_C||SKey_{CB})$
$SKey_{BR}$,
N_B,
puk_R
$\mathcal{E}_{puk_R}(ID_R||ID_B||ID_C|| \mathcal{E}_{puk_R}(ID_R||Passwd_C)||N_B||SKey_{BR})$
puk_B
$\mathcal{E}_{puk_B}(ID_R||ID_B||ID_C||(N_B+1))$
N_C,
puk_C
$\mathcal{E}_{puk_C}(N_C+1)$

Now we describe the details of authentication procedure.

Step 1: Authentication request message from C

When C needs to authenticate itself for accessing services of R, it invokes the distribution of authentication procedure by sending the authentication request message $AUTH_C$ to AHS. This process is achieved as follows:

1. C generates the following:

   - A timestamp $TStamp_C$, a nonce number N_C and a session key $SKey_{CB}$.
   - $AUTH_C = \mathcal{E}_{puk_B}(ID_B||ID_R||ID_C|| \mathcal{E}_{puk_R}(ID_R||Passwd_C)||TStamp_C||N_C||SKey_{CB})$ with the public key puk_B of AHS.

2. C sends $AUTH_C$ to AHS as its authentication request message.

Step 2: Authentication request message from AHS

When AHS receives the request message $AUTH_C$, it decrypts the message to verify the requester C according to the information that have been received, and builds its authentication request message, which will be sent to R. It does the following:

1. Decrypts the message with its private key prk_B.

2. Checks whether ID_B is its identity to make sure that the message is not modified and for itself.

3. Checks the validity of timestamp $TStamp_C$ and saves $SKey_{CB}$ and N_C for future use.

4. Finds the public key puk_R of the service provider R corresponding to the identity ID_R.

5. Generates a nonce number N_B and a session key $SKey_{BR}$.

6. Generates $AUTH_B = \mathcal{E}_{puk_R}(ID_R \| ID_B \| ID_C \| \mathcal{E}_{puk_R}(ID_R \| Passwd_C) \| N_B \| SKey_{BR})$ with puk_R.

7. Sends $AUTH_B$ to R that is identified by ID_R.

Step3: Authentication response message from R

Upon receipt of the $AUTH_B$, R decrypts that message and verifies the membership of C and if the messages are received from the intended AHS and sent to it, and then builds the authentication response message and sends it back to AHS. R does the following:

1. Decrypts the message with its private key prk_R.

2. Checks if the message is from AHS and for it by comparing ID_B and ID_R with the corresponding data that it has.

3. Confirms that $\mathcal{E}_{puk_R}(ID_R \| Passwd_C)$ is not altered by checking the correctness of the decrypted data ID_R.

4. Verifies the validity of membership of C with $(ID_C, Passwd_C)$.

5. Saves the session key $SKey_{BR}$ for future communication between AHS and R and generates $RAUTH_R = \mathcal{E}_{puk_B}(ID_R \| ID_B \| ID_C \| (N_B+1))$.

6. Sends $RAUTH_R$ back to AHS as the authentication response message.

Step 4: Authentication response message from AHS

Upon receipt of the authentication response message $RAUTH_R$ from R, AHS decrypts that message and verifies if the message is received from the intended R and sent to it, and finds the client whom the authentication response message is for. Then, AHS builds the authentication response message and sends it to C, and C verifies the response message. This process is achieved as follows:

1. AHS decrypts the message with its private key prk_B.

2. AHS checks if the message is from AHS and for itself by comparing ID_B and ID_R with the corresponding data that it has.

3. AHS checks the correctness of N_B+1 with the aid of N_B that it generated to make sure that the response message is not modified.

4. AHS applies the public key puk_C of the client C identified by ID_C and the value N_C that it received to generates $RAUTH_B = \mathcal{E}_{puk_C}(N_C+1)$ and sends the message to C.

5. C decrypts $\mathcal{E}_{puk_C}(N_C+1)$ with its private key prk_C, which is derived by decrypting the encrypted private key $\mathcal{E}_{Passwd_C}(prk_C)$ with its password $Passwd_C$, to get the value (N_C+1) and verifies if the value is correct by comparing it with N_C generated by itself.

6. If the result of verification is correct, then the authentication is successful.

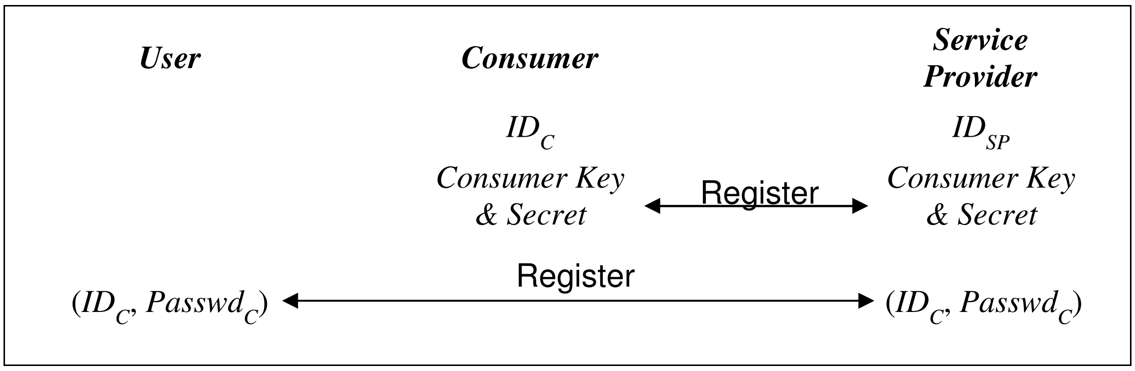
C. Two protocols available in market

C.1 OAuth protocol

The OAuth protocol enables websites or applications (Consumers) to access protected resources from a web service (Service Provider) via an API, without requiring Users to disclose their Service Provider credentials to the Consumers. More generally, OAuth creates a freely-implementable and generic methodology for API authentication.

OAuth does not require a specific user interface or interaction pattern, nor does it specify how Service Providers authenticate Users, making the protocol ideally suited for cases where authentication credentials are unavailable to the Consumer.

C.1.1 Setup

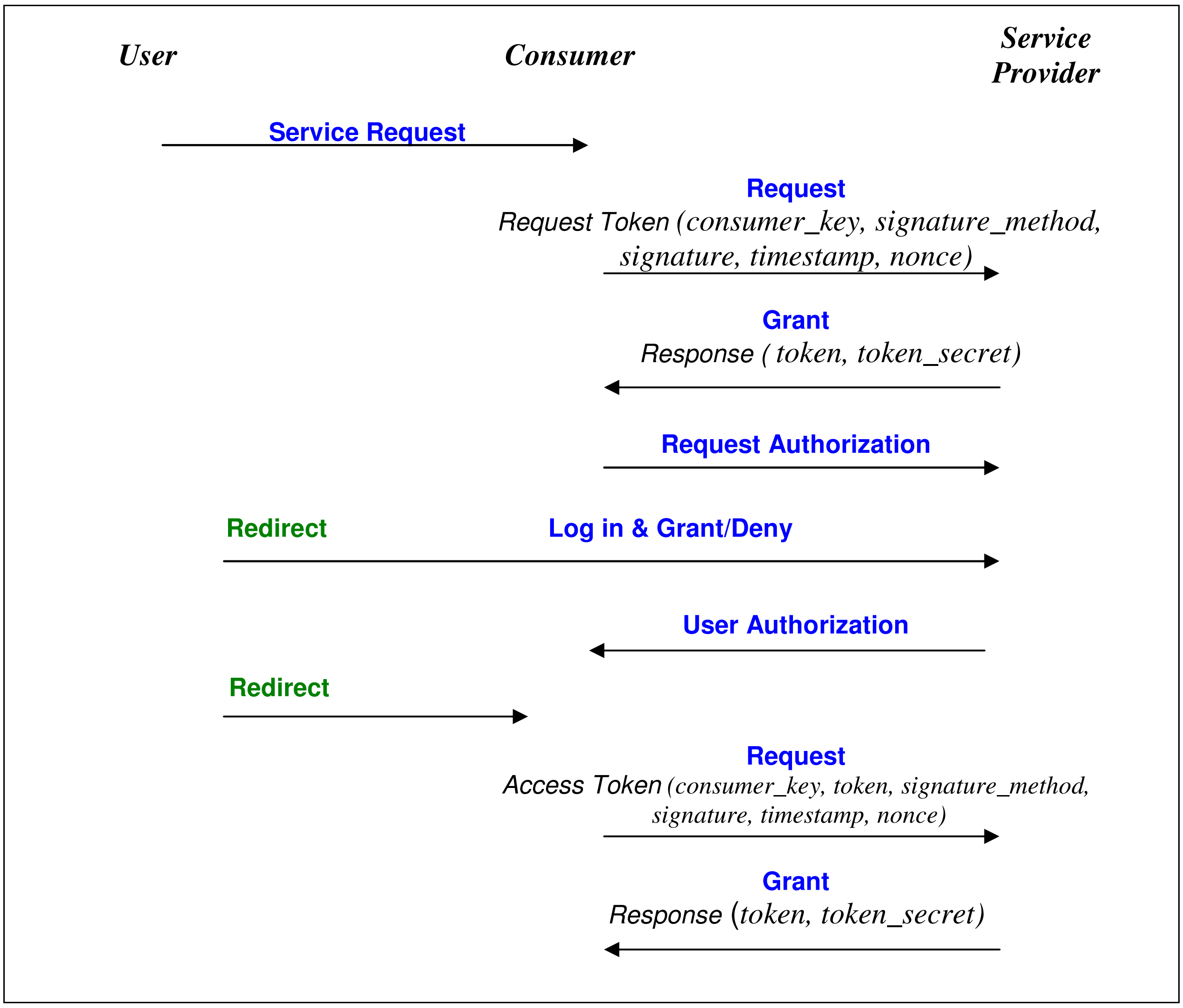


The details of procedure are as follows.

1. Consumer registers in Service Provider with a Consumer Key and matching Consumer Secret that together authenticate the Consumer (as opposed to the User) to the Service Provider. Consumer-specific identification allows the Service Provider to vary access levels to Consumers (such as un-throttled access to resources).

2. User registers in Service Provider with her identity and password.

C.1.2 The protocol


User
Consumer
Service Provider
Service Request
Request
Request Token (consumer_key, signature_method, signature, timestamp, nonce)
Grant
Response (token, token_secret)
Request Authorization
Redirect
Log in & Grant/Deny
User Authorization
Redirect
Request
Access Token (consumer_key, token, signature_method, signature, timestamp, nonce)
Grant
Response (token, token_secret)

Authentication is done in three steps:

1. The Consumer obtains an unauthorized Request Token.
2. The User authorizes the Request Token.
3. The Consumer exchanges the Request Token for an Access Token.

For the details of procedure, refer to [6].

Comparing with the SAP use case, Consumer here refers to AHS and Service Provider refers to R. The difference between these two cases is that, AHS knows clients at the stage of setup, but it is not necessary for Consumer to know Users before the authentication starts. The services that Consumer provides could be treated as external services that Service Provider provides to the User.

C.2 OpenID protocol

OpenID is an open, decentralized, free framework for user-centric digital identity. OpenID eliminates the need for multiple usernames across different websites, simplifying user online experience. OpenID takes advantage of the existing internet technology (URI, HTTP, SSL, Diffie-Hellman) and realizes that people are already creating identities for themselves whether it be at their blog, photostream, profile page, etc. With OpenID you can easily transform one of these existing URIs into an account which can be used at sites which support OpenID logins.

For businesses, this means a lower cost of password and account management, while increasing site visitor registration conversion rates. OpenID lowers user frustration by letting users have control of their login.

There are three parties in OpenID application system, End user, Relying Party and OpenID Provider. The data flow is described in Figure 6.

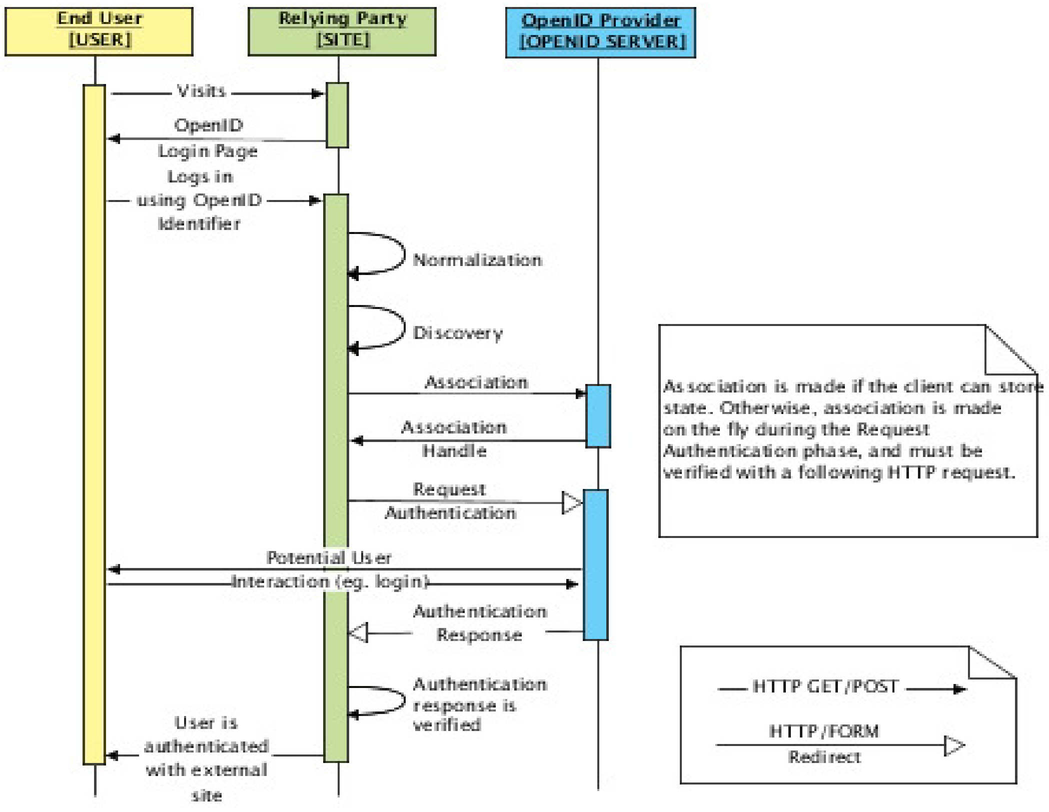

End User [USER]
Relying Party [SITE]
OpenID Provider [OPENID SERVER]
Visits
OpenID Login Page
Logs in using OpenID Identifier
Normalization
Discovery
Association
Association Handle
Request Authentication
Potential User Interaction (eg. login)
Authentication Response
Authentication response is verified
User is authenticated with external site
Association is made if the client can store state. Otherwise, association is made on the fly during the Request Authentication phase, and must be verified with a following HTTP request.
HTTP GET/POST
HTTP/FORM Redirect

Figure 6: The data flow in OpenID

For the details of the procedure, refer to [7].

The reasons why it is not suitable to the SAP use case.

1. OpenID looks like an agent-based single sign-on scheme.
2. It does not build the key agreement.

3. If AHS works as an OpenID Provider, this protocol does not provide any mechanism for secure communications between OpenID and End User or Relying Party. Otherwise, the fourth party is needed.
4. In the SAP use case, clients never communicate with R directly, and any data exchange must pass AHS, but in an OpenID application system, end users exchange data with Relying Party directly.

D. Comparison of the protocols

Because OpenID protocol does not suit the SAP use case, we do not discuss it further. We compare other four protocols in Table 2.

Table 2: Comparison of four protocols

Security Function	Protocol 1	Protocol 2	Protocol 3	OAuth
Mutual Authentication	√	√	√	
Freshness of the agreed session key	√	√	√	
Confidentiality	√	√	√	√ (with TLS or SSL)
Integrity	√	√	√	√
Symmetric encryption	√	√	√	√
Asymmetric encryption			√	√
Digital signature				√
Need TLS or SSL				√
Risks of redirection (e.g. Phishing attacks)				√
Number of secret keys of a client(user)	1	2	2	1
Number of Rounds	2	2	2	4
Computation Cost	low	low	medium	high

Based on the above table, we give summary on the differences among these four protocols, especially between the three proposed protocols and the OAuth protocol.

1. All the proposed protocols offer the mutual authentication, while the OAuth protocol only accommodates one-way authentication. Therefore, there exists a main issue in the OAuth protocol, that is, the user and the Service Provider cannot be sure that the user who initiated the OAuth flow (and thus has logged in at the user side) is the same user who logged into the Service Provider during the authorization process. If something akin to SAML's SSO model were in play (where identities of the principal at the consumer & SP sites are federated in a privacy-preserving manner – meaning no correlation issue), then ensuring it is the same user would be a no brainer.

2. Comparing to the proposed protocols, we did not find a mechanism in OAuth for refreshing the session key if it does not integrate with SSL.

3. As the authors declaimed in OAuth Core 1.0 that "The OAuth specification does not describe any mechanism for protecting Tokens and secrets from eavesdroppers when they are transmitted from the Service Provider to the Consumer". This means, to protect the confidentiality of the exchanged data, OAuth must be used in conjunction with a transport-layer security mechanism such as TLS or SSL which does provide such protection.

4. Protocols 1 and 2 apply only a symmetric encryption to protect data, but Protocol 3 and OAuth integrate symmetric and asymmetric encryptions. That means both of them need a PKI. In particular, users in OAuth have to create two digital signatures (instead of using the credentials) with a signing algorithm to obtain the access permission from the Service Provider, and the user still needs the third digital signature to retrieve the services provided by the Service Provider. So, from the perspective of Computation Cost, Protocols 1 and 2 are more efficient than Protocol 3 and OAuth.

5. Although a client in Protocols 2 and 3 needs to keep two secret keys, in fact, the client only needs to remember a password, as the second secret key is encrypted with the first one. Therefore, in these four protocols, a client only needs to remember one secret key.

6. To obtain the authentication, the three proposed protocols need two rounds (note: one round means a party sends a request to another party and receives a response from that party), while the OAuth protocol needs four rounds. Hence, OAuth might have a longer delay than the proposed three protocols.

7. Because a user in the OAuth protocol is redirected twice, this may cause Users to become inured to the practice of being redirected to websites where they are asked to enter their passwords. If Users are not careful to verify the authenticity of these websites before entering their credentials, it will be possible for attackers to exploit this practice and steal Users' passwords (i.e., phishing attacks). However, in the three proposed protocols, no redirection is needed.

8. OAuth was designed for web applications and requires the public-key algorithms, which are regarded as inefficient in security computation. Adopting OAuth implies that the computational overhead has to be increased. Although it should be fine for the current versions of iPhone, we are not sure how it will perform for less powerful smart phones.

9. OAuth requires a trusted third party who signs public-key certificates. Although this party generally is an external party such as VeriSign, we could assume that an internal server such as the R server signs certificates. This assumption requires an additional security code to be written.

10. Compared with our protocols 1 and 2, OAuth is a much more complex software package, which requires modifications in order to meet all desirable security requirements.

11. Our protocols 1 and 2 can be easily developed in terms of coding. Basically, we only require hashing and encryption, whose codes can be found from open sources.

12. In Protocol 2, if an attacker installs malicious software in the mobile to replace the original software of processing password, the attacker would know the client's password and session key. To avoid such an attack, the application in a mobile phone can generate a hash code of running software of processing password along with time stamp and send the encrypted hash code to AHS. Since AHS knows the running software of processing password and the time stamp, it can verify whether the original software is replaced or not.

E. New protocol with a different setting

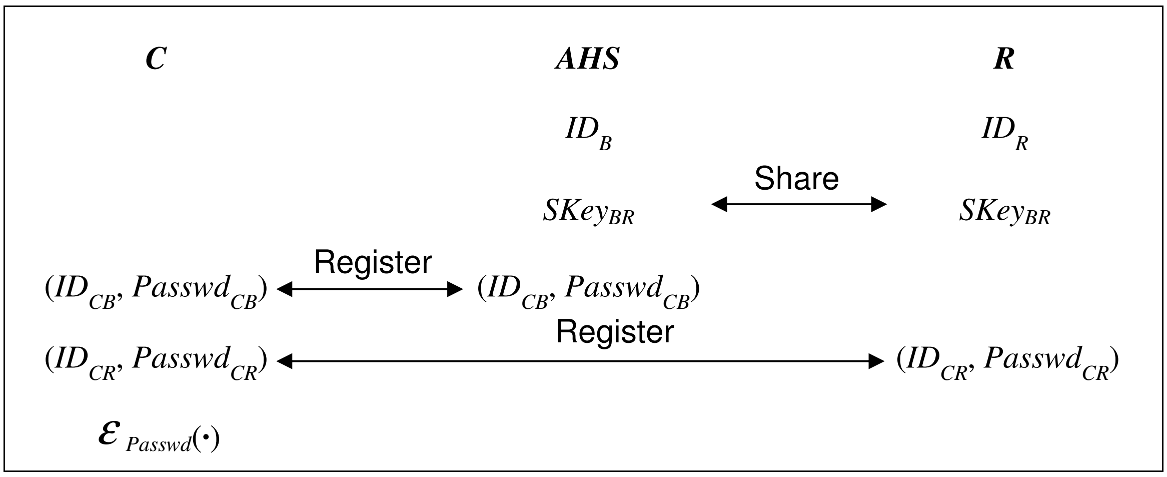
The above three proposed protocols are based on the setting that clients register only in R. Now we propose a new authentication protocol based on a different setting that all clients have to register in both AHS and R. The security requirement is as follows:

1. The communication channel between AHS and R is secure with their secret session key $SKey_{BR}$.

2. After the authentication, the channel between C and AHS is secure with their secret session key $SKey_{CB}$ that is built in the procedure of authentication.

3. The session key $SKey_{CB}$ between C and AHS is refreshed after a service access is finished.

E.1 Setup



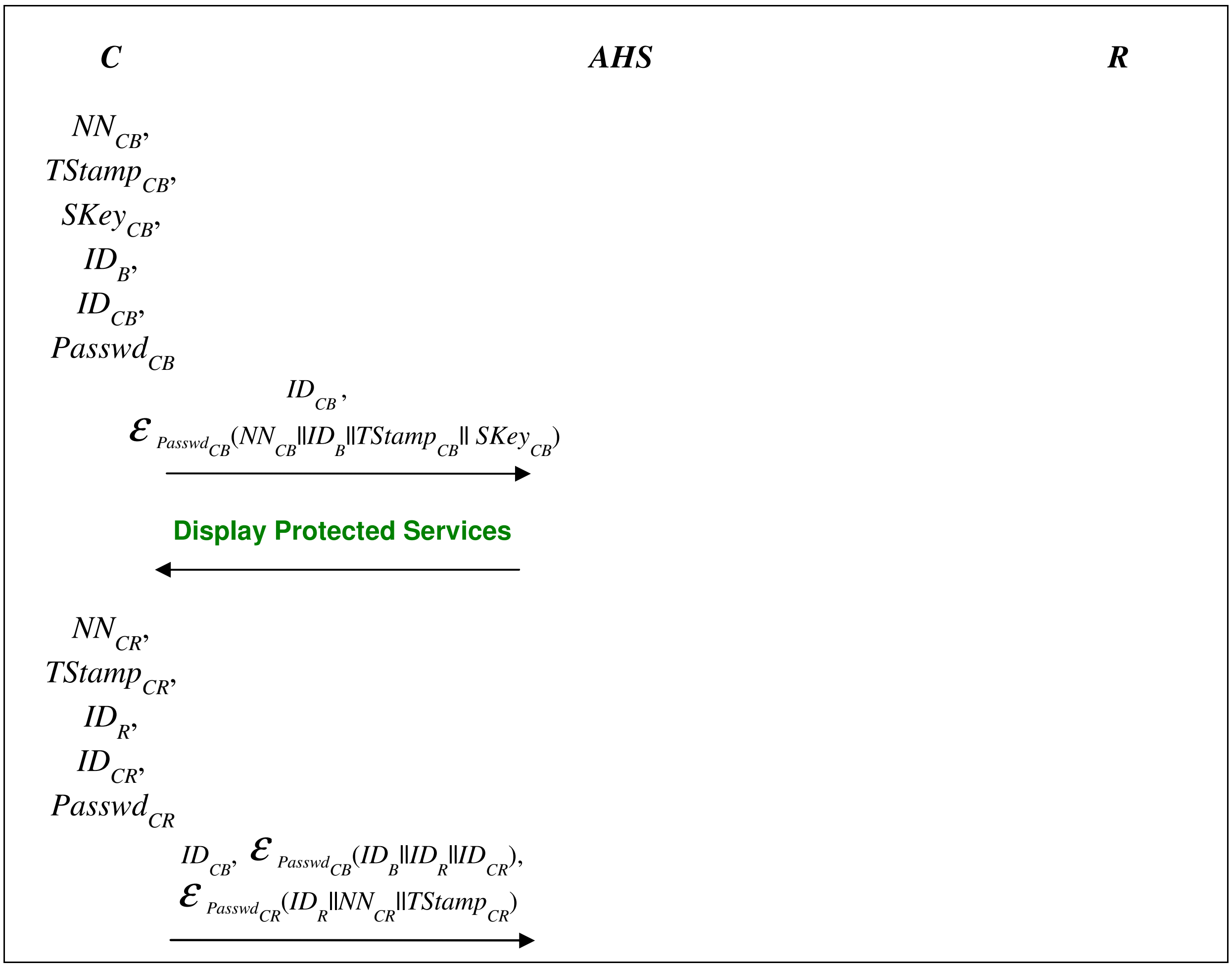
The details of procedure are as follows.

1. AHS has its identity ID_B known to all consumers and R, and shares a secret session key SK_{BR} with R.

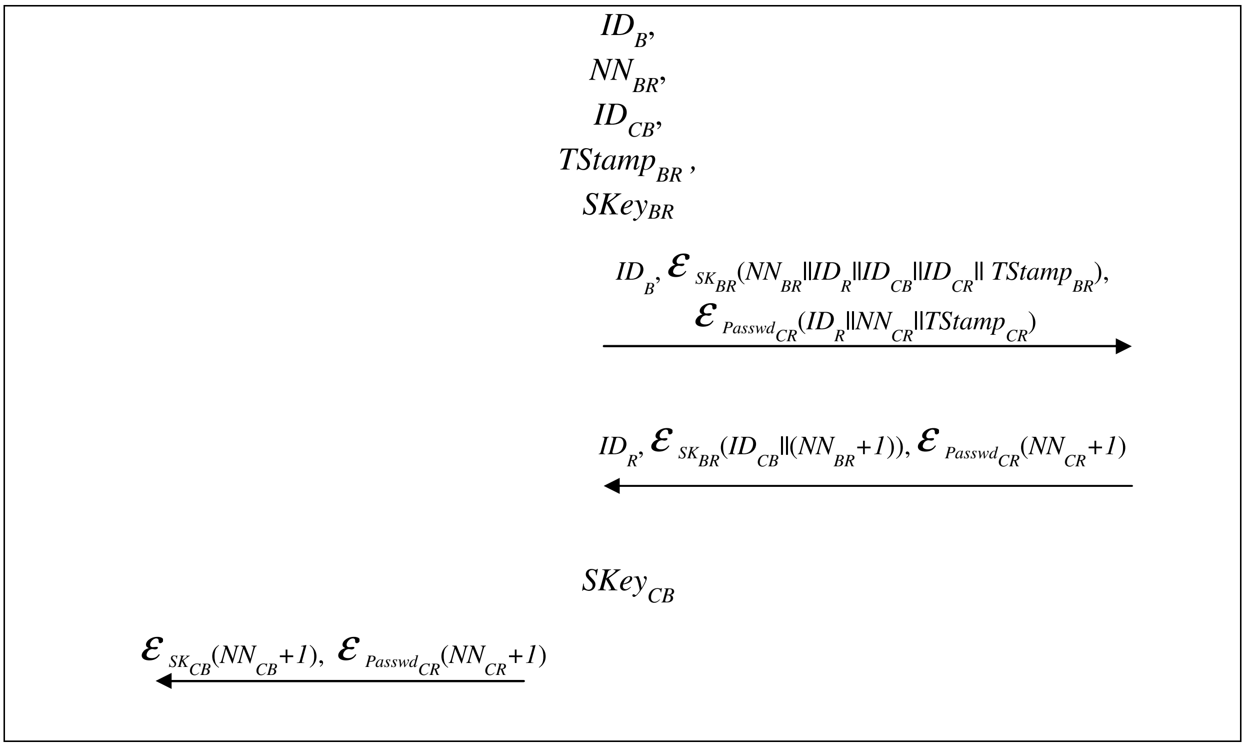
2. R has its identity ID_R, which is known to all consumers and AHS, and shares a secret session key $SKey_{BR}$ with AHS.

3. Every client C registers in AHS and R for their protected services, and obtains the identities and corresponding passwords in AHS and R, *i.e.,* $(ID_{CB}, Passwd_{CB})$ and $(ID_{CR}, Passwd_{CR})$, where every password can be used as a secret encryption key in a symmetric cryptosystem $\mathcal{E}(\cdot)$ that both AHS and R know.

E.2 The protocol



C
AHS
R
NN_{CB}, $TStamp_{CB}$, $SKey_{CB}$, ID_B, ID_{CB}, $Passwd_{CB}$
ID_{CB}, $\mathcal{E}_{Passwd_{CB}}(NN_{CB}||ID_B||TStamp_{CB}||SKey_{CB})$
Display Protected Services
NN_{CR}, $TStamp_{CR}$, ID_R, ID_{CR}, $Passwd_{CR}$
ID_{CB}, $\mathcal{E}_{Passwd_{CB}}(ID_B||ID_R||ID_{CR})$, $\mathcal{E}_{Passwd_{CR}}(ID_R||NN_{CR}||TStamp_{CR})$


ID_B,
NN_{BR},
ID_{CB},
$TStamp_{BR}$,
$SKey_{BR}$
ID_B, $\mathcal{E}_{SK_{BR}}(NN_{BR}||ID_R||ID_{CB}||ID_{CR}||TStamp_{BR})$,
$\mathcal{E}_{Passwd_{CR}}(ID_R||NN_{CR}||TStamp_{CR})$
ID_R, $\mathcal{E}_{SK_{BR}}(ID_{CB}||(NN_{BR}+1))$, $\mathcal{E}_{Passwd_{CR}}(NN_{CR}+1)$
$SKey_{CB}$
$\mathcal{E}_{SK_{CB}}(NN_{CB}+1)$, $\mathcal{E}_{Passwd_{CR}}(NN_{CR}+1)$

The details of authentication procedure are as follows.

Step 1: Authentication request message for AHS from C

When C needs to authenticate itself to AHS to see the protected services, C invokes the distribution of authentication procedure by sending the authentication request message, $AUTH_{CB}$, to AHS. C does the following:

1. Generates the following:

   - The nonce numbers NN_{CB} and a timestamp $TStamp_{CB}$.
   - A new session key $SKey_{CB}$.
   - $AUTH_{CB} = (ID_{CB}, \mathcal{E}_{Passwd_{CB}}(NN_{CB}||ID_B||TStamp_{CB}|| SKey_{CB}))$.

2. Sends $AUTH_{CB}$ to AHS.

Step 2: Authentication request message for R from C to AHS

When C needs to authenticate itself to R to access the locked services, C invokes the distribution of authentication procedure by sending the authentication request message, $AUTH_{CR}$, to AHS. C does the following:

1. Generates the following:

   - The nonce numbers NN_{CR} and a timestamp $TStamp_{CR}$.
   - $AUTH_{CR} = (ID_{CB}, \mathcal{E}_{Passwd_{CB}}(ID_B||ID_R||ID_{CR}), \mathcal{E}_{Passwd_{CR}}(ID_R||NN_{CR}||TStamp_{CR}))$.

2. Sends $AUTH_{CR}$ to AHS

Step 3: Authentication request message from AHS.

When AHS receives the request messages $AUTH_{CR}$, it decrypts a part of the message $AUTH_{CR}$, and rebuilds its authentication request message, which will be sent to R. It does the following:

1. Finds the password $Passwd_{CB}$ of the client identified by ID_{CB}.

2. Decrypts the $\mathcal{E}_{Passwd_{CB}}(ID_B||ID_R||ID_{CR})$ with the key $Passwd_{CB}$ to derive ID_B, ID_R and ID_{CR}, checks the correctness of ID_B to make sure that the message is not altered and the sender is the user.

3. Generates a timestamp $TStamp_{BR}$ and a nonce numbers NN_{BR}.

4. Finds the secret encryption key SK_{BR} corresponding to the identity ID_R.

5. Generates $AUTH_{BR} = (ID_B, \mathcal{E}_{SK_{BR}}(NN_{BR}||ID_R||ID_{CB}||ID_{CR}|| TStamp_{BR}), \mathcal{E}_{Passwd_{CR}}(ID_R||NN_{CR}||TStamp_{CR}))$.

6. Sends $AUTH_{BR}$ to R that is identified by ID_R.

Step 4: Authentication response message from R

Upon receipt of the $AUTH_{BR}$, R decrypts that message, verifies the membership of C, and checks if the messages are sent by AHS and the original request is from C. Finally R builds the authentication response message and sends it back to AHS. R does the following:

1. Finds the session key SK_{BR} identified by ID_B.

2. Decrypts the $\mathcal{E}_{SK_{BR}}(NN_{BR}||ID_R||ID_{CB}||ID_{CR}|| TStamp_{BR})$ to get NN_{BR}, ID_R, ID_{CB}, ID_{CR} and $TStamp_{BR}$, checks the validity of the timestamp $TStamp_{BR}$ and the correctness of ID_R to make sure that the sender is AHS and the message is valid and unaltered.

3. Finds the password $Passwd_{CR}$ corresponding to the identity ID_{CR} and decrypts the $\mathcal{E}_{Passwd_{CR}}(ID_R||NN_{CR}||TStamp_{CR})$ to verify the correctness of ID_R and the validity of the timestamp $TStamp_{CR}$ to make sure that the sender is C and the message is valid and not unaltered.

4. Generates $RAUT_{BR} = (ID_R, \mathcal{E}_{SK_{BR}}(ID_{CB}||(NN_{BR}+1)), \mathcal{E}_{Passwd_{CR}}(NN_{CR}+1))$.

5. Sends $RAUT_{BR}$ back to AHS as an authentication response message.

Step 5: Authentication response message from AHS

Upon receipt of the authentication response message $RAUT_{BR}$ from R, AHS decrypts that message and verifies if the message is received from the intended R, finds the client whom the authentication response message is for. Then, AHS builds the authentication response message and sends it to C. C verifies the response message to confirm that the authentication response message is from the intended senders. This process is achieved as follows:

1. AHS decrypts $\mathcal{E}_{SK_{BR}}(ID_{CB}||(NN_{BR}+1))$ by the session key SK_{BR} identified by ID_R to get ID_{CB}, $(NN_{BR}+1)$, and checks the correctness of $NN_{BR}+1$ with the aid of NN_{BR} that it generated for the client C identified by ID_{CB} to make sure that the response message is from R and for the client C.

2. AHS generates $RAUT_{CB} = (\mathcal{E}_{SK_{CB}}(NN_{CB}+1), \mathcal{E}_{Passwd_{CR}}(NN_{CR}+1))$ with the session that it received from C and sent it as the authentication response message to C.

3. C decrypts $\mathcal{E}_{SK_{CB}}(NN_{CB}+1)$ by the session key SK_{CB} to get the values of $NN_{CB}+1$, and verifies if the value of $NN_{CB}+1$ is correct by comparing it with NN_{CB} generated by itself.

4. C decrypts $\mathcal{E}_{Passwd_{CR}}(NN_{CR}+1)$ by the password $Passwd_{CR}$ to get the values of $NN_{CR}+1$, and verifies the correctness of $NN_{CB}+1$ by comparing it with NN_{CR} generated by itself.

5. If both results of verifications are correct, then the authentication is successful.

F. Security Discussion

The proposed protocols can fulfill the following security requirements: mutual authentication, consumer anonymity, non-repudiation, session key's freshness, end-to-end security, and data confidentiality and integrity.

The proposed protocols utilize the symmetric and asymmetric encryption, hash technology and "challenge-response" mechanism to implement the mutual authentication while keeping consumers' identity privacy.

To provide consumer anonymity, during the real time authentication stage of our protocols, the consumers' actual identities is not transported in plaintext mode. The passive attacker cannot eavesdrop on the authentication messages to get the consumer's real identity and launch the tracking attack. Additionally, AHS cannot know the consumers' identities that were registered in e-health service providers, and any e-health service provider cannot know the consumers' identities that were registered in AHS.

Consider the requirement of non-repudiation. Our protocols uses the one-way hash chaining function (e.g. MD5 or SHA) to achieve non-repudiation.

For each authentication, the session key between the consumer and AHS is reset differently. This ensures the session key's freshness.

The requirement of end-to-end security is also addressed in the proposed protocol. Since the full communication path is protected under encryptions, data confidentiality and integrity are both achieved.