Universal Journal of Communications and Network 3(4): 89-98, 2015 http://www.hrpub.org
DOI: 10.13189/ujcn.2015.030403

Security Protocols with Privacy and Anonymity of Users

Nazri bin Abdullah[1,*], Sead Muftic[2]

[1]Department of Communication Systems, Royal Institute of Technology, Sweden
[2]SETECS, Inc., USA

Abstract Security is very important for many Internet and mobile applications, but with the recent proliferation of tracking and profiling of users and unauthorized distribution of their personal data, user *privacy* is also becoming a very important issue. In addition, new technologies are introducing the possibility of innovative applications that require *user anonymity*. Privacy means that user identities and transactions are revealed and known only to transaction partners and only during the execution of the transaction. Anonymity, on the other hand, means that even transaction partners do not have access to that information. The requirements for providing standard security services, that require sharing of user identities and security credentials, are contrary to the requirements of privacy and anonymity. This paper describes an innovative solution to this problem: a design of extensions of standard security protocols – user authentication, key exchange protocol, and authorization protocol – to include privacy and even anonymity of users. The solution is based on the concept of a secure application proxy server, special cryptographic protocols, and encapsulated security objects.

Keywords Security, Privacy, Anonymity, Secure Proxy, Cryptography, Protocol, Strong Authentication, Access Control, Authorization, Single Sign-on

1. Introduction: Problems and Motivations

The research results described in this paper address the issues of the security, privacy, and anonymity of users and their transactions when using Internet applications. Applications addressed in this paper are either Web applications, accessed by users using standard browsers, or mobile applications, accessed through smartphones. When performing secure transactions with such applications, users use their identity and security credentials, perform various actions, and generate various transactions data.

When providing standard security services to users, such as authentication or authorization, using current security technologies and protocols, users are required to share their identities, security credentials, actions, and transactions data with application services providers (ASPs). This practice exposes users' private and sensitive data to various threats, including not only theft by hackers, but also impersonation and violation of privacy by legitimate services providers. In addition to that problem, standard practice of most ASPs today is to track users by placing cookies in their browsers, profile users by collecting their sensitive personal data, and use those data for various commercial purposes without users' consent and most often, even without their knowledge ([13], [14]). This practice violates *user privacy*, which is becoming a more and more serious issue and an important requirement for Internet applications and transactions [15].

On the other hand, new mobile technologies are very convenient for implementation of new, innovative personal and business applications that require complete *user anonymity*. Examples of such applications are personal payments, digital notaries, voting systems, and medical applications. These types of applications require identification and authentication of users, but transactions should be anonymous. The technology of a blockchain as a public ledger for Bitcoin payments has introduced just an initial concept of anonymous, personal, distributed, and community-based peer-to-peer transactions.

In this paper we consider privacy and anonymity of users to be the following concepts [4]:

(*Internet Privacy*): An Internet application or a broader Internet environment provides *privacy of users* if the identities of transaction parties and their actions are not accessible (revealed) to any party other than transaction partners directly involved in the transaction. This means that transaction partners – individuals or Web Servers obtain and can validate user's identity credentials and know about user's actions, but can not share them with any other party.

(*Internet Anonymity*): An Internet application or a broader Internet environment provides *anonymity of users* if the identities of transaction parties and also transaction metadata are not accessible (revealed) to any party, including even direct transaction partners. An example of such application is Bitcoin payments, where identities of transaction parties are completely anonymous.

Providing standard security services in combination with user privacy and anonymity may at first seem contradictory or even impossible. For example, to be authenticated, the user must first provide his/her identity and then actively participate in the protocol to prove its correctness and validity. This approach, obviously, does not provide user privacy and anonymity. To be granted authorization to perform some action, users must provide their roles and other authorization attributes. Similarly, to exchange encrypted resources, users must first exchange crypto keys using protocols that also require explicit user identification and authentication. These examples of standard security protocols clearly indicate that if they are performed using current methods and techniques, user privacy and anonymity cannot be guaranteed.

Some applications that require privacy and anonymity can be performed without user identification and authentication. An example is anonymous payments [11]. As long as the recipient can validate that the payment transaction is valid, correct, and legal, the identity of the payee is not required. However, many applications that require anonymity and privacy also require user identification and authentication. An example is the digital voting system. Voting transactions require anonymity, but at the same time, the voter must be recognized (not identified) as a legitimate registered voter and one who has not yet cast his/her ballot.

This paper presents a solution to provide standard security services in combination with privacy and anonymity by designing security infrastructure based on the use of *security proxies*. These are specifically designed servers, located between users and ASP servers that act as user agents for transactions. In addition to channelling transactions and resources between users and ASP servers, these proxy servers also perform two very important additional functions. First, they keep all user security credentials (crypto keys, certificates, security tokens, etc.) and use them to preform cryptographic transformations on user data objects and resources. In this way, all of these objects and resources are in an encrypted form when stored on the ASP side of the secure proxy and transformed to a clear form when presented to users. Second, these security proxies access the ASPs' servers on behalf of users, eliminating the possibility to place cookies onto users' browsers and at the same time also hiding user locations. Thus, security proxies effectively protect user data and the privacy of user locations and actions.

The concept and operations of the secure proxy server has already been described in our report [1] and in our research paper [4]. In those documents, we described how the secure proxy server provides privacy and anonymity only against ASPs but not against security service providers (SSPs). In this paper, we extend the concept of the secure proxy server and its protocols to provide privacy and anonymity even against SSPs. Furthermore, in our previous version, the control of user data was shared between the user and the secure proxy server, while in this version, all sensitive data are fully under user control and shared only with the user's explicit permission and consent.

Another problem that this research solves is *anonymity of users* when performing transactions using applications that require anonymity. At the time of this research, there were no effective mechanisms or protocols that enabled users to perform anonymous transactions. On the contrary, requirements by ASPs to share users' data and credentials and for users to be explicitly identified eliminate the possibilities to perform transactions anonymously. This research created some initial solutions for such transactions. But, providing privacy and anonymity with transactions that also require standard security services, is a topic for further research. The solutions described in this paper are based on the use of so-called anonymizing networks located between secure proxies and ASP servers [8] and the use of pseudonyms as user identities.

In summary, this research solves the problems of providing user privacy and anonymity when performing standard security protocols. Our results overcome the controversy between, on one hand, the explicit sharing of identities and credentials for security and, on the other hand, prevention of that sharing for privacy and anonymity. The protocols described in this paper can be used with all new applications that require the privacy and anonymity of validated users.

2. Literature Review

This paper describes specific solutions for providing privacy and anonymity in combination with standard security protocols: user authentication, key exchange, and user authorization. For *user privacy*, the suggested solutions are *(a)* secure proxies that separate and hide users from ASPs and SSPs and *(b)* special cryptographic protocols that these proxies perform. For *user anonymity* and the anonymity of their actions, the paper suggests the use of anonymous user identities based on use of cryptographic credentials.

Similar ideas have been presented in the literature, but none of these solutions provides privacy and anonymity in combination with standard security protocols.

A group of researchers from the University of California and Brigham Young University introduced *Delegate*, a proxy-based architecture for secure website access from untrusted machines [3]. Delegate is designed to address the security vulnerabilities of untrusted machines, such as key logging, password sniffing, shoulder surfing, and session hijacking. The objectives of the architecture are: *(a)* authenticate users who access Internet service providers from untrusted machines with temporary credentials to avoid these users from being manipulated in the future; *(b)* detect and prevent session hijacking to stop malware from performing unauthorized transactions while users perform their regular operations; *(c)* limit the scope of potential damages by reducing the attack surface while users access the Internet from an untrusted machine; and *(d)* minimize the modifications required by an Internet server or the user to deploy the system. There are four major components of the

Delegate architecture: *(i)* trusted proxy, *(ii)* Internet server, *(iii)* untrusted computer, and *(iv)* trusted mobile device. The authentication method in the trusted proxy is a one-time password or PIN, which is used to authenticate users before the user navigates to an untrusted machine. It has an auto login function based on prior setup credentials, which are stored in a database. However, the authors did not describe the protection of user credentials in the database. Other functions in this design validate HTTP requests and remove sensitive user information. Overall, the architecture of Delegate is concerned only with the confidentiality of user information but not providing user privacy while on the Internet. Its trusted proxy has several shortcomings, as it does not apply any trust framework, such as Public Key Infrastructure (PKI) or trusted computing. Without the use of public key cryptography, it is very difficult to enforce user control and consent, including the privacy of users and their data. All systems based on use of only secret key cryptography must share symmetric keys between users and security services providers, usually as master keys ([16], [17]). Thus, the solution presented in this paper addresses certain aspects of security but not user privacy.

Jian Wu and Zhimin Huang from Zhejiang University of China in their paper entitled "*Proxy-based Web Services Security*" presented a proxy-based web services security protocol that authenticates users based on PKI and authorizes them based on privilege management infrastructure (PMI) [9]. However, the proxy that they introduced is not an independent third party, but a service located in both the client and the server. The service captures both HTTP requests and response messages and then extracts messages formatted in compliance with the Simple Object Access Protocol (SOAP) from the HTTP message body. SOAP messages are encrypted, signed, authenticated, and authorized by the service provider in each client and server machine. Users can customize their security-related parameters (e.g., encryption and signature methods and public and private keys) using the GUI tool. However, the architecture does not define user certificates and where/how they are generated, key storage, or identity management.

The paper "*A Proxy-based Security Architecture for Internet Applications in an Extranet Environment*" by Andy Dowling and John G. Keating proposed an approach to proxy-based security that is similar to what we present in this paper [2]. The major components in their architecture are: *(a)* proxy-based security services and *(b)* advanced authorization and access controls. Security proxies are deployed at client and server sites to filter networked applications' communications for vulnerability scanning. They use standard digital certificates, such as the X.509 public-key certificate (PKC), to securely bind a set of access privileges to an identity when a single identity is accessing different resources. However, the Certification Authority (CA) in their concept is used in the offline mode, so real-time validation of certificates, especially using Certificate Revocation Lists (CRLs), is not possible. The operations performed by CA, such as issuing, revoking, and updating CRLs, are performed manually. This implies that there are time gaps in the process of managing certificates, which introduces time periods when the system is vulnerable to impersonations. Certificate trust parameters (certificate policies) are stored in local security databases maintained at the client and server proxy sites, which contradicts well-established practice and recommendations that these policies be maintained and enforced by a special type of CA servers, usually called policy CA servers.

There are several papers that treat privacy and anonymity of users in different networking environment and applications. One of the very first papers that addressed these aspects and even specified precise concept of privacy and anonymity was [18]. The definitions of sender's and receiver's privacy in this paper were adopted form that paper. Most of the research papers treat privacy in the context of peer-to-peer or group communications and not many can be found that consider the aspects when users are communicating with Application Servers. Very important is that we could not find any research paper solving the aspects of privacy when performing security protocols and accessing security services providers.

Research papers [19], [20] and [21] consider different aspects of privacy when communicating with users groups. They are all based on the same principle –using randomized protocol in a group to provide privacy and anonymity between the source and target of a transaction. In [19] there is no specific application, in [20] the application is a question and answers system, while in [21] the application is polling application. Significant difference of our solution compared with these ideas is that is it is applicable to group applications, but also to communications of users with applications and security servers. Paper [22] considers anonymity of users when sharing files, but it is based on the prerequisite of trust with users participating in the protocol. The system is implemented as adaptive peer-to-peer network of nodes that query one another to store and retrieve data files. The most important weakness of all such "group" anonymizing networks is that trust must be placed in the first connecting notes. When the source user/node initiates the transaction, the first accessed node must maintain privacy of the connection. More advanced ideas for sharing files are based on the concept and use of Bitcoin blockchain ([23], [24], [25]). This concept does not depend on the trust in any party.

The most important difference of our design with all these ideas is that our solution does not depend on any third party, it does not assume trust in any participant of the transaction, and it is applicable to wide range of applications.

3. Federated Security Architecture

Our security architecture is proxy-based and its conceptual model involves seven components in the process of the private and autonomous authentication and authorization of users, as shown in Figure 1.

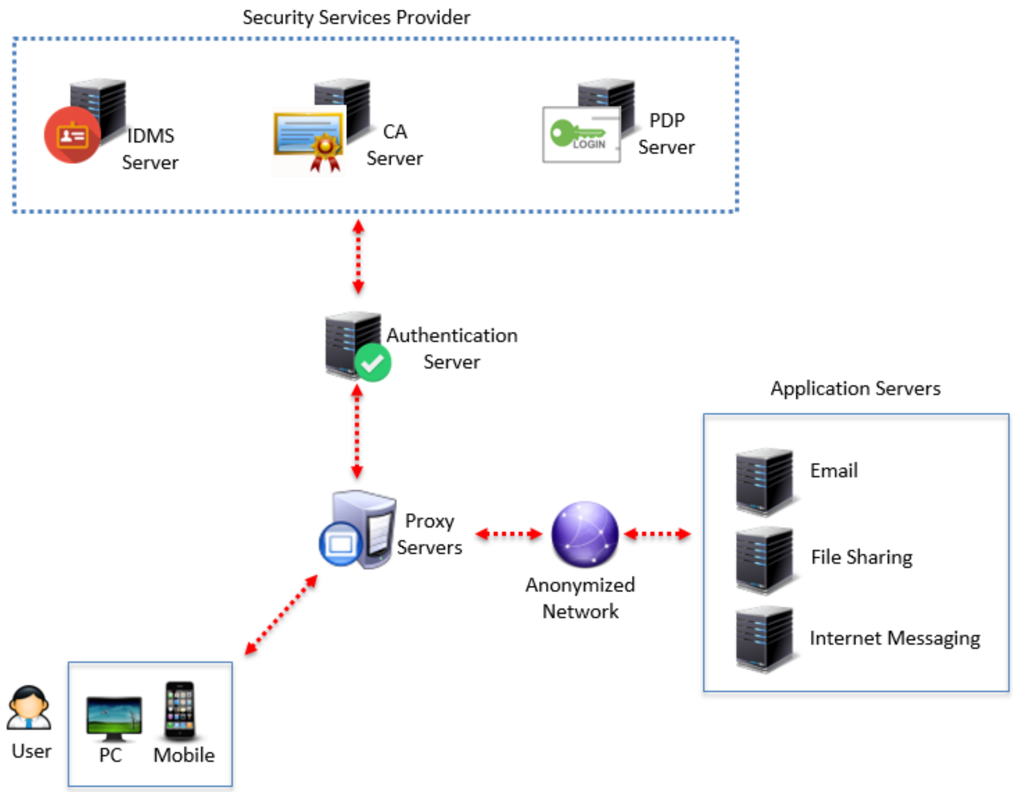

Figure 1. Conceptual Model of the Proxy-based Security Architecture

(a) *User station*: This is personal computer or mobile device used to enter user credentials for the authentication process;

(b) *Security Proxy (SP) Server*: A server to re-direct user requests to ASPs and SSPs for application and security services. The server stores users' public security credentials and in that way assists users to perform security protocols instead performing them with Application or Security Services Providers. For its cryptographic operations it uses locally attached hardware security module (HSM);

(c) *Strong Authentication (SA) Server*: The server that performs user authentication by receiving user credentials during the authentication process, verifying them, performing strong authentication protocol, and if successfully completed, issuing an Single Sign-On (SSO) ticket for the user and sending it to the proxy server for the subsequent SSO protocol;

(d) *Identity Management System (IDMS) Server*: The server that manages user identities, i.e., their creation, storage, distribution, validation, updates, and removal;

(e) *Certificates Authority (CA) Server*: The server that manages user certificates by issuing, distributing, validating, and revoking them;

(f) *Policy Decision Point (PDP) Server*: This server makes access and authorization decisions based on user's role(s) and the appropriate authorization policies;

(g) *Applications Services Provider (ASP) Servers*: These servers host web services or applications and provide corresponding application services.

The essence of the designed protocols is to provide authentication and authorization of users to ASPs when users access these servers, but instead of each ASP performing its own verification, the security architecture is based on the standard concept of shared SSPs. However, the distinguishing features of the protocols described in this paper are that they are all extended with user privacy and anonymity.

In order to provide the highest level of assurance (Assurance Level 4) and therefore eliminate the need for any trust in the Security Proxy Server, all user private security credentials are stored in user's hardware security token [6]: *"Level 4 authentication is based on proof of possession of a key through a cryptographic protocol. At this level, in-person identity proofing is required. Level 4 is similar to Level 3 except that only "hard" cryptographic tokens are allowed. The token is required to be a hardware cryptographic module validated at Federal Information Processing Standard (FIPS) 140-2 Level 2 or higher overall with at least FIPS 140-2 Level 3 physical security. Level 4 token requirements can be met by using the PIV authentication key of a FIPS 201 compliant Personal Identity Verification (PIV) Card"*

With this approach, the next important issue of the trust in the Security Proxy Server is its correct operations. This aspect is addressed by two methods: *(a)* validation of its operations and functionality by the independent third party, and *(b)* assurance that correct operations will always be performed by protecting its executable code using cryptography. This approach requires special, so called

Security Java Class Loader, which has been described in our previous research [10]. With these approach there is no need to place any trust in operations of the Secure Proxy.

The final remaining issue is that the Security Proxy Server represents single point of failure. This is the case if the conceptual model is at the same time also the deployed architecture. But, our security architecture is in fact distributed infrastructure of federated Secure Proxies. This means that our architecture comprises multiple Proxies, each serving local users / environment, but at the same time all of them are transparently mutually federated. User stations are simultaneously linked to multiple Secure Proxies, so in case of one being down, some other Server will support user operations. This multi–connection feature is equivalent to the approach used with Bitcoin Wallet and can be achieved using Web redirection. Federated architecture of multiple Security Proxies is shown in Figure 2.

Federation protocol between Security Proxies comprises two types of exchanges: first, the Proxies exchange security credentials of each user after registration, and second, through Web redirection they exchange user parameters for security protocols. Security credentials distributed between Secure Proxies are user identity attributes, X.509 certificates, and encrypted authentication and authorization tokens. Parameters for security protocols are initial message during strong authentication, single sign-on token, and authorization ticket.

With the described federated security architecture, all Security Proxies collectively represent in fact distributed and federated communication infrastructure, supporting security, privacy, and anonymity protocols and transactions without any need to place trust in their availability, functionality, reliability, and continuous availability.

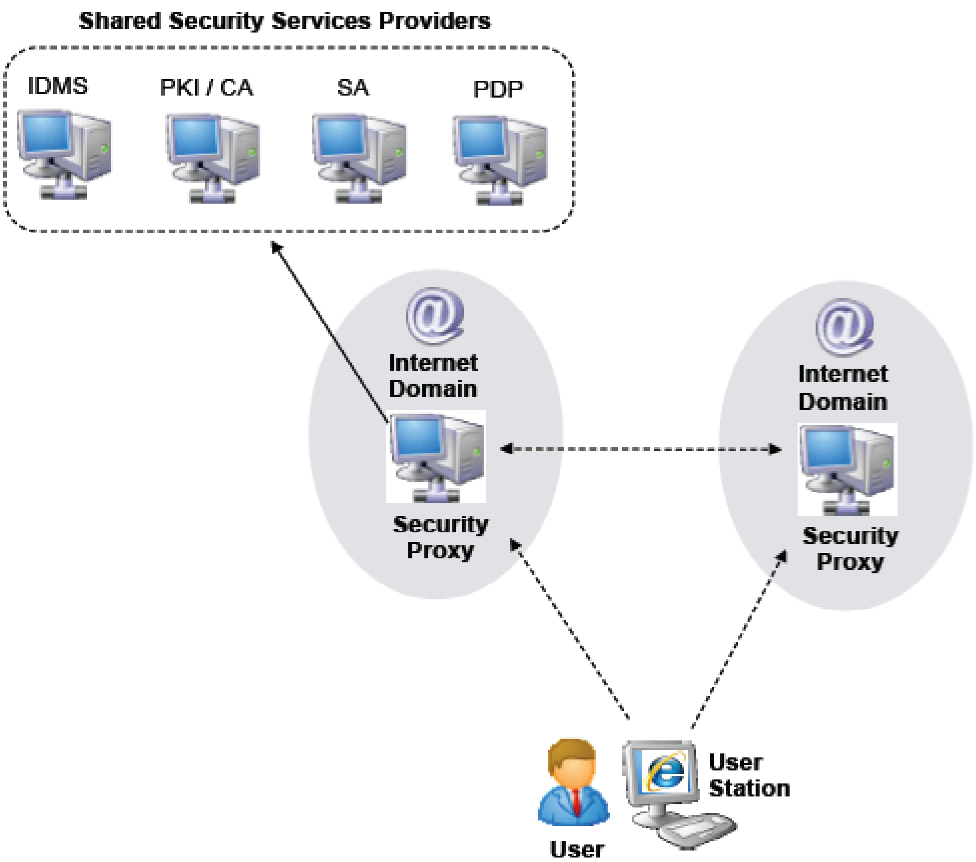

Figure 2. Federated Security Architecture

4. Security Protocols with Privacy and Anonymity

4.1. Modified Protocols Based on Secure Proxies

User authentication procedure is initiated when the user accesses an ASP to request an application service. With the current standard approach, such a request is performed by users directly accessing an ASP server. If shared SSPs are deployed, the targeted ASP server redirects that request to the corresponding SSP server.

With our security architecture, however, users access all ASP and SSP servers only indirectly, through the secure proxy [4]. The proxy assists users with all security protocols and in that way extends all security services with privacy and anonymity against both ASPs and SSPs.

Three protocols are required to perform user authentication: the user registration protocol, the initial authentication protocol, and the single sign-on protocol. The registration protocol is performed between the user station, the trusted proxy server, and the IDMS server. After registration, the two final steps are issuing certificates to users by the CA server and registering user role(s) in the PDP server.

The user station, the Secure Proxy server, and the Strong Authentication server participate in the initial authentication protocol, while the IDMS server, the CA server, and PDP server assist the Strong Authentication server to validate the user's identity, certificate, and authorization role(s). This protocol is compliant with the National Institute of Standards and Technologies (NIST) strong authentication standard and is based on the challenge/response protocol [5]. If the user is successfully authenticated, the SSO ticket is issued to the user by the PDP server and stored in the trusted proxy server.

Later, when the user wants to access various ASP servers, he/she first accesses the Secure Proxy server. The Proxy authenticates the user by receiving his/her SSO ticket, which is then forwarded to the PDP server to validate the user's authorization. If everything is successful, the user's request is then passed to the corresponding ASP server.

Compared to the current standard version of the registration process, the essential extension of our three protocols is that the user registration credentials (identities) provided by the users are substituted with the pseudonyms created by the Secure Proxy server and stored in the IDMS system. The pseudonyms in our system are cryptographically encapsulated public keys of the user's private/public key pair. This approach is equivalent to that used for Bitcoin addresses. With these extensions, our protocols, in addition to security, also provide user privacy and anonymity.

Generation of Bitcoin addresses is shown in the following Figure [11]:

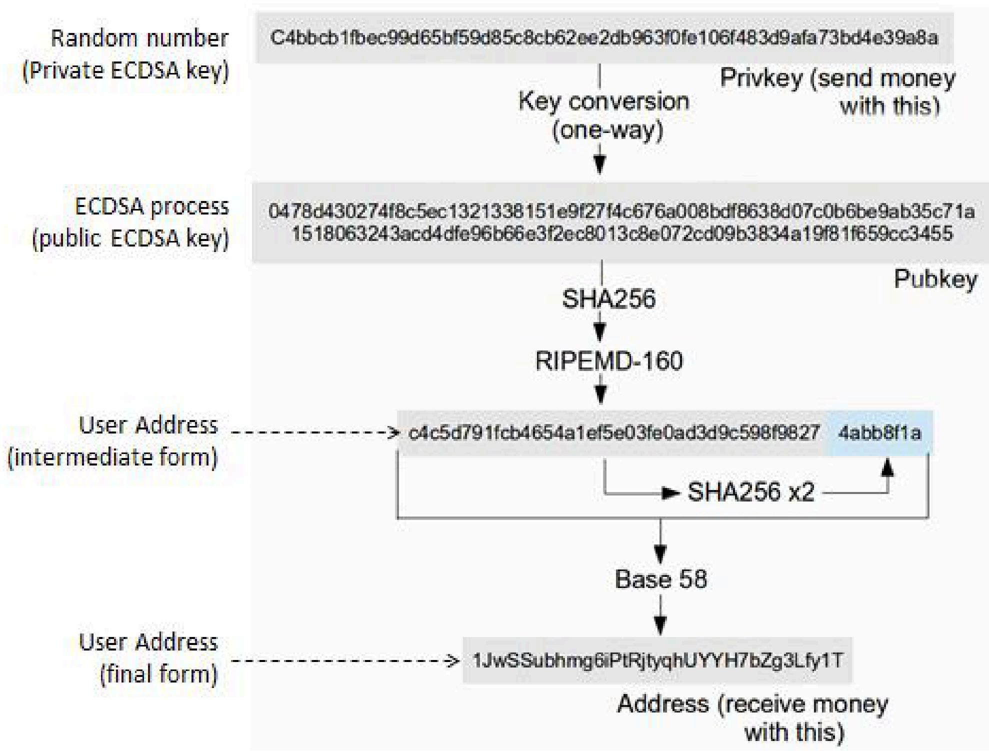

Figure 3. Generation Protocol for Bitcoin Addresses

The protocols with random pseudonyms are convenient to use because on the user's side of the Secure Proxy all credentials are in a human readable form and familiar attributes (such as first name, last name, e-mail address, etc.), while on the ASP and SSP side, they are completely unrecognizable pseudonyms.

It is important to emphasize, based on the design of the security architecture described in section 3 that Secure Proxy *(a)* does not store any user's security credentials, and *(b)* does not perform any security operations on behalf of users. These functions are performed by user's security token, i.e. smart card. Therefore, the collection of federated Proxies is in fact simple communication infrastructure, passing user requests between ASPs, SSP, and even other Proxies, without any possibility to interfere with user security credentials used in these protocols.

4.2. User Registration Protocol

The user registration protocol is performed by users who are not yet registered in the system. In order to meet Assurance Level 4, the prerequisite is that users meet face-to-face with the system administrator – security authority and get their smart card with the PN determined by the user. With that card in user's possession, registration protocol is performed as the sequence of the following five steps:

Step 1: The protocol is initiated when the new user inserts smart card in the local smart card reader and visits the SP server's home page and clicks on the "*Register*" button.

Step 2: The SP server returns to the user security servlet (Java servlet or ActiveX Module) that displays fields to enter the user registration credentials: user identity attributes and smart card PIN. User identity credentials are stored in the smart card. At the same time, the card generates a pair of public/private keys. Registration data are cryptographically enveloped using user's public key and together with Certificate Request passed to the SP Server. If such enveloped user's record is not found in the IDMS server, the SP server creates a new user identity as a random number (user's pseudonym).

Step 3: The SP server forwards Certificate Request to the CA server using user's pseudonym as the value of the attribute Common Name in the user's Distinguished Name.

Step 4: The CA server *(a)* generates the user certificate based on the SP's request and *(b)* returns it to the SP Server. The SP Server then stores it in its local certificate database and also returns it to the user.

Step 5: User stores the certificate in his/her smart card.

The described protocol works with current standard Certification Authorities, because they do not validate and verify the value of the Certificate Request attributes. As a result of the user registration protocol: *(a)* user registration data enveloped with user's public key are stored in the IDMS database table, *(b)* random user identifiers – pseudonyms are created (for privacy), *(c)* public crypto keys are used as alternative identifiers (for anonymity), *(d)* user's X.509 certificates are generated, and *(e)* the encryption key for the complete registration record is user's private key, so that use of registration attributes is fully under user's consent and control.

Random ID numbers and public keys are used as private and anonymous users' identifiers. Public/private key pairs and X.509 certificates are used for the strong authentication protocol. Registration data can only be decrypted if users' private keys are available, which provides additional assurance that users' registration data are only used with their explicit consent and control. Users use real identities for themselves and when communicating with their partners on the user side of the SP server. But all identifiers are in the pseudonymous and anonymous form on the ASP and SSP side of the SP server. Therefore, the described protocol handles the security credentials in the form that supports private and anonymous user authentication.

With this approach "reverse engineering" of user private and secure credentials is not possible, as they are all *(a)* stored in a tamper-resistant media (smart card chip) and *(b)* not publicly revealed in any of the security protocols.

It may be emphasized that this approach of enveloping user registration data stored in database tables prevents also their theft by hackers in case of intrusion. Furthermore, user security and privacy preserving credentials, stored and used in the smart card chip, can be used with all applications that require privacy and anonymity, such as e-Voting, as user identity and authorization may be verified using functionalities of the smart card, without revealing user's personal and sensitive attributes to any other party.

4.3. Initial User Strong Authentication Protocol

The initial user authentication protocol is performed by users who are registered in the system, but do not have an SSO ticket, as the sequence of the following ten steps:

Step 1: The protocol is initiated when the registered user visits the SP server's home page and clicks "*Login*" button.

Step 2: The SP server returns the security servlet (Java servlet or ActiveX module) displaying the field for the user to enter his/her smart card PIN (see Figure 4).

Step 3: If verification of the PIN by the smart card is successful, smart card is opened; user certificate is retrieved from the card and sent to the SA server (via SP server), representing the first identification message of the strong authentication protocol, in accordance with the FIPS 196 standard.

Step 4: The SA server validates the certificate using the certificate chain up to the top of the PKI and against all CRLs.

Step 5: If the certificate is valid, the SA server returns a challenge to the SP server in the form of the random value encapsulated with the public key extracted from the user's certificate, together with its own certificate. SP server forwards the challenge to the user's security servlet (Java servlet or ActiveX module)

Step 6: The servlet passes the challenge to the smart card. The card uses the user's private key to open the encapsulated challenge and then uses the SA server's public key, extracted from its certificate, to encapsulate the challenge and returns it to the SA server (via SP server).

Step 7: The SA server opens the encapsulation using its own private key and verifies the value of the response against its originally generated challenge.

Step 8: If the response is correct, the SA server contacts the PDP server, which issues the SSO ticket and returns it to the SA server. The server returns the ticket to the TP server as a positive notification, encapsulated with user's public key.

Step 9: The SP server forwards the ticket to the user's security servlet (Java servlet or ActiveX module) that stores in in the smart card.

Figure 4. Client Module for the Strong Authentication Protocol using Smart Card

As a result of the protocol: *(a)* the user is authenticated by the SA server, *(b)* his/her SSO ticket is issued by the PDP server, *(c)* the ticket is encapsulated with the user's public key, and *(d)* the ticket is stored in user's smart card for use in the subsequent SSO authentication protocol.

The validity period of the ticket is determined by the security policy enforced by the system. Managing and enforcing such a policy is outside of the scope of this paper.

4.4. Single Sign-On Authentication Protocol

The single sign-on authentication protocol is performed by users who are registered in the system and have valid SSO tickets, as the sequence of the following five steps:

Step 1: The protocol is initiated when the registered user visits the SP server's home page and clicks on the "*Link*" button for one of the ASP applications enabled for the SSO protocol. The buttons for these applications are extended with the functionality to send security servlet (Java servlet or ActiveX module) to the user's browser.

Step 2: Security servlet displays a panel on Figure 3, user enters PIN, PIN is verified by the smart card, SSO ticket is retried from the card and submitted to the SP Server.

Step 3: If the SSO ticket is still valid, the SP Server redirects user's request to the PDP Server.

Step 4: PDP Server validates the ticket and if valid it verifies whether user request to access specific ASP is in the Authorization Policy. If yes, PDP Server updates SSO ticket and returns it to the SP Server together with its access decision "*Permit*".

Step 5: SP Server forwards the new SSO Ticket to user's security servlet, which stores it in user's smart card, and at the same time redirects the user to the home page of the selected ASP Server,

As a result of the protocol: *(a)* the user is authenticated by the SA server, *(b)* his/her action request is verified against Authorization Policy, *(c)* his/her SSO ticket is updated, and *(d)* the user is directed to the application's home page.

4.5. Fine-Grained Authorization Protocol

If the activated application has certain sub-functions, special data, or sensitive and restricted actions, it can be registered in the Authorization Policy file compliant to the Extended Access Control Markup Language (XACML) standard. This is handled by the PDP Server, together with the rules for their access and use.

For that reason, whenever the user clicks on a new link on the application's home page, that request is captured by the SP Server and sent for authorization verification and approval to the PDP Server. The authorization request contains the user's SSO ticket. Because this ticket was issued by the PDP Server, it will be used to recognize the user and his/her roles and authorization will be approved or rejected based on the local authorization policy.

The XACML Policy, requests/responses, PDP decisions, and all other details of the authorization protocol are based on the XACML standard.

5. Implementation

Our current implementation is based on the concept of the Business Information Exchange (BIX) proxy system [4]. The technology used is PrimeFaces [7]. All panels are implemented as JavaServer Pages (JSP) dialogs, and the processing modules are Java Beans. The front-end server is a standard Tomcat server and the back-end server is a standard SQL server. Because all components of the architecture in Figure 1 have their own certificates, all communications are protected by the SSL protocol.

Cryptographic engines are based on use of OpenSSL libraries that are FIPS 140-2 certified, and all crypto keys are of the largest size: 256 bits for symmetric key cryptography and 2048 bits for public key cryptography. A version with Personal Identity Verification (PIV) smart cards for users is also available, which makes the system able to support all protocols at the NIST assurance level 4 [6].

Current version of the Secure Proxy Server supports three security–enhanced applications: Secure E-mail, Secure Sharing of Documents, and Secure Instant Messages. Since our Secure Proxy Server I sonly "pass through" server, it is linked to all major standard Web-based E-mail systems: Google Mail, Yahoo Mail and Hotmail. For sharing of documents, the Proxy is linked to major File Storage providers, such as Google Disk and DropBox. The components of such Business Information Exchange (BIX) System are shown in Figure 5:

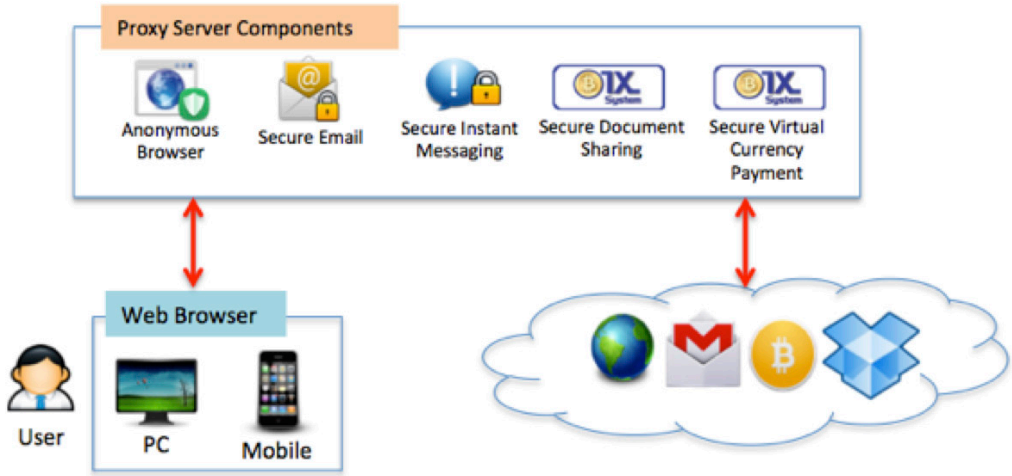

Figure 5. Applications of the Business Information Exchange (BIX) System

Very important advantages of this approach, compared to some similar implementations, are that *(a)* users do not need to install anything on their workstation, they can use only standard browsers; *(b)* users do not need to change applications that they are currently using, and *(c)* security for these applications is extended to the highest assurance level without the need for any updates of native Internet applications. Thus means that the described set of secure applications is very easy to activate and use. An important feature that significantly contributes to this property is that all applications share the same user identities, security and authorization credentials.

6. Conclusions

In this paper we have described our research and design solutions for the two most popular security services—authentication and authorization, extended with anonymity and privacy. Our solutions are based on the core principle of the Bitcoin system (the use of crypto credentials as anonymous identities), so our protocols can also be used to enhance the security of the Bitcoin and many other applications that require privacy and anonymity in addition to security.

Providing privacy and anonymity services to users, in combination with standard security services, is very difficult task as the methods for implementing these two groups of services are often mutually exclusive. We have solved that problem with the concept of federated security architecture, comprising Secure Proxy servers and two groups of their protocols: mutual federation and security protocols with users. Significant features of the described solution are that there is no need to place any trust in the Secure Proxies and that all sensitive parameters are under user control and strong protection. The strength of the solution is at the highest assurance level (Assurance Level 4). The system is easy to deploy, as instances of the Secure Proxy Servers are standard Web Applications.

One important aspect of our protocols is that for authentication o fusers and validation of transactions they use centralized security components – security servers. As extensions of that concept, we are already working on the design of equivalent protocols based on a fully distributed public ledger, which would provide the same security, privacy, and anonymity services to users, based on strict peer–to–peer transactions and without reliance on any third party [12]. With this concept all publicly available data, i.e. user pseudonyms and transactions, are not handled by any server, but instead they are distributed using the concept of public transactions ledger.

At the time of this publication the first instance for the Secure Proxy Server is available in the US at the URL: **www.bixsystem.com**. Deployment of several other instances is in preparation, in Brazil, South Africa, and Malaysia.

Acknowledgements

The authors would like to thank anonymous reviewers for their valuable comments and suggestions that have greatly improved the content and quality of the paper.

REFERENCES

[1] Bin Abdullah, N., "Security Architecture and Protocol for Protection, Privacy, and Anonymity of Users and Transactions", Licentiate Report, Royal Institute of Technology, Stockholm, Sweden, 2015

[2] Dowling, A. and Keating, J.G., "A Proxy-based Security Architecture for Internet Applications in an Extranet Environment", J Syst Software, vol. 58, no. 2, pp. 107–118, Sept 2001

[3] Jammalamadaka, R.C., van der Horst, T.W., Mehrotra, S., Seamons, K.E., and Venkasubramanian, N., "Delegate: A Proxy Based Architecture for Secure Website Access from an Untrusted Machine", in the Proceedings of the Computer

Security Applications Conference 2006. ACSAC ’06. 22nd Annual, 2006, pp. 57–66

[4] Muftic, S., bin Abdullah, N., Kounnelis, I., “Business Information Exchange (BIX) System with Security, Privacy, and Anonymity”, submitted to the Journal of Electrical and Computer Engineering, Special Issue on Innovations in Communications Security, Hindawi Publishing Corp, October 2015

[5] NIST, “Federal Information Processing Standard 196, Entity Authentication using Public Key Cryptography”, February 18, 1997, http://www.nist.gov

[6] NIST, “Electronic Authentication Guidelines”, SP 800–63–2, August 2013, http://www.nist.gov

[7] “PrimeFaces: Ultimate UI Framework for Java EE”, http://www.primefaces.org/

[8] “TOR Project: Anonymity Online”, [Online]. Available: https://www.torproject.org/. [Accessed: 10 September 2015]

[9] Wu, J. and Huang, Z., “Proxy-based Web Service Security”, in Proceedings of the IEEE Asia-Pacific Services Computing Conference, 2008. APSCC ’08, 2008, pp. 1282–1288

[10] Abbasi, A.G., “CryptoNET: Generic Security Framework for Cloud Computing Environments”, Ph.D. Dissertation, Royal Institute of Technology, Stockholm, Sweden, June 2011 (http://kth.diva-portal.org/smash/get/diva2:411892/FULLTEXT01)

[11] Bitcoin, https://en.bitcoin.it/wiki, 2010

[12] Muftic, S., “BIX Certificates: Cryptographic Tokens for Anonymous Transactions based on Certificates Public Ledger”, unpublished manuscript, submitted to the Ledger Journal (www.ledgerjournal.org), 2015

[13] The Washington Post, “Saving the Internet”, Editorial, 23 August 2014

[14] Business Week, “The Cookies You Can’t Crumble”, Editorial, 2014

[15] European Commission, “Protection of Personal Data”, http://ec.europa.eu/justice/data-protection/index_en.htm

[16] NIST, “Recommendation for Cryptographic Key Generation”, NIST SP 800–133, http://dx.doi.org/10.6028/NIST.SP.800-133

[17] Trust in Digital Life (TDL) Consortium, “Architecture serving complex Identity Infrastructures”, TDL Report, http://www.trustindigitallife.eu/uploads/Architecture%20serving%20complex%20Identity%20Infrastructures.pdf

[18] Pfitzmann, A., Waidner, M., “Networks without User Observability”, Computers & Security 2, 6, 158{166, 1987

[19] Reiter, M.K., Rubin, A.d., “Crowds: Anonymity for Web Transactions”, ACM Transactions on Information and System Security

[20] Fleming, S., Chalmers, D., Wakeman, I., “A Deniable and Efficient Question and Answer Service over Ad-hoc Social Networks”, Information Retrieval, v.15 n.3-4, p.296-331, June 2012

[21] Kacimi, M., Ortolani, S., Crispo, B., “Anonymous Opinion Exchange over untrusted Social Networks”, Proceedings of the Second ACM EuroSys Workshop on Social Network Systems, p.26-32, March 2009, Nuremberg, Germany

[22] Clarke, I., Sandberg, O., Wiley, B., Hong, Th.W., “Freenet: A Distributed Anonymous Information Storage and Retrieval System”,International Workshop on Designing Privacy-Enhancing Technologies: Design Issues in Anonymity and Unobservability, p.46-66, July 25-26, 2000, Berkeley, California, USA

[23] Wilkinson, Sh., et al., “Storj: A Peer-to-Peer Cloud Storage Network”, in collaboration. with Tome Boshevski, Josh Brandoff, and Vitalik Buterin, http://storj.io/storj.pdf

[24] Buterin, V., “Secret Sharing and Erasure Coding: A Guide for the Aspiring Dropbox Decentralizer”, https://ethereum.org

[25] Svensson, D., at al., “SecuRES: Secure Resource Sharing System”, M.Sc. Thesis, (adviser: S. Muftic), ICT/KTH, June 2015