A Framework for Secure Single Sign-On

Bernardo Machado David[1], Anderson C. A. Nascimento[1], Rafael Tonicelli[1]

Department of Electrical Engineering
University of Brasilia, Brazil
bernardo.david@redes.unb.br, andclay@ene.unb.br, tonicelli@redes.unb.br

Abstract. Single sign-on solutions allow users to sign on only once and have their identities automatically verified by each application or service they want to access afterwards. There are few practical and secure single sign-on models, even though it is of great importance to current distributed application environments. We build on proxy signature schemes to introduce the first public key cryptographic approach to single sign-on frameworks, which represents an important milestone towards the construction of provably secure single sign-on schemes. Our contribution is two-fold, providing a framework that handles both session state across multiple services and granular access control. The intrinsic centralized access control functionality adds no additional cost to the single sign on protocol while providing an easy way to manage access policies and user rights revocation. Moreover, our approach significantly improves communication complexity by eliminating any communication between services and identity providers during user identity and access permission verification. Relying on simple primitives, our methods can be easily and efficiently implemented using standard cryptography APIs and libraries. We base our constructions on standard cryptographic techniques and a threat model that captures the characteristics of current attacks and the requirements of modern applications. This is the first approach to base single sign-on security on public key cryptography and associate such a practical application to proxy signatures.

1 Introduction

The growing complexity of current corporate and internet based application ecosystems poses unprecedented access control challenges. Both the new cloud computing paradigm and Service Oriented Architecture (SOA) advocate that applications and systems be run as distributed interdependent services, which may be accessed remotely by users as they need them. Thus, many applications and systems that once ran locally have been transferred to remote distributed systems, each of them with different characteristics and access control methods. The proliferation of different remote applications and services makes it impractical for large corporations to manage each separate authentication system or for users to hold individual access credentials to each of them. This raises the need for centralized user identity management solutions that increase security while decreasing management complexity and costs.

1.1 Single Sign-on

Most of current application architectures require the user to memorize and utilize a different set of credentials (*e.g* username/password or tokens) for each application he/she wants to access. However, this approach is inefficient and insecure with the exponential growth in the number of applications and services a user has to access both inside corporative environments and at the Internet. Mainly, it is difficult for a corporation to manage potentially multiple authentication solutions and databases individually used by each application. Furthermore, most users tend to rely on the *same* set of credentials for accessing all of their systems, posing a serious security threat since an attacker who discovers these credentials can easily access all of the user's applications.

In a single sign-on platform, the user performs a single initial (or primary) sign-on to an *identity provider* trusted by the applications he wants to access. Later on, each time he wants to access an application, it automatically verifies that he is properly authenticated by the identity provider without requiring any direct user interaction. Single sign-on solutions eliminate the need for users to repeatedly prove their identities to different applications and hold different credentials for each applications. Furthermore, a well designed and implemented single sign-on solution significantly reduces authentication infrastructure and identity management complexity, consequently decreasing costs while increasing security.

1.2 The Problem: Secure Single Sign-On

Current corporate environments require employees to access multiple systems on a daily basis in order to perform their activities. Similarly, the number of personal web applications regularly accessed by millions of users grows each day (*e.g.* social networks, instant messaging, music streaming, webmail and digital content providers). Usually, users have to individually sign on to each system before being able to acces their data or perform any actions.

Each individual authentication process is commonly handled by a classical username and password credential mechanism, requiring the user to remember multiple credentials. In most cases, users tend to assign the same username/password credentials to all their user accounts in different systems. Thus, it is easy for an attacker to compromise all the other systems after obtaining credentials for only one of them. Alternatively, One Time Password (OTP) [8] methods are used, introducing an extra authentication factor which renders such attacks ineffective. However, such methods require the user to carry specific tokens or devices and contribute to increase sign on complexity.

The goal of a single sign on platform is to eliminate individual sign on procedures by centralizing user authentication and identity management at a central identity provider. Ideally, in a single sign-on solution, the user should seamlessly authenticated to his multiple user accounts (across different systems) once he proves his identity to the identity provider. Nevertheless, in many current solutions, the user is required to repeat sign on for each service using the same set of credentials, which are validated at the identity provider by each service.

For example, Google Accounts allows a user to sign on to different services provided by Google using the same username/password pair. Another infamous example is RSA SecurID [14], which a two factor authentication solution based on a OTP token and classical username/password credentials, allowing a user to sign on to several SecurID enabled services using the same token. However, a recent attack to EMC facilities exposed the overall fragility of this heuristic system. Even though their security was unaffected by current attacks, both solutions still require the user to repeatedly perform the sign on procedure.

In most of current transparent single sign-on architectures [6, 13], the user receives some kind of "authentication ticket" after he successfully signs on to the identity provider. When the user desires to sign on, he sends this ticket to the intended service provider or application, which then verifies it's validity by direct communication with the identity provider. This approach has several drawbacks, such as complex management and the requirement of secure online communication between applications and identity providers, which increases network traffic and processing loads.

1.3 Our Contributions

We propose a novel proxy signature based approach for single sign-on architectures that is both efficient and cryptographically sound. The proposed framework enjoys the following characteristics:

- Seamless and transparent session state maintenance and user identity verification after the initial sing-on.
- Granular access control and permissions enforcement without any additional cost.
- Easily manageable centralized access policies and user rights revocation.
- No communication required between service providers and identity providers.
- Efficient implementation based on Delegate-by-Certificate proxy signatures and standard cryptographic APIs.

The security of this framework can be reduced to the security of the underlying proxy signature scheme, assuming that the user's secret is kept secure. However, for the sake of brevity we present a detailed security analysis, leaving the formal security definitions and the complete proof to the full version of this work. To the best of our knowledge there is no other single sign-on framework based on public key primitives (specially proxy signatures) and that also provides granular centralized access control. Such nice results are achieved through a clever association between recent cryptographic results on proxy signatures and the problem of delegated identity verification.

This is the first public key cryptography based approach to single sign on platforms and the first solution to associate proxy signatures schemes to this practical application. Although this association may seem clear, it had not been pointed out in current literature yet. Our contributions pave the way for the design of provably secure and efficient practical single sign-on architectures, providing a flexible framework for building centralized authentication solutions.

1.4 Roadmap

In section 2, we analyse several existing single sign-on solutions, observing their strengths and weaknesses in different scenarios. In section 3, we introduce the threat model under which our scheme's security will be analysed. In section 4, we give a brief introduction to proxy signatures and describe the framework for single sign-on. Finally, in section 5, we summarize our results and conclude with directions for future research.

2 Related Works

Several approaches have been proposed in current literature to address the problem of single sign-on in different scenarios. In this section we analyse classical and promising methods for achieving secure single sign-on.

2.1 Network services single sign-on: Kerberos

Kerberos [15] was one of the first single sign-on solutions proposed in the literature and implemented as a network service. It is formally described as a network authentication system, initially designed for providing single sing-on to network services.

A Kerberos "realm" infrastructure is composed by an *Authentication Server*, a *Ticket Granting Server* and a set of service providers. The Authentication Server is responsible for verifying the user's identity while the Ticket Granting Server generates tickets for authenticated users. The service providers are simply networked servers that authenticated users are allowed to access. The two servers act together as an identity provider, handing the user an authentication ticket that he can use to sign-on to the relying service providers. In fact, the sign-on process in Kerberos is extremely complex, requiring several interactions between the user and the servers (which can be combined into an identity provider).

Although it provides a nice practical single sign-on solution, Kerberos infrastructure management is extremely complex, being prone to several mistakes that may severely compromise security. Both the identity provider (composed by the Kerberos servers) and all the service providers must be tightly time synchronized. This rules out the utilization of Kerberos as a single sign-on framework for distributed applications that may reside in the internet or the cloud. Furthermore, Kerberos relies solely on unproven symmetric encryption mechanisms to authenticate users and maintain session state. It may also be possible to impersonate users and steal authentication tickets through simple network based attacks.

2.2 Web applications single sign-on: OpenID

Among the many commercial single sign-on solutions, one of the most successful is OpenID [12], which provides a framework for deploying flexible centralized

user authentication for web based applications. In OpenID, the user can choose from a variety of identity providers, which may be any website or web based application where he already has an user account (*e.g.* Google). In order to sign-on to a given web based application that supports OpenID, the user first signs on to the identity provider of his choice and OpenID exchanges the necessary authentication data between the identity provider and the application.

However, it may be possible to compromise a given identity provider or the session state maintenance mechanism using simple social engineering techniques and client side or network based attacks. In order to transfer authentication information from the identity provider to relying applications, Open ID relies on a complex mechanism involving authentication information stored as cookies in the user's machine and background HTTP requests. This mechanism can be attacked through network based techniques (such as DNS poisoning) and methods based on client side website vulnerabilities (such as cross site scripting).

2.3 Secure login for network and web applications: Snap2Pass

The most promising approach for provably secure single sign-on in current literature is Snap2Pass, which was recently introduced in [7]. Snap2Pass allows users to sign-on to different web based or networked services using their mobile phones as credentials. In this framework, the users first shares a secret key with a service provider, storing this key in a mobile phone running a sign-on application.

Each time the user wishes to log in to the service provider, he issues an authentication request and receives a random challenge encoded as a QR-Code [9]. The user then launches the log in application and acquires the QR-Code with the mobile phone's camera. The application generates an HMAC [2] of the random challenge under the user's shared secret key and sends it back to the service provider through the internet (using 3G networks or Wi-Fi). The service provider accepts the user's sign-on if the HMAC is valid. The authors also propose a public-key based approach where a digital signature scheme is used instead of the HMAC.

Although this approach seems secure, the protocols proposed in [7] require the mobile phone to have direct online communication links to the identity provider, which increases costs and may severely affect system performance. Also, it may still be attacked by an adversary that controls the user computer or internet connection. A simple man-in-the-middle attack would consist in luring the user into log in to an arbitrary service by modifying the authentication challenge, which is not signed or verified through the HMAC. An attacker could set up a fake website or email message compelling the user to login to some trusted service (*e.g.* his internet banking website) but instead handling the user authentication challenges from other services.

2.4 Session state maintenance: Secure Cookie Protocol

A "Secure Cookie Protocol" for maintaining session state through authenticated cookies is presented in [10]. This protocol can be used for both maintaining

session state and controlling access to different areas inside the same web service. The authors propose to check the authenticity and integrity of authentication data cookies using the HMAC. In their solution, the cookies act as a secure token that the remote service providers and applications use to verify user's identity. This solution is also the first to propose the introduction of access control information in the authentication tickets (in this case, cookies).

However, no detailed security analysis is given and it may be possible to subvert this protocol to obtain unauthorized access to applications. It cannot be used as a mechanism for maintaining session state in a global single sign-on framework, since it is based on cookies and thus only suited for web applications. Moreover, managing the symmetric keys required for the HMAC algorithm is complex.

3 Threat Model

It is necessary to define a threat model in order to analyse the security of a given sign-on solution. Such a model must comprise the several threats and attacks to which the solution may be submitted. First of all, we consider an infrastructure model where the user accesses services and applications from his personal computer, communicating to the identity provider and respective service providers through an Internet connection.

A threat model that captures the security requirements and threats of current internet services and application is presented in [7]. We base our security analysis on a similar threat model that considers attackers with powers to completely control all the communications links between he user's machine, the identity provider and the application/service providers. The protocol introduced in this paper is designed to be secure against the following threats:

Phishing: An attacker may try to lure a user into disclosing his access credentials or accidentally performing unwanted sign-on operations [16]. The attacker may set up spoofed websites and email messages or use other social engineering techniques to compel the user into performing actions that he would not normally carry out.

Network attacks: The attacker has complete control over the user's internet link and overall network, disrupting communication or altering data as he wants. Such attacks can be carried out by adversaries who are naturally in a privileged "gateway" position in the network (which can be achieved, for example, by infecting firewalls). Furthermore, ARP spoofing techniques may be used to divert traffic from the user's machine through the adversary's machine and back to the original destination, effectively giving control the user's link.

The adversary is also given complete control over the communication links between the identity provider and the individual service providers.

This threat model captures most of the attacks directed against current single sign-on platforms, only excluding malware and trojan horses. However, it does not make sense to design a session state and access control protocol secure against malware or trojan horses, which can assume complete control of the user's machine. Notice that, even if this protocol was secure against attackers that completely control the user's machine, the attacker could simply wait for the user to legitimately sign-on and then perform arbitrary operations on the application. Thus, it is irrelevant to consider attacks that allow the adversary to obtain unrestricted control over the user's machine.

Considering this model, for the sake of security analysis, we can assume the user's secret key (and any other sensitive information) to be securely stored in his machine. Note that it is also possible to keep user's secret keys in an external device that is able to exchange information with his personal computer, such as a mobile phone or a token. In this case it is possible to rely on the assumption that the user's mobile phone is not controled by malware, which is also relied upon in [7].

4 The Framework

In this section we introduce our framework for maintaining session state across the various application servers that trust the identity provider. While most single sign-on solutions keep session state by means of application servers that directly communicate to the identity provider, we introduce a novel approach where the user himself proves to each application server he wants to access that he is already authenticated by the identity provider. Moreover, our approach allows for centralized application access control by the identity provider, *i.e.* the service providers are able to verify whether the user possesses the right access permissions.

The main ingredient in our protocol is a Proxy Signature scheme, a primitive first introduced in [11]. Proxy signatures allow the owner of a private key to delegate to a third party the right to sign *specific* messages on its behalf, allowing anyone to verify whether the signature is valid. Even though they enjoy these interesting properties, proxy signatures can be realized from standard digital signatures, allowing easy and efficient implementation from current cryptographic APIs and libraries.

In this framework, each user, application server (service provider) and the identity provider are considered to have key pairs whose public keys are known to each other. The identity provider also holds an access control list (ACL) containing a description of the applications that each user is allowed to access. Considering the threat model presented in the previous session, the user's secret key is stored in his personal computer, which also runs an application responsible for computing the necessary cryptographic operations.

Basically, after a successful sign-on, the identity provider sends the user an *identity ticket*, delegating to the user the right to sign messages containing only his application access permissions (*e.g.* the names of the applications he is al-

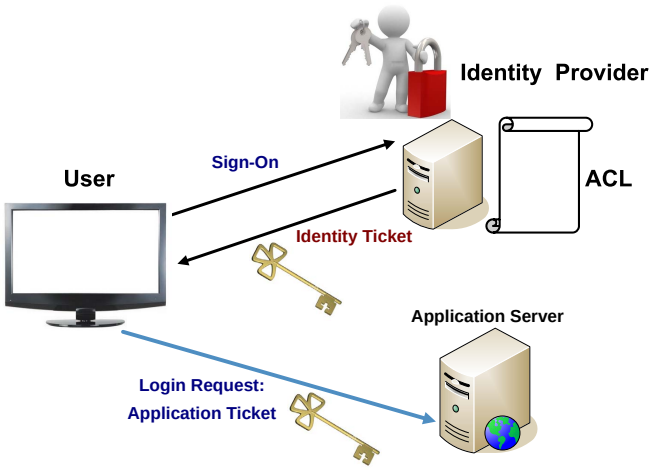

Fig. 1. Interactions between User, Identity Provider and Application Server

lowed to access). Therefore, when the user wants to access some application, he simply generates an *application ticket*, which is a message containing the appropriate access credentials on behalf of the identity provider, and sends it to the application server, who verifies whether the signature is valid. The framework is depicted in Fig. 1. Notice that no communication is takes place between the Identity Provider and the Application Server.

Different sign-on methods can be used for the initial sign-on to the identity provider, but it would be nice to couple this framework with a sign-on method that relies on the same credentials (*i.e.* key pair). For this purpose we point out that cryptographic identification protocols [3]. Such protocols enable one party to prove the possession of a secret key to another party without revealing it (*i.e.* a zero-knowledge proof of key possession). Hence, by proving the he possesses a the secret key corresponding to a pre-exchanged public key, a user can sign-on to the identity provider. Such a scenario is illustrated in Fig. 2.

We assume that the required cryptographic operations are carried out by an *Identity Broker* application running on the user's machine, since it would be unnatural to require the user to perform the signature generation and verification steps. The identity broker has access to the user's private key, which is securely stored in the user's machine (considering the threat model of section 3, refer to that section for further discussion). Such an identity broker could be easily implemented as a web browser plug-in or as a local proxy that captures access attempts to applications and automatically performs the necessary operations.

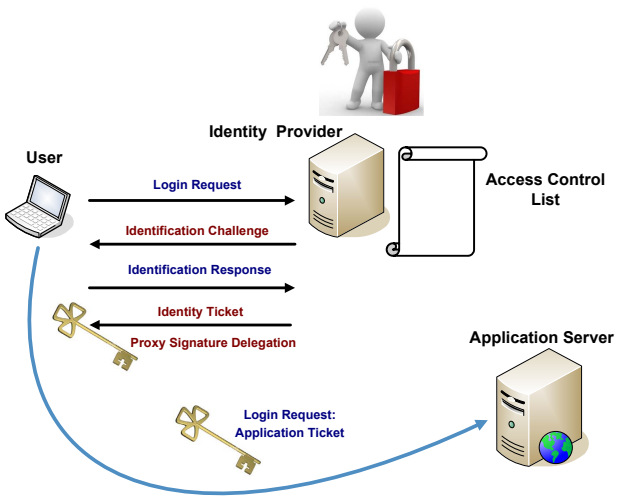

Fig. 2. Interactions between User, Identity Provider and Application Server considering and identification based sign-on

4.1 Proxy Signatures

In a proxy signature scheme, a party called the designator or original signer delegates to a party called the proxy signer the right to sign messages inside an specific message space. The basic (informal) security properties of proxy signatures stated in [1] are the following:

Strong Unforgeability: The original signer and third parties who are not designated as proxy signers cannot create a valid proxy signature.

Verifiability: From proxy signature a verifier can be convinced of the original signer's agreement on the signed message either by a self-authenticating form or by an interactive form. (The proxy signer can only sign messages inside a message space specified by the designator)

Strong Identifiability: Anyone can determine the identity of the corresponding proxy signer from a proxy signature.

Strong Undeniability: Once a proxy signer creates a valid proxy signature for an original signer, the proxy signer cannot repudiate his signature creation against anyone.

Although proxy signatures have been extensively studied, the first formal security definitions that capture the above properties and provably secure con-

structions were only introduced in [4]. Apart from the basic algorithm in digital signature schemes ($KeyGen(1^n)$, $Sign(sk, m)$ and $Verify(pk, m, sig)$), proxy signature schemes comprise four more components:

Delegation Algorithms: $\mathtt{D}(pk_i, sk_i, j, pk_j, \omega)$ is run by the designator, who inputs his key pair (pk_i, sk_i) along with a proxy signer ID i and public key pk_i. It also inputs a descriptor of the delegated message space. $\mathtt{P}(pk_j, sk_j, pk_i)$ is run by the proxy signer in order to obtain the proxy signing key skp and takes as input the proxy signer's key pair and the designator's public key.

Proxy signing algorithm: $\mathtt{PS}(skp, m)$ takes as input a proxy signing key and a message $m \in \omega$, outputting a proxy signature $psig$.

Proxy verification algorithm: $\mathtt{PV}(pk, psig, m)$ takes as input a public key pk, a proxy signature $psig$ and a message m, outputting 1 if the signature is valid for m and pk. Otherwise, it outputs 0.

Proxy identification algorithm: $\mathtt{ID}(psig)$ takes as input a proxy signature and outputs the identity of the proxy signer.

The security definitions for proxy signatures are formalized in [4], and it is shown that proxy signature schemes can be obtained from any digital signature schemes. Moreover, it is shown that aggregate signature schemes [5] can be used in such constructions in order to obtain shorter signatures. One should notice that it is possible to add a timespan parameter to the proxy delegation algorithm, allowing the designator to specify a timespan during which the proxy signer may use the key. This new functionality can be added by a simple modification of the Delegate-by-certificate and Aggregate Signature Based proxy signature schemes in [4], requiring only minor (trivial) modifications in the security proofs. The altered delegation algorithm is denoted by $\mathtt{D}(pk_i, sk_i, j, pk_j, \omega, t)$, where t is the timespan information.

4.2 The session state and access control framework

In this public key based framework, we assume that the identity provider, the user and the application servers share their public keys. However, we do not require any Certificate Authority functionalities or a full blown Public-Key Infrastructure since application and user public keys are centrally stored and controlled by the identity provider, which is already trusted by the applications and users. Also, in actual networked systems, public keys can be exchanged during user account registration. Our framework has four main components:

ACL: An access control list that is stored at the identity provider and contains unique IDs of the applications a user is allowed to access. It is denoted by $ACL_{uid} := \{AppID_0, \cdots, AppID_n\}$ for each user identified by uid.

Identity ticket: A proxy signature key generated by the identity provider. It delegates signing rights of a message space representing the ACL_{uid} during a session timespan t to a user uid. It is represented as $IDt_{uid} := \mathtt{D}(pk_i, sk_i, j, pk_{uid}, ACL_{uid}, t)$, where (pk_i, sk_i) is the identity provider's key pair.

Application ticket: A message containing a given application's unique ID and a proxy signature of this message generated by a user who has an identity ticket. It is denoted by $APPt := \langle \mathtt{PS}(IDt_{uid}, AppID_i), AppID_i \rangle$, where $AppID_i \in ACL_{uid}$.

Identity broker: An application that runs on the user's computer and manages identity tickets and application ticket requests.

After the user successfully signs on to the identity provider, it receives an identity ticket. The application sign-on procedure is as follows:

1. The identity broker in the user's computer stores the identity ticket and waits for application sign-on requests.
2. Upon receiving an application sign-on request from the user, the identity broker generates an application ticket and sends it to the application server.
3. The application server verifies that the application ticket is valid by running $\mathtt{PV}(pk_i, APPt)$. If the ticket is valid, it confirms that the user is already authenticated by the identity provider and that he has the proper access rights. Otherwise, it aborts.
4. The application server establishes a session with the user, relying on standard cryptographic methods for ensuring secure communications (*e.g.* establishing a secure channel or signing messages).

Once again notice that an application sign-on requests can be easily captured by watching local network traffic, enabling the identity broker to be implemented as a web browser plug-in or a local proxy.

4.3 Security Analysis

The security of this protocol can be proved based on standard assumptions under the threat model proposed. The basic technique is to show that any adversary who is able to sign-on to an application server without a valid identity ticket is also able to break the underlying proxy signature scheme, considering that legitimate user secret keys are safely stored in user's machines. However, for the sake of brevity and simplicity we present a concise but detailed security analysis. The full security proof will be presented in the full version of this paper.

First, remember that in the threat model for this protocol, the adversary is assumed not to control the user's machine. Notice that only the user can generate proxy signatures using his identity ticket, since it is necessary to use his secret key (which is safely stored in his computer) for proxy signing. Hence, even if an attacker sniffs the network and captures the identity ticket, it remains

useless. The scheme is also secure against replay attacks, since current signature schemes [5] output fairly random signatures that also work as nonces in our scheme, allowing applications to detect whether they have received the same application ticket more than once.

Now we analyze this approach according to the proxy signature's security properties. By the Strong Unforgeability property, an attacker cannot forge an application ticket on behalf of the identity provider in order to obtain access to applications. By the Verifiability property, the application server can verify that the user has the necessary access rights (*i.e.* he can really sign the message containing the application's ID on behalf of the identity provider). The Strong Identifiability property allows the application to obtain the user's identity (possibly using it for further fine grained access control). The Strong Undeniability property adds a nice benefit for digital forensic analysis, since it is possible to determine that a user has really accessed an application.

5 Conclusion

We presented a new approach for practical efficient and secure single sign-on frameworks based on proxy signature schemes. The proposed framework provides seamless and transparent single sign-on without undermining overall network security and without requiring *any* online communications between service providers and the identity provider. Additionally, it allows for fine grained access control without any increase in the protocol's computational or communication complexity. Moreover, it offers simple access policy and user revocation management while providing nice forensic and audit data by building on common proxy signature strong unforgeability and undeniability properties. Our framework is also the first to apply public key cryptography techniques to the problem of practical single sign-on. We remark that our approach is the first to associate proxy signature schemes to this practical problem, even though the association seems. These results represent an important step towards the formalization of single sign-on and user authentication protocols, and the construction of provably secure schemes for these practical applications.

As a future work, the integration of our approach with other cryptographic protocols remains to be studied. Another promising research direction is obtaining efficient and secure proxy signature key revocation protocols in order to implement better single sign-off mechanisms. It is still an open problem to obtain a session state and access control protocol that remains secure if the adversary is given control of the user's computer. A promising direction for solving this issue is to adapt our protocol to rely on an external tamper-proof token or mobile device to run the identity broker.

References

1. B. Lee, H.K., Kim, K.: Strong proxy signature and its applications. In: Proc. of the 2001 Symposium on Cryptography and Information Security (SCIS'01). vol. 2, pp. 603–608 (2001)

2. Bellare, M., Canetti, R., Krawczyk, H.: Keying hash functions for message authentication. In: Proceedings of the 16th Annual International Cryptology Conference on Advances in Cryptology. pp. 1–15. CRYPTO '96, Springer-Verlag, London, UK (1996), http://portal.acm.org/citation.cfm?id=646761.706031
3. Bellare, M., Fischlin, M., Goldwasser, S., Micali, S.: Identification protocols secure against reset attacks. In: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques: Advances in Cryptology. pp. 495–511. EUROCRYPT '01, Springer-Verlag, London, UK (2001), http://portal.acm.org/citation.cfm?id=647086.715697
4. Boldyreva, A., Palacio, A., Warinschi, B.: Secure proxy signature schemes for delegation of signing rights. Journal of Cryptology pp. 1–59 (2010), http://dx.doi.org/10.1007/s00145-010-9082-x, 10.1007/s00145-010-9082-x
5. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Proceedings of the 22nd international conference on Theory and applications of cryptographic techniques. pp. 416–432. EUROCRYPT'03, Springer-Verlag, Berlin, Heidelberg (2003), http://portal.acm.org/citation.cfm?id=1766171.1766207
6. De Clercq, J.: Single sign-on architectures. In: Davida, G., Frankel, Y., Rees, O. (eds.) Infrastructure Security, Lecture Notes in Computer Science, vol. 2437, pp. 40–58. Springer Berlin / Heidelberg (2002)
7. Dodson, B., Sengupta, D., Boneh, D., S., L.M.: Secure, consumer-friendly web authentication and payments with a phone. In: Proceedings of the Second International ICST Conference on Mobile Computing, Applications, and Services (MobiCASE) (2010)
8. Haller, N., Metz, C., Nesser, P., Straw, M.: A One-Time Password System. No. 2289 in Request for Comments, Internet Engineering Task Force, IETF (Feb 1998), http://www.ietf.org/rfc/rfc2289.txt
9. ISO 18004:2005: Information technology – Automatic identification and data capture techniques – QR Code 2005 bar code symbology specification Automatic. ISO, Geneva, Switzerland
10. Liu, A., Kovacs, J., Huang, C.T., Gouda, M.: A secure cookie protocol. In: Computer Communications and Networks, 2005. ICCCN 2005. Proceedings. 14th International Conference on. pp. 333 – 338 (oct 2005)
11. Mambo, M., Usuda, K., Okamoto, E.: Proxy signatures for delegating signing operation. In: Proceedings of the 3rd ACM conference on Computer and communications security. pp. 48–57. CCS '96, ACM, New York, NY, USA (1996), http://doi.acm.org/10.1145/238168.238185
12. OpenID: www.openid.net
13. Pashalidis, A., Mitchell, C.: A taxonomy of single sign-on systems. In: Safavi-Naini, R., Seberry, J. (eds.) Information Security and Privacy, Lecture Notes in Computer Science, vol. 2727, pp. 219–219. Springer Berlin / Heidelberg (2003)
14. SecurID, R.: http://www.rsa.com/node.aspx?id=1156
15. Steiner, J.G., Neuman, C., Schiller, J.I.: Kerberos: An authentication service for open network systems. In: in Usenix Conference Proceedings. pp. 191–202 (1988)
16. Yu, W.D., Nargundkar, S., Tiruthani, N.: A phishing vulnerability analysis of web based systems. In: IEEE Symposium on Computers and Communications (ISCC 2008). pp. 326–331. IEEE Computer Society, Marrakech (Jul 2008), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4625681