Implementing Strong Authentication Interoperability with Legacy Systems

Jan Zibuschka and Heiko Roßnagel

Johann Wolfgang Goethe University Frankfurt,
Chair for Mobile Business and Multilateral Security, Gräfstraße 78,
60054 Frankfurt am Main, Germany,
zibuschka@m-lehrstuhl.de, mail@heiko-rossnagel.de

Abstract. In a WWW environment, users need to come up with passwords for a lot of different services, e.g. in the area of e-commerce. These authentication secrets need to be unrelated if the user does not want to make himself vulnerable to insider attacks. This leads to a large number of passwords that a user has to generate, memorize, and remember. This password management is quite straining for users. Single sign on systems provide a solution for this dilemma. However, existing solutions often require the implementation of specific interfaces by the individual service providers, and usually do not support existing strong authentication factors, e.g. smart cards, without protocol extensions or modification of implementations. In this paper we propose a different approach that generates strong passwords using electronic signatures. Our approach builds on existing smart card infrastructures to achieve strong authentication, while at the same time it provides an interface to legacy password authentication systems.

1 Introduction

In a modern web environment, users need to come up with passwords for a lot of different services. Examples are web based mail, e-commerce sites and discussion forums. Passwords are also widely used for authentication in email, operating system login, remote shells, databases and instant messaging. This leads to a large number of passwords that a user has to generate, memorize, and remember. However, remembering a lot of randomly selected, independent passwords is quite straining for users, especially if some passwords are used only occasionally. Users tend to either choose weak passwords [4], or choose related passwords for several or even all accounts [2], which makes the authentication system vulnerable to cross-service attacks [11].

Furthermore, forgotten passwords are a major problem and an economic factor. A recent study estimates that help desk staff has to reset user passwords manually in 82% of cases [14]. This procedure often takes more than 5 minutes. As forgotten passwords are a common usability problem, this may result in high help desk costs. Additionally, the distraction and the time spent on resetting the password will reduce the productivity of users [14].

There are several authentication systems that are generally expected to offer a stronger security than passwords, such as public key cryptography in conjuncture with tokens. However, market penetration and usage of such systems has not lived up to expectations, and they often require the implementation of specific interfaces by the individual service providers.

We present a solution that integrates one such infrastructure, namely signature-capable smart cards, with password-based authentication mechanisms, offering single sign on functionality to the user, without requiring the implementation of specific interfaces by service providers. Implementing such protocols obviously consumes some resources. Also, replacing passwords as an authentication mechanism may add complexity, and as of such entrance barriers, thus losing the service provider users. Additionally, it may not be in service provider's best interest to standardize in the area of authentication due to the underlying network effects, based on phenomena such as lock-in and network externalities [21]. Still, compatibility is a major issue with regard to the overall value of the network. As [6] points out, "links in networks are potentially complementary but it is compatibility that makes complementary actual".

This paper is structured as follows: In section 2 we analyze requirements such a system has to meet. Based on that knowledge, we present our implementation of our system in section 3. We then discuss the advantages and disadvantages of our approach in section 4 before we conclude our findings in section 5.

2 Requirements

Apart from storing passwords in an encrypted form, it is also possible to generate them on the fly, using strong cryptography. However, such methods have to meet several requirements to guarantee their usefulness for user and service provider.
Our system's main concern is building a secure, interoperable authentication infrastructure on legacy systems (smart card infrastructures and password authentication mechanisms), rather than e.g. aiding anonymous service usage [8]. Several requirements can be derived from this scenario, and will be listed in this section.

- **Consistency:** For each web site, each user should be transparently provided with a consistent password.
- **Security of Generated Passwords:** The service passwords must be pseudorandom and independent. Additionally, the system must not leak information about the central authentication secret (in this case, the private signature key). Furthermore, a service password for any site must not give any information on any other service password. As a corollary, generated passwords should be chosen from a suitably large set, and should be ideally equidistributed, to avoid efficient (e.g. dictionary) attacks focusing on a specific, more probable subset.
- **Single Secret:** Given a single secret, e.g. a secret signature key or master password, the system should generate secure, distinct passwords for each web site.

The central secret should be protected using the strongest available measures, as it is also a single point of failure [14]. Preferably, the usage of a single secret should not be enforced, but it should rather be an option for the user to employ several secrets, if he deems it necessary.

- **Compliance with Password Policies:** Each generated password needs to be accepted by the web site, e.g. it must comply with the service's password policy.
- **Interoperability:** The architecture should be able to build on existing smart card or public key infrastructures (PKIs). To make the system easily deployable on top of e.g. an existing signature card infrastructure, we would prefer to use only algorithms that are present on all smart cards that can produce digital signatures. Additionally, service passwords should not be stored on the smart card, as publicly accessible memory is not present on all cards, and generally is less wide-spread than signature-related smart card applications.
- **Pervasiveness:** We aim for inter-device mobility, making storage of authentication information on the device impractical. Additionally, the implemented authentication mechanism should be performable on mobile devices, in spite of their limited performance.
- **Minimal Insider Attacks:** Unlike protocols employing an authentication proxy, we aim to realize a protocol that cannot be executed by a third party alone, to thwart insider attacks. The same holds true for the individual service providers, who in the classic password scenario could leverage the tendency of users to reuse passwords at several services [14] for cross-service attacks [3].
- **Usability:** The system should require minimal user interaction, as each necessary interaction step significantly reduces the acceptance of security-related systems. It has been stated that "The user base for strong cryptography declines by half with every additional keystroke or mouse click required to make it work." ("Ellison's Law") [3].
- **Minimal Provider Costs:** We aim to minimize necessary server infrastructure, while still meeting the interoperability and usability requirements.

3 Implementation

We implemented a prototype of our proposed password generator in Java. This allows for easy porting of the core components and easy deployment on many platforms, including mobile devices and various host applications, for example web browsers. The implementation uses signatures for the generation of account specific passwords. For signature creation we used a SIM card that was developed during the WiTness project sponsored by the European Union. This was chosen as an arbitrary existing SIM card infrastructure, demonstrating the adaptability of our system. It offers strong cryptographic algorithms, namely it is capable of creating RSA signatures [15] and also provides 3DES encryption. Other features of the SIM, like the encryption capability, might also have been used. However, the focus of deployed cards seems to be on digital signature capabilities, so these were also used for the password creation process.

3.1 Overview

We implemented a pluggable architecture, designed to make adding new smart card APIs or other authentication factors easy. Also, password encoder and transmission components have been designed to be easy to replace (see Figure 1). An additional benefit of this modular approach is a small, portable core containing the key algorithms.

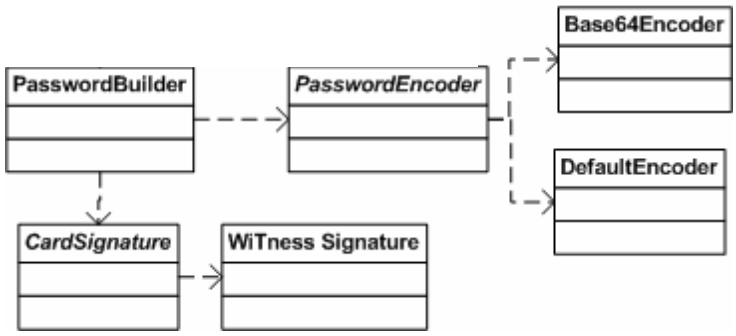

Fig. 1. Schematic Class Diagram

The basic data flow can be summarized in four steps [17]:

1. Define a scheme for deriving service identifiers for the different service providers the user might want to authenticate to. This can be implemented by concatenating several attributes of the service, such as a service name, URL, user's login name, IP address, and so on.
2. Combine the identifier for the service with the user's master password using strong cryptography [1] [8].
3. Transform the resulting value into a pseudorandom account password. This step is described in more detail in section 3.2.
4. Transfer the password to the appropriate service login form. This may be realized by an application that integrates with current browsers, maybe as a plug-in [9] [16]. Other implementations that generate passwords for additional services, such as database access or remote login are also possible.

Several cryptographic primitives, such as hash functions [1], signatures or a combination of algorithms [8], are suitable for step 2. As already mentioned, this paper focuses on signatures, for deployment reasons. Also, unkeyed hash functions have no secret that could be stored on the smart card.

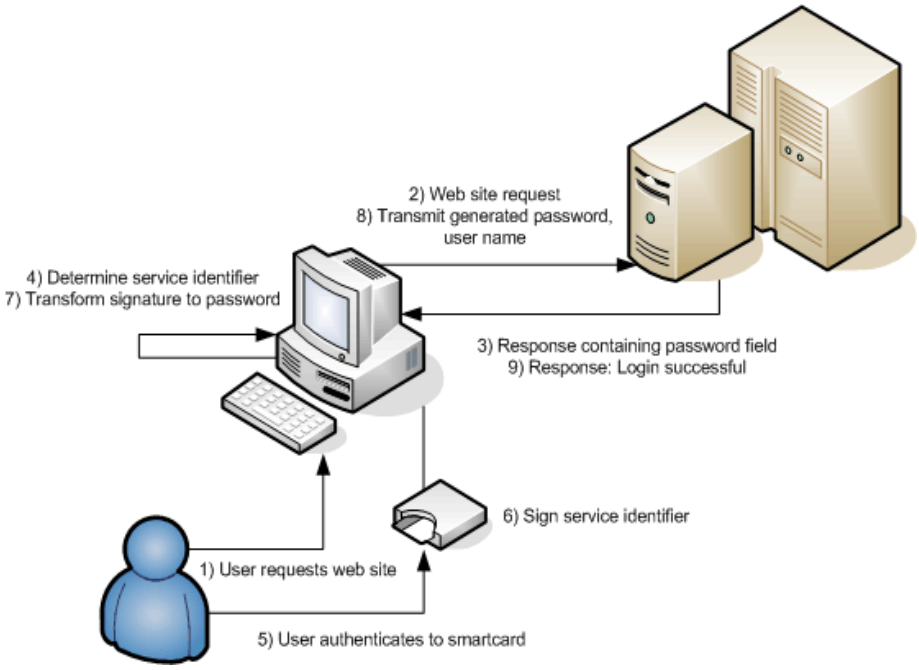

Fig. 2. Password Generation Using Smart Cards [17]

Like hash functions, electronic signatures can be used to generate strong service passwords for the user. Unlike hash functions, digital signatures have the security property of unforgeability, meaning that an attacker can't produce the user's signature for any text if he does not have the secret key, even if he is given the user's public key and several examples of signed messages. This would also translate to passwords. An attacker cannot compute any of the user's service passwords without knowing the secret key stored on the smartcard, even if he knows the user's passwords for several other accounts. The whole process is illustrated in Figure 2.

When the user needs to authenticate to an e-commerce site (1-3), the local system first derives the service identifier from the available context information, such as the accessed service's domain name and IP address (4). The user authenticates to the smart card using his PIN, thus unlocking the private signature key (5). The service identifier for the relevant account is then signed by the signature card using the private key, producing an electronic signature (6). The resulting value is encoded as a password (7). This is a critical step. While unforgeability is guaranteed due to the fact that signatures are used, the distribution and set size of generated passwords are also a factor for security of the system - it needs to output passwords chosen from a suitably large set, and may not employ an overly skewed selection algorithm. The transcoded signature is transmitted to the service provider requiring authentication, along with the user's login name (8). Access to the protected resources is then granted (9).

One advantage of this approach is that the central secret – the user's private key – is actually stored on the smart card and not directly dependent upon a user chosen password. Guessing the PIN will only allow access to this key if the attacker is also in possession of the token.

It has to be noted that the consistency requirement can only be achieved using deterministic signatures. This limits the theoretical strength of the system; however, it is an obvious requirement when interfacing with password authentication mechanisms. Additionally, the bulk of deployed infrastructures use deterministic algorithms like RSA or DSA.

The usage of passwords derived from signatures links the user's identity to his intent to use the service. The signature, encoded as a password, may be verified by the service provider using the user's public key. To realize this, the service provider first decodes the password to the original signature, and then follows the usual verification procedure.

Of course, signatures in this scenario are not linked to individual transactions. This is due to the fact that the widely deployed password systems do not perform user authentication on a transaction level.

3.2 Transcoding Signatures to Passwords

This can be done using e.g. a simple Base64 encoding, although more complicated schemes may be used to ensure the compliance of passwords with service policies [16]. In our solution, more complex transcoding algorithms are implemented to make up for restrictions of the used signature algorithms (e.g. signature distributions dependent on public keys, such as the RSA modulus) and requirements of service authentication mechanisms (e.g. password policies); to optimize compliance with password policies, output domain and statistic distribution of generated passwords.
As service passwords are generated pseudorandomly, the widely enforced password policies form an obstacle to this approach. They are usually not available in a format that would be machine readable by e.g. a SSO browser extension. The user may manually edit the produced passwords to suit the policy; however, this is error-prone and awkward.

Policy parameters might be input by the user for each service. However, to facilitate this approach, the number of parameters should be quite small, preferably with sensible defaults. Password length, distinction of upper-case and presence of numbers or special characters seem like a sensible set of parameters. To minimize user interaction, we will assume that policy information is supplied by a server, or a default policy suiting most services is employed. For a detailed overview of a policy input UI, see [7]. A complete overview of the set of parameters used in our system is given in Table 1.

While this set is not suitable to precisely describe all password policies in use, it is able to model an acceptable subset for each policy to the best knowledge of the authors, and a similar approach was successfully employed for [16]. In [9], another approach is proposed, but it cannot guarantee presence of certain character classes, and uses a significantly larger parameter set.

Table 1. Parameters for Password Policy Configuration

Password Policy Parameter	Legal Values
relevantUpperCase	true, false
hasNumberCharacter	true, false
hasSpecialCharacter	true, false
passwordLength	positive integer {…6, 7, 8, 9….}
specialCharactersUsed	String (set of special characters in use)

Note that the character sets marked as true are not only possible, but required for all generated service passwords. Having separate parameters for this is possible, but would lead to additional clutter of the user interface for inputting policies. Passwords generated in this way will still be stronger than user passwords, because of the length and pseudorandomness of generated service passwords.

To realize random password generation based on those parameters, the simple armoring step (using e.g. Base64) for transformation in the encoding step 3 will have to be replaced by a more sophisticated encoding scheme. We propose a 2 step process.

In a first step, the generated signature material is adjusted to be equidistributed in a range needed by the next stage, e.g. it has to be transformed from pseudorandomly chosen over the signature output domain, defined by e.g. the RSA modulus in the case of RSA, to pseudorandomly chosen from a set of legal passwords, or several pseudorandomly chosen characters. For this, a stream base conversion method similar to the one described in [19] is employed.

In the second step, the renormalized pseudorandom material output by the first step is then transformed to part of a password (e.g. a character), parameterized with the service provider's password policy. One representative for each used character subset is generated, and then the remainder of the password is filled with random characters from the entire used character domain. In pseudo-code, using parameter names from Table 1:

```
password = ''
password.append(random_lower_case_character())
if (relevantUpperCase)
  password.append(random_upper_case_character())
if (hasNumberCharacter)
  password.append(random_number_character())
if (hasSpecialCharacter)
  password.append(
    random_special_character(specialCharactersUsed)
  )
while (password.length<passwordlength)
  password.append(random_legal_character())
do_random_permutation(password)
```

The last operation, do_random_permutation, signifies a permutation of the password's characters. While this will widen the set of possible outputs (e.g. not all output passwords start with a lower case letter), it somewhat skews the equidistribution. While the algorithm by does not offer a perfect distribution of service passwords, and is also restricted to a subset of service passwords allowed

under certain password policies. Still, it offers a notable improvement over typical user-chosen passwords, and reaches password length and randomness that is hard to match even by careful users. Equidistribution of created passwords can be reached by omitting the permutation step. However, this will further reduce the output password subset.

3.3 User Interface

The prototype user interface is quite straightforward, as shown in Figure 3. There are input fields for a service identifier (which may be e.g. the domain name of the service in the web surfing case) and the login the user chose for the service. Additional components, e.g. the service IP address, may also be used. The input may be provided by the user, or the fields may be filled automatically using data from the application used to contact the service (for example, by embedding it in a web browser plug-in [16]). In this case, the only things the user has to provide are the signature card and the PIN necessary to authenticate to it.

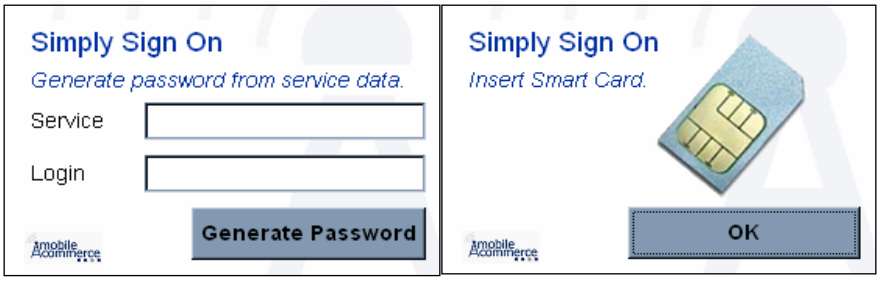

Fig. 3. Application screenshots

After the card is activated, an identification string for the relevant account is created by concatenating service identifier and user login. This string representation of the account is then signed using the existing signature mechanism. The account identifier is digitally signed in the usual fashion. The signature value is then transformed into a password string, as described in section 3.2.

The resulting password is copied to the system clipboard for transfer to an arbitrary host application.

We presume that a simple user interface such as described here will be best suited for meeting the simplicity requirement raised in section 2. However, enabling transparent (or, at least, minimally invasive) support for changing password policies on the server side will require additional effort. In [12], manually adjusting the generated password to meet the needs of the service is proposed. However, this will be straining for the user. For the time being, we just assume that a system similar to the service meta-information server described in [7] will be in place, and the formatting information be retrieved from there (or from a local cache, although that

solution will hinder cross-system availability and thus pervasive applicability of the authentication solution).

4 Discussion

In contrast to conventional smart card solutions that store encrypted passwords on the token, our system can be deployed on top of already existing signature card infrastructure, thus limiting costs for the user and also the amount of authentication tokens the user has to manage.

Recently, governments across Europe have issued [5] [10] [20] or plan to issue signature capable identity cards as an e-government measure. For our prototype implementation we used a signature capable SIM card, demonstrating the viability of the algorithm in a mobile scenario. However, identity cards could also be used for the signature generation. Therefore, our solution could be implemented on top of e-government infrastructures already deployed in some European countries, such as the ones mentioned above. Furthermore, by using a SIM card for signature creation our solution could be implemented on a mobile phone, leveraging infrastructure provided by mobile operators [18]. It may be based either on already deployed SIM cards (if the algorithms present on the card allow for appropriate security guarantees), or rolled out together with signature-enabled SIMs to add a compelling use case.

The generated service passwords are directly dependent upon the user's cryptographic signature keys. If the user's key pair needs to be replaced, because e.g. it has been revoked, all the generated service passwords will change. While this poses a serious usability barrier in the described basic version of the system, saving account meta-information on a server can improve user experience during this and other use cases. The system is able to iterate over all the accounts, using the same architecture as the master password changing assistant in [7]. Note that, while the revoked key pair's signatures can no longer be verified, they may of course still be encoded and submitted as passwords.

Of course, in a SSO system, loss of the central secret – the secret key on the smart card token - means loss of all derived passwords. This paper does not discuss mechanisms for ensuring the robustness with regard to lost passwords in detail. However, most services offer a means to reset or retrieve a lost password. Additionally, conventional methods, like saving encrypted password lists to a secure storage as a backup, may be used.

As is pointed out in [1] [8], unlinkable user pseudonyms may also be generated in a similar fashion, which would be especially useful when combined with anonymous communication channels, based on e.g. TOR [8].

The card is portable but it is – in many cases, for example where signature cards are deployed as part of e-government initiatives - not obvious that it is used as a SSO token, so the security risks of portability are partially mitigated.

Also, the portability of the smart card token, along with the pervasiveness offered by the algorithm's operability without saved passwords, suggest implementing the system on mobile terminals. There are functions on standard-issue GSM SIMs that

may take up the role of the signature algorithm presented in this paper. However, real-life implementations of these functions are dependent on the individual mobile operator. Also, keys are often shared between subscriber and mobile operator. So, while the system may be easily implementable if a signature-capable card is implemented in a mobile terminal, employing standard-issue SIMs for a similar functionality will require additional investigation, and - at least in some cases - additional authentication secrets.

Using the SSO system does not require trust towards third parties, as opposed to systems based on an authentication proxy or similar architecture. The authentication secret is only handled by user and service, with the central authentication secret remaining on the user side – more specifically, on the token - at all times. The system offers an alternative to hash functions for the purpose of generating passwords on the fly. In addition to the capabilities of hash function based systems, the presented implementation makes use of the strength of smart card based two factor authentication. It also meets the technical requirements outlined in section 2.1, offering a mobile and interoperable dynamic authentication infrastructure built on legacy systems.

As the user can employ the solution for password management, we estimate that the perceived usefulness should be quite high. This in turn might ameliorate the acceptance of electronic signatures [13] [18], leading to a wider usage of signature cards and readers and to a more secure, multi-factor authentication infrastructure.

5 Conclusion

In this paper, a possible solution for the password management challenge users face today was presented, using electronic signatures for password generation. An overview of the architectural components of such a system was provided, and security requirements and guarantees were investigated. A prototype implementation that uses a signature capable SIM card was presented. The underlying architecture may also be used with other signature cards, like electronic id-cards, that are being rolled out in several member states of the European Union. Integrating electronic signatures with password legacy systems would increase the number of transactions where signatures are used. Therefore, the technology acceptance of electronic signatures might also be increased.

References

1. Abadi, M., Bharat, K. and Marais, J. (1997) System and method for generating unique passwords, United States Patent 6141760.
2. Adams, A., Sasse, M. A. and Lunt, P. (1997) Making Passwords Secure and Usable, Proceedings of HCI on People and Computers XII, Bristol, UK, Springer, 1-19.

3. Ball, P. (2001) Hacktivism and Human Rights: Using Technology to Raise the Bar, Panel Discussion, DEF CON 9, Las Vegas, USA.
4. Brown, B. J. and Callis, K.(2004) Computer Password Choice and Personality Traits Among College Students, Southeast Missouri State University, Cape Girardeau, Missouri, USA.
5. De Cock, D., Wouters, K. und Preneel, B. (2004) Introduction to the Belgian EID Card, S. K. Katsikas, S. Gritzalis und J. Lopez (Eds.), Public Key Infrastructures, Berlin Heidelberg, Springer, 1-13.
6. Economides, N. (1996) The Economics of networks, International Journal of Industrial Organization, 14, 673-699.
7. Fraunhofer SIT (2006) Der PasswordSitter, White Paper.
8. Gabber, E., Gibbons, P., Matias, Y. and Mayer, A. (1997) How to Make Personalized Web Browsing Simple, Secure and Anonymous, Proceedings of the First International Conference on Financial Cryptography, Anguilla, British West Indies, Springer, 17-32.
9. Halderman, J. A., Waters, B. and Felten, E. W. (2005) A convenient method for securely managing passwords, WWW '05: Proceedings of the 14th international conference on World Wide Web, Chiba, Japan, ACM Press, 471-479.
10. Hvarre, J. (2004) Electronic signatures in Denmark: free for all citizens, e-Signature Law Journal, 1, 1, 12-17.
11. Ives, B., Walsh, K. and Schneider, H. (2004) The Domino Effect of Password Reuse, Communications of the ACM, 4, 47, 75-78.
12. Karp, A.H. (2003) Site-Specific Passwords, Technical Report, HP Laboratories Palo Alto.
13. Lopez, J., Opplinger, R. and Pernul, G. (2005) Why Have Public Key Infrastructures Failed So Far? Internet Research, 15, 5, 544 - 556.
14. RSA Security (2005) RSA Security Survey Reveals Multiple Passwords Creating Security Risks and End User Frustration: Press Release, http://www.rsasecurity.com/press_release.asp?doc_id=6095, September 27.
15. Rivest, R. L., Shamir, A. and Adleman, L. (1978) A Method for Obtaining Digital Signatures and Public Key Cryptosystems, Communications of the ACM, 21, 2, 120-126.
16. Ross, B., Jackson, C., Miyake, N., Boneh, D. and Mitchell, J. C. (2005) Stronger Password Authentication Using Browser Extensions, Proceedings of the 14th Usenix Security Symposium, Baltimore, Maryland.
17. Roßnagel, H., Zibuschka, J. (2007) Integrating Qualified Electronic Signatures with Password Legacy Systems, Digital Evidence Journal, 4, 1, 1-10.
18. Roßnagel, H. (2007) Mobile Qualifizierte Elektronische Signaturen: Analyse der Hemmnisfaktoren und Gestaltungsvorschläge zur Einführung, unpublished doctoral dissertation, Department of Business Administration and Economics, Johann Wolfgang Goethe University, Frankfurt am Main.
19. Savard, J. (1999) Keystream Base Conversion and Random Bit Unbiasing, A Cryptographic Compendium.
20. Secure Information Technology Center – Austria (2006) The Austrian Citizen Card, http://www.buergerkarte.at/index_en.html, February 28.

21. Weitzel, T. (2003) A Network ROI, Proceedings of the MISQ Academic Workshop on ICT standardization, ICIS 2003, Seattle WA, USA.