Password-Enabled PKI:

Virtual Smartcards versus Virtual Soft Tokens

Ravi Sandhu

SingleSignOn.Net Inc., and
George Mason University
rsandhu@singlesignon.net

Mihir Bellare

SingleSignOn.Net Inc., and
University of California, San Diego
mihir@cs.ucsd.edu

Ravi Ganesan

SingleSignOn.Net Inc.,
11417 Sunset Hills Road, Reston, VA
ravi@singlesignon.net

Abstract

Recently there has been considerable interest among PKI vendors and researchers in the concept of password-enabled PKI. Several viable proposals and products have emerged. Fundamentally there are two distinct methods for using passwords with private keys. One method is to use the password to retrieve a private key, while the other uses the password as one component of the private key. We motivate the names virtual soft tokens for the former and virtual smartcards for the latter. The major characteristics of these two approaches are identified and contrasted.

1. Introduction and Motivation

The notion of a password-enabled PKI sounds like an oxymoron to those of us who have lived through the last decade of discussion on PKI and its rosy prospects. PKI was supposed to do away with passwords. By all logic and forecast, passwords should be a relic of the stone age of cyberspace and should no longer be in use today. PKI was expected to replace them with private keys securely generated and forever safe in tamper-proof smartcards. In the coming brave new world, these private keys would be activated by appropriate biometrics securely embedded and captured by the smartcard. The reality of 2002, however, is that passwords are used in cyberspace on a scale scarcely imagined a decade ago. There are hundreds of millions, perhaps even billions, of instances of password usage in cyberspace every day. Conservatively, consider tens of millions of users each invoking ten instances of password usages per day. In contrast one has to look high and low to find actual uses of smartcards, even in laboratory or pilot situations.

Simply stated, it is an indisputable empirical fact that smartcards have not happened.[1] If the original vision of smartcards with ubiquitous readers had become reality there would be no need to talk about password-enabled PKI. All the same, it is worth mentioning that the vision of smartcards has not faded completely, and they may still happen some day. At this moment the DoD is engaged in a major rollout of smartcards in numbers that make sense in the scale of today's Internet.[2] Not a few thousand or even a few hundred thousand but in the scale of 2 to 5 million. The discipline and resources of the DoD have few parallels in the world. This is a fascinating experiment to watch. It may finally prove the feasibility of a large-scale deployment of smartcards. Nonetheless it will be hard for Federal Government agencies, corporations, educational institutions, etc. to emulate this scale of deployment of smartcards. Note that the difficulty is not so much in the process and cost of issuing the smartcards per se, but much more in deploying smartcard readers on each and every computer in use by the user population. Proliferation of devices such as PDAs and wireless phones further compounds the problem. Moreover, we have needs at larger scale than the DoD experiment. The Federal Government often deals with 100's of millions of users. It is not unusual for large corporations to be in touch with 10's of millions of users. It is certainly within their vision to be in touch with 100's of millions and even billions of users in the future. Given the multi-year deployment of DoD smartcards, one wonders how the truly large scale will ever be realized in this mode. It seems rather unlikely that we will have a national scale deployment of smartcards in the near future, let alone a global scale deployment. Organizations whose PKI strategy depends entirely on smartcards happening very soon are making a rather risky bet.

[1] This statement should be understood in context of the use of smartcards for PKI on the Internet via widely available desktops and laptops. The use of smartcards in specialized applications has seen considerable success, more so in Europe and Asia than in North America. In these applications the smartcard is often embedded in a device such as a wireless telephone or a television-set top box, or in a credit-card with specialized readers.

[2] http://www.defenselink.mil/news/Oct2000/n10102000_20001010 7.html

If smartcards are not available where do we store the user's private key and how do we make it portable? Most systems today store the private key encrypted with a user-selected password on the hard disk. Portability is achieved by transporting the encrypted private key on removable media, such as a floppy disk. Currently one cannot guarantee availability of floppy disk readers, or other media-specific devices, on every computer. In general the portability of any media-specific transport is questionable in a truly open environment. All the same the notion of a "soft token", that is a private-key encrypted with a password, in contrast to a "hard token", that is a private-key which never leaves a smartcard, has been around for over a decade and has been deployed in several systems. From the user's perspective it is a natural progression to store the soft token on a network server rather than having to carry it around. This is very easily achieved by copying the contents of the soft token onto a server.

Password-enabled PKI relies on passwords to enable the use of private keys. Passwords are extremely easy to use and are easily usable from multiple computers. Users continue to express frustration with passwords, mainly because they have too many of them and are often required to change them too often. Password-enabled PKI alleviates both of these problems. PKI facilitates use of the same digital identity at multiple relying parties, including those with whom the user has had no prior contact. Thus a user need not be burdened with a separate password for every relying party. With a dramatically reduced number of passwords to remember, users can be reasonably persuaded (or gently enforced) to choose passwords of adequate complexity without having to write them on paper as a memory aid. Gentle enforcement of password complexity rules is more user-friendly than the current conventional wisdom of constantly chasing users to change passwords as a countermeasure to selection of weak passwords (or the writing on paper of many complex passwords).

To a large extent password-enabled PKI has happened in spite of PKI orthodoxy which calls for smartcard-enabled PKI wherein the private key never leaves the smartcard. As such the concept of password-enabled PKI has not really been studied systematically. Instead a number of proposals have been published and implemented, each one motivated by its own principal considerations. One of the goals of this paper is to provide a systematic analysis from security, functionality and operational perspectives. We specifically assume that the underlying cryptographic protocols are secure. This is a reasonable assumption since in many cases proofs of security or at least strong informal arguments have been provided. Empirically, we can say that it is quite feasible to get the cryptography correct. Our goal is to understand the overall security that is achieved and the functional and operational implications of specific approaches.

2. Password Vulnerabilities

It is generally agreed that password-enabled PKI will not provide the same level of security as smartcard-based[3] or biometric-based PKI.[4] All the same there is considerable confusion about the actual security vulnerabilities of passwords. So we begin with a brief discussion of password vulnerabilities before turning to the main topic of the paper.

There are some inherent vulnerabilities of password-based systems. A password can be compromised without knowledge of the legitimate owner. There is no physical evidence of theft. The possibility for undetected compromise is further enhanced if users reuse the same password at poorly protected sites, who may do something silly like storing passwords in the clear (or even less extreme). This is almost as bad as writing the password on paper and displaying it in a public place. Conversely a password can be easily shared. A common example is a corporate executive who shares her password with her secretary. In absence of other convenient mechanisms for this purpose, sharing of a password is a simple means to provide the secretary access to the executive's email. These inherent vulnerabilities cannot be completely addressed without cooperation and education of users. However, technology to mitigate these problems does exist. Misuse detection systems can help in identifying occurrences of misuse due to compromise or sharing. The concept of a trusted path can be used to ensure that passwords are revealed to trusted entities and not to software that spoofs the look and appearance of trusted entities. Even more effective is the use of protocols that do not reveal passwords but instead prove knowledge of the password for authentication.

Passwords are also susceptible to guessing attacks. On-line guessing requires the attacker to try

[3] It should be noted that not all smartcards are equal. It is possible to do smartcards very badly so they are not tamperproof. For purpose of this paper we assume that smartcards can be made tamperproof. In practice this is a difficult goal.

[4] Currently there is considerable interest in biometrics for authentication, especially following the events of September 11, 2001. Biometric-enabled PKI, with or without the use of smartcards, is a fascinating possibility for higher assurance than achieved by password-enabled PKI or even smartcard-enabled PKI. Consideration of biometric-enabled PKI is beyond the scope of this paper.

password guesses directly against the protected system and see if the guessed password works successfully. Enforcement of password complexity rules makes these attacks harder. The threat is further mitigated by throttling schemes which slow down the rate at which such attacks can be pursued. With a simple "three guesses and out" rule it is possible to introduce denial of service vulnerabilities but more sophisticated approaches are possible. For our purpose we assume that on-line guessing is taken care of in some such manner.

The most serious threat to existing password-based systems is the possibility of off-line dictionary attacks. In these attacks the attacker has knowledge of the outcome of some cryptographic operation which uses the password as a "key". The precise knowledge available and attendant attack varies from system to system. We will generically call this information as known plaintext.[5] We will also use the shorter term dictionary attack to specifically mean off-line dictionary attack. Known plaintext is sufficient to allow an attacker to verify if a password guess is correct or not. The crucial aspect is that the guesses can be verified off-line. By trying large numbers (tens or hundreds of thousands) of commonly used passwords from a so-called dictionary the attacker can succeed without searching the entire key space. This problem has been well known since at least 1979 [MT79] but it continues to be a major vulnerability of existing password-based systems [WU99]. We can distinguish between network-based and server-based offline dictionary attacks. In the former case the required known plaintext is obtained from the network protocol, possibly by network sniffing or more directly by simply running the protocol. Server-based attacks require capture of this information by server penetration in some way. In particular system administrators of the server will typically have easy access to the requisite known plaintext.

To complicate dictionary attacks a password is typically salted before it is used as a "key". The salt is a random number which is usually not kept secret. Different users with different salts will generate different known plaintext making the dictionary attack more time consuming. In particular the attacker cannot precompute known plaintext values from the dictionary passwords alone, but must do so separately for each value of the salt. This makes precomputation of the dictionary attack infeasible since the space of possible salts is very large.

[5] It should be noted that known plaintext can be known structure rather than known content.

3. Password-based Cryptographic Protocols

The Kerberos system [KN93, NT94] was one of the first to use passwords as a basis for cryptographic protocols. Susceptibility of Kerberos to network-based dictionary attacks is well-known [BM91, WU99].[6] A number of password-based cryptographic protocols immune to network-based dictionary attacks have since been published. Notable amongst these are the EKE [BM92], SPEKE [JAB96] and SRP [WU98] protocols, but there are many others. All these protocols use public-key cryptography in some way, a requirement that has been shown to be theoretically necessary [HK99b]. We can reasonably claim that, since about 1992, we know how to construct password-based cryptographic protocols immune to network-based dictionary attacks.[7]

In the above protocols both the client and the server store the password. Server compromise is however a real threat, and in this case it immediately yields the password to the attacker. In the augmented EKE [BM93] and SNAPI-X [MPS00] protocols, the server holds a hash of the password rather than the password, so server-compromise does not immediately yield the password to the adversary, but the attacker, having compromised the server, can still mount a dictionary attack based on the password hash. Immunity to server-based dictionary attack is not so easy to achieve. An approach based on multiple servers has recently emerged. The user's password is used to retrieve shares of a secret from multiple servers without exposure to network-based dictionary attacks. The secret is then assembled at the client computer from its shares. This long random secret can then be used for a variety of cryptographic purposes. Ford and Kaliski [FK00] present an elegant n-of-n scheme for this purpose, and suggest using 2-of-2 in practice. In general all n servers need to be penetrated by an attacker. Compromise of (n-1) is not sufficient. Jablon [JAB01] proposes schemes with additional desirable properties.

In practice schemes with multiple servers impose operational requirements to keep additional servers online and available. Moreover these servers may be subject to common-mode security failures.

[6] Kerberos failure to server attacks is complete and absolute obviating the need to do a server-based dictionary attack. It is interesting to note that Kerberos employs the user name and realm name as salt in its string_to_key function [KN93].

[7] Security analysis of such protocols is however subtle, and definitions of security goals, together with proven secure protocols, including a proof for the core of EKE, have emerged only more recently [BPR00, BMP00].

Once an attacker knows how to break one server, likelihood of success on the other is quite significant in practice. Possibility of insider attacks could be reduced due to requirement of insider collusion across multiple servers, but outsider attacks may not be significantly mitigated. At the same time operational quality may be degraded. Security infrastructure is expected to be more robust than the infrastructure it protects. Each security server would generally be replicated for reliability purposes. Each additional server therefore counts as two. Appropriate hardening of a single server with suitable separation of duties and least privilege could present a more viable approach to outsider and insider attacks.

4. Password-enabled PKI

With this background and motivation we now address the main topic of this paper. There are fundamentally two distinct ways to implement password-enabled PKI.

1. Employ the user's password as a means to securely retrieve the user's private key on to any computer from where it can then be used without further online interaction.
2. Employ the user's password as a component of the user's private key which can be used only in conjunction with another component which, in turn, can only be used on an online server.

The principal distinction between these two approaches is whether or not the user's private key is completely resident on the user's computer in a usable form. In the first case the user's key is available in the clear on the user's computer and can be used independent of any further interaction with the online server. Network-based storage of a user's private key in this manner is analogous to storage of an encrypted private key on a soft token. Once this private key is decrypted on a computer it can be used indefinitely without continued need for the soft token. Because of this analogy we call this approach a **virtual soft token** (or network-based soft token).

In the second approach the password only enables usage of the private key without bringing the entire private key together in one place. The overall private key is split into two components. One component is computed from the user's password. The other is resident on a secure online server. Let us call the former component the password component and the latter component the server component. Both components are required whenever the user wishes to use her overall private key but they are never brought together in one place. Instead an interactive protocol is carried out to achieve that result. Network-based usage of a component of the user's private key in this manner is analogous to usage of a private key in a smartcard. Just as the private key never leaves the smartcard, the server component of a user's overall private key never leaves the network server. Because of this analogy we call the second approach a **virtual smartcard** (or network-based smartcard).[8]

In the remainder of this paper we identify major characteristics of these two approaches to password-enabled PKI and compare them.

4.1. Virtual Soft Tokens

Virtual soft tokens enable retrieval of a user's private key onto any computer of the user's choice. A simplistic approach to this task would be to store each user's key encrypted with the user's password on an online server. Anyone could retrieve any of these encrypted keys, but without knowledge of the correct password would not be able to directly decrypt them. The virtual soft token is simply a substitute for the physical soft token.[9] Unfortunately this scheme is susceptible to dictionary attacks. An attacker who has access to the encrypted private key can verify guesses for the password by decrypting the private key with the guess and verifying success or failure with respect to the known public key.[10]

A virtual soft token therefore cannot be freely accessible for download. Instead it must be protected from network-based dictionary attacks by one of the password-based protocols such as EKE, SPEKE or equivalent.

Virtual soft tokens were first proposed in the SPX system [TA91]. The designers of SPX did not feel comfortable downloading the user's long-term private key to the client machine. Instead they proposed creation of a short-term private key whose public-key certificate was signed by the user's long-term private key. Only the short term key would be downloaded to the client machine for unrestricted use within its short life. In a sense this proposal is stronger than a physical soft token since compromise

[8] It should be noted that the term virtual smartcard has been used for schemes that are virtual soft tokens in our terminology. This is inappropriate since an essential characteristic of a smartcard is that the private key never leaves the smartcard.

[9] People who use soft tokens can trivially virtualize them in this manner by simply copying the soft token to some server from where it is accessible.

[10] Hoover and Kausik [HK99a] suggest that this dictionary attack can be avoided by protecting disclosure of the public key. Unfortunately their approach of "cryptographic camouflage" negates the main advantage of PKI where the public key does not need to be confidential. Technically, Hoover and Kausik also require elimination of redundancy in encryption of the private key so "known structure" attacks are not possible.

of the client leads to compromise of a short-term key with a life of say 8 hours. Compromise of a client with a long-term key with a life of say 1 year is more devastating. In another sense the SPX soft token is weaker with respect to non-repudiation. The user's long-term key is exposed on the SPX server where it is needed to construct the certificate for the short-term key. The SPX server can therefore impersonate the user via knowledge of the long-term key. Novell's NetWare v4 deployed a similar process for downloading a temporary private key [KPC95] (although it used a different set of underlying cryptographic algorithms).

Recent proposals for virtual soft tokens have returned to the idea of retrieving the long-term private key on to the client. As we have seen we know how to prevent network-based dictionary attacks in this context. A number of protocols for this purpose were recently presented by Perlman and Kaufman [PK99]. There are some significant differences in detailed properties of these protocols. Nonetheless from our vantage they all share a common core of security properties: exposure of long-term private keys on the client and immunity to network-based dictionary attacks.

Ford and Kaliski [FK00], and later Jablon [JAB01], propose solutions to server-based dictionary attacks. As discussed earlier these solutions require additional servers which may degrade operational quality while the gain in security may be diminished due to common-mode failures.

4.2. Virtual Smartcards

Virtual Smartcards are based on split private keys. In classical 2-key RSA the public and private keys for given n are related by the following equation.

$$e*d = 1 \bmod \varphi(n)$$

The splitting of d into d1 and d2 is computed as follows.

$$d1*d2 = d \bmod \varphi(n)$$

The fundamental operation of exponentiation in RSA then gives us the following equations.

$$(M^{d1})^{d2} \bmod n =$$
$$(M^{d2})^{d1} \bmod n =$$
$$M^{d1*d2} \bmod n =$$
$$M^{d} \bmod n$$

This idea can be extended to more than two splits of the original private key d if so desired. It can also be applied to an additive rather than multiplicative split. These ideas were first published by Colin Boyd [BOY89]. Their first application to virtual smartcards is due to Ganesan [GAN95, GAN96]. Ganesan's innovation was to realize that one of the split keys, say d1, can come from a password and therefore easily remembered and carried around mentally by a user. Nonetheless security of d2 is equivalent to security of a traditional RSA private key.[11]

To summarize, in a 3-key RSA system there are 2 private keys whose multiplication mod φ(n) is equivalent to a single overall private key. One of these keys d1 is computed from the user's password and known only to the user. It is the password component of the overall private key. The server component of the overall private key is d2 which is stored and used only on a secure online server. The server component constitutes a virtual smartcard which can be used only if knowledge of d1 is demonstrated. The overall private key d is never reconstructed on the client or the server. Every use of d involves an online interaction between the client and server.[12]

An immediate benefit of virtual smartcards is the ability to do instant revocation. The server resident d2 can be revoked at any time rendering the password component d1 completely useless. From here on d1 cannot be used to generate a signature even if the certificate for (e,n) continues to be valid. The network-based virtual smartcard will refuse to participate in the signing protocol. This is a tremendous benefit relative not only to virtual soft tokens but also to local smartcards. Another benefit is potential for misuse detection by monitoring usage of the virtual smartcard. Note that these benefits continue to accrue even if d1 is stored on a local smartcard rather than computed from a password. As such virtual smartcards provide valuable additional services even when we reach the age of ubiquitous smartcards (and smartcard readers).

MacKenzie and Reiter [MK01] have an interesting variation on the use of split-key RSA. They show how to make the loss of a local smartcard safe in that there is no private key within the smartcard that can be extracted. Also the smartcard is useless without knowledge of the user's password. In a nutshell the password component of a user's password is much the same as in Ganesan's scheme.

[11] This notion is formally proved in Appendix A.

[12] Contrast this with SPX discussed above where the entire private key is resident on the server. SPX thereby fails to provide non-repudiation.

The server component, however, is stored encrypted with the server's public on the smartcard, i.e., d2 encrypted with the server's public key. Cooperation of the server is therefore required whenever the smartcard is used. This is much like the virtual smartcard scheme. However, revocation is done out of band and requires the servers to maintain the equivalent of revocation lists. Mobility in this scheme is achieved by moving the device from computer to computer which requires a suitable reader or interface. This is a characteristic of conventional local smartcards.

5. Conclusion

In this paper we have identified two approaches to password-enabled PKI. We have motivated the reasons for calling these virtual soft tokens versus virtual smartcards. Virtual smartcards remove exposure of the user's private key on a client computer while allowing for misuse detection and instant revocation. Conversely, virtual soft tokens expose the user's private key on client computers and cannot support misuse detection or instant revocation. These are substantial differences.

As we look to the future, PKI thinking must depart from its conventional reliance on smartcards as the technology which will make PKI real. With hundreds of millions of computers deployed all over the world today retrofitting smartcard readers on each one is a formidable task. A variety of wireless and personal computing devices are also proliferating. Uniform availability of smartcard readers across all these devices is extremely unlikely. Instead we should look to an environment where virtual smartcards are pervasive with local smartcards and biometrics being used for higher assurance situations.

The recent big push for identity services on the Internet has veered away from PKI to proposals that are entirely password based and make extensive use of symmetric cryptography. In the past year we have seen a number of such initiatives from big players in the Information Technology arena. PKI still offers considerable advantages over symmetric technology. But if the PKI community is not alert and adaptive to industry trends we may find the baby is thrown out with the bath water.

References

[BM91] Bellovin, S and Merritt, M. "Limitations of the Kerberos authentication system." Proceedings of the Winter USENIX Conference, 1991, pp 253-267.

[BM92] Bellovin, S and Merritt, M. "Encrypted key exchange: password-based protocols secure against dictionary attacks." Proceedings of the IEEE Symposium on Security and Privacy, 1992, pp. 72 –84.

[BM93] Bellovin, S and Merritt, M. "Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise." Proceedings of the ACM Conference on Computer and Communications Security, 1993, pp. 244 – 250.

[BPR00] M. Bellare, D. Pointcheval and P. Rogaway. "Authenticated Key Exchange Secure Against Dictionary Attacks." Advances in Cryptology - Eurocrypt 2000 Proceedings, Lecture Notes in Computer Science Vol. 1807, B. Preneel ed, Springer-Verlag,2000.

[BMP00] V. Boyko, P. MacKenzie and S. Patel. "Provably Secure Password Authenticated Key Exchange Using Diffie-Hellman." Advances in Cryptology - Eurocrypt 2000 Proceedings, Lecture Notes in Computer Science Vol. 1807, B. Preneel ed, Springer-Verlag, 2000.

[BOY89] C. Boyd. Digital multisignatures. In Cryptography and Coding, H. Beker and F. Piper eds., Oxford University Press, 1989, pp. 241-246.

[BS01] M. Bellare and R. Sandhu. "The security of practical two-party RSA signature schemes." Manuscript, November 2001. Manuscript available via http://www.cse.ucsd.edu/users/mihir.

[FK00] Ford, W. and Kaliski, B. "Server-assisted generation of a strong secret from a password." Proceedings 9th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000, pp 176 – 180.

[GY94] R. Ganesan and Y. Yacobi. A Secure Joint Signature and Key Exchange System. Bellcore Technical Report TM-24531, October 1994.

[GAN95] Ravi Ganesan. Yaksha: Augmenting Kerberos with public-key cryptography.

Proceedings of the ISOC Network and Distributed Systems Security Symposium, 1995.

[GAN96] R. Ganesan. Yaksha: Towards Reusable Security Infrastructures. PhD Thesis. Johns Hopkins University, Baltimore, MD, 1996.

[HK99a] D. Hoover and B. Kausik, "Software smart cards via cryptographic camouflage." Proceedings of the IEEE Symposium on Security and Privacy, 1999.

[HK99b] S. Halevi and H. Krawcyzk. "Public-key cryptography and password protocols." ACM Transactions on Information and System Security (TISSEC) Volume 2 , Issue 3 (August 1999), Pages: 230 – 268.

[JAB96] D. Jablon, "Strong password-only authenticated key exchange." ACM Computer Communications Review, October 1996.

[JAB01] D. Jablon, "Password authentication using multiple servers." Proceedings RSA Conference: Cryptographers' Track, 2001 San Francisco, CA, April 8-12, 2001, Springer LNCS 2020.

[KPC95] C. Kaufman, R. Perlman and M. Speciner, "Network Security: Private Communication in a Public World." Prentice-Hall, 1995.

[KN93] J. Kohl and C. Neuman. The Kerberos Network Authentication Service (V5). RFC 1510, September 1993.

[MPS00] P. MacKenzie, S. Patel and R. Swaminathan. "Password Authenticated Key Exchange based on RSA." Advances in Cryptology - Asiacrypt 2000 Proceedings, Lecture Notes in Computer Science Vol. 1976, T. Okamoto ed, Springer-Verlag, 2000.

[MR01] P. MacKenzie and M. Reiter. "Networked cryptographic devices resilient to capture." Proceedings of the IEEE Symposium on Security and Privacy, 2001.

[MT79] Robert Morris and Ken Thompson, "Password Security." Communications of the ACM, Volume 22 Issue 11, November 1979.

[NT94] C. Neuman and T. Ts'o. "Kerberos: An Authentication Service for Computer Networks." IEEE Communications, 32(9):33-38. September 1994.

[PK99] R. Perlman and C. Kaufman, "Secure password-based protocols for downloading a private key." Proceedings of the ISOC Network and Distributed Systems Security Symposium, 1999.

[TA91] J. Tardo and K. Alagappan, "SPX: global authentication using public key certificates" Proceedings of the IEEE Symposium on Security and Privacy, 1991, pp. 232-244.

[WU98] T. Wu, "The Secure Remote Access Protocol." Proceedings of the ISOC Network and Distributed Systems Security Symposium, 1998.

[WU99] Thomas Wu, "A Real-World Analysis of Kerberos Password Security." Proceedings of the ISOC Network and Distributed Systems Security Symposium, 1999.

Appendix A: Equivalence of 3-key RSA To 2-key RSA

We show that the security of 2-key RSA is equivalent to the security of 3-key RSA, following Ganesan and Yacobi [GY94] who first established this conjecture of Boyd [BOY89].[13]

A traditional 2-key RSA pair is generated as follows.

1. Generate two large, distinct primes p, q of roughly equal bit-length
2. Compute n=p*q
3. Select e such that gcd(e,φ(n))=1 and 1<e<φ(n), where φ(n)=(p-1)*(q-1)
4. Compute d, such that 1<d<φ(n) and e*d =1 mod φ(n)

[13] We note that this argument only reflects key-recovery attacks. Security arguments for our schemes that consider forgery attacks are more involved, and provided in [BS01].

5. Destroy p, q
6. Public key is e, n and private key is d

In the password-based 3-key system steps 1-4 are as above, followed by the steps given below.

5. Ask user to select a password Pwd that meets password selection rules
6. Pick an iteration count IC
   Repeat
   6.1 Pick a random SALT
   6.2 Compute d1 = Expand(Pwd,SALT,IC)
   Until (gcd(d1, φ(n))=1 and 1<d1<φ(n))
   [The function Expand is specified via PKCS5. The IC value and the final SALT value are accessible for subsequent use by the user.]
7. Compute d2 such that 1<d2<φ(n) and d1*d2 = d mod φ(n).
8. Destroy p, q, d
9. Public key is e, n; user's private key component is d1 (user remembers password Pwd from which d1 is computed) and appliance private key component for that user is d2.

We claim that the expected number of iterations of the repeat loop in Step 6 is around 2, so that the loop terminates quite fast. (Assume the Expand function is random and has range $\{0,1\}^k$ where $2^{k-1} \leq n < 2^k$. Then the expected number of iterations is at most (2*n)/φ(n) which is very close to 2.)

The strength of the split-key setting is that it provides as much security as RSA even if the user password is compromised, in the following sense: the problem of computing d2 given n, e, d1 is as hard as the traditional RSA problem of computing the secret exponent given the public key in the standard setting.

To detail this claim, we recall that the traditional RSA problem is defined as follows:

Given: n, e
Compute: d such that e*d=1 mod φ(n) and 1<d<φ(n)

We define the split-key RSA problem as follows:

Given: n, e, d1
Compute: d2 such that e*d1*d2 =1 mod φ(n) and 1<d2<φ(n)

We claim that if the split-key RSA problem is tractable, then so is the traditional RSA problem. To justify this claim, we assume we are given a method of solving the split-key RSA problem relative to a password generation process (formally, randomized algorithm) P that models the client's choice of password. The following code shows how we can then solve the traditional RSA problem. Explanations follow the code.

Given n, e,

1. Run P to obtain a password Pwd
2. Pick random SALT, and IC, and compute d1 = Expand(Pwd,SALT,IC)
3. Run the given split-key RSA solving method on input n, e, d1 to obtain d2
4. Let m = e*d1*d2 - 1
5. Use m, e to factor n [see later text for why this is possible]
6. Use the factorization of n to compute φ(n)
7. Let d be the inverse of e modulo φ(n)
8. Output d

Note that the value d1 chosen in Step 2 may not be relatively prime to φ(n) and in that case the algorithm will probably not succeed. However, d1 as chosen in step 2 has probability around 1/2 of being relatively prime to φ(n) and hence the success probability of the algorithm above is about one-half that of the given method for solving the split-key RSA problem.

The value m computed in Step 4 is a multiple of φ(n), because, modulo φ(n) we have:

$$e*d1*d2 - 1 = e*d1*d2 - e*d = e*[d1*d2-d] = 0.$$

Step 5 uses a well-known fact, namely that given a multiple of φ(n) it is possible to factor n.

One might ask why the algorithm does not, after step 3, simply compute d = d1*d2, output d and halt, since this d satisfies e*d mod φ (n) = 1. However this d may not satisfy 1<d<φ(n).