Security implications in Kerberos by the introduction of smart cards

Nikos Mavrogiannopoulos Andreas Pashalidis Bart Preneel
KU Leuven, ESAT/SCD/COSIC – IBBT
Kasteelpark Arenberg 10
Leuven, Belgium
{firstname.lastname}@esat.kuleuven.be

ABSTRACT

Public key Kerberos (PKINIT) is a standardized authentication and key establishment protocol which is used by the Windows active directory subsystem. In this paper we show that card-based public key Kerberos is flawed. In particular, access to a user's card enables an adversary to impersonate that user even after the adversary's access to the card is revoked. The attack neither exploits physical properties of the card, nor extracts any of its secrets. We propose protocol fixes and discuss properties that authentication and/or key establishment protocols should provide in order to be better equipped against the threats that arise due to the usage of smart cards.

Categories and Subject Descriptors

C.2.0 [**Computer-communication networks**]: General—*Security and protection*; C.2.2 [**Computer-communication networks**]: Network protocols; C.3 [**Special-purpose and application-based systems**]: Smartcards

Keywords

security protocols, smart-cards, key agreement

1. INTRODUCTION

Consider the following real-world scenario. On his way to a customer, Bob realizes that he forgot his smart card in the drawer of his desk. Knowing that he needs a document that is stored on his company's file server, which he can access only using the card, he calls his colleague Alice for help. He explains his situation, and asks her to retrieve the document using his smart card from the drawer, and email it to him. He also tells her the required PIN. Alice follows his instructions, and Bob's meeting goes well. On the next day, after thanking Alice for her help, Bob takes back his card and, as a security precaution, also changes its PIN.

In the above scenario, Bob expects Alice and, for that matter, anyone else, to be unable to impersonate him from the moment he regains control of his card. Bob's expectation is reasonable, and stems from the fact that smart cards are possession-based authentication tokens; non-possession must lead to the inability to authenticate.

In this paper, we describe an attack against the Diffie-Hellman (DH) variant of the public key Kerberos protocol [33], that enables Alice to continue impersonating Bob even after he has regained control of his card and changed his PIN. Our attack, which assumes that smart cards are used for user authentication, shows that, contrary to reasonable expectation, smart card-based public key Kerberos fails to provide authentication based on 'something you have'. Moreover, it demonstrates that deployments where smart cards are protected with a PIN for the purposes of two-factor authentication, in fact do not provide any authentication factor. This is because Alice is able to continue impersonating Bob even after a PIN change. Our attack neither exploits the physical properties of the card, nor extracts any of its secrets. Note that smart card-based public key Kerberos is a deployed protocol [12, 14, 21], and has been proven secure under certain models [2, 8, 11, 26]. Also note that implementations that conform to the specification [33] are required to support the DH variant.

The models under which public key Kerberos was shown to be secure do not support the outsourcing of certain functions to a removable smart card which is subject to attack independently from the user's terminal. Finally, we propose a fix to the protocol, and discuss desirable protocol properties.

The rest of this paper is organized as follows. The next section surveys related work. Section 3 describes our attack and proposes a protocol fix. Section 4 discusses certain protocol properties and explains why they are particularly desirable when smart cards are used as secure computation devices. Section 5 discusses whether existing formal analysis are sufficient to identify the attack. Finally, section 6 concludes.

2. BACKGROUND AND RELATED WORK

Kerberos is an authentication and key distribution protocol originally proposed in 1988, and has a long history of attacks and updates (see, for example, [2, 11] and the references therein). Its main goal is to enable users to log into multiple servers that belong to a common infrastructure. To this end, a user first requests an electronic 'ticket' from a central Key Distribution Center (KDC). The ticket then enables authenticated users to log into any server that is part of the infrastructure.

ASIACCS '12, May 2–4, 2012, Seoul, Korea.

The focus of this paper is the DH variant of the public key Kerberos protocol (PKINIT), as specified in [25, 33]. This protocol naturally lends itself to an implementation where the user's private signing key is stored in a smart card [12, 14]. In the resulting combined protocol, the smart card functions as a signature device that signs the user's ticket requests. However, introducing a smart card to a security protocol presents risks. This is because the system is now divided into a larger number of entities, each of which is subject to attack [19, 28]. A large body of literature discusses, for example, attack strategies that involve corrupting the user's terminal [1, 6, 9, 31].

Several models, such as the BAN logic [10, 20], process calculus-based logics [13], and complexity-theoretic analysis techniques (e.g. [4]) capture protocol security guarantees on authentication and key exchange. The adversary they consider is typically a Dolev-Yao-style adversary [17], i.e. is assumed to control the entire network, but is not able to corrupt participants. Other models that consider stronger adversaries, i.e. ones that may also corrupt participants by extracting their private keys, have been used to capture the notion of perfect forward secrecy [16, 7, 29].

The public key Kerberos protocol, including its DH variant, has been proven secure under a Dolev-Yao-style adversary model [2, 8, 11, 26]. However, the adversary models used in the above studies do not consider adversaries with attack abilities such as temporary access to a user's card. That is, the currently used models for protocol verification do not capture our attack and, more generally, their positive results of protocol verification do not carry over to a smart card setting.

Despite the fact that smart cards have been introduced to a multitude of protocols, to the best of our knowledge, only few works provide formal treatments of smart card-based protocols against an adversary that is able to attack a user's terminal independently from his smart card (see, for example, [3, 18, 30]). Our attack underlines the necessity of such models in verifying a protocol's security guarantees when smart-cards are involved.

3. OUR ATTACK

This section briefly revisits the DH variant of public key Kerberos with smart cards [12, 21, 25], describes our attack, and proposes a fix. The attack was disclosed to the IETF Kerberos mailing list in June 2011 [23].

3.1 DH public key Kerberos with smart cards

The protocol specifies three types of player: users, servers and a central entity called the Key Distribution Center (KDC). A user is equipped with a terminal and a smart card. The card contains an asymmetric key pair and a certificate, signed by an authority, that binds the public key to the user's identity. Moreover, the card provides an interface over which the terminal can ask the card to sign messages using the private key. Note that the card may require a PIN in order to respond to signature requests from the terminal.

One of the main goals of the protocol is to establish fresh session keys between users and servers. Figure 1 provides a high-level overview of the protocol. When the user decides to log into a server (step 0 in the Figure) his terminal constructs an `AS_REQ` message as specified in [25] and then calculates its hash value h. Then it chooses a DH group, randomly generates an ephemeral DH secret $x \in \mathbb{Z}_p$, and computes g^x, where p is a large prime and g is the generator of the group. The terminal also chooses a nonce n and stores the current time. It then provides the values g^x, n, τ and h to the smart card for signing (step 1). Note that, depending on the implementation, it may only provide a hash of these values to the smart card. If the smart card is PIN-enabled, then the user must provide his PIN prior to this operation. Depending on the implementation, the PIN is inserted either to the terminal or the smart card reader. Figure 1 shows the case where the user provides his PIN to the terminal in step 0, and where this gets sent to the card in step 1.

The signature σ, output by the smart card (step 2), is then used by the terminal to construct an augmented version of the `AS_REQ` message, which we denote by `AS_REQ`*. This message, which contains g^x, n, τ, σ and `AS_REQ` as a substructure, is sent to the KDC (step 3) which, among other things, verifies the signature.[1] If verification succeeds, then the KDC chooses a random $y \in \mathbb{Z}_p$, computes g^y and the ephemeral secret $\kappa = g^{xy}$, and constructs a response `AS_REP` according to [25]. This message contains a ticket which is encrypted with κ. Finally, it augments this message with fields containing the value of g^y and n signed with server's private key. The resulting message, denoted `AS_REP`*, is sent to the terminal (step 4).

Using the value g^y and its ephemeral secret x, the terminal recovers the key κ and is therefore able to decrypt the ticket. This ticket enables the terminal to complete the subsequent message exchange with the desired server (step 5). At the end of this exchange, which follows the standard [25] and does not involve the smart card, the user and the server will have established a session key. The exact details of this exchange are not relevant to our attack.

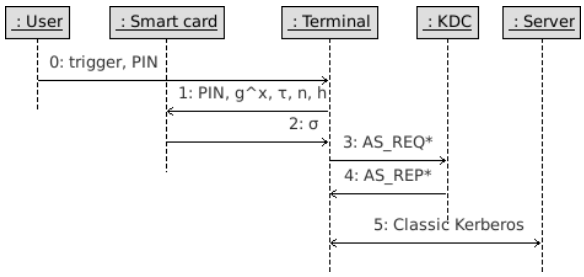

Figure 1: The DH variant of public key Kerberos with smart card (simplified).

3.2 The attack

Our attack is based on the observation that the KDC has no means to verify whether or not an incoming `AS_REQ`* message is fresh. That is, while the KDC checks that the timestamp τ indicates approximate current time, this does not guarantee that the `AS_REQ`* message was constructed recently. In fact, the `AS_REQ`* message could have been generated in the distant past.

In order to mount our attack, the adversary first obtains access to a victim's smart card. This can be done either by

[1] Note that n and τ are used for replay protection. In particular, the KDC stores the received nonce n until a particular expiry time, which is calculated based on τ. All `AS_REQ`* messages that (a) are received until the time of expiry and (b) contain the same nonce, are considered replays and are rejected.

compromising the victim's terminal, or by stealing the card and its PIN. The adversary then fabricates an AS_REQ message, calculates its hash h, chooses a nonce n, a random DH secret x, and chooses a timestamp τ indicating a particular future point in time. It then sends the values g^x, τ, n, and h to the card in order to obtain the signature σ. Using this signature, the adversary constructs an AS_REQ* message.
Note that this fabricated AS_REQ* message will be accepted by the KDC as a genuine ticket request from the victim at time indicated by τ. Since neither the victim's himself, nor his smart card is required in the remainder of the protocol, the message enables the adversary to impersonate the victim to the KDC at time τ. With the ticket in the KDC' response, the adversary will further be able to impersonate the victim to the server of his choice until the ticket expires.
Note that, in order to be able to impersonate a victim at, say, approximately 20:00 of every Monday in a two-year period, the adversary must fabricate about 104 AS_REQ* messages as described above and, for each such message, obtain a signature from the card. In other words, a few minutes of access to a victim's card are sufficient for the adversary to be able to impersonate the victim, on a regular basis, for years.

3.3 Proposed fixes to the protocol

If the user's smart card was required in a later protocol stage, our attack may have been avoided. The RSA public key encryption variant of public key Kerberos, for example (also specified in [33]), does not appear to be affected by our attack because, in this variant of Kerberos, the smart card is required to decrypt the KDC' reply. Thus, the adversary would not be able to decrypt replies from the KDC without access to the smart card. However, switching to this Kerberos variant may not be desirable because it allows violation of the key separation principle: the same RSA key pair is used for both signing and decrypting [22, 33]. Moreover, switching to this Kerberos variant is not an option for smart cards that are based on the DSA or ECDSA cryptosystems [32, 33], since they not support decryption.
We propose two fixes, one that completely defends against the attack at the cost of an additional message exchange, and a fix that limits the effectiveness of the attack to a specific time period. None of the proposed fixes requires already deployed cards to be replaced, assuming that the cards conform to widely used standards [24, 27], and can therefore can sign arbitrary data. They also do not require involvement of the user's smart card after the initial message is generated, but instead provide the means for the KDC to verify the freshness of incoming AS_REQ* messages.
The first fix requires the KDC to initially send a nonce n_S to the terminal. This nonce is then added to the data signed by the smart card and included in AS_REQ* (see step 1 in Figure 1). On reception of the AS_REQ* message, the KDC must also ensure that the number n_S in the message matches the number it generated. Note that this fix requires a message from the KDC to the terminal carrying the nonce, and a change to the AS_REQ* message to accommodate the additional nonce.
An alternative fix does not change the two-message negotiation of Kerberos but requires the client to obtain a fresh 'cookie' from the KDC at a regular intervals, selected by the KDC (e.g. once per day). The cookie is randomly generated by the KDC periodically and is used to ensure the freshness of the incoming AS_REQ* messages. For that, the cookie must be covered by the signature of the AS_REQ* message. Note that now an adversary can successfully launch our attack within the cookie validity period.
Finally, it is perhaps worth mentioning that, albeit in reference to a previous version of Kerberos, weaknesses by the use of timestamps were already pointed out, and replacing them with nonces was already suggested, more than twenty years ago [5].

4. DESIRABLE PROPERTIES FOR SMART CARD PROTOCOLS

Authentication and session key establishment protocols that make use of smart cards are exposed to threats that arise due to fact that certain functions are performed by the terminal while others take place on the smart card.[2] The most important threats are adversaries that (a) have temporary access to smart cards, and (b) compromised user terminals. Smart card-based protocols should resist these threats by automatically recovering when a temporary compromise is over. That is, if the adversary's access to the smart card is revoked, or when a non-compromised terminal is used, then the protocol should provide the same level of security as if no compromise ever happened.
In the following, we list four protocol properties that are aimed to address the above threats. Although they do not constitute a formal security model, we believe that they offer a practical check list for protocol designers and implementers. Note that we list only properties that address the smart card-specific threats. As such, our list complements the list of desirable protocol properties from [7].
The following properties ensure that a temporary compromise only affects the secrecy of sessions that fall inside the time window of the compromise.

- *SC perfect forward secrecy:* Session keys that were established with a user's smart card over a non-corrupted terminal remain secret, even if an adversary later obtains access to that user's smart card, as long as the adversary does not extract any long-term secrets from the card.
- *SC backward secrecy:* Session keys that are established with a user's smart card over a non-corrupted terminal remain secret, even if an adversary had previously accessed the user's smart card although, without extracting any long-term secrets.

Note that perfect forward secrecy as defined in [7, 16] requires session keys to remain secret even if the adversary has knowledge of long-term keys. *SC perfect forward secrecy* is, therefore, a weaker notion; however, we argue that it is more relevant in practice to smart card-based protocols because the cost of obtaining access to a victim's smart card and simply performing computations with it, is typically much lower than the cost of extracting its long-term secrets. Of course, if a protocol provides the stronger notions of forward/backward secrecy, then it is likely to also satisfy the above notions.
The following properties ensure that a temporary compromise is not adequate for the adversary to impersonate legitimate users outside the time window of the compromise.

[2]While we use the term 'smart card', the discussion in this section also applies to protocols that make use of other security tokens or modules.

- *SC key-compromise impersonation:* An adversary with access to a user's smart card, but without having extracted the long-term secrets stored in the card, should not be able to impersonate other entities to that user, as long as the user uses a non-corrupted terminal.
- *Possession-based authentication:* An adversary with access to a user's smart card, but without having extracted the long-term secrets stored in the card, should be able to impersonate that user only for as long as it has access to his card.

SC key-compromise impersonation is a weaker notion than key-compromise impersonation as defined in [7], because the adversary is not given the user's long-term secrets. Finally, *possession-based authentication* captures the 'ownership' of the card, in particular that performing operations with the keys stored in the card implies physical possession of the card. A protocol that does not offer the above properties, fails to meet reasonable expectations of any system that is supposed to provide authentication based on 'something you have'.

5. EXISTING VERIFICATION METHODS

The existing models used for the verification of public key Kerberos [2, 8, 11, 26] consider a Dolev-Yao style adversary where, in addition to the network, the adversary may also control statically corrupted parties. Since they do not consider the smart card setting, these models do not capture an adversary with temporary access to a smart card.

Shoup and Rubin present such an approach in [30]. Their model considers an adversary that can perform operations with a user's smart card, and the applicability on the Kerberos protocol can be easily demonstrated, showing that existing protocol verification methods are sufficient to model the requirements of smart card-based protocols.

6. CONCLUDING REMARKS

This paper demonstrates that seemingly straight-forward protocol extensions, such as the migration of the user's private key from his computer to a smart card, can have severe implications to the security of the overall system. Even though previous work has argued for the need of a separate model for smart card protocols [28, 30], the case of public key Kerberos shows that the usage of smart cards is sometimes still handled as a deployment issue rather than a fundamental change to the protocol. As a result, the proofs of security for public key Kerberos, which were based on the protocol specification without smart cards, become effectively irrelevant.

We showed that the DH variant of public key Kerberos, when used with smart cards, enables an adversary with access to a user's smart card, to impersonate that user even after access to his smart card is revoked. We also proposed fixes for the protocol. Clearly there is need to formally verify smart card-based public key Kerberos, with our proposed fixes, under a smart card-aware model such as [30]. Such a treatment would not only verify, or disqualify, the effectiveness of our fix, but may also uncover other unidentified issues in the overall protocol.

More generally, there is a need to formally verify all smart card-based protocols, especially those that have been implemented and deployed, such as TLS with PKCS#11 smart cards [15, 27], under an appropriate model.

7. ACKNOWLEDGMENTS

The authors would like to thank Jens Hermans for his comments which improved the manuscript. This work was supported in part by the Institute for the Promotion of Innovation through Science and Technology in Flanders (IWT Vlaanderen) SBO project, the Research Council K.U.Leuven: GOA TENSE (GOA/11/007), by the IAP Programme P6/26 BCRYPT of the Belgian State (Belgian Science Policy), and in part by the European Commission through the support action with contract number 258630 GINI SA and CIP thematic network with contract number 270901 SSEDIC.

8. REFERENCES

[1] N. Asokan, H. Debar, M. Steiner, and M. Waidner. Authenticating public terminals. *Computer Networks*, 31(8):861 – 870, 1999.

[2] M. Backes, I. Cervesato, A. D. Jaggard, A. Scedrov, and J.-K. Tsay. Cryptographically sound security proofs for basic and public-key kerberos. *International Journal of Information Security*, 10(2):107–134, 2011.

[3] G. Bella. Inductive verification of smart card protocols. *Journal of Computer Security*, 2003.

[4] M. Bellare, R. Canetti, and H. Krawczyk. A modular approach to the design and analysis of authentication and key exchange protocols. Cryptology ePrint Archive, Report 1998/009, 1998.

[5] S. M. Bellovin and M. Merritt. Limitations of the kerberos authentication system. *SIGCOMM Comput. Commun. Rev.*, 20:119–132, October 1990.

[6] I. Z. Berta, L. Butty, and I. Vajda. A framework for the revocation of unintended digital signatures initiated by malicious terminals. *IEEE Transactions on Dependable and Secure Computing*, 2:268–272, 2005.

[7] S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. In M. Darnell, editor, *Crytography and Coding*, volume 1355 of *Lecture Notes in Computer Science*, pages 30–45. Springer Berlin / Heidelberg, 1997.

[8] B. Blanchet, A. D. Jaggard, A. Scedrov, and J.-K. Tsay. Computationally sound mechanized proofs for basic and public-key kerberos. In *Proceedings of the 2008 ACM Symposium on Information, Computer and Communications Security, ASIACCS 2008, Tokyo, Japan, March 18-20, 2008*, pages 87–99, 2008.

[9] A. Bottoni and G. Dini. Improving authentication of remote card transactions with mobile personal trusted devices. *Computer Communications*, 30(8):1697 – 1712, 2007.

[10] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on computer systems*, 8:18–36, 1990.

[11] I. Cervesato, A. D. Jaggard, A. Scedrov, J.-K. Tsay, and C. Walstad. Breaking and fixing public-key Kerberos. *Information and Computation*, 206(2-4):402 – 424, 2008. Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA '06).

[12] M. K. Consortium. PKINIT configuration, 2011. Available at: `http://k5wiki.kerberos.org/wiki/Pkinit_configuration`.

[13] A. Datta, A. Derek, J. C. Mitchell, and A. Roy.

Protocol composition logic (PCL). *Electron. Notes Theor. Comput. Sci.*, 172:311–358, April 2007.

[14] J. de Clerq. Microsoft TechNet: Smart Cards, 2011. Available at: `http://technet.microsoft.com/en-us/library/dd277362.aspx`.

[15] T. Dierks and E. Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246 (Proposed Standard), Aug. 2008.

[16] W. Diffie, P. C. V. Oorschot, and M. J. Wiener. Authentication and authenticated key exchanges. *Des. Codes Cryptography*, 2:107–125, June 1992.

[17] D. Dolev and A. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198 – 208, Mar. 1983.

[18] J. Furukawa, K. Sako, and S. Obana. IC card-based single sign-on system that remains secure under card analysis. In *Proceedings of the 5th ACM workshop on Digital identity management*, DIM '09, pages 63–72, New York, NY, USA, 2009. ACM.

[19] H. Gobioff, S. Smith, and J. D. Tygar. Smart cards in hostile environments. In *In Proceedings of the 2nd USENIX Workshop on Electronic Commerce*, pages 23–28, 1995.

[20] L. Gong, R. Needham, and R. Yahalom. Reasoning about belief in cryptographic protocols. In *Proceedings 1990 IEEE Symposium on Research in Security and Privacy*, pages 234–248. IEEE Computer Society Press, 1990.

[21] B. Hill. Weaknesses and best practices of public key kerberos with smart cards. Technical report, iSEC Partners, 2009.

[22] J. Jonsson and B. Kaliski. Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1. RFC 3447 (Informational), Feb. 2003.

[23] N. Mavrogiannopoulos. PKINIT with smart-cards, June 2011. IETF Kerberos mailing list post. Available at `http://www.ietf.org/mail-archive/web/krb-wg/current/msg02781.html`.

[24] Microsoft. Cryptographic service providers, 2011.

[25] C. Neuman, T. Yu, S. Hartman, and K. Raeburn. The Kerberos Network Authentication Service (V5). RFC 4120 (Proposed Standard), July 2005.

[26] A. Roy, A. Datta, and J. Mitchell. Formal proofs of cryptographic security of diffie-hellman-based protocols. In G. Barthe and C. Fournet, editors, *Trustworthy Global Computing*, volume 4912 of *Lecture Notes in Computer Science*, pages 312–329. Springer Berlin / Heidelberg, 2008.

[27] RSA Laboratories. PKCS #11: Cryptographic token interface standard v2.30, 2009.

[28] B. Schneier and A. Shostack. Breaking up is hard to do: Modeling security threats for smart cards. In *In First USENIX Symposium on Smart Cards*, 1999.

[29] V. Shoup. On formal models for secure key exchange, 1999. Appeared in the Theory of cryptography library and has been included in the ePrint Archive. sho@zurich.ibm.com 10500 received April 19, 1999. Revised November 15, 1999.

[30] V. Shoup and A. D. Rubin. Session key distribution using smart cards. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, pages 321–331, 1996.

[31] T. Stabell-kulø, R. Arild, and P. H. Myrvang. Providing authentication to messages signed with a smart card in hostile environments. In *Usenix Workshop on Smart Card Technology*, pages 10–11, 1999.

[32] L. Zhu, K. Jaganathan, and K. Lauter. Elliptic Curve Cryptography (ECC) Support for Public Key Cryptography for Initial Authentication in Kerberos (PKINIT). RFC 5349 (Informational), Sept. 2008.

[33] L. Zhu and B. Tung. Public Key Cryptography for Initial Authentication in Kerberos (PKINIT). RFC 4556 (Proposed Standard), June 2006.