On the Resilience of Key Agreement Protocols to Key Compromise Impersonation

Maurizio Adriano Strangio

University of Rome "Tor Vergata", Rome, Italy
strangio@disp.uniroma2.it

Abstract. Key agreement protocols are a fundamental building block for ensuring authenticated and private communications between two parties over an insecure network. This paper focuses on key agreement protocols in the asymmetric trust model, wherein parties hold a public/private key pair. In particular, we consider a type of known key attack called key compromise impersonation that may occur once the adversary has obtained the private key of an honest party. This attack represents a subtle threat that is often underestimated and difficult to counter. Several protocols are shown vulnerable to this attack despite their authors claiming the opposite. We also consider in more detail how three formal (complexity-theoretic based) models of distributed computing found in the literature cover such attacks.

Keywords: key compromise impersonation, key agreement protocols.

1 Introduction

Key agreement protocols are a fundamental building block for ensuring authenticated and private communications between two parties over an insecure network. We consider the asymmetric trust model wherein parties hold a long-term private/public key pair (and thus can establish each others true identity by exchanging digital certificates issued by a trusted Certification Authority).

The private key must be handled with care (e.g. stored on a tamper-proof storage token) to prevent it from falling into the hands of a malicious party since it is used across all communication sessions. On the other hand, ephemeral session-specific data is used once and is therefore simply discarded when (or even before) a session terminates.

A (two-party) key agreement protocol should provide some form of assurance regarding the authenticity of the session key as well as ensuring that the key is known only by the two intended participants at the end of the protocol execution. This includes the requirement that the protocol must also provide some form of entity authentication (e.g. by adding the digital certificates to the message flows).

The notions of "authenticated key agreement" considered within the compass of this article are the following: *Authenticated Key* (AK) agreement whereby a party A is assured that no one aside from the identified principal B (the intended partner of A in the communication) can possibly learn the value of the session key and, *AK with key Confirmation* (AKC), whereby party A is assured that B (and/or vice versa) has actually computed (or knows how to compute) the session key (see [8, 5] for further discussion).

A.S. Atzeni and A. Lioy (Eds.): EuroPKI 2006, LNCS 4043, pp. 233–247, 2006.

We will consider only implicitly authenticated AK protocols, i.e. authentication is complete when both principals have successfully concluded a run of the protocol and have then proved knowledge of the session key in subsequent communication.

As usual, we designate the two generic parties participating in a protocol run as Alice and Bob. Suppose an adversary (say Eve) was able to obtain the private key of Alice either by compromising the machine running an instance of the protocol (e.g. when the key is stored in conventional memory as part of the current state) or perhaps by cloning Alice's smart card while she inadvertently left it unattended. Eve may now be able to mount the following attacks against the protocol:

1. impersonate Alice in a protocol run;
2. impersonate a different party (e.g. Bob) in a protocol run with Alice;
3. obtain previously generated session keys established in honest-party runs of the protocol.

In case 1. Eve can send messages on behalf of Alice and these will be accepted as authentic, in case 2. Eve could establish a session with Alice while masquerading as another party; this is known as Key Compromise Impersonation (KCI) and seems to appear for the first time in [17]. For example, Eve could impersonate a banking system and cause Alice to accept a predetermined session key and then obtain her credit card number over the resulting private communication link. In case 3. Eve may be able to decrypt the data exchanged by Alice and Bob in previous runs of the protocol (provided the transcripts are known).

The discussion above demonstrates that long-term key compromise can lead to undesirable consequences (at least until the involved principal discovers that his key was compromised). However, protocol designers are often concerned with forward secrecy and seem to ignore key compromise impersonation.

The main thesis of this paper is that key compromise impersonation is not less important than forward secrecy; one should require that a secure key agreement protocol be also KCI-resilient since this security attribute is also related to party corruption.

In Section 4 we show that several implicitly authenticated key agreement protocols found in the literature do not withstand KCI attacks despite the authors claims. In order to offer a simplified and uniform treatment, all the protocols considered are specified in an elliptic curve setting since most of them were originally conceived in EC-based groups. In Section 5 we consider in more detail how three formal (complexity-theoretic based) models of distributed computing cover such attacks.

2 Notation and Mathematical Background

Given two strings s_1, s_2, the symbol $s_1 \| s_2$ denotes string concatenation. If X is a finite set then $x \stackrel{R}{\leftarrow} X$ denotes the sampling of an element uniformly at random from X. If α is neither an algorithm nor a set $x \leftarrow \alpha$ represents a simple assignment statement. The hash function $\mathcal{H}$ is used as a key derivation function (see [15] for practical KDFs).

The protocols we consider are based on EC cryptosystems. Let $\mathbb{F}_q$ denote the finite field containing q elements, where q is a prime power ($q = p$ or $q = 2^m$). An elliptic