Cryptanalysis of an Efficient Diffie-Hellman Key Agreement Protocol Based on Elliptic Curves

Shengbao Wang[1], Zhenfu Cao[1], Maurizio Adriano Strangio[2], and Lihua Wang[3]

[1] Department of Computer Science and Engineering,
Shanghai Jiao Tong University, China
{shengbao-wang,cao-zf}@cs.sjtu.edu.cn

[2] Department of Mathematics,
University of Rome "Roma Tre", Italy
strangio@mat.uniroma3.it

[3] Information Security Research Center,
National Institute of Information and Communications Technology, Japan
wlh@nict.go.jp

Abstract. In 2005, Strangio proposed protocol ECKE-1 as an efficient Diffie-Hellman two-party key agreement protocol based on elliptic curves. In this letter, we show that the protocol is vulnerable to a key-compromise impersonation attack. We also present a revised version of the protocol — ECKE-1N, which can withstand such attacks. The new protocol enjoys this property at the expense of a higher computational workload with respect to the original version.

Key Words. Key agreement, protocols, elliptic curves, key-compromise impersonation attack.

1 Introduction

Key agreement protocols allow parties to establish a secret *session key* over an insecure network. The session key may subsequently be used to achieve security goals such as confidentiality or integrity.

Protocols affording *implicit key authentication* (IKA) enable one party to be assured that no other party aside from its intended peer may learn the established secret key. A key agreement protocol that provides mutual IKA between the two parties is called an *authenticated key agreement* (AK) protocol [1].

Since the Diffie-Hellman key exchange was first introduced [2], a large number of key agreement protocols have been proposed (see [1] and Chapter 12.6 of [5] for comprehensive surveys). However, the design of secure and efficient key agreement protocols is notoriously a non trivial task. Indeed, sometimes flaws are found even years after a protocol was published.

In 2005, Strangio [6] proposed an efficient two-pass (1-round) elliptic curve key agreement protocol (ECKE-1). It is claimed that protocol ECKE-1 achieves all the desirable security attributes of an AK protocol, such as known-key security (KK-S), forward secrecy (FS), unknown key-share resilience (UKS-R), key control (KC), and in particular, *key-compromise impersonation resilience* (KCI-R).

A KCI-R protocol does not allow the adversary, that has obtained the long-term private key of an honest entity A, to successfully impersonate the peer entity B in a run of the protocol. In this letter we show that Strangio's protocol ECKE-1 is insecure against key-compromise impersonation attacks.

The rest of the paper is organized as follows. We first review protocol ECKE-1 [6] in Section 2. In Section 3, we give the details of the key-compromise impersonation attack.

In Section 4, we present a new version of protocol ECKE-1 — ECKE-1N, which is key-compromise impersonation resilient. Our concluding remarks are in Section 5.

2 Review of Protocol ECKE-1

We briefly review Strangio's protocol ECKE-1 [6]. The protocol is defined on an elliptic curve $E(\mathbb{F}_q)$ over a finite field $\mathbb{F}_q$, where q is a prime power. The number of points on $E(\mathbb{F}_q)$ is denoted by $\#E(\mathbb{F}_q)$. We use the same notations as in [6]:

- P $(= (P.x, P.y))$: the base point of large prime order in $E(\mathbb{F}_q)$;
- P_∞: the identity point in $E(\mathbb{F}_q)$;
- n: the order of P in $E(\mathbb{F}_q)$;
- h: the cofactor of n, i.e., $h = \#E(\mathbb{F}_q)/n$;
- (w_U, W_U): the long-term key pair of user U, where $w_U \in_R [1, n-1]$ is the private key and $W_U (= w_U P)$ is the public key;
- id_U: a user's identity information;
- $\mathcal{F}_1, \mathcal{F}_2$: two independent hash functions satisfying $\mathcal{F}_1, \mathcal{F}_2 : \{0,1\}^* \to \mathbb{F}_q$;
- $\mathcal{G}$: a key derivation function satisfying $\mathcal{G} : \mathbb{F}_q \to \{0,1\}^*$.

We assume that A and B are two honest parties running the protocol and that their public keys are exchanged via certificates issued by a trusted certification authority (CA). The protocol works as outlined below (cf. Fig.1):

$$\begin{array}{rl}
\hline
\multicolumn{2}{l}{A(w_A, W_A), B(w_B, W_B)} \\
\hline
A: & r_A \in_R [1, n-1] \\
 & e_A = \mathcal{F}_1(r_A, w_A, id_A) \\
 & Q_A = (r_A + e_A w_A)P \\
A \to B: & Q_A \\
B: & r_B \in_R [1, n-1] \\
 & e_B = \mathcal{F}_1(r_B, w_B, id_B) \\
 & Q_B = (r_B + e_B w_B)P \\
B \to A: & Q_B \\
A: & d_A = w_A \mathcal{F}_2(Q_A.x, Q_B.x, id_A, id_B) \\
 & T_A = h((r_A + e_A w_A)Q_B + d_A W_B) \\
 & sk = \mathcal{G}(T_A.x) \\
B: & d_B = w_B \mathcal{F}_2(Q_A.x, Q_B.x, id_A, id_B) \\
 & T_B = h((r_B + e_B w_B)Q_A + d_B W_A) \\
 & sk = \mathcal{G}(T_B.x) \\
\hline
\end{array}$$

Fig. 1. Protocol ECKE-1.

1. A picks a random r_A such that $r_A \in [1, n-1]$ and then computes $e_A = \mathcal{F}_1(r_A, w_A, id_A)$. Analogously, B picks a random r_B such that $r_B \in [1, n-1]$ and then computes $e_B = \mathcal{F}_1(r_B, w_B, id_B)$;
2. A computes $Q_A = (r_A + e_A w_A)P$. Symmetrically, B computes $Q_B = (r_B + e_B w_B)P$;
3. If $Q_A = P_\infty$ (resp. $Q_B = P_\infty$), A (resp. B) returns to step 2. Otherwise, A initiates a protocol run with B by sending Q_A to B;
4. B invokes a procedure to perform public-key validation of Q_A, namely to verify that Q_A is actually a point in the group $E(\mathbb{F}_q)$, and aborts the protocol run if the validation fails. Otherwise, B sends Q_B to A as the response message;
5. A invokes a procedure to perform public-key validation of Q_B and aborts the protocol run if the validation fails;

6. A and B compute, respectively, the points T_A and T_B;
7. Both A and B terminate holding the session key sk (see also Fig.1).

Correctness of the protocol follows from the fact that for honest parties we have $T_A = T_B$, therefore A and B will both compute the same secret session key sk from $h(r_A r_B + r_B e_A w_A + r_A e_B w_B + e_A e_B w_A w_B + d w_A w_B)P$[1], in which $d = \mathcal{F}_2(Q_A.x, Q_B.x, id_A, id_B)$.

The scalar multiplication by the cofactor h prevents the small-subgroup attack [3]. The basic structure of protocol ECKE-1 is similar to the well-known MQV protocol [3]. The author claimed that protocol ECKE-1, just like the MQV protocol, achieves key-compromise impersonation resilience. Unfortunately, the attack presented in the next section invalidates this security claim.

3 A Key-Compromise Impersonation Attack on Protocol ECKE-1

Let us consider the case that the long-term private key of an entity is compromised. Obviously, an adversary (denoted by E) who obtains this key is able to impersonate the corrupted party to other entities, since the latter is exactly identified by the long-term private key. However, it is desirable that such an event does not enable the adversary to impersonate other entities to her. Accordingly, a *key-compromise impersonation attack* is an attack whereby E, with some entity's long-term private key on hand, masquerades as a different entity and tries to establish a valid session key with the compromised party. Note that key-compromise impersonation attack represents a serious threat since an entity may not even be aware that her computer was corrupted and that an adversary has obtained her long-term private key.

Suppose that A's long-term private key (i.e. w_A) is disclosed. We now show that ECKE-1 is vulnerable to the key-compromise impersonation attack depicted in Fig.2, in which $E(B)$ indicates that E is impersonating B to A. The attack runs as follows:

$$
\begin{array}{rl}
\hline
\multicolumn{2}{l}{A(w_A, W_A), B(w_B, W_B)} \\
\hline
A: & r_A \in_R [1, n-1] \\
 & e_A = \mathcal{F}_1(r_A, w_A, id_A) \\
 & Q_A = (r_A + e_A w_A)P \\
A \rightarrow B: & Q_A \\
E(B): & r_{E(B)} \in_R [1, n-1] \\
 & Q_{E(B)} = r_{E(B)} P \\
E(B) \rightarrow A: & Q_{E(B)} \\
A: & d_A = w_A \mathcal{F}_2(Q_A.x, Q_{E(B)}.x, id_A, id_B) \\
 & T_A = h((r_A + e_A w_A) Q_{E(B)} + d_A W_B) \\
 & sk = \mathcal{G}(T_A.x) \\
E(B): & d_{E(B)} = w_A \mathcal{F}_2(Q_A.x, Q_{E(B)}.x, id_A, id_B) \\
 & T_{E(B)} = h(r_{E(B)} Q_A + d_{E(B)} W_B) \\
 & sk = \mathcal{G}(T_{E(B)}.x) \\
\hline
\end{array}
$$

Fig. 2. KCI attack on protocol ECKE-1.

1. $E(B)$ first picks a random $r_{E(B)}$ such that $r_{E(B)} \in [1, n-1]$ and then computes $Q_{E(B)} = r_{E(B)} P$;
2. A first picks a random r_A such that $r_A \in [1, n-1]$ and then computes $e_A = \mathcal{F}_1(r_A, w_A, id_A)$ and subsequently, $Q_A = (r_A + e_A w_A)P$;

[1] Notice that there are in fact some typos in [6], where the author mistakenly writes this term as $h(r_A r_B + r_B e_A w_B + r_A e_B w_A + e_A e_B w_A w_B + d_A d_B w_A w_B)P$.

3. If $Q_A = P_\infty$, A returns to step 2. Otherwise, A initiates a protocol run by sending Q_A, with B being the intended recipient;
4. $E(B)$ intercepts Q_A and sends $Q_{E(B)}$ to A, purportedly as B's response message;
5. A (resp. $E(B)$) invokes a procedure to perform public-key validation of $Q_{E(B)}$ (resp. Q_A) and aborts the protocol run if the validation fails;
6. A and $E(B)$ compute, respectively, the points T_A and $T_{E(B)}$;
7. Both A and $E(B)$ terminate holding the session key sk (see also Fig.2).

The above attack is successful since the two points computed by A and $E(B)$, i.e. T_A and $T_{E(B)}$, are equal, thus they will both compute the same secret session key sk. We prove this as follows (Obviously, we have $d_{E(B)} = d_A$):

$$\begin{aligned}
T_A &= h((r_A + e_A w_A)Q_{E(B)} + d_A W_B) \\
&= h((r_A + e_A w_A) r_{E(B)} P + d_A W_B) \\
&= h(r_{E(B)}(r_A + e_A w_A)P + d_{E(B)} W_B) \\
&= h(r_{E(B)} Q_A + d_{E(B)} W_B) \\
&= T_{E(B)}.
\end{aligned}$$

Therefore, when A wants to initiate a secure communication with any specific entity, E can always intercept the first protocol message Q_A and subsequently impersonate the specific entity to A, until the compromise is detected and the long-term key is revoked.

4 A Revised Version of Protocol ECKE-1

In this section we present a modified version of protocol ECKE-1 that is key-compromise impersonation resilient. The specification of the protocol, denoted as ECKE-1N, is presented in Fig.3.

$$\begin{array}{rl}
\hline
\multicolumn{2}{l}{A(w_A, W_A), B(w_B, W_B)} \\
\hline
A: & r_A \in_R [1, n-1] \\
 & e_A = \mathcal{F}_1(r_A, w_A, id_B) \\
 & Q_A = e_A P \\
A \to B: & Q_A \\
B: & r_B \in_R [1, n-1] \\
 & e_B = \mathcal{F}_1(r_B, w_B, id_A) \\
 & Q_B = e_B P \\
B \to A: & Q_B \\
A: & T_{A1} = h e_A Q_B \\
 & T_{A2} = h(w_A Q_B + e_A W_B) \\
 & T_A = T_{A1} + T_{A2} \\
 & sk = \mathcal{G}(T_A.x) \\
B: & T_{B1} = h e_B Q_A \\
 & T_{B2} = h(w_B Q_A + e_B W_A) \\
 & T_B = T_{B1} + T_{B2} \\
 & sk = \mathcal{G}(T_B.x) \\
\hline
\end{array}$$

Fig. 3. Protocol ECKE-1N.

Correctness of the protocol is obvious since we have $T_{A1} = T_{B1}$ and $T_{A2} = T_{B2}$, and therefore $T_A = T_B$.

With respect to the original protocol there is a degradation in performance since on-line computation for each principal requires performing four scalar multiplications (compared to the three required by protocol ECKE-1) and evaluating the hash functions $\mathcal{F}_1, \mathcal{G}$.

Protocol ECKE-1N is essentially based on the exchange of unauthenticated Diffie-Hellman ephemeral keys. The session key is derived by a construction similar to that of the MTI/A0 protocol [4].

It is easy to verify that the protocol is key-compromise impersonation resilient. Indeed, this follows from the formal proof of KCI resilience recently provided for the MTI/A0 protocol (cf. [7]).

The main security attributes of the protocol can be seen to hold in a straightforward manner. Forward secrecy is achieved by means of the term $e_A e_B P$ and holds due to the intractability of the computational Diffie-Hellman problem (note that this property does not hold for the MTI/A0 protocol). The terms e_A, e_B include the identities of the intended peers as opposed to the original protocol. This allows resistance to UKS attacks since the identity of the peer is involved in the calculation of the session key and therefore an replacement of certificates (for the same public key) during a run of the protocol would not allow the communication to take place (the parties would accept different keys).

5 Conclusion

Key agreement protocols play a central role in achieving secure communications, however, protocol design is extremely error-prone due to the inherent complexity of the problem. In this letter we have shown that, contrary to Strangio's security claim, protocol ECKE-1 [6] is insecure against key-compromise impersonation attacks. We have also proposed a new protocol construction that can withstand such attacks at the expense of a performance degradation.

Acknowledgments

The first author would like to thank Rongxing Lu for many constructive discussions on this work. This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant Nos. 60673079 and 60572155.

References

1. S. Blake-Wilson and A. Menezes, "Authenticated Diffie-Hellman key agreement protocols," in *Proc. Selected Areas in Cryptography 1998*, LNCS 1556, pp. 339-361. Springer-Verlag, 1999.
2. W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theory* vol.22, no.6, pp. 644-654, 1976.
3. L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone, "An efficient protocol for authenticated key agreement," Dept. C & Q, Univ. of Waterloo, CORR 98-05, 1998.
4. T. Matsumoto and Y. Takashima and H. Imai, "On seeking smart public-key distribution systems," in Transactions of IEICE, vol.E69-E, no.2, pp.99-106, 1986.
5. A. Menezes, P.C. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography. CRC Press, Boca Raton, 1997.
6. M. A. Strangio, "Efficient Diffie-Hellmann two-party key agreement protocols based on elliptic curves," in *Proc.* 20^{th} *ACM Symposium on Applied Computing (SAC)*, pp. 324-331, ACM Press, 2005.
7. M. A. Strangio, "On the resilience of key agreement protocols to key compromise impersonation," Cryptology ePrint Archive, Report 2006/252, 2006.