Revisiting an efficient elliptic curve key agreement protocol

Maurizio Adriano Strangio

University of Rome "Roma Tre", ITALY
strangio@mat.uniroma3.it

Abstract. A recent paper by Wang *et al.* has revealed a vulnerability in the ECKE-1 key agreement protocol. In particular, contrary to the author's claims, protocol ECKE-1 is shown to be susceptible to a key-compromise impersonation attack. This attack was also independently pointed out by the author in another recent paper published in the EURASIP Journal on Embedded Systems. Here we present a revised version of the protocol, ECKE-1R, which is key-compromise impersonation resilient and also (slightly) more efficient than the original protocol ECKE-1.

Key words:key compromise impersonation, key agreement protocol, elliptic curves

1 Introduction

In general, a secure two-party key agreement protocol should not allow an adversary, eavesdropping or manipulating message flows in any finite number of protocol runs, to subvert the security goals (e.g. obtain information on the secret session key, engage in a successful protocol run while masquerading as a legitimate principal, etc). However, designing a "good" key agreement protocol that is both efficient and secure is far from being a simple task; there are so many details involved (including the complicated interactions with the environment) that the designer can never be assured that the protocol is infallible. In practice, the degree of confidence accompanying a protocol (as with many other cryptographic primitive) increases with time as the underlying algorithms (and assumptions) survive many years of public scrutiny without any significant flaws being discovered.

A recent paper by Wang *et al.* [8] has revealed a weakness in the ECKE-1 [6] key agreement protocol. In particular, contrary to the author's claims, protocol ECKE-1 is shown to be susceptible to a key-compromise impersonation attack (cfr [7] for a discussion on KCI attacks). This attack was also independently pointed out by the author in a recent paper [2].

In this work, we present protocol ECKE-1R, a revised version of ECKE-1, which is key-compromise impersonation resilient. The new protocol is computationally more efficient with respect to the original version since there is one less hash function computation; however, it exhibits an increase in the communication (bit) complexity.

2 The original protocol ECKE-1

In this section we review protocol ECKE-1 [6]. The protocol is defined on a (subgroup of) elliptic curve $E(\mathbb{F}_q)$ over a finite field $\mathbb{F}_q$ with q a prime power (the protocol can also be specified in the generic multiplicative group).

Consider two parties A and B with private-public key pairs (w_A, W_A), (w_B, W_B) and digital certificates $cert_A, cert_B$, respectively. Long term keying material are associated with a set of domain parameters $\Phi_{EC} = (q, FR, S, a, b, P, n, h)$. Recall that q is the underlying field order, FR (*field representation*) is an indication of the method used to represent field elements in $\mathbb{F}_q$, the *seed* S is for randomly generated elliptic curves, the *coefficients* $a, b \in \mathbb{F}_q$ define the equation of the elliptic curve E over $\mathbb{F}_q$ ($E(\mathbb{F}_q)$), the *base point* $P = (P.x, P.y)$ in $E(\mathbb{F}_q)$, the prime order n of P and the cofactor $h = \sharp E(\mathbb{F}_q)/n$.

The parameters should be appropriately chosen so that no efficient algorithms exists that solve the Discrete Logarithm Problem (DLP) or the Computational Diffie-Hellman Problem (CDHP) in the subgroup $\langle P \rangle$. The domain parameters must also undergo a validation process proving the elliptic curve has the claimed security attributes [3].

We also need two (collision resistant) hash functions, namely $\mathcal{F}_1, \mathcal{F}_2 : \{0,1\}^* \rightarrow \mathbb{F}_q$. The function `kdf` represents a standard key derivation function (see [4] for examples of practical kdfs).

The main actions of protocol ECKE-1 are described as follows (refer to Figure 1 for the details):

1. A (resp. B) selects a random r_A (resp. r_B) in $[1, n-1]$ and computes e_A (resp. e_B);
2. If $Q_A \equiv P_\infty$ (resp. $Q_B \equiv P_\infty$), A (resp. B) repeats step 1 otherwise, in the role of initiator, A sends Q_A to B;
3. B invokes a procedure to perform public-key validation of Q_A and aborts the protocol if the validation fails;
4. B, in the role of responder, sends Q_B to A;
5. A invokes a procedure to perform public-key validation of Q_B and aborts the protocol if the validation fails;
6. A and B compute, respectively, the points T_A and T_B;
7. The protocol completes successfully if both A and B accept the same session key sk.

Correctness of the protocol is determined by the fact that in any honest execution $T_A \equiv T_B$, therefore A and B will both compute the same session key $sk = h(r_A r_B + r_A e_B w_B + r_B e_A w_A + e_A e_B w_A w_B + c w_A w_B)P$ with $c = \mathcal{F}_2(Q_A.x, Q_B.x, id_A, id_B)$. The cofactor h is needed in the scalar multiplication to prevent the small-subgroup attack [5].

On-line computation for each principal requires performing three scalar multiplications and evaluating both the hash functions $\mathcal{F}_1, \mathcal{F}_2$. The on-line computational complexity for a principal (say A) may be reduced by pre-computation of the 3-tuple (r_A, e_A, Q_A). The resulting workload will be two scalar multiplications and one hash value computation.

$$
\begin{array}{l}
\hline
A, B \\
\hline
\quad A : r_A \stackrel{R}{\leftarrow} [1, n-1] \\
\qquad e_A \leftarrow \mathcal{F}_1(r_A, w_A, id_A) \\
\qquad Q_A \leftarrow (r_A + e_A w_A)P \\
A \rightarrow B\text{: } Q_A \\
\quad B : r_B \stackrel{R}{\leftarrow} [1, n-1] \\
\qquad e_B \leftarrow \mathcal{F}_1(r_B, w_B, id_B) \\
\qquad Q_B \leftarrow (r_B + e_B w_B)P \\
B \rightarrow A\text{: } Q_B \\
\quad A : d_A \leftarrow w_A \mathcal{F}_2(Q_A.x, Q_B.x, id_A, id_B) \\
\qquad T_A \leftarrow h((r_A + e_A w_A)Q_B + d_A W_B) \\
\qquad sk \leftarrow \mathtt{kdf}(T_A.x) \\
\quad B : d_B \leftarrow w_B \mathcal{F}_2(Q_A.x, Q_B.x, id_A, id_B) \\
\qquad T_B \leftarrow h((r_B + e_B w_B)Q_A + d_B W_A) \\
\qquad sk \leftarrow \mathtt{kdf}(T_B.x) \\
\hline
\end{array}
$$

Fig. 1. Protocol ECKE-1

3 The revised protocol ECKE-1R

In previous work [6], Strangio claimed that the security attributes of protocol ECKE-1 included key-compromise impersonation resilience, forward secrecy, unknown key-share resilience and partial key control. In practice, however, the protocol suffers from a vulnerability that exposes it to key-compromise attacks.

Recall that a KCI attack involves an adversary that has obtained the private key of an honest party. Although direct impersonation of that party would then be straightforward the adversary may instead want to exploit the long-term key to capture valuable information about the “corrupted” party (e.g. credit card number). To this end, by impersonating a legitimate principal, the adversary attempts to establish a known session key in a run of the protocol with the target principal using the compromised private key.

The details of the KCI attack against protocol ECKE-1 are presented in [8, 2]. However, we point out that the conclusion drawn by the authors of [8], by which the adversary does not require a valid long-term key pair for such an attack to succeed is not correct. In fact, to compute the value $d_{E(B)}$ the adversary E must first obtain either w_A or w_B.

In this section we present protocol ECKE-1R, a revised version of protocol ECKE-1, which is key-compromise resilient and also slightly improves the overall efficiency by eliminating the need for computing the values d_A, d_B (while still maintaining all the relevant security attributes). The main actions of protocol ECKE-1R are shown in Figure 2.

Let $\mathcal{H} : \{0,1\}^* \rightarrow \mathbb{F}_q$ denote a collision resistant hash function.

Again, correctness of the protocol follows from the fact that in any honest execution $T_A \equiv T_B$, therefore A and B will both compute the same session key $sk = h(r_A r_B + e_A e_B w_A w_B)P$.

On-line computation for each principal requires performing three scalar multiplications and evaluating once the hash function $\mathcal{H}$. The on-line computational complexity

A, B
$A : r_A \stackrel{R}{\leftarrow} [1, n-1]$
$e_A \leftarrow \mathcal{H}(r_A, w_A, id_A)$
$Q_A \leftarrow r_A P$
$A \rightarrow B$: Q_A, e_A
$B : r_B \stackrel{R}{\leftarrow} [1, n-1]$
$e_B \leftarrow \mathcal{H}(r_B, w_B, id_B)$
$Q_B \leftarrow r_B P$
$B \rightarrow A$: Q_B, e_B
$A : T_A \leftarrow h(r_A Q_B + e_A e_B w_A W_B)$
$sk \leftarrow \mathtt{kdf}(T_A.x)$
$B : T_B \leftarrow h(r_B Q_A + e_A e_B w_B W_A)$
$sk \leftarrow \mathtt{kdf}(T_B.x)$

Fig. 2. Protocol ECKE-1R

for a principal (say A) may be reduced by pre-computation of the 3-tuple (r_A, e_A, Q_A). The resulting workload will be of only two scalar multiplications.

Notice that the KCI attack on protocol ECKE-1 as described in [8] no longer applies to protocol ECKE-1R. Indeed, the adversary (with knowledge of w_A) impersonating B is unable to construct a valid e_B, using a nonce of her own choice, since this requires knowledge of the long-term secret key w_B.

4 Security of protocol ECKE-1R

Protocol ECKE-1R is a key agreement protocol designed to provide implicit key authentication (IKA); the session key established in a run of the protocol should be known only by the two uncorrupted parties involved in the communication (since computation of the key by each party requires knowledge of their long-term private keys). We briefly (and informally) review the main security attributes of the protocol below.

Key privacy. The adversary is unable to compute the session key established by two honest parties in a run of the protocol assuming the intractability of the CDHP in the underlying group (and the session key, in the best case, is a randomly distributed value in $\{0, 1\}^\ell$ with $\ell \geq 128$).

Key independence. An adversary with known session keys (e.g. previously established by the same parties) has a negligible probability of mounting a successful attack against the protocol since session keys are uncorrelated (except for the possibility of nonces repeating, which is extremely unlikely).

Forward secrecy. An adversary that holds one or both of the long-term private keys w_A, w_B needs either r_A or r_B to derive the secret key of the target session (i.e. a completed session for which the corresponding ephemeral public keys Q_A, Q_B are also known). However, recovering these nonces is computationally infeasible assuming the intractability of the DLP (intractability of the CDHP precludes deriving the term $r_A r_B P$ used to compute the session key).

Key-compromise impersonation resilience. Suppose an adversary has obtained A's private key w_A; she can now easily impersonate A in a run of the protocol. The adversary (impersonating B) may succeed in a KCI attack against A if she is able to produce a valid e_B; however this is computationally unfeasible (due to the security properties of the hash function $\mathcal{H}$) unless w_B is known (statistically the odds are $2^{-|\mathcal{H}(\cdot)|}$ of guessing the correct value of e_B).

Unknown key-share resilience. An adversary posing as E cannot deceive A into believing that messages received from E were actually issued by B. Indeed, A and B make use of the values e_A, e_B which bind the identities id_A, id_B to the exchanged messages Q_A, Q_B by means of their respective private keys; therefore, A may (mistakenly) believe the session key is shared with E (the adversary), but will not derive the same session key as B, who is (correctly) convinced the session key is shared with A.

5 Conclusions and future work

In this paper we presented protocol ECKE-1R, a revised version of its predecessor ECKE-1 which was found to be vulnerable to key compromise impersonation attacks. It was informally shown that the new protocol is key-compromise impersonation resilient and also enjoys common security properties such as forward secrecy, key independence and unknown key-share resilience.

Future work includes the development of a formal proof of security in an appropriate model of distributed computing (e.g. in the model of Canetti-Krawczyck [1]).

References

1. R. Canetti and H. Krawczyk. Analysis of key exchange protocols and their use for building secure channels. *Advances in Cryptology-EUROCRYPT 2001*, LNCS 2045:453–474, 2001.
2. R. Duraisamy, Z. Salcic, M. Strangio, and M. Morales-Sandoval. Supporting symmetric 128-bit aes in networked embedded systems: an elliptic curve key establishment protocol-on-chip. *EURASIP Journal of Embedded Systems*, 2007:9, 2007.
3. D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer Professional Computing, New York, 2004.
4. IEEE-P1363.2/D15. Standard specifications for password-based public key cryptographic techniques. Institute of Electrical and Electronics Engineers, 2004.
5. L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, 28:119–134, 2003.
6. M. Strangio. Efficient Diffie-Hellmann Two-Party Key Agreement Protocols based on Elliptic Curves. *20th ACM Symposium on Applied Computing - Security Track*, pages 324–331, 2005.
7. M. A. Strangio. On the Resilience of Key Agreement Protocols to Key Compromise Impersonation. *Cryptology ePrint Archive, Report 2006/252*, http://eprint.iacr.org/2006/252.pdf, 2006.
8. S. Wang, Z. Cao, and R. Lu. Cryptanalysis of an efficient diffie-hellman key agreement protocol based on elliptic curves. *Cryptology ePrint Archive, Report 2007/26*, http://eprint.iacr.org/2007/026.pdf, 2007.