Efficient Diffie-Hellmann Two-Party Key Agreement Protocols based on Elliptic Curves

Maurizio Adriano Strangio
Department of Computer Science, Systems and Production
University of Rome "Tor Vergata"
Rome, ITALY
strangio@disp.uniroma2.it

ABSTRACT

Key agreement protocols are of fundamental importance for ensuring the confidentiality of communications between two (or more) parties over an insecure network. In this paper we review existing two-party protocols whose security rests upon the intractability of Diffie-Hellmann and Discrete Logarithm problems over elliptic curve groups. In addition, we propose a new two-party mutual authenticated key agreement protocol and collectively evaluate the security and performance of all the schemes considered. Elliptic curve techniques are used to minimise the computational workload on resource-constrained devices and to afford security levels with possibly fewer bits.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Public Key Cryptosystems

Keywords

Elliptic curves, cryptography, key agreement, protocols

1. INTRODUCTION

A basic security requirement to ensure the confidentiality and/or integrity of messages exchanged by two parties over an insecure network (such as the Internet) during a communication session is the establishment of a shared secret key. Minimal security requirements to be met by correctly executed key agreement protocols encompass: (1) termination with identical session keys; (2) (mutual) authentication of one party to the other; (3) generation of good quality (random) cryptographic keys (4) protection of session key secrecy against third parties.

Session key validity is required to span through the duration of a unique communication session since this may limit exposure in the event of key compromise, guarantee independence across communication sessions, simplify management and avoid vulnerabilities deriving from the use of a large number of keys for many different purposes.

In general, such protocols require users having at their disposal (symmetric or asymmetric) authentic cryptographic keys securely distributed or established in an initial setup phase. The correct management of such keys is a crucial issue since it may depend on human controlled procedures, on the existence of trusted third parties and in some cases on usage of expensive tamper-proof devices.

In a public network (e.g. the Internet), it can be assumed that messages exchanged in any protocol run by honest parties are susceptible to eavesdropping by an adversary, an attack which may be relatively easy to setup in practice. Such a scenario typically configures a *passive attack* wherein adversaries can perform off-line analysis of data extracted from valid protocol transcripts.

An *active attack* involves a more powerful adversary with enhanced capabilities; it is commonly assumed that she can modify, insert, delete, inject and reuse messages in on-line instances of the protocol. This leads to a number of possible attack scenarios where corresponding objectives include (a) impersonation of a party; (b) gaining knowledge of a session key; (c) misleading parties about the identity of another party they are communicating with.

We stress that the security analysis conducted here only considers network viable attacks against protocols, other security models (including non-cryptographic issues) may be required for protection of user communication equipment and is considered a local implementation issue.

Designing secure key agreement protocols is not a simple task; underlying assumptions must be made clear and security objectives precisely stated to allow them to be fulfilled, furthermore, the gap observed between cryptographic theory and practice makes the matter more complicated. Indeed, actual implementations of well known protocols have revealed fatal flaws years after they were first proposed: (a) incorrectness or lack of integrity preserving and authentication mechanisms; (b) susceptibility to replay and impersonation attacks; (c) not enough uniqueness for session ids (SIDs); (d) insufficient padding for packets and fields; (e) questionable usage of hash functions and/or ciphers; (f) buffer overflow attacks; (g) unknown message types; (h) resistance to side channel attacks; (i) leakage of keying material that does not compromise any information in past protocol message flows (forward secrecy), etc.

An important direction followed by recent research efforts in protocol security is the development of provably secure

SAC'05, March 13-17, 2005, Santa Fe, New Mexico, USA

schemes [5, 24, 6, 8] (in complexity-theoretic frameworks). Caution must be exercised since achieving meaningful results requires making correct and well defined assumptions. In addition, security proofs may be difficult to comprehend for the general public (including application developers).

Another approach in protocol design is based on the encapsulation of desirable security attributes [6] into the protocol, often tailored to the application, based on rules and techniques derived from the best practices and expertise of cryptographic professionals. Although heuristic arguments are used in the security analysis one hopes that all claims sum up to an acceptable security guarantee.

In this paper, we investigate two-party key agreement schemes based on asymmetric keys and on elliptic curve cryptosystems. With elliptic curve groups, in addition to the more efficient use of computational resources, it seems possible to achieve equivalent security with respect to other groups albeit with smaller key sizes. Cryptographic techniques based on elliptic curves have been recently introduced in standards endorsed by the IEEE and ANSI [3, 13, 14] and are now widely supported by these institutes which are conducting an ongoing standardisation effort to promote their acceptance and deployment.

Motivation for this work stems from the need to review existing elliptic curve based protocols in terms of security and efficiency to eventually identify those versions that are eligible for deployment on low power network capable devices such as cellular phones, PDAs, pagers and smart cards.

In section 2 we give an overview of elliptic curve cryptography and of basic concepts needed to evaluate the security of key agreement protocols. Section 3 recalls the security attributes of existing key agreement protocols founded on the conjectured intractability of Diffie-Hellmann problems (and discrete logarithms) instantiated in elliptic curve groups. We then propose a new elliptic curve based key agreement protocol and evaluate its security attributes, respectively, in sections 4 and 5. Finally, in the last part of the paper (section 6) we attempt a deeper insight into all the examined protocols by comparing performance and efficiency figures.

2. PRELIMINARIES

2.1 Notation

Let Σ denote the set $\{0,1\}$; then Σ^* is the set of finite binary strings, Σ^t is the set of finite binary strings of length t and Σ^∞ is the set of infinite binary strings. For the set S, $\sharp S$ denotes the number of elements in S.

The notation $Pr[x_1 \leftarrow C_1; \ldots x_k \leftarrow C_k : E]$ represents the probability of occurrence of event E in an experiment involving the variables $x_1, \ldots, x_k$ which are, respectively, assigned the values $C_1, \ldots, C_k$ (the C_is may sometimes represent sets, this should be clear from context). Instantiations of the random variable x_i (endowed with probability distribution D_i) over the set X_i is denoted by $x_i \overset{D_i}{\leftarrow} X_i$ (or by $x_i \overset{R}{\leftarrow} X_i$ if x_i is 0-uniform). Note that the above probability will be evaluated over the random variables x_i and on the other sources of randomness characterising event E.

We use the usual C-language style construct $(c?s1 : s2)$ as an abbreviation for "*if c then s1 else s2*".

2.2 Elliptic Curve Cryptography

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 curve $\mathrm{E}(\mathbb{F}_q)$ over the field $\mathbb{F}_q$ is a set of points $P \equiv (P.x, P.y)$ which satisfy a (Weierstrass) equation of the form

$$y^2 + a_1xy + a_3y = x^3 + a_2x^2 + a_4x + a_6$$

where $a_1, a_2, a_3, a_4, a_6 \in \mathbb{F}_q$. For simplicity assume $q = p$ prime ($p > 3$), then with an admissible change of variables in the preceding equation we obtain a more familiar representation:

$$y^2 = x^3 + ax + b, \quad 4a^3 + 27b^2 \neq 0$$

Care must be taken in choosing the coefficients $a, b \in \mathbb{F}_q$ since a poor choice could generate a curve which is easier to attack for the cryptanalyst.

The number of points $\sharp\mathrm{E}(\mathbb{F}_q)$ in an elliptic curve satisfies Hasse's bound:

$$|\sharp\mathrm{E}(\mathbb{F}_q) - (q+1)| \leq 2\sqrt{q}.$$

The set $\mathrm{E}(\mathbb{F}_q)$ endowed with the operation of *point addition* $Q = P + R$ defined according to a *chord-and-tangent* rule and the point at infinity P_∞ serving as the identity element forms an (abelian) group structure. Repeated addition of a point P to itself x times is known as *scalar multiplication* $Q = xP$, this operation often dominates the execution time of elliptic curve based cryptographic schemes.

The public elliptic curve domain parameters over $\mathbb{F}_q$ define an 8-tuple:

$$EC\_D = (q, FR, S, a, b, P, n, h)$$

where 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 two *coefficients* $a, b \in \mathbb{F}_q$ define the equation of the elliptic curve E over $\mathbb{F}_q$, the *base point* $P = (P.x, P.y)$ in $\mathrm{E}(\mathbb{F}_q)$, the prime order n of P and the cofactor $h = \sharp\mathrm{E}(\mathbb{F}_q)/n$. When needed procedures for *Domain Parameter Validation* are available [12].

Each user U in an elliptic curve cryptosystem is assigned a key pair (w_U, W_U) once a set of valid domain parameters EC_D has been defined. In practice, U's *private key* is a value w_U selected at random in $[1, n-1]$ and the corresponding *public key* is the elliptic curve point $W_U = w_U P$. When key pairs are bound to a user for a relatively long period of time they are called *long-term or static* key pairs, conversely, key pairs used for a single run of a key agreement protocol are typically known as *short-term or ephemeral* key pairs. The process of verifying that a public key W_U is actually a point in the group $E(\mathbb{F}_q)$ is called *Public Key Validation* [12].

The security of many cryptographic primitives is based on the conjecture that solving the discrete logarithm problem (DLP) on some finite cyclic groups is intractable. Let $E(\mathbb{F}_q)$ be an elliptic curve defined over a finite field and P a point in $E(\mathbb{F}_q)$ with prime order n. Let $\langle P \rangle$ be the cyclic subgroup of $E(\mathbb{F}_q)$ generated by P, then associated to n and P are the Elliptic Curve Discrete Logarithm (ecdl) function and its inverse defined by:

$$\mathsf{ecdl}_{n,P}(x) = xP \in \langle P \rangle,$$

$$\mathsf{ecdl}^{-1}_{n,P}(Y) = \log_P Y = x \in [1, n-1]$$

where $x \in [1, n-1]$, $Y \in \langle P \rangle$. The commonly accepted conjecture is that $\mathsf{ecdl}_{n,P}(x)$ is a one-way function and thus computing the inverse $\mathsf{ecdl}^{-1}_{n,P}(Y)$ is believed to be (computationally) hard, this is known as the Elliptic Curve Discrete Logarithm Problem. More formally, we make the following:

ASSUMPTION 1 (ECDLP). *Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$ and P a point in $E(\mathbb{F}_q)$ of prime order n. The Elliptic Curve Discrete Logarithm problem is intractable if for every probabilistic polynomial time Turing Machine $\mathcal{A}$ the probability of success $(\mathsf{Succ}^{\mathsf{ecdl}}_{P,E(\mathbb{F}_q)}(\mathcal{A}))$ in computing $\mathsf{ecdl}^{-1}_{n,P}(Y)$ is negligible:*

$$\mathsf{Succ}^{\mathsf{ecddh}}_{P,E(\mathbb{F}_q)}(\mathcal{A}) = \Pr\left[\begin{array}{c} x \xleftarrow{R} [1, n\text{-}1]; \\ Q \leftarrow \mathsf{ecdl}_{n,P}(x) : \\ \mathcal{A}(n, P, Q) = \mathsf{ecdl}^{-1}_{n,P}(Q) \end{array}\right] \leq \varepsilon$$

where the probability is taken over the coin tosses of $\mathcal{A}$ (and random choice of x).

The fastest algorithm known for the ECDLP is *Pollard's rho algorithm* which has an expected running time of $\sqrt{n\pi}/2$. For additional information on algorithms to solve the ECDLP refer to [12, 9].

To resist the most efficient known attacks on the ECDLP some important requirements that must be fulfilled are that $\sharp E(\mathbb{F}_q)$ be divisible by a sufficiently large prime $n > 2^{160}$ and that $\sharp E(\mathbb{F}_q) = hn$ with a small h. Other criteria for selecting appropriate curves are given in [22, 23, 11].

We now recall the intractability assumptions for Diffie-Hellmann problems over elliptic curve group structures. Let $\langle P \rangle$ be the cyclic subgroup of $E(\mathbb{F}_q)$ generated by P, then associated to n and P is the Elliptic Curve Diffie-Hellmann (ecdh) function defined by:

$$\mathsf{ecdh}_{n,P}(X, Y) = xyP \in \langle P \rangle$$

where $(X = xP, Y = yP) \in \langle P \rangle \times \langle P \rangle$ for unknown $x, y \in [1, n-1]$. The Diffie-Hellmann computational problem is based on the intractability of computing the function ecdh. More formally, we make the following:

ASSUMPTION 2 (ECCDHP). *Let E denote an elliptic curve defined over a finite field $\mathbb{F}_q$ and P a generator with prime order n. The Elliptic Curve Computational Diffie-Hellman problem is intractable if for every probabilistic polynomial time Turing Machine $\mathcal{A}$ the probability of success $(\mathsf{Succ}^{\mathsf{eccdh}}_{P,E(\mathbb{F}_q)}(\mathcal{A}))$ in computing $Q = rsP$ from $R, S \in \langle P \rangle$ is negligible:*

$$\mathsf{Succ}^{\mathsf{ecddh}}_{P,E(\mathbb{F}_q)}(\mathcal{A}) = \Pr\left[\begin{array}{c} r, s \xleftarrow{R} [1, n\text{-}1]; \\ R \leftarrow rP; S \leftarrow sP; Q \leftarrow rsP : \\ \mathcal{A}(n, P, R, S, Q) = \mathsf{ecdh}_{n,P}(R, S) \end{array}\right] \leq \varepsilon$$

where the probability is taken over the coin tosses of $\mathcal{A}$ (and random choices of r, s).

If $r, s \in [1, n-1]$ s.t. and $r = \mathsf{ecdl}^{-1}_{n,P}(R), s = \mathsf{ecdl}^{-1}_{n,P}(S)$, then, if an efficient algorithm exists for computing $\mathsf{ecdl}^{-1}_{n,P}(Y)$ (i.e. the ECDLP is tractable) then one must exist for solving the ECCDHP. However, the converse is not known to hold, i.e., knowledge of an algorithm for solving the ECCDHP does not imply the existence of one for solving the ECDLP.

ASSUMPTION 3 (ECDDHP). *Let E be an elliptic curve defined over a finite field $\mathbb{F}_q$ and P a generator with prime order n. The Elliptic Curve Decision Diffie-Hellman problem is intractable if for every probabilistic polynomial time Turing Machine $\mathcal{A}$ the probability of success $(\mathsf{Succ}^{\mathsf{ecddh}}_{P,E(\mathbb{F}_q)}(\mathcal{A}))$ in distinguishing the distributions (P, R, S, Q) and (P, R, S, T) is negligible:*

$$\mathsf{Succ}^{\mathsf{ecddh}}_{P,E(\mathbb{F}_q)}(\mathcal{A}) = \Pr\left[\begin{array}{c} r, s, t \xleftarrow{R} [1, n\text{-}1]; \\ R \leftarrow rP; S \leftarrow sP; \\ Q \leftarrow rsP; T \leftarrow tP; \\ b \xleftarrow{R} 0, 1; \\ (b = 0 ? U \leftarrow T : U \leftarrow Q) : \\ \mathcal{A}(n, P, R, S, U) = b \end{array}\right] < 1/2 + \varepsilon$$

where the probability is taken over the coin tosses of $\mathcal{A}$ (and random choices of r, s, t).

2.3 Security Requirements for Key Agreement Protocols

Generic two-party protocols participants (principals) are denoted by the letters A, B and are endowed with unique identities (id_A, id_B) drawn from the set $\mathsf{NAMESPACE} \subseteq \Sigma^{\ell}$ (say $\ell \geq 64$). For simplicity assume that the set $\mathsf{NAMESPACE}$ refers to a domain managed by the Certification Authority (CA) responsible for issuing certificates (which will include the identity information).

In [6] two basic key authentication models are defined for key agreement protocols: Authenticated Key (AK) agreement and AK with Key Confirmation (AKC). An AK protocol substantially defines a user-oriented goal and provides weak key authentication or Implicit Key Authentication (IKA), while the inclusion of a key confirmation phase (no key-related information must be revealed) accounts for strong key authentication known as Explicit Key Authentication (EKA):

Authenticated Key (AK). In an honest execution of an AK protocol an entity A is assured that no one aside from an identified principal B can possibly learn the value of a session key sk. Actually, A is not given concrete evidence that B has effectively computed the session key sk but there must be a mechanism for which a principal can effectively authenticate the other party participating in the protocol;

AK with Key Confirmation (AKC). A stronger goal is achieved by adding key confirmation to an AK protocol, i.e. in honest executions of an AKC protocol, principal A is assured that B (or vice versa) has actually computed (or knows how to compute) the session key. In this case a party receives assurance as to the binding of a session key to an authenticated protocol participant.

A principal A can initiate any number of protocol conversations with B, the i-th running instance initiated by A is represented by the symbol $\Pi^i_{A,B}$. Adversaries have complete control over the communication link, therefore they can interact with every protocol instance and are allowed to inject, modify, delete and replay (etc) message flows between communicating instances.

Every protocol instance $\Pi^i_{A,B}$ is characterised by key-related and state information whose protection, in practical protocol realizations, we consider a local implementation matter. Stored data includes input to the protocol; PIDs (Partner IDs) with the identity of conversation partners; SIDs (Session IDs) defined as the (concatenation) sequence of messages exchanged by communicating instances at a given time; private data such as the established session key sk; long-term keying material (asymmetric key pairs, symmetric keys, passwords, etc).

The set denoted by $\mathsf{SK} \subseteq \Sigma^\ell$, with $\ell \geq 128$, is the domain of definition for session keys sk; a good key agreement protocol should (ideally, in the presence of a passive adversary) output a 0-uniform random variable defined over SK.

Following work in [16] we define basic security attributes that are desirable for a generic key exchange protocol:

Known-Key Security (KK-S): For two honest principals A, B participating in any execution of the key agreement protocol, the associated instances $\Pi^i_{A,B}$ and $\Pi^j_{B,A}$ must terminate with a unique and matching session key sk. Furthermore, the above condition must hold even if session are revealed to an adversary (an event which may occur due to protocol flaws);

Forward Secrecy (FS): If the long-term private key of A and/or B are revealed to an adversary, then the secrecy of previously established session keys sk, in honest protocol executions, should not be affected;

Key-Compromise Impersonation Resilience (KCI-R): If an instance $\Pi^i_{A,B}$ is corrupted with loss of A's long-term private key, an adversary can impersonate A to any other principal. It is required that an adversary cannot masquerade to A as another principal;

Unknown Key-Share Resilience (UKS-R): A cannot be coerced into sharing a key with B without A's knowledge, i.e., when A believes the key is shared with some principal $E \neq B$, and B (correctly) believes the key is shared with A. The primary objective of the adversary need not be that of actually obtaining the session key. In public key based protocols, to avoid unknown key-share attacks where an adversary E simply chooses $W_E = W_A$ and attempts to obtain a valid certificate $cert_E$, the certification authority should actually verify whether the requester has knowledge of the private key before issuing a certificate (still, in some cases it may be insufficient to prevent this type of attack);

Key Control (KC): Neither A nor B can predetermine any portion of the session key sk being established;

Identity Assurance (IA): A principal must be able to authenticate the partner participating in any protocol execution. This may be achieved by linking the long-term static key and the identifier of a principal.

3. AK PROTOCOLS

In this section we briefly review some recently proposed key agreement protocols. In the sequel $\mathcal{G}(\cdot)$ represents a key derivation function, (w_U, W_U) a private/public key pair generated for a generic principal U and cert_U a digital certificate issued by a Certification Authority binding U's identity to the long-term public key W_U. In all protocol descriptions we assume that public keys are exchanged via certificates.

3.1 MTI/A0 Protocol

The MTI/A0 key agreement protocol described in Figure 1 was proposed by Matsumoto *et al.* [19]. The conjectured security attributes are KK-S, KCI-R.

The key is derived from the expression $h(r_A w_B + r_B w_A)P$. The protocol does not possess the FS attribute since the session key can be derived from the equivalent expression $h(w_B Q_A + w_A Q_B)$ and it suffices that both w_A, w_B are revealed given that Q_A, Q_B are publicly known. Also, the UKS-R attribute is not possessed by this protocol (see [16]).

3.2 UM Protocol

The Unified Model key agreement protocol described in Figure 2 was proposed by Ankney *et al.* [2, 13]. The conjectured security attributes are KK-S, FS, UKS-R. The key is derived from the expression $(w_A w_B)P \| (r_A r_B)P$. The protocol does not afford the KCI-R attribute since knowledge of either of the long-term static keys w_A, w_B is sufficient to compute ZS_A/ZS_B).

3.3 MQV Protocol

The MQV key agreement protocol described in Figure 3 was proposed by Law *et al.* [16]. The conjectured security attributes are KK-S, FS, KCI-R.

In this protocol $f = \lfloor \log_2 n \rfloor + 1$; if Q is a finite elliptic curve point and $\overline{x}$ denotes the integer obtained from the binary representation of $Q.x$, then $\overline{Q}$ is the integer given by $(\overline{x} \bmod 2^{\lceil f/2 \rceil}) + 2^{\lceil f/2 \rceil}$. The key is derived from the expression $h(r_A r_B + r_A w_B \overline{Q}_B + r_B w_A \overline{Q}_A + w_A w_B \overline{Q}_A \overline{Q}_B)P$ which in turn is derived (by A) from $hs_A(Q_B + \overline{Q}_B W_B)$.

A particular feature of this protocol is due to the factor $Q_B + \overline{Q}_B W_B$ which allows efficient scalar multiplications; however, it is computed from publicly available information (Q_B, W_B) and this fact has been exploited to mount successful unknown key share attacks [15]. Furthermore, practical cryptanalysis of this scheme has recently been described in [17].

3.4 LLK Protocol

The LLK key agreement protocol described in Figure 4 was proposed by Lee *et al.* [18]. The conjectured security attributes are KK-S, FS, KCI-R, UKS-R. The key is derived from the expression $h(r_A w_B + r_B w_A)P$.

3.5 SK Protocol

The SK key agreement protocol described in Figure 5 was proposed by Song *et al.* [25]. The conjectured security attributes are KK-S, FS, KCI-R, UKS-R. The key is derived from the expression $h(r_A w_B + r_B w_A + r_A r_B)P$.

3.6 SSEB Protocol

The SSEB key agreement protocol described in Figure 6 was proposed by Al-Sultan *et al.* [1]. The conjectured security attributes are KK-S, FS, KCI-R, UKS-R.

The key is derived from the expression $h(r_A r_B + w_A w_B)P$. Inversion of field elements can be avoided if computation of sk (say for A) is modified as $h(r_A Q_B + w_A W_B)$ with $Q_B = r_A P$.

Care must be taken with this protocol (and others) to avoid attacks containing degenerate values of short-term private keys. To illustrate this, an adversary with knowledge of w_B responds to message Q_A with $Q_E = P$ (E sets $r_E = 1$) and thus trivially defeats the KCI-R attribute by computing a valid session key from $T_A = r_A Q_E + w_A W_B = Q_A + w_E W_A (= T_E)$ since Q_A, W_A are publicly available data. As a countermeasure, A should verify that $Q_E \neq P$ (or that $nQ_E = P_\infty$).

4. ECKE-1 PROTOCOL

4.1 Protocol Specification

A high level description of the two-pass (1-round) elliptic curve AK agreement protocol ECKE-1 is depicted in Figure 7. It is assumed that the domain parameters EC_D are verified either by an explicit validation procedure [3], by certificates in a preliminary one time setup phase or by use of

A		B
$r_A \xleftarrow{R} [1, n-1]$		$r_B \xleftarrow{R} [1, n-1]$
$Q_A \leftarrow r_A P$		$Q_B \leftarrow r_B P$
	$\xrightarrow{Q_A}$	
	$\xleftarrow{Q_B}$	
$T_A \leftarrow h(r_A W_B + w_A Q_B)$		$T_B \leftarrow h(r_B W_A + w_B Q_A)$
$sk \leftarrow \mathcal{G}(T_A.x)$		$sk \leftarrow \mathcal{G}(T_B.x)$

Figure 1: Protocol MTI/A0

A		B
$r_A \xleftarrow{R} [1, n-1]$		$r_B \xleftarrow{R} [1, n-1]$
$Q_A \leftarrow r_A P$		$Q_B \leftarrow r_B P$
	$\xrightarrow{Q_A}$	
	$\xleftarrow{Q_B}$	
$ZS_A \leftarrow w_A W_B$		$ZS_B \leftarrow w_B W_A$
$ZE_A \leftarrow r_A Q_B$		$ZE_B \leftarrow r_B Q_A$
$sk \leftarrow \mathcal{G}(ZS_A \| ZE_A)$		$sk \leftarrow \mathcal{G}(ZS_B \| ZE_B)$

Figure 2: Protocol UM

A		B
$r_A \xleftarrow{R} [1, n-1]$		$r_B \xleftarrow{R} [1, n-1]$
$Q_A \leftarrow r_A P$		$Q_B \leftarrow r_B P$
	$\xrightarrow{Q_A}$	
	$\xleftarrow{Q_B}$	
$s_A \leftarrow r_A + \overline{Q}_A w_A$		$s_B \leftarrow r_B + \overline{Q}_B w_B$
$T_A \leftarrow hs_A(Q_B + \overline{Q}_B W_B)$		$T_B \leftarrow hs_B(Q_A + \overline{Q}_A W_A)$
$sk \leftarrow \mathcal{G}(T_A.x)$		$sk \leftarrow \mathcal{G}(T_B.x)$

Figure 3: Protocol MQV

A		B
$r_A \xleftarrow{R} [1, n-1]$		$r_B \xleftarrow{R} [1, n-1]$
$Q_A \leftarrow h r_A W_B$		
	$\xrightarrow{Q_A}$	
		$R_B \leftarrow h r_B W_A$
		$T_B \leftarrow Q_A + R_B$
		$Q_B \leftarrow R_B + r_B w_B^{-1} Q_A$
	$\xleftarrow{Q_B}$	
$T_A \leftarrow Q_A + (w_A(w_A + r_A)^{-1}) Q_B$		
$sk \leftarrow \mathcal{G}(T_A.x)$		$sk \leftarrow \mathcal{G}(T_B.x)$

Figure 4: Protocol LLK

A		B
$r_A \xleftarrow{R} [1, n-1]$		$r_B \xleftarrow{R} [1, n-1]$
$Q_A \leftarrow r_A P$		$Q_B \leftarrow r_B P$
	$\xrightarrow{Q_A}$	
	$\xleftarrow{Q_B}$	
$T_A \leftarrow h(r_A W_B) + h(w_A + r_A) Q_B$		$T_B \leftarrow h(r_B W_A) + h(w_B + r_B) Q_A$
$sk \leftarrow \mathcal{G}(T_A.x)$		$sk \leftarrow \mathcal{G}(T_B.x)$

Figure 5: Protocol SK

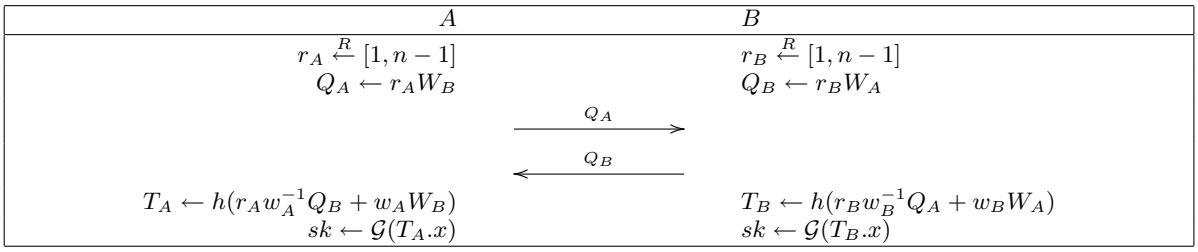

Figure 6: Protocol SSEB

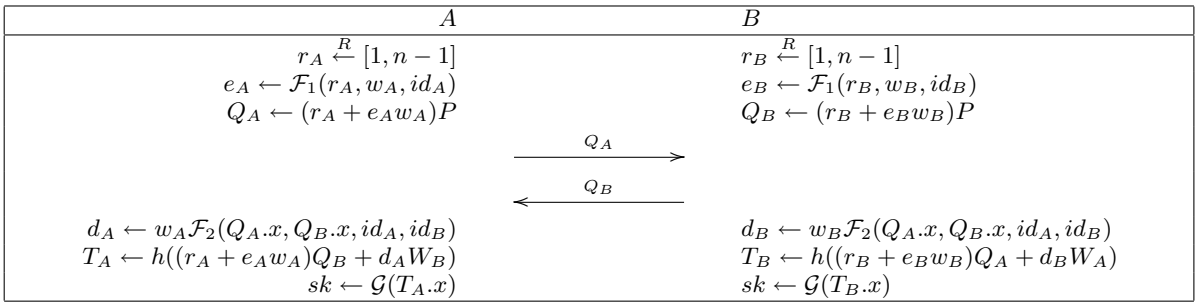

Figure 7: Protocol ECKE-1.

a cryptographic device (e.g. smart card) issued by a trusted key distribution center.

We define two independent (collision resistant) hash functions, namely $\mathcal{F}_1, \mathcal{F}_2 : \Sigma^* \rightarrow \mathbb{F}_q$ and also (*key derivation function*) $\mathcal{G} : \mathbb{F}_q \rightarrow \Sigma^*$.

The main protocol actions are as follows:

1. A selects a random r_A in $[1, n-1]$ and computes e_A from the 3-tuple (r_A, w_A, id_A). B selects a random r_B in $[1, n-1]$ and computes e_B from the 3-tuple (r_B, w_B, id_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. Both A and B terminate holding the session key $sk \in \mathsf{SK}$.

If the 3-tuples (r_A, e_A, Q_A), (r_B, e_B, Q_B) can be precomputed, on-line work, for each entity, is reduced to exactly two scalar multiplications and one hash function computation.

The key derivation function $\mathcal{G}(\cdot)$ is used to derive a session key sk from $T_A.x$ ($\equiv T_B.x$) and also to protect against prediction of weak bits. Standard key derivation functions are specified in [13].

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 secret key sk from $h(r_A r_B + r_B e_A w_B + r_A e_B w_A + e_A e_B w_A w_B + d_A d_B w_A w_B)P$.

4.2 Security Analysis

In the following discussion hash and key derivation functions will be modeled as random oracles [4]). Under this hypothesis, in the presence of a passive adversary (and honest principals), any run of the protocol can be easily seen to generate a uniformly distributed session key (in SK).

Any principal A with access to an authentic copy of $cert_B$ can verify B's identity contained therein (id_B) in a straightforward manner. However, A should verify B's identity (on-line) during any protocol run to be assured that she will actually share a session key only with B (In practice, this would require access to a CRL maintained by the CA). A weak form of identity assurance (IA) may be obtained through construction of the expression $r_A + e_A w_A$ (recall that we assume $\mathcal{F}_1$ is a collision resistant hash function) binding id_A to r_A, w_A. No direct verification procedure is available to B, nonetheless, the availability of an authentic certificate $cert_A$ would prove that id_A is indeed A's identity and that it is linked to w_A. In addition, since only B has knowledge of the private keys w_B, r_B it follows that the protocol must possess the IKA attribute. Conversely, no key authenticator message (confirming that the other principal has computed a valid session key) is exchanged in the message flow therefore the protocol does not possess the EKA attribute. Key confirmation can be added to the protocol using, for example, the compiler of [6]. Specifically, a one-time key (ek) is derived from T_A using a key derivation function (different from $\mathcal{G}(\cdot)$) and used to construct mutual authenticators with a (secure) keyed message authentication scheme (MAC).

We now examine protocol ECKE-1 in relation to the security attributes defined in section 2; informal arguments are provided to support our claims:

$\mathsf{KK\text{-}S}$: Every session key sk is uniquely derived from the random ephemeral keys r_A, r_B and the static private keys w_A, w_B (through construction of $e_A, e_B, d_A d_B$). Therefore, an adversary with known session keys but no information

on r_A, r_B, w_A, w_B, has a negligible probability of mounting a successful attack against any run of the protocol since extracting this data requires (at least) tractability of the Elliptic Curve discrete logarithm problem (ECDLP).

FS: For any adversary with knowledge of the long-term private key w_A (and/or w_B), to derive a session key r_A, r_B are required. Furthermore, d_A, d_B, e_A, e_B are random values (in the random oracle assumption) and the session key also depends on the Diffie-Hellmann secret $r_A r_B P$. It is computationally infeasible to recover these values assuming the intractability of the Elliptic Curve Diffie-Hellmann problems.

KCI-R: If an adversary comes into possession of A's static private key w_A the protocol will be exposed to attacks through impersonation (of A). An adversary (impersonating E) to compute a valid session key (shared with A) must solve a non linear equation, in r_A derived from the condition $(r_A+e_A w_A)Q_E+d_A W_E = T_E$ where $T_E = (r_E+e_E w_E)Q_A + d_E W_A$ and $Q_E = (r_E + e_E w_E)P$. Here we assume that the adversary knows a valid key pair (w_E, W_E) (for the adversary without this information the problem is harder).

UKS-R: An adversary masquerading as E cannot deceive A into believing that messages received from E are originated from B. Indeed, the adversary cannot use $cert_E = cert_B$ with id_E in any protocol run because A can easily verify if $id_E \in cert_E$ (given that a CA verifies possession of private keys prior to issuing a certificate). Furthermore, even if the adversary could obtain a valid certificate $cert_E$, for some (w_E, W_E), the attack would fail since session keys are generated from values (d_A) that bind identities (id_A), extracted from valid certificates, to exchanged messages (Q_A). Observe also that the attack can be prevented by adding key confirmation to the protocol.

KC: The principal acting as the responder (say B) in Diffie-Hellmann based key agreement protocols has a potential advantage in that he may choose a predictable ephemeral private key (r_B) [20]. However, the ephemeral public keys exchanged by principals (Q_A, Q_B) are built on non linear expressions in $r_A, r_B, e_A, e_B, w_A, w_B$ where e_A, e_B are random (since $\mathcal{F}_2$ is a random oracle), thus any responder in practice can achieve *partial* or *imperfect* key control (IKC) (Note that a commitment mechanism for short-term private keys could be added to the protocol to counter such threats at the cost of increased communication complexity).

5. IMPLEMENTATION ISSUES

The efficiency of computations in elliptic curve based cryptographic schemes is mainly influenced by the efficiency of point multiplications which in turn depend on the arithmetic of the underlying finite field.

The IEEE standards [13, 14] restrict the underlying finite field to $\mathbb{F}_q$ where q is either an odd prime $(q = p, p > 3)$ or a power of 2 $(q = 2^m)$ and allow extension fields $(q = p^m)$ of odd characteristic including OEFs (Optimal Extension Fields). A basic security recommendation is that the order of n of the generator point P should be greater than 2^{160}.

The arithmetic operations of $\mathbb{F}_p$ yield significantly slower algorithms with respect to those in $\mathbb{F}_{2^m}$ since the latter does not suffer from carry-dependent inefficiencies and leads to hardware implementations with less area occupation. Furthermore, the choice of the representation basis and of the irreducible polynomials are important factors in the achievement of performance efficiency.

Some properties often used to evaluate protocol performance are the following:

Round (Pass) Complexity: the total number of messages exchanged in a protocol execution. The AK protocols in this paper achieve minimum (one) round complexity;

Communication Complexity: the total number of bits exchanged in a protocol execution (bandwidth required by messages). Elliptic curve based techniques can lead to more efficient implementations since they use considerably shorter key sizes and offer security levels analogous to other traditional cryptosystems. Indeed, according to [21] for EC and RSA based cryptosystems it is possible to conceive of public key size ratios approximately equal to 1:12, 1:20 and 1:30 considering (EC) keys of, respectively, 256, 384 and 512 bits;

Message Independence: the messages exchanged by the principals involved in a protocol execution are unrelated (non-interactive). It is straight forward to verify that all protocols treated here possess this property;

Computation Efficiency: the group operations that dominate the processing time of each entity participating in a protocol execution. In an elliptic curve setting these operations are point multiplications. Protocol ECKE-1 requires 3 point multiplications per entity, this figure can be reduced by one if pre-computation is possible (cfr § 4.1).

Performance figures (in terms of operation count per single user) of the one-round AK protocols treated in this paper are summarised in Table 1. The first two columns contain the number of scalar multiplications, respectively, without (SM) and with precomputations $(SM-pre)$. Column three refers to hash computations, key derivation functions are not considered since they apply to all protocols (Note that one hash computation can occur off-line). Finally, column four counts field inversions.

Table 1: Computational efficiency comparison

$\downarrow Prot./Op\rightarrow$	*SM*	*SM-pre*	*Hash*	*Fld-inv*
MTI/A0	3	2	0	0
UM	3	2	0	0
MQV	2.5	1.5	0	0
LLK	2	1	0	1
SK	3	2	0	0
SSEB	3	2	0	1
ECKE-1	3	2	2	0

Standard documents [13] recommend the use of well known conventional cryptographic hash functions such as SHA-1, RIPEMD-160, etc. In general, it is hard to design secure practical hash functions and although they are completely characterised in formal security models by well known techniques [4] results do not immediately translate to the real world [7]. Nonetheless, hash based cryptosystems can turn out to be very efficient schemes.

The conjectured security attributes of all the protocols are summarised in Table 2. It is easily verified that they all possess the IKA attribute but not the EKA attribute.

6. CONCLUSIONS AND FUTURE WORK

We have proposed a new AK protocol ECKE-1 and compared it with other known elliptic curve public-key based protocols. Elliptic curve techniques may result in smaller

Table 2: Conjectured security attributes

↓Prot./Attr.→	*KK-S*	*FS*	*KCI-R*	*UKS-R*	*IKC*
MTI/A0	yes	no	yes	no	no
UM	yes	yes	no	yes	yes
MQV	yes	yes	yes	no	no
LLK	yes	yes	yes	yes	yes
SK	yes	yes	yes	yes	no
SSEB	yes	yes	yes	yes	no
ECKE-1	yes	yes	yes	yes	yes

system parameters, low-power requirements, bandwidth efficient and faster implementations. All protocols are of comparable efficiency.

Security claims are based on elliptic curve instances of Diffie-Hellman assumptions. The ECKE-1 protocol uses hash functions as a main building block, this allows important security attributes to be fulfilled (albeit in the random oracle model).

Future work includes the actual implementation of a working prototype of protocol ECKE-1 to provide performance figures and to evaluate the effective (real-world) security features conjectured in this paper. Rationale for the conjectured security attributes is based on heuristic arguments, for this reason we also plan to perform a more effective (provable) security analysis in a formal (computational) model of security. In addition, implementations with detailed performance and security analyses will be provided for all the protocols reviewed in this paper.

7. ACKNOWLEDGEMENTS

The author is grateful to the anonymous reviewers for their helpful comments and suggestions.

8. REFERENCES

[1] K. Al-Sultan, M. Saeb, M. Elmessiery, and U.A.Badawi. A new two-pass key agreement protocol. *Proceedings of the IEEE Midwest 2003 Symp. on Circuits, Systems and Computers*, 2003.

[2] R. Ankney, D. Hohnson, and M. Matyas. The unified model. *Contribution to X9F1*, 1995.

[3] ANSI-X9.62-1998. Public key cryptography for the financial services: The elliptic curve digital signature algorithm (ECDSA). American National Standards Institute, 1999.

[4] M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. *In 1st Conference on Computer and Communications Security*, pages 62–73, 1993.

[5] M. Bellare and P. Rogaway. Entity authentication and key distribution. *In Proceedings of CRYPTO 1993*, LNCS 773:232–249, 1994.

[6] S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. *In Proceedings of the 6th IMA Int.l Conf on Cryptography and Coding*, LNCS 1355:30–45, 1997.

[7] R. Canetti, O. Goldreich, and S. Halevi. The random oracle methodology, revisited. *In 30th STOC*, pages 209–218, 1998.

[8] R. Canetti and H. Krawczyk. Analysis of key exchange protocols and their use for building secure channels. *Advances in Cryptology-EUROCRYPT 2001*, pages 453–470, 2001.

[9] Certicom. ECC challenge. http://www.certicom.com/resources/ecc_chall/challenge.html, 1997.

[10] W. Diffie and M. Hellmann. New directions in cryptography. *IEEE Transactions in Information Theory*, pages 644–654, 1976.

[11] FIPS-186-2. Digital signature standard. American National Standards Institute, 2000.

[12] D. Hankerson, A. Menezes, and S. Vanstone. *Guide to Elliptic Curve Cryptography*. Springer Professional Computing, New York, 2004.

[13] IEEE-P1363-2000. Standard specifications for public key cryptography. Institute of Electrical and Electronics Engineers, 2000.

[14] IEEE-P1363a/D12. Draft standard specifications for public key cryptography-amendment 1: Additional techniques. Institute of Electrical and Electronics Engineers, 2003.

[15] B. Kaliski. An unknown key share attack on the MQV key agreement protocol. *ACM Transactions on Information and System Security*, pages 36–49, 2001.

[16] L. Law, A. Menezes, M. Qu, J. Solinas, and S. Vanstone. An efficient protocol for authenticated key agreement. *Designs, Codes and Cryptography*, pages 28:119–134, 2003.

[17] P. Leadbitter and N. Smart. Analysis of the insecurity of ECMQV with partially known nonces. *Proceedings ISC 2003*, LNCS 2851:240–251, 2003.

[18] C. Lee, J. Lim, and J. Kim. An efficient and secure key agreement. *IEEE p1363a draft*, 1998.

[19] T. Matsumoto, Y. Takashima, and H. Imai. On seeking smart public-key distribution systems. *Transactions of IEICE*, VolE69:99–106, 1986.

[20] C. Mitchell, M. Ward, and P. Wilson. Key control in key agreement protocols. *Electronic Letters*, Vol. 34:980–981, 1998.

[21] NIST-SP800-57. Recommendation for key management. part 1: General guideline. National Institute of Standards and Technology, 2003.

[22] SEC1. Elliptic curve cryptography - version 1.0. Standards for Efficient Cryptography Group, 2000.

[23] SEC2. Recommended elliptic curve domain parameters - version 1.0. Standards for Efficient Cryptography Group, 2000.

[24] V. Shoup. On formal models for secure key exchange. Technical Report RZ 3120, IBM Research, 1999.

[25] B. Song and K. Kim. Two-pass authenticated key agreement protocol with key confirmation. *Progress in Cryptology - Indocrypt 2000*, LNCS 1977:237–249, 2000.