Modern Key Agreement Techniques*

Rainer A. Rueppel† Paul C. van Oorschot‡

29 April 1994

Abstract

We present a survey of modern key agreement techniques, and discuss distinguishing characteristics, including identity (entity) authentication, implicit key authentication, key confirmation, and key freshness.

1 Introduction

The security of all cryptographic mechanisms depends upon the proper management of keying material, and thus the management of cryptographic keys is of central importance. *Key establishment* mechanisms seek to make a secret key available to two (or more) authorized parties for subsequent cryptographic use. These can be broken down in two broad categories: *key transfer* mechanisms, in which a key is created by one party, and securely transmitted to another; and *key agreement* mechanisms, in which two parties contribute information which jointly establishes a shared secret key. In this paper, we consider only the latter. We survey modern key agreement mechanisms which employ public-key techniques and do not require the use of an online server or trusted third party.

This paper is organized as follows. In Section 2 we discuss in general various properties and characteristics that differentiate key agreement protocols. In Sections 3, 4 and 6, respectively, we discuss the basic two-pass Diffie-Hellman key agreement protocol [3], ElGamal's one-pass variation of it [5], and a three-pass variation based on ISO 9798-3 [8]. Section 5 considers the one-pass key agreement mechanism of Nyberg and Rueppel [11]. Section 7 examines several protocols of Matsumoto et al. [10]. Section 8 discusses protocols based on self-certifying public keys of Girault [6]. In Section 9 we consider a protocol based on zero-knowledge techniques, originating from work by Brandt et al. [1]. Section 10 considers identity-based public key schemes and implicitly certified public keys used in key agreement algorithms, and examines the requirements for trust in third parties in such schemes, following ideas of Günther [7], Girault [6] and Nyberg-Rueppel [11]. Many of the protocols discussed are in the process of becoming standardized in ISO/IEC CD 11770-3 [9].

2 Properties of Key Agreement Protocols

Key agreement mechanisms can be categorized by many criteria. We focus on protocols involving two parties, without the requirement of an online trusted third party. An obvious distinguishing characteristic is the number of message exchanges required between the parties involved in the protocol, called *passes*. We consider one-pass, two-pass and three-pass such techniques; protocols with four or more passes exist, but are often both undesirable and avoidable. Another important property is the *freshness* of the derived key – whether the key is new (fresh) for this protocol. If not, the protocol may be subject to replayed messages resulting in the reuse of an old key from a previous protocol run, potentially to the advantage of an adversary. The freshness of a derived key may be influenced or controlled by one or both parties;

*A version of this paper will appear in *Computer Communications*, Special Issue on Security (July 1994)
†$\mathcal{R}^3$ Security Engineering AG, CH-8607 Aathal, Switzerland. email: rueppel@r3.ch
‡Bell-Northern Research, P.O. Box 3511 Station C, Ottawa, Canada K1Y 4H7; and until July 31, 1994: School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6. email: paulv@bnr.ca.

the influence of a particular party on the value of a jointly derived key may be dynamic (based on a per-session parameter) or static (based on a longer-term or fixed parameter).

Another distinguishing feature is the *type* or *degree* of authentication that results. The basic objective of all two-party key agreement techniques is that two parties jointly derive a shared key, suitable for use in subsequent cryptographic communications with each other, in such a manner that no (potentially malicious) third party be able to deduce the same key. Protocols which satisfy this objective may be said to provide "secure key establishment". However, as just described, this concept is somewhat imprecise. It is clearly important that each party know or learn the true *identity* of the other party sharing the key, and it is usually assumed that secure key establishment involves such definite identification; this should be addressed explicitly. Furthermore, secure key establishment as described above admits two possibilities: while a second party is presumed by a first to be the only possible party who could feasibly derive the joint key, that party may or may not *actually* possess the key, as far as the first party knows.

The feature of *entity (identity) authentication* provides assurance, in the form of some positive corroboration, of the identity of another party in a protocol; typically it also establishes that that party is *live* (or active) at the time the protocol is executing. Some key agreement protocols provide *key confirmation* – corroboration that another party actually knows the jointly established key, involving one party receiving a message from another demonstrating the latter possesses that key. Such demonstration might be via a message authentication code (MAC), encryption of a known quantity, one-way hash value on the key, or by a zero knowledge technique. Key confirmation establishes a type of mutual belief in the shared secret. Key confirmation seems useful only if key freshness is guaranteed.

Some protocols provide (only) *implicit* key authentication – namely assurance to one party that no one aside from a specifically identified second party could possibly have acquired the shared key, although possession of such key by such party is not actually known. (As opposed to implicit key authentication, note that "implicit entity authentication" is a meaningless concept in this terminology, as entity authentication is by our definition explicit, i.e. based on positive corroboration.) Entity authentication, implicit key authentication, and key confirmation may be provided to both parties (*mutual*) or to only one party (*unilateral*). An intermediate property between key confirmation and secure key establishment via implicit key authentication is possible, wherein one party claims to possess the joint key, without actually demonstrating such possession. However, such claimed possession of key does not appear useful in practice, and is not considered further here.

In the public key agreement techniques we consider, three types of public keys are potentially employed: public *encryption keys* (e.g. RSA keys [12], typically used for key transport); public *signature keys* (used for verifying digital signatures); and public *key-agreement keys* (e.g. Difie-Hellman exponentials used for Diffie-Hellman key agreement). In some cases, the public keys are made available to other parties through use of public key *certificates*, in which case further distinction can be made based on the nature of the certificates. Certificates bind a user identity to a public key, in some manner by which other parties can verify the authenticity of the public key. *Conventional* public-key certificates, for example as used in X.509 [2], use digital signatures of a trusted third party to achieve verifiability. *Identity-based certificates* and *self-certifying public keys*, as proposed by Günther [7]) and Girault [6], provide alternatives; both are methods which make available copies of implicitly authenticated public keys.

The type of public key certificate used implies certain requirements regarding the degree of trust required in a trusted third party. Identity-based systems, originally proposed by Shamir [13], involved a trusted third party T providing a (public, private) key pair (K_U, ${K_U}^{-1}$) to each user U. The public key K_U is simply U's identification information (e.g. name and address). The private key ${K_U}^{-1}$, computed by T and given to U, is a function of U's identification information and some secret information known only to T (e.g. T's private key of a public key pair); thus the third party T must be trusted not to disclose U's private key. Self-certifying public keys [6] are similar, but each user creates its own private key ${K_U}^{-1}$, and T computes the public key K_U from U's identifying information, a one-way function of ${K_U}^{-1}$, and some secret information known only to T. In this case, the trusted party does not learn a user's secret key itself (but only a one-way function of it).

Finally, we discuss a more subtle point regarding authentication. Distinction can be made regarding whether or not one party, say the first, in an authentication protocol, has as an *a priori* intention the objective of authenticating itself to and/or exchanging a key with only a specific second party, or is willing to do so with any second party. In one case, the service is identity *corroboration*; in the other, the service is both *extraction and corroboration* of the identity of an unidentified party. In the latter case, once the identity becomes known as a protocol outcome, that information could be used to decide whether to proceed to use the derived key. For example, in mutual authentication, the initiator typically has a specific second party in mind, but the responder is usually willing to carry out the protocol with any (possibly unidentified) initiator, the responders' intended target being no more specific than the (possibly unidentified) party who initiated. The question of the intended protocol objective may be clouded by the fact that in authenticated key agreement protocols which specify a target party, it is often unclear whether targeting originated as a technical detail necessary for the integrity of identity corroboration, or was an original design goal.

Key agreement protocols may thus differ in many characteristics. A summary of the distinguishing features is given by the following list of potential properties, each of which may be considered from the point of view of either party in the protocol.

1. number of passes (one; two; three)
2. entity/identity authentication (mutual; unilateral; none)
3. key confirmation (mutual; unilateral; none)
4. implicit key authentication (yes; no)
5. key freshness (mutual control; unilateral control; replay not controlled)
6. public key certificate type used (conventional; identity-based; self-certifying; none)

3 Basic Diffie-Hellman Protocol

The Diffie-Hellman key agreement protocol [3] establishes in two passes a shared secret key between entities A and B without prior exchange of keying information. However, neither entity can be sure about the other's identity - that is, the mechanism provides neither implicit key authentication nor (explicit) entity authentication.

Let $GF(p)$ denote the Galois field with p elements, and let g be a primitive element of the field. The protocol runs as follows:

1. A randomly and secretly chooses $r_A \in \{1, \ldots, p-2\}$, computes the key token
$$g^{r_A} \bmod p$$
and sends it to B.
2. B randomly and secretly generates $r_B \in \{1, \ldots, p-2\}$, computes the key token
$$g^{r_B} \bmod p$$
and sends it to A.
3. A computes the shared key as
$$K_{AB} = (g^{r_B})^{r_A} = g^{r_A r_B} \bmod p$$
4. B computes the shared key as
$$K_{AB} = (g^{r_A})^{r_B} = g^{r_A r_B} \bmod p$$

A generic version of the DH-protocol is described in ISO CD 11770-3 [9], where a generic commutative function $F(h, g)$ is used in place of modular exponentiation as above. More specifically, $F(h, g)$ is an appropriate function conjectured to be a one-way function of h, and commutative with respect to h under composition, i.e. $F(h_A, F(h_B, g)) = F(h_B, F(h_A, g))$.

The Diffie-Hellman key agreement mechanism has the following properties. The protocol requires two passes. The protocol does not provide entity authentication or implicit key authentication. However, it may be useful in environments where the authenticity of the key tokens is verified using other means. For instance, a hash code of the resulting key may be exchanged between the entities using a second communication channel, which offers authenticity. The mechanism provides no key confirmation and does not protect against replay of the key tokens. Each entity can assure freshness of the key, since each of them provides one of the inputs of the shared key generation. We assume here that each party checks that the other's exponential is not degenerate, i.e. the other party did not choose (or an adversary did alter messages such that) $r_A = 0$ or $r_B = 0$. By the one-wayness of discrete exponentiation, neither party can control the value of the key in advance.

A simple variation of the basic Diffie-Hellman scheme allows mutual authentication. Rather than using per-session random numbers, consider fixing g^{r_A} and g^{r_B} as the long-term public keys of A and B, respectively, and allowing for their authenticity to be verifiable (e.g. via public key certificates). This allows a zero-pass key agreement mechanism (if we assume public keys are available *a priori*) with mutual implicit key authentication. Unfortunately, the derived key $K = g^{r_A r_B}$ is time-invariant for the user pair (A, B), and as such would be best not used as a session key itself. One might consider using this K in a symmetric cryptographic technique, such as an encryption function E or a keyed hash function H, to create a time-variant session key: for example, A might choose a random per-session number r, send to B $E_K(r)$, and use r as the session key; or simply send r cleartext to B, and use $H_K(r)$ as a session key. Both of these one-pass schemes, however, suffer from the same disadvantage - subsequent compromise of either r_A or r_B compromises the master key K, which is catastrophic as it potentially exposes all past session keys.

4 ElGamal Variant of DH-Protocol

ElGamal [5] proposed a slight variation of the basic DH-protocol. This variant establishes in one pass a shared secret key between A and B with implicit key authentication from B to A, but no authentication from A to B (i.e. B does not know with whom it has established the shared secret key). The protocol offers no entity authentication. Again a finite field $GF(p)$ and primitive element g are fixed. It is assumed that each entity X has a private key agreement key h_X and a corresponding public key $y_X = g^{h_X} \bmod p$. Moreover, each entity has access to authenticated copies of the public key agreement keys of the other entities. The protocol runs as follows:

1. A randomly and secretly chooses $r_A \in \{1, \ldots, p-2\}$, computes the key token
$$g^{r_A} \bmod p$$
and sends it to B.

2. A computes the shared key as
$$K_{AB} = (y_B)^{r_A} \bmod p = g^{h_B r_A} \bmod p$$

3. B computes the shared key as
$$K_{AB} = (g^{r_A})^{h_B} = g^{h_B r_A} \bmod p$$

The ElGamal variant of the DH-protocol has the following properties. It requires one pass (assuming other party's public key is available *a priori*). A generic version of the protocol is described in ISO CD 11770-3 [9]. The mechanism provides implicit key authentication of B (B is the only entity other than A who can compute the shared secret key), but provides no key confirmation. The mechanism does not prevent the replay of the key token. An opponent may attempt to force B into using an old shared secret key. Note however that in the absence of a compromise, the key will only be known and in use between A and B. A can assure freshness of the key, since it is the entity supplying the random value r_A. However, no guarantee of K_{AB}'s freshness is given to B by the mechanism. A may precompute the key K_{AB} before it sends the key token to B. However, A cannot control the value of the key in advance (by the one-wayness of discrete exponentiation).

5 Nyberg-Rueppel Key Agreement

The basic Diffie-Hellman key exchange protocol requires two passes and does not give authentication. The ElGamal public key encryption system requires one pass but authenticates only the receiver. In this section we show that using the signature system of Nyberg and Rueppel [11], the ElGamal protocol can be extended to achieve mutual authentication in one pass. In this context, authentication of the receiver B is implicit key authentication - B is the only party that could possibly compute the secret key correctly from the public message received from the sender A. The authentication of A sender in the basic protocol is also implicit key authentication. Entity authentication can be provided if timestamps or sequence numbers are used in addition, but is not provided by the basic protocol outlined below.

Let p be a prime number, and let $q = p-1$ or a large integer factor of $p-1$; $\alpha \in GF(p)$ is an element of order q; $s_A \in Z_q$ is the private key of user A, with corresponding public key $k_A = \alpha^{-s_A} \bmod p$. It is assumed that each user has access to authenticated copies of the public keys of the other users. The protocol runs as follows:

1. User A initiates the key agreement protocol. A generates two secret random numbers r and $R \in Z_q$, computes

$$\begin{aligned} e &= \alpha^{R-r} \bmod p \\ y &= r + s_A e \bmod q, \end{aligned}$$

   and sends (e, y) to B.

2. A computes the shared secret key as

$$K = {k_B}^R \bmod p$$

3. Upon receiving (e, y) from A, user B computes the shared secret key

$$(\alpha^y k_A^e e)^{-s_B} \bmod p = \alpha^{-Rs_B} \bmod p = K.$$

The Nyberg-Rueppel key agreement has the following properties. It requires one pass (assuming other party's public key is available *a priori*). The basic mechanism provides mutual implicit key authentication, but no key confirmation. If a cryptographic hash code of the key is sent to B then the mechanism also provides unilateral key confirmation. If a timestamp or sequence number is included in the key token, then the mechanism provides (explicit) entity authentication of A to B through the signature of the key token, and prevents the replay of the key token. A may precompute the key K_{AB} before it sends the key token to B. However, A cannot control the value of the key in advance (by the one-wayness of discrete exponentiation). A can assure freshness of the key, since it is the entity supplying the random value r. But no guarantee of K_{AB}'s freshness is given to B. A generic version of the protocol is described in ISO CD 11770-3 [9].

6 Key Agreement using ISO 9798-3

In this section we describe a key agreement mechanism based on the three-pass authentication mechanism of ISO 9798-3 [8]. The mechanism described involves three passes, and establishes a shared secret key between A and B with mutual entity authentication. We assume the basic Diffie-Hellman setting with a prime p and exponentiation modulo p. In addition, it is assumed that each entity X has a personal asymmetric signature system (S_X, V_X), consisting of a signature transformation S_X and a signature verification transformation V_X. We also assume that each entity has access to authenticated copies of the public verification transformations V_X of all other entities. The key agreement protocol runs as follows:

A1 A randomly and secretly generates $r_A \in \{1, \ldots, p-2\}$, computes $g^{r_A} \bmod p$, constructs the key token

$$KT_{A1} = g^{r_A} \bmod p$$

and sends it to B.

B1 B randomly and secretly generates $r_B \in \{1, \dots, p-2\}$, computes $g^{r_B} \bmod p$, constructs the signed key token

$$KT_{B1} = S_B(g^{r_B} \bmod p, g^{r_A} \bmod p, A)$$

and sends it back to A.

A2 A verifies B's signature on the key token KT_{B1} using B's public verification transformation V_B, and checks that the received value $g^{r_A} \bmod p$ agrees with the one sent in step (A1). If the check is successful, A proceeds to compute the shared key as

$$K_{AB} = (g^{r_B})^{r_A} = g^{r_A r_B} \bmod p$$

Then A constructs and sends to B the signed key token

$$KT_{A2} = S_A(g^{r_A} \bmod p, g^{r_B} \bmod p, B)$$

B2 B verifies A's signature on the key token KT_{A2}, using A's public verification transformation V_A, and checks that the received value $g^{r_B} \bmod p$ agrees with the one sent in step (B1). If the check is successful, B proceeds to compute the shared key as

$$K_{AB} = (g^{r_A})^{r_B} = g^{r_A r_B} \bmod p$$

The Diffie-Hellman version of the three-pass ISO 9798-3 mechanism requires two types of algorithms: the Diffie-Hellman key agreement scheme and a public key signature scheme (such as RSA). To save implementation overhead, both schemes may be based on the same basic function. For instance, the Digital Signature Algorithm (DSA) and Diffie-Helman are both based on exponentiation mod p.

In its basic form, the mechanism provides mutual entity authentication, but no key confirmation. Using the shared secret key K_{AB} in a cryptographically secure way in KT_{B1} or KT_{A2} (e.g. by including a cryptographic hash value) may provide mutual key confirmation. Each entity can assure freshness of the key, since each of them provides one of the inputs of the shared key generation. The mechanism conforms with ISO 9798-3 [8] – KT_{A1}, KT_{B1}, and KT_{A2} are identical to the tokens sent in the three pass authentication mechanism described in Clause 5.2.2 of ISO 9798-3, when the data fields are used as follows: (a) the data field R_A (which is present in all three tokens of ISO 9798-3, Clause 5.2.2) transmits the random function value $g^{r_A} \bmod p$; (b) the data field R_B (which is present in all three tokens of ISO 9798-3, Clause 5.2.2) transmits the random function value $g^{r_B} \bmod p$. The complete protocol is described in ISO CD 11770-3 [9]. Further discussion of properties of this protocol may be found in Diffie et al. [4], where the protocol, modified to include key confirmation, is called the Station-to-Station (STS) protocol.

7 Key Agreement Mechanisms of Matsumoto et al.

In this section we consider several two-pass key agreement protocols discussed by Matsumoto et al. [10]. All provide mutual implicit key authentication, and are role-symmetric (i.e. both parties carry out analogous actions).

As before, let g be a primitive element for $GF(p)$. To describe a key agreement between parties be A and B, let s_A denote A's long-term private key and $y_A = g^{s_A}$ be A's long-term public key; as above, it is assumed that each party has access to authenticated copies of the public keys of other users prior to the start of the protocol. r_A is a per-session random number chosen by A, and z_{AB} is the value A sends to B in the protocol. Corresponding definitions are made for B, and the resulting key agreement key is denoted K. A selection of two-pass key agreement schemes examined by Matsumoto et al. is presented in the following table.

Table 1. Description of key agreement protocols of Matsumoto

Scheme	z_{AB}	z_{BA}	K computed by A	K computed by B	K
A(0)	g^{r_A}	g^{r_B}	${z_{BA}}^{s_A} {y_B}^{r_A}$	${z_{AB}}^{s_B} {y_A}^{r_B}$	$g^{s_B r_A + s_A r_B}$
B(0)	${y_B}^{r_A}$	${y_A}^{r_B}$	${z_{BA}}^{{s_A}^{-1}} g^{r_A}$	${z_{AB}}^{{s_B}^{-1}} g^{r_B}$	$g^{r_A + r_B}$
C(0)	${y_B}^{r_A}$	${y_A}^{r_B}$	${z_{BA}}^{{s_A}^{-1} r_A}$	${z_{AB}}^{{s_B}^{-1} r_B}$	$g^{r_A r_B}$
C(1)	${y_B}^{r_A s_A}$	${y_A}^{r_B s_B}$	${z_{BA}}^{r_A}$	${z_{AB}}^{r_B}$	$g^{s_A s_B r_A r_B}$

Scheme A(0) has the advantage that it requires no additional passes if the public keys of users must be exchanged, e.g. via certificates, within the protocol itself. Examining the computational complexities of the above schemes (see [10]), Schemes A(0) and B(0) require 3 exponentiations, whereas Schemes C(0) and C(1) require only 2. Scheme C(1) has the additional advantage over Schemes B(0) and C(0) that no inverses are needed; however, the inverses required are fixed long-term and can be precomputed in a one-time process. Matsumoto et al. also examine the computational complexity of passive eavesdropping attacks on the session key K. However, these results do not take into account active attacks nor potential time dependencies between the session keys.

The four schemes of Matsumoto et al. described above share the following properties. They require two passes. They provide mutual implicit key authentication, but no key confirmation. Assuming that A begins the protocols, use of additional fields in the message from B to A, containing a hash value of the shared secret key, provides unilateral key confirmation from B to A. The basic mechanisms do not provide detection of replayed tokens. Each entity can assure freshness of the key, since each of them provides one of the inputs of the shared key generation. ISO 11770-3 [9] contains a generic version of the C(1) protocol.

8 Key Agreement Mechanisms based on Self-Certifying Keys

In this section we consider key agreement mechanisms based on self-certifying public keys, as proposed by Girault [6].

Let n be an RSA integer [12], and α an element of maximal order in Z_n. T is a trusted authority which knows the factorization of n, and has a pair of integers e, d which are a (public, private) RSA key pair relative to n. Let user A have an identifying string I_A (e.g. A's name and address). A chooses a private key s_A, and provides $\alpha^{-s_A} \bmod n$ to T, in some authenticatable manner. T computes A's public key (which also serves as A's public key certificate here) as

$$P_A = (\alpha^{-s_A} - I_A)^d \bmod n$$

Consequently, the following is true:

$$P_A{}^e + I_A = \alpha^{-s_A} \bmod n$$

Similarly, B has private and publicly computable values s_B and $\alpha^{-s_B} \bmod n$. Note that any party can recover, from publicly available information, the quantities α^{-s_A}, α^{-s_B}, α^{s_A}, and α^{s_B}.

The idea is to use these publicly computable quantities as the public keys in Diffie-Hellman type protocols; the corresponding private keys s_A and s_B are known only to A and B, respectively. The above relations can be used to create authenticated key exchange protocols for users A and B as described below. Protocol G1 is from Girault [6]. It consists of basic Diffie-Hellman modified to use long-term certified key agreement keys as discussed at the end of Section 3; the novel feature is that the certification of these keys is provided by Girault's technique. Protocols G2 and G3 are obvious implementations of the ElGamal variation of Diffie-Hellman, and Matsumoto Scheme A(0), using the same idea.

Protocol G1. This protocol requires zero passes (assuming public keys are available *a priori*). A and B create the time-invariant joint key

$$K = (P_A{}^e + I_A)^{s_B} = (P_B{}^e + I_B)^{s_A} = \alpha^{-s_A s_B} \bmod n$$

Protocol G2. This protocol requires one pass. A chooses a random integer r_A and sends to B the value $\alpha^{r_A} \bmod n$. B computes $P_A{}^e + I_A = \alpha^{-s_A} \bmod n$, allowing A and B to each compute (via different methods as shown) the time-variant joint key

$$K = (\alpha^{s_A}\alpha^{r_A}(P_A{}^e + I_A))^{s_B} = (P_B{}^e + I_B)^{-r_A} = \alpha^{r_A s_B} \bmod n$$

Protocol G3. This protocol requires two passes. A chooses a random integer r_A and sends to B the value $\alpha^{-r_A} \bmod n$. Analogously, B chooses a random integer r_B and sends to A the value $\alpha^{-r_B} \bmod n$. A and B each compute (via different methods as shown) the time-variant joint key

$$K = (\alpha^{-r_A})^{s_B}({P_A}^e + I_A)^{r_B} = (\alpha^{-r_B})^{s_A}({P_B}^e + I_B)^{r_A} = \alpha^{-s_B r_A - s_A r_B} \bmod n$$

The key agreement mechanisms G1, G2 and G3 have properties analogous to those to which they correspond (as noted above) which do not use self-certifying public keys. Their status regarding standardization is as per the corresponding mechanisms. Mechanisms G1, G2 and G3 require zero, one and two passes, respectively. G1 provides mutual implicit key authentication; G2 provides unilateral implicit key authentication; G3 provides mutual implicit key authentication. In the basic form, they provide no key confirmation and no replay protection. G1 provides no freshness (it is designed to reuse the same key); in G2 freshness is controlled by A; in G3 freshness is mutually controlled.

9 Key Agreement based on Zero-knowledge Techniques

This key transport mechanism securely transfers in three passes two secret keys, one from A to B and one from B to A, which may be combined to form a key agreement. It is derived from the protocol known as COMSET and is based on zero-knowledge techniques - neither entity learns anything from the execution of the mechanism that it could not have computed itself (see Brandt et al. [1] for theoretical background). Both entities are authenticated and obtain (a type of) key confirmation about their respective keys. More precisely, there is confirmation that other enciphered plaintext in the same message as the key was properly recovered by the far end party, implying that the key itself was almost surely similarly recovered properly. The key confirmation here thus might be said to be *indirect*; this is in line with the zero-knowledge objectives of the protocol. It is assumed that each entity X has a personal asymmetric encipherment system with a public encryption transformation E_X and a private decryption transformation D_X and that each entity has access to authenticated copies of the public encipherment transformations of all other entities. The protocol is as follows.

A1 A wants to transfer its key K_A securely to B. A constructs a key block consisting of its distinguished name A, the key K_A, two randomly chosen numbers r_{1A} and r_{2A} and a time stamp or sequence number TVP_A. A enciphers the key block using entity B's public encipherment transformation E_B and appends the random number r_{1A} as validator. The following key token is then sent to B:

$$KT_{A1} = E_B(A, K_A, r_{1A}, r_{2A}, TVP_A), r_{1A}$$

B1 B also wants to transfer its key K_B securely to A. In the same way as A, B constructs a key block consisting of its distinguished name B, the key K_B, two randomly chosen numbers r_{1B} and r_{2B} and a time stamp or sequence number TVP_B. B enciphers the key block using entity A's public encipherment transformation E_A and appends the random number r_{1B} as validator to form the key token

$$KT_{B1} = E_A(B, K_B, r_{1B}, r_{2B}, TVP_B), r_{1B}$$

B2 B extracts the enciphered key block from the received key token KT_{A1} and deciphers it using its private decipherment transformation D_B. Then B verifies that A's distinguished name is present and that the validator r_{1A} sent openly in KT_{A1} is consistent with the random number r_{1A} in the deciphered key block. Then B checks the TVP_A for timeliness. If the verification is successful, B extracts the answer r_{2A} from the deciphered key block and sends back to A the key token KT_{B2}

$$KT_{B2} = KT_{B1}, r_{2A}$$

A2 A verifies that the answer r_{2A} extracted from KT_{B2} is consistent with the random number r_{2A} sent enciphered in KT_{A1} to entity B. If the verification is successful, A has obtained corroborating evidence that the token KT_{A1} has reached entity B intact; this provides a type of (indirect) key confirmation of key K_A.

A extracts the enciphered key block from the received key token KT_{B2} and deciphers it using its private decipherment transformation D_A. Then A verifies that B's distinguished name is present and that the validator r_{1B} contained openly in KT_{B1} is consistent with the random number r_{1B} in the deciphered key block. Then A checks the TVP_B for timeliness. If the verification is successful, A extracts the answer r_{2B} from the deciphered key block and sends the key token

$$KT_{A3} = r_{2B}$$

back to B. A may compute a shared secret key K_{AB} using a one-way combination of K_A and K_B.

B3 B verifies that the answer r_{2B} extracted from KT_{A3} is consistent with the random number r_{2B} sent enciphered in KT_{B1} to entity A. If the verification is successful, B has obtained corroborating evidence that the token KT_{B1} (and therefore almost surely key K_B) has reached entity A securely, and may compute a shared secret key K_{AB} using a one-way combination of K_A and K_B.

By use of the time-variant parameters, this mechanism provides mutual entity authentication, mutual (indirect) key confirmation, and prevents the replay of old key tokens. If timestamps are used, secure and synchronized timeclocks are required; otherwise the ability to maintain and verify pairwise sequence numbers is required. A can assure freshness of the key K_A, since it is the originating entity. But no guarantee of K_A's freshness is given to B by the mechanism. Analogously, B can assure freshness of the key K_B. But no guarantee of K_B's freshness is given to A by the mechanism.

10 Identity-based Public Keys without Restrictions in Trust

The basic idea of identity-based public key systems is that after the registration with a Key Center a user is able to authenticate himself to any other user without further communication with the Key Center. However, this property typically requires that the users have to trust the Key Center also to generate their private keys. In [7], Günther proposed an identity based public key system, where the Key Center creates *identity-based certificates* for identified users with distinguished names. These certificates have the following properties.

1. A user's public key can be retrieved by correct combination of the user's name and the certificate.
2. The authenticity of the certificate is not directly verified, but the correct public key can only be recovered from an authentic identity-based certificate.

Günther's system is very efficient, the certificates are short, but it has one drawback: for the computation of a certificate the Key Center also has to generate the corresponding private key. That is, the Key Center requires complete trust by the users. Nyberg and Rueppel [11] have presented a public key system with properties 1 and 2 that allows the users to generate their own private keys. Let $\mathcal{N}$ be the set of names or, more generally, the set of user credentials. Let $F : \mathcal{N} \rightarrow \mathbb{Z}_p$ be a one-way hash function or a redundancy generating function. Let S_C be the Key Center's private signature transformation and let V_C be the corresponding verification transformation with message recovery (for details of the new signature system please refer to [11]).

User U who wishes to obtain its public key token from the Key Center is adequately identified to have $name_U \in \mathcal{N}$ when presenting its public key $y_U = g^{x_U} \bmod p$ to the Key Center. Then the Key Center

1. Computes: $(r_U, s_U) = S_C(F(name_U)y_U \bmod p)$.
2. Delivers the public key token (r_U, s_U) to U or any other node in the network.

From $name_U$ and (r_U, s_U) the public key y_U is recovered as

$$y_U = F(name_U)^{-1}V_C(r_U, s_U) \bmod p$$

It is hard for anybody but the Key Center to produce a valid signature $S_C(m)$ for a given $m = F(name_U)y_U$. If a forger starts with a valid pair $(m, S_C(m))$ he is faced with the equation

$$m = F(name_U)y_U = F(name_U)g^{x_U} \bmod p$$

to solve for $name_U$ and x_U. If the forger fixes $name_U$ first he will not be able to retrieve the valid private key x_U. The function F has to be chosen in such a way that it is not possible to solve for x_U

and $name_U$ simultaneously. As in [7] the identity-based certificates can be used in combination with a suitable signature system to create identity based signatures. Similarily a secret session key establishment mechanism can be made identity-based if the parties use their identity-based certificates to transfer their public keys to the other party.

The ideas of section 8 are similar to those presented here (unlike Günther's scheme, Girault's technique does not result in the trusted authority knowing a user's private key). Both make use of certificates (actually, public keys) which are indirectly, or implicitly, authenticated by their construction and usage and where the party's secret key itself need not be given to the trusted party.

11 Summary

Table 2 surveys the features entity authentication, implicit key authentication and key confirmation provided by a key agreement protocol, from a unilateral viewpoint. Example protocols, discussed in the paper, are listed for reference.

Table 2. Types of authentication in key agreement protocols

type	entity authentication	implicit key authentication	key confirmation	protocol example
Type 0	–	–	–	Diffie-Hellman (basic)
Type 1	yes	–	–	ISO 9798-3 (key agreement version)
Type 2	–	yes	–	ElGamal; Matsumoto; Nyberg-Rueppel
Type 3	yes	yes	–	Nyberg-Rueppel (modified)
Type 4	–	–	yes	Diffie-Hellman + one-way hash of key
Type 5	yes	–	yes	Station-to-Station (STS)
Type 6	–	yes	yes	Type 2 protocol + key confirmation
Type 7	yes	yes	yes	Type 3 protocol + key confirmation

In Type 1 protocols, a critical point is that the protocol should verify that the party whose identity is corroborated by entity authentication is the same other party who has knowledge of the established shared key. This underlines the need to examine the relationship between entity authentication (column 2 in the table) and secure key agreement in protocols of Types 3, 5 and 7 also. It is notable that entity authentication is not necessary to obtain a securely established key, as illustrated by Type 2 protocols; indeed, implicit key authentication does not establish whether or not the party in question is operative at execution time. In Type 6 and 7 protocols, entity authentication is an inherent consequence of the other two features, if key freshness is a protocol feature; in this case, entity authentication is redundant in the Type 7 protocol.

Regarding key confirmation, given any key agreement mechanism, unilateral key confirmation can be obtained by appending to the final protocol message an extra field containing a cryptographic hash of the derived key; and mutual key confirmation can similarly be obtained by an extra message containing a distinct such hash. Adding key confirmation to a Type 0 protocol (yielding a Type 4 protocol) provides questionable benefit, in the absence of other means, such as an authenticated channel, for verification of the identities of the parties involved. Adding key confirmation to a Type 1 protocol (yielding a Type 5 protocol) must be done with care – as noted above, it is desirable to ensure that the party with whom the secret key is established, which will be the source of the key confirmation, is the same party whose identity follows from the entity authentication.

Table 2 also helps distill a meaning for a companion to the term implicit key authentication, namely *explicit key authentication* – that is, that a specifically identified party actually possesses (i.e., has effectively demonstrated possession of) a specified key. This differs from key confirmation which by itself, as defined herein, provides no evidence of the identity of the party involved; thus key authentication, rather than key confirmation alone, is the property actually desired in practice. Explicit key authentication may be achieved by any combination of two of the three columns in Table 2: (1) implicit key authentication plus key confirmation; (2) implicit key authentication plus entity authentication; or (3) key confirmation plus entity authentication. In each of cases (2) and (3), however, it is also required that there be verification by some means that the party implied by the two types of authentication is the same.

Table 3 summarizes some of the properties of the key agreement protocols that have been discussed. Protocols in which one party requires a certified copy of another party's public key can accomplish this via conventional certificates, out-of-band authentic channels, or via self-certifying keys as discussed in Section 8. The protocols listed listed in Table 3, or variations thereof, are all under consideration for standardization within ISO/IEC [9].

Table 3. Summary of key agreement protocols and properties[a]

protocol	passes[b]	EA	IKA	KC	KF[c]	reference
DH (basic Diffie-Hellman)	2	no	no	no	yes(A,B)	[3]
DH with fixed keys	0†	no	mutual	no	no	§3
ElGamal variant of DH	1†	no	unilateral	no	yes(A)	[5]
Nyberg-Rueppel	1†	optional[d]	mutual	no	yes(A)	[11]
ISO 9798-3 key agreement	3	mutual	no	optional[e]	yes(A,B)	§6
Matsumoto A(0) B(0) C(0) C(1)	2†	no	mutual	no	yes(A,B)	[10]
COMSET-based protocol	3†	mutual	mutual	indirect[f]	yes(A,B)	§9; [1]

[a] EA = entity authentication; IKA = implicit key authentication; KC = key confirmation; KF = key freshness
[b] assuming public key of other party is available *a priori* for entries marked "†"
[c] under control of indicated party (x)
[d] unilateral entity authentication possible if timestamp used
[e] mutual key confirmation possible with optional fields, without additional passes (STS protocol [4])
[f] indirect mutual key confirmation, as discussed in Section 9

References

[1] J. Brandt, I. Damgård, P. Landrock, T. Pedersen, *Zero-knowledge authentication scheme with secret key exchange* (extended abstract), Advances in Cryptology - Crypto'88, Lecture Notes in Computer Science, Springer-Verlag, pp.583-588. See also: P. Landrock, *A zero-knowledge protocol for identification with secret key exchange*, COMSET protocol version Feb.25/93, ISO SC27/WG2.N195.

[2] *CCITT Recommendation X.509*, The Directory - Authentication Framework, 1988 (revised 1993); also ISO 9594-8.

[3] W. Diffie, M.Hellman, *New directions in cryptography*, IEEE Transactions on Information Theory, Vol.22, 1976, pp.644-654.

[4] W. Diffie, P.C. van Oorschot, M.J. Wiener *Authentication and Authenticated Key Exchanges*, Designs, Codes and Cryptography, Vol.2, 107-125, 1992.

[5] T. ElGamal, *A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms*, IEEE Transactions on Information Theory, Vol.31, 1985, pp.469-472.

[6] M. Girault, *Self-certified public keys*, Advances in Cryptology - Eurocrypt'91, D.W. Davies (ed.), Lecture Notes in Computer Science 547, Springer- Verlag, 490-497.

[7] C. G. Günther, *An Identity-Based Key Exchange Protocol*, Advances in Cryptology - Eurocrypt'89, Lecture Notes in Computer Science 434, Springer- Verlag, pp.29-37.

[8] ISO/IEC IS 9798-3, Entity authentication mechanisms – Part 3: Entity authentication using a public-key algorithm, 1993.

[9] ISO/IEC CD 11770-3: Key Management – Part 3: Key management mechanisms using asymmetric techniques, 1993.

[10] T. Matsumoto, Y. Takashima, H. Imai, *On Seeking Smart Public-Key-Distribution Systems*, Trans. of the IECE of Japan, vol.E69 no.2, Feb.1986 pp.99-106.

[11] K. Nyberg, R. Rueppel, *A new signature scheme based on DSA giving message recovery*, Proc. 1st ACM Conference on Computer and Communications Security, pp.58-61, Nov. 1993.

[12] R. Rivest, A. Shamir, L. Adleman, *A method for obtaining digital signatures and public key cryptosystems*, Communications of the ACM, Vol.21, 1978, pp.120-126.

[13] A. Shamir, *Identity-based cryptosystems and signature schemes*, Advances in Cryptology - Crypto'84, G.R. Blakley and D. Chaum (eds.), Lecture Notes in Computer Science 196, Springer- Verlag, pp.47-53.