SPDH – A Secure Plain Diffie–Hellman Algorithm

Henrik Tange and Birger Andersen

Center for Wireless Systems and Applications/CTIF-Copenhagen, Copenhagen University College of Engineering, Lautrupvang 15, 2750 Ballerup, Denmark; e-mail: {heta, bia}@ihk.dk

Abstract

Secure communication in a wireless system or end-to-end communication requires setup of a shared secret. This shared secret can be obtained by the use of a public key cryptography system. The most widely used algorithm to obtain a shared secret is the Diffie–Hellman algorithm. However, this algorithm suffers from the Man-in-the-Middle problem; an attacker can perform an eavesdropping attack listen to the communication between participants A and B. Other algorithms as for instance ECMQV (Elliptic Curve Menezes Qo Vanstone) can handle this problem but is far more complex and slower because the algorithm is a three-pass algorithm whereas the Diffie–Hellman algorithm is a simple two-pass algorithm. Using standard cryptographic modules as AES and HMAC the purposed algorithm, Secure Plain Diffie–Hellman Algorithm, solves the Man-in-the-Middle problem and maintain its advantage from the plain Diffie–Hellman algorithm. Also the possibilities of replay attacks are solved by use of a timestamp.

Keywords: secure Diffie–Hellman algorithm, AES, HMAC, Man-in-the-Middle attacks, replay attacks.

Journal of Cyber Security and Mobility, Vol. 1, 143–160.

1 Introduction

Secure communication between two parties over an unsecure channel in a network in general requires confidentiality, data integrity, data origin authentication, non-repudiation and entity reputation.

The use of confidentiality ensures that data are secret for other than the two participants. Data integrity ensures that data have not been altered passing over the unsecure channel. To ensure that the sender of a message is the sender data origin authentication is used. The goal of non-repudiation is to make it able for the receiver to document that the message is sent from the sender. At last, entity reputation convinces the participants of each other's identity.

Furthermore, cryptographic systems can be divided into symmetric key systems and public key cryptography. Symmetric key systems are used for encryption and decryption of a message that should be kept secret. The encryption and decryption is done using a shared secret key. The public key cryptography is used to transport the shared secret key in a safe manner. The public key cryptography must provide a way to solve the key distribution problem also known as a need for private and authenticated key transport over an unsecure channel. This problem is partly solved by the Diffie–Hellman key exchange which makes it possible to obtain privacy in the key exchange. The advantage of the Diffie–Hellman algorithm is that, it is a lightweight two-pass protocol with only a public key transport from participant A to participant B and again from B to A. In the Diffie–Hellman algorithm the public key is used on both sides to calculate the shared secret. The problem is that the Diffie–Hellman algorithm is vulnerable against Man-in-the-Middle attacks. In this paper we show how the Diffie–Hellman algorithm can be protected against Man-in-the-Middle attacks and still function as a lightweight two-pass protocol. We show how the protocol can be used with one or more authentication centers with limited overhead and finally we verify the security of the protocol.

2 Background

Since the first establishment of electronic networks security has been an issue in order to keep information secret between parties. In this background we look into some important algorithms in public key cryptography, symmetric algorithms and message authentication.

2.1 The RSA Algorithm

There are several public key cryptography algorithms. One of the first public key cryptosystems was the RSA (Rivest, Shamir and Adleman) algorithm invented in 1977 [1, pp. 6]. The number theoretical problem behind the RSA algorithm is the integer factorization problem [1, pp. 6–7]. The integer factorization problem is the problem of calculating the plaintext m in the encryption and signature scheme:

$$m^{ed} \equiv m(\mathrm{mod})\, n,$$

where e is an encryption exponent and d is a private key or decryption exponent. m^{ed} is equal to the ciphertext c. Decryption of the ciphertext c is done by using the private key $d(c^d)$ due to the fact that

$$c^d \equiv (m^e)^d \equiv m(\mathrm{mod}\, n).$$

However, a number of possible attacks on the RSA algorithm have been described in the past years [2] such as factorization of n.

2.2 The Discrete Logarithm Problem

The next type of public key cryptography protocols is built on the discrete logarithm problem. The discrete logarithm problem was described by Diffie and Hellman in 1976. El Gamal described in 1984 the DL (Discrete Logarithm) public key encryption and signature system. The DL algorithm uses a set of public domain parameters: p, q, g. In this set p is a prime, q is a prime divisor of $p - 1$. g has order q and is selected in $[1, q - 1]$.

The discrete logarithm problem is built on the fact [1, p. 9] that

$$y = g^x \bmod p,$$

where x is a private key. The problem is to determine x from y.

2.3 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) is one of the newest cryptographic systems. Though elliptic curves have been known for the last 150 years it was first discovered to be used in public key cryptography in 1985. Since then ECC has been used for this purpose and is more interesting because it is substantially smaller than RSA [1, p. 19] but definitely more secure. Elliptic Curve systems are built on finite cyclic groups. An Abelian group $(G, *)$ is defined as a set G with a binary operation $* : GxG \rightarrow G$. The Abelian group

G match the following properties [1, pp. 11–12]: *Associativity, existence of an identity, existence of inverses and commutiativity*. In ECC the group operations are normally addition (+) or multiplication ($\cdot$). The identity element of addition is 0. The additive inverse value of a group b is $-b$. In the multiplicative group the identity element is normally 1. The inverse of the multiplicative group b is b^{-1}. A group is finite if G is a finite set: $\{0, 1 \ldots p-1\}$ and the number of elements in the group G is the *order of G*. A *finite field* is defined as $(F_{p,}+, \cdot)$. If G has an element g of n order then g is a generator of G which then is a *cyclic multiplicative group*.

The mathematical problem is the *discrete logarithm problem* and can be shown as extracting a private key from a public key

$$y = g^x,$$

where x is a private key and y is a public key. g is a multiplicative cyclic group and a part of the domain parameters. Also the order n is a domain parameter and the private key x is careful selected in $[1, n-1]$.

2.4 Non-Adjacent Form (NAF) and τ-adic Non-Adjacent Form (TNAF)

An elliptic curve E over F_p is an equation defined as

$$E : y^2 = x^3 + ax + b.$$

The calculation of a public key using a private key can be done with the NAF (Non-Adjacent Form) [1, p. 98] method for point multiplication. NAF represents the private key k as a signed digit representation

$$k = \sum_{i=0}^{l-1} k_i^{2^i} \quad \text{where} \quad k \in \{0, \pm 1\}.$$

This representation is guaranteed to be unique, and has an average density of non-zero digits around 1/3 of the length. As an input the NAF method takes a private key k and a point P on the elliptic curve. The point p is a base point which is a public known point defined according to the field size. After calculating the representation the NAF method uses k to calculate a new point on the elliptic curve. The outcome of the NAF representation algorithm is now used for calculating a new point on the elliptic curve. This is done as follows: If k_i is 0 $Q \leftarrow 2Q$. If $k_i = 1$ then first $Q \leftarrow 2Q$ and $Q \leftarrow Q + P$.

If $k_i = -1$ then first $Q \leftarrow 2Q$ and $Q \leftarrow Q - P$. The calculation $Q \leftarrow 2Q$ is called point doubling and is fairly heavy algorithm which consists of 15 steps.

Anomalous binary curves, also known as Koblitz curves, come in two equations:

$$E_0 : y^2 + xy = x^3 + 1,$$
$$E_1 : y^2 + xy = x^3 + x^2 + 1.$$

The big advantage of Koblitz curves is that point doubling can be avoided. A Koblitz curve E_a uses a cofactor h and a prime n and if $nh = \#E_a(F_{2m})$, where h is 4 if $a = 0$ and h is 2 if $a = 1$. Furthermore the Koblitz curve can define the Frobenius map [8] $\tau : E_a(F_{2m}) \rightarrow E_a(F_{2m})$ defined by

$$\tau(\infty) = \infty, \quad \tau(x, y) = (x^2, y^2).$$

This means that the Frobenius map is simple and fast to compute since squaring can be done in hardware near manner.

The Koblitz curve uses a τ-adic non-adjacent form (TNAF) [1, p. 116] instead of NAF explained above. The TNAF algorithm also produces a representation of private key $k : u_i \in \{0, \pm 1\}$. The TNAF (k) gives also a unique representation of k. If the length of the TNAF representation has the length l, then the average density of non-zero digits will be $l/3$. The TNAF algorithm can also guarantee that a non-zero digit is followed by zero. The digits produced in the main algorithm of TNAF are calculated by repeatedly dividing k by τ and τ^2 in the following manner:

Let $\alpha = r_0 + r_1\alpha$

If r_0 is even and α is divisible by τ then:

$$\alpha/\tau = (r_1 + \mu r_0/2) - (r_0/2)\tau.$$

Only if $r_0 \equiv 2r_1 \pmod 4$ then α is divisible by τ^2.

The TNAF algorithm is a perfect match for *point multiplication* in Elliptic Curve Cryptography in order to create a public key multiplied with a private key. The TNAF algorithm returns a representation of the private key $k \in \{0, \pm 1\}$. First a point Q is set to ∞. Now the sequence above is run through by first setting $Q = \tau Q$. If and only if $u_i = 1$ then $Q \leftarrow Q + P$. If and only if $u_i = -1$ then $Q \leftarrow Q - P$. The outcome is a public key Q.

Participants:

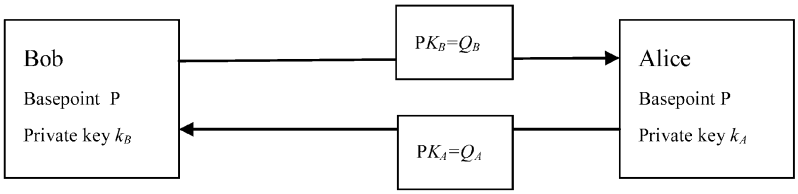

Figure 1 Diffie–Hellman key exchange algorithm (ECC version).

2.5 The Diffie–Hellman Algorithm – The ECC Version

The Diffie–Hellman public key exchange algorithm is a simple protocol [3, pp. 8–10] using exchange of public keys in order to obtain a common shared which can be used in a symmetric cryptographic system.

As stated above the Diffie–Hellman public key exchange is absolutely well fitted to the use of Elliptic Curve Cryptography. The simplicity of the protocol can be seen in Figure 1.

Now Bob wants to make a key exchange with Alice in order to obtain a shared secret. Bob multiplies his private key k_B with the basepoint P. Outcome is a new Point Q_B. This point is now sent to Alice. Receiving this point Alice now does the same as Bob multiplying the private key k_A with the basepoint P resulting in a new point Q_A. Q_A is then sent to Bob. The common shared secret for Bob is $k_B Q_A$ and for Alice $k_B Q_A$, which is the same key. Now this key can be used in a symmetric algorithm as for instance AES (Advanced Encryption System)

As it can be seen, this algorithm is very easy and only requires two communication steps in order to obtain a shared secret.

The problem with this algorithm is that the algorithm is open to Man-in-the-Middle attacks [4] where a third person Eve acts on behalf of Bob exchanging key with Alice and acts as Alice exchanging key with Bob.

2.6 The ECMQV Algorithm

The Man-in-the-Middle problem can be solved with a three-pass protocol as for instance the ECMQV (Elliptic Curve Menezes Qo Vanstone) algorithm. This protocol is using long-term static key pairs [3, p. 10]. Furthermore it is required to calculate two key pairs for each participant in the communication path. As the protocol is a three-pass algorithm it requires three

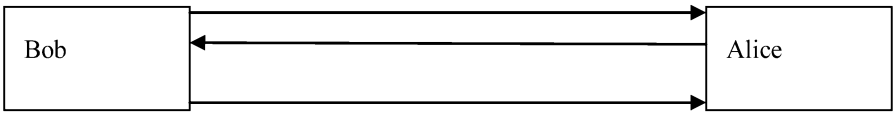

Figure 2 The ECMQV algorithm.

communication paths to obtain a shared secret. The protocol can be seen in Figure 2.

The algorithm has the following calculating sequences:

1. Bob calculates A, R_B (ephemeral key pair) which is sent to Alice
2. Alice sends B, R_A, $t_A = \text{MAC}k_1(2, B, A, R_A, R_B)$ to Bob
3. Bob sends $t_B = \text{MAC}k_1(3, A, B, R_B, R_A)$ to Alice.

This protocol is far heavier and requires extra network bandwidth and time to obtain the shared common.

2.7 TLS (Transport Layer Security)

The goal of TLS is to setup private keys for communication in an insecure network. TLS version 1.2 was standardized in August 2008.

The TLS handshake protocol involves a five step communication [5, p. 35] flow between server and client. The protocol can be seen in Figure 3.

The five steps start with a Client Hello sent to the server. This message contains information about the version number, some random generated data to be used to generate a master secret, a session ID, information about the cipher suite and the compression algorithm.

The server responds with a Server Hello. This message includes the chosen version number. A Server Random value is generated which is used in the master secret. The session ID is chosen: Either a new Session ID is created or a session ID is resumed, or a null value. The strongest possible cipher suite will be selected and at last the compression algorithm is selected.

The server sends a server certificate (server's public key) to the client along with a client certificate request. This part of the communication is ended with a Server Hello Done message.

Now the client responds to the server by returning a client certificate. After calculating the premaster secret the client sends a client key exchange to the server. Both the client and server will compute the master secret locally and derive the session keys from it. The Change Cipher Specification message

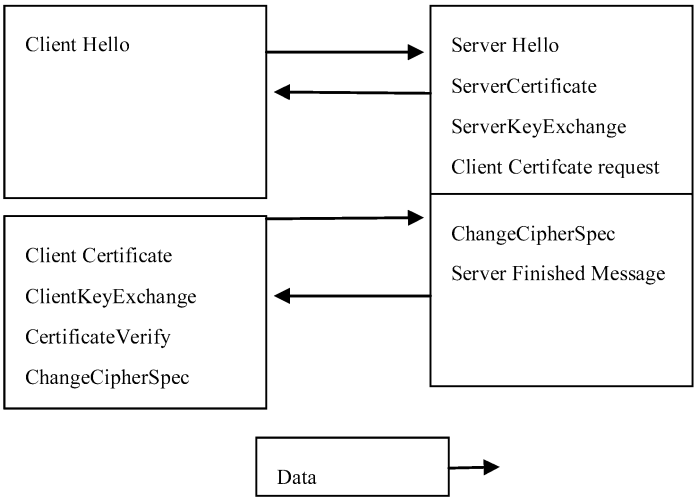

Figure 3 TLS handshake protocol.

tells the server that the following messages will be encrypted. The client ends this part by sending a client finished message.

The server's final response to the client is a Change Cipher Specification message and a server finished message. As it can be seen, this is a very heavy weight protocol.

2.8 AKE (Authenticated Key Exchange) Protocols

Huang and Cao have proposed an ID based AKE protocol [6]. The protocol is based on bilinear pairing using two cyclic groups of prime order $q(e = GxG \rightarrow GT)$. $P \in G$ is the generator of group G.

Furthermore the protocol uses three hash functions. There are two participants, A and B. When A is initiating the communication, A chooses an ephemeral private key $x \in Z$. Next A computes an ephemeral public key $X = xP$. X is sent to B. B is doing similarly and sends $Y = yP$ to A. When receiving B verifies that $X \in G$. B computes:

$$\begin{aligned} Z1 &= e(X + QA1; yZ + dB1), \\ Z2 &= e(X + QA2; yZ + dB2), \\ Z3 &= yX \quad \text{and} \\ SK &= H(Z1; Z2; Z3; \text{sid}), \quad \text{sid} = (X; Y; A; B). \end{aligned}$$

B keeps SK as a session key. A performs similar calculation to obtain the session key.

Jooyoung Lee and Je Hong Park have proposed a new AKE protocol [7] called the NAXOS+ protocol which is a modified NAXOS protocol as proposed by LaMacchia, Lauter and Mityagin.

The NAXOS+ protocol involves a CA (Certificate Authority). The CA checks if the public static key is contained in G^*. Via a certificate a participant obtains knowledge about each other.

For each session an ephemeral public/private key pair is generated. In order to obtain a MAC of the message a three-pass algorithm NAXOS+C.

2.9 AES (Advanced Encryption Standard)

The Advanced Encryption Standard (AES) became standard in 2001 and is described in the FIPS-197 document from NIST (National Institute of Standards and Technology).

At the time AES is considered to be one of the most secure symmetric algorithms.

AES is non-feistel algorithm; not the same algorithms are used for encryption and decryption.

As in normal block cipher, AES can be used in several modes [8, p. 151]: CBC, EBC, CFB, CTR and OFB.

The AES algorithm is using a fixed block size of 128 bits and uses different key sizes of 128, 192 or 256 bits depending on the security level. In AES four operations are used: AddRoundKey, SubBytes, ShiftRows and MixColumns. The order of four operations follows a well-known described scheme in the main algorithm consisting of rounds: In the initial round AddRoundKey is performed. In the following rounds SubBytes, ShiftRows, MixColumns and AddRoundKey are performed. In the last round only SubBytes, ShiftRows and AddRoundKey are performed. If the key size is 128 bits 10 rounds are executed, if the key size is 192 bits the number of rounds is 12 and finally if the key size is 256 bits the number of rounds is 14.

2.10 HMAC (Hash-based Message Authentication Code)

The HMAC is from 1997 and is publish in the RFC 2104 document. According to the document the HMAC algorithm can be used with any iterative cryptographic hash functions. At the time being there exists two standard implementations: SHA-1 and SHA-2. SHA stands for Secure Hash Standard.

SHA-3 implementation is to be decided in 2012. HMAC consists of a cryptographic hash function H and a secret key K. Also two fixed strings are used; ipad (inner) and opad (outer). Ipad contains the byte 0x36 B times, where B is the length of K. *opad* contains the byte 0x5C B times.

The HMAC is now computed [9, p. 3] over some text *aText*:

$$H(K\ XOR\ opad), H(K\ XOR\ ipad, aText)$$

The SHA-2 cryptographic function is rather a family of functions: SHA-224, SHA-256, SHA-384, and SHA-512, where the number denotes the digest length in bits. The SHA-2 algorithms were published in 2001 in FIPS PUB 180-2 designed by the NIST (National Institute of Standards and Technology). The SHA-2 is considered substantially more secure than the SHA-1.

3 Secure Plain Diffie–Hellman Algorithm

In this section we introduce the Secure Plain Diffie–Hellman algorithm (SPDH). The SPDH algorithm can be used where an AuC (Authentication Center) is present.

3.1 Introduction to the Secure Plain Diffie–Hellman Algorithm

The algorithm addresses the following issues:

- It is a lightweight two pass algorithm.
- It uses basic Diffie–Hellman (Elliptic Curve) approach.
- It is secured against Man-in-the-Middle attacks.
- It is secured against replay attacks.
- Security relies on ECC, HMAC and AES.
- It can use personal IDs – not only hardware IDs.
- It can be routed on a network without being exposed.
- Initial keys ready from start.
- Initial keys can be renewed.
- Secure transport of Diffie–Hellman keys.
- It uses well-known and tested algorithms.

3.2 Secure Plain Diffie–Hellman Algorithm

INPUT: A basepoint P, private key k, a private key x_{AES}, a private key y_{HMAC}, a personal ID *pid*, Timestamp ts

PARTICIPANTS: A, B and Authentication Center (AuC)

OUTPUT: A shared secret G

1. A calculates $P_{(A)k}$
2. A calculates HMAC(AES($PA_{(A)k} + pid_A + ts$)) using $x_{AES(A)}$ and $y_{HMAC(A)}$
3. $\rightarrow$ AuC Checks (2) (Recalculates HMAC and checks that ts is newer than listed and reliable)
4. AuC unpack HMAC(AES($PA_{(A)k} + pid_A + ts$)) using $x_{AES(A)}$ and $y_{HMAC(A)}$
5. AuC recalculates HMAC(AES($PA_{(A)k} + pid_A + ts$)) using $x_{AES(B)}$ and $y_{HMAC(B)}$
6. $\rightarrow$B Checks (5) (Recalculates HMAC and checks that ts is newer than listed and reliable)
7. B unpack (6) using $x_{AES(B)}$ and $y_{HMAC(B)}$
8. B calculates $P_{(B)k}$ and stores $\mathrm{P}_{(A)k}$
9. B calculates HMAC(AES($P_{(B)k} + pid_B + ts$)) using $x_{AES(B)}$ and $y_{HMAC(B)}$
10. $\rightarrow$ AuC checks (9) (Recalculates HMAC and checks that ts is newer than listed and reliable)
11. AuC unpack HMAC(AES($\mathrm{P}_{(B)k} + pid_B + ts$)) using $x_{AES(B)}$ and $y_{HMAC(B)}$
12. AuC recalculates HMAC(AES($\mathrm{P}_{(B)k} + pid_B + ts$)) using $x_{AES(A)}$ and $y_{HMAC(A)}$
13. $\rightarrow$ A Checks (12) (Recalculates HMAC and checks that ts is newer than listed and reliable)
14. A unpack (13) using $x_{AES(A)}$ and $y_{HMAC(A)}$ and stores $P_{(B)k}$
15. Common shared secret is $P_{(A)k}P_{(B)k}$

It is assumed that the private k is generated by the participant, the private keys for A: $x_{AES(A)}$, $y_{HMAC(A)}$ and $x_{AES(B)}$, $y_{HMAC(B)}$ are delivered from authentication center (AuC) in the initial phase during setup. The personal ID is also generated by the participant and is well known by other participants.

As it can be seen from this algorithm not even the AuC can calculate the common shared secret $P_{(A)k}P_{(B)k}$ since only the participants themselves know the private key k.

The key exchange is secured further. The HMAC algorithm solves the problem of malicious altering of the key exchange and will also prevent replay attacks. The message uniqueness is guaranteed by using Timestamp ts.

The algorithm has some further requirements. The following is assumed:

1. A and B are participants in the system and the authentication procedure between AuC and the participants are done correctly.
2. The keys $x_{AES(A)}$, $y_{HMAC(A)}$ and $x_{AES(B)}$, $y_{HMAC(B)}$ are created in a challenge-response manner with AuC.
3. En elliptic curve base point is created along with other domain parameters.
4. A pid is selected.
5. A private key k for each participant is selected.
6. The Timestamp ts is created locally for both participants A and B plus the AuC.
7. A, B and AuC maintains a list of timestamps used for checking that a message is new and never received before. The Timestamp is also encrypted.
8. The ECC, HMAC and AES software are installed.

3.3 Network Topology – Involving More Than One AuC

Since the content is secret during transport on the network involving more than one AuC in a network (roaming) does not impose further problems. It is assumed that the involved AuC know and trust each other and have exchanged encryption/decryption keys. In that way a single AuC can transport encrypted information without having knowledge about the shared secret. It is not demanded that the transport of the content is secured previously for instance using SSL/TLS.

The main flow can be shown as described below.

3.4 The Routed Secure Plain Diffie–Hellman Algorithm

INPUT: A basepoint P, Private key k, a private key x_{AES}, a private key y_{HMAC}, a personal ID pid, Timestamp ts, a network routing table rt, receiver info ri.

PARTICIPANTS: A, B and Authentication Centers ($\text{AuC}_i^{i=0}$, AuC_{i+1} ... AuC_{i+n})

OUTPUT: A shared secret G

Note: The input parameter rt is a routing table. This routing table routes the Diffie–Hellman setup information through a network that can change depending on transmission time, cost or other parameters. The routing table

is typically created as a matrix in two or more dimensions. The ri input parameter is a unique information about the receiver of the information (i.e. roaming information).

1. A calculates $P_{(A)k}$
2. A calculates HMAC(AES($PA_{(A)k} + pid_A + ts$)) using $x_{AES(A)}$ and $y_{HMAC(A)}$
3. A sends(2) to AuC_0
4. AuC_i Checks (3) (Recalculates HMAC and checks that ts is newer than listed and reliable)
5. AuCi unpack HMAC(AES($PA_{(A)k} + pid_A + ts$)) using $x_{AES(A)}$ and $y_{HMAC(A)}$
6. AuC_i recalculates HMAC(AES($PA_{(A)k} + pid_A + ts + rt + ri$)) using $x_{AES(AuCi)}$ and $y_{HMAC(AuCi)}$
7. AuC_i Sends (6) to AuC_{i+1}
8. AuC_{i+1} (Recalculates HMAC and checks that ts is newer than listed and reliable)
9. AuC_{i+1} unpack (8) using $x_{AES(AuCi)}$ and $y_{HMAC(AuCi)}$
10. AuC_{i+1} recalculates HMAC(AES($PA_{(A)k} + pid_A + ts + rt + ri$)) using $x_{AES(AuCi+2)}$ and $y_{HMAC(AuCi+2)}$
11. AuC_{i+1} sends (10) toAuC_{i+2}
12. For length of rt: Step 4 to 12
13. AuC_{i+n} Checks (12) (Recalculates HMAC and checks that ts is newer than listed and reliable)
14. AuC_{i+n} unpack HMAC(AES($PA_{(A)\,k} + pid_A + ts + rt + ri$)) using $x_{AES(AuCi+n)}$ and $y_{HMAC(AuCi+n)}$
15. AuCi+n recalculates HMAC(AES($PA_{(A)k} + pid_A + ts$)) using $x_{AES(B)}$ and $y_{HMAC(B)}$
16. AuC_{i+n} reads ri (now empty) and sends (15) to B
17. B checks (16) (Recalculates HMAC and checks that ts is newer than listed and reliable)
18. B checks that pid_A is well known and accepted
19. B calculates $P_{(B)k}$ and stores $\mathrm{P}_{(A)k}$
20. B calculates common shared secret $G = (P_{(A)k}\ \ P_{(B)k})$
21. B calculates HMAC(AES($P_{(B)k} + pid_B + ts$)) using $x_{AES(B)}$ and $y_{HMAC(B)}$
22. B sends (21) to AuCi *in reverse order* AuC_{i+n} to AuC_0
23. AuCi checks (22) (Recalculates HMAC and checks that ts is newer than listed and reliable)

24. AuCi unpack HMAC(AES($P_{(B)k} + pid_B + ts$)) using $x_{AES(AuCi+1)}$ and $y_{HMAC(AuCi+1)}$
25. AuCi recalculates HMAC(AES($P_{(B)k} + pid_B + ts + rt + ri$)) using $x_{AES(A)}$ and $y_{HMAC(A)}$
26. AuCi sends (25) to A
27. A checks (26) (Recalculates HMAC and checks that ts is newer than listed and reliable)
28. A checks that pid_B is well known and accepted
29. A unpack (27) using $x_{AES(A)}$ and $y_{HMAC(A)}$ and stores $P_{(B)k}$
30. Common shared secret is $G = (P_{(A)k}\ \ P_{(B)k})$

4 Experimental Results and Analysis

In order to test the performance of the algorithm, the system consisting of Elliptic Curve Cryptography, HMAC and AES has been implemented in C++.

4.1 Software Implementation

The software implementation of the Elliptic Curve Cryptography is built in three layers: (1) a basic layer for operations on bit strings, (2) a second layer for handling fields; and (3) the third layer is handling Elliptic Curve Cryptography operations. The Elliptic Curve Cryptography implementation is an anomalous binary curve also known as a Koblitz curve.

HMAC is implemented as described in FIPS 180-2 (Federal Information Processing Standards Publications) from National Institute of Standards and Technology (NIST) and is the SHA-256 (Secure Hash Algorithm) version.

The flow is as described in Section 3.2. First A is calculating a point on the curve using the private key. Next a message encryption of the point along with the personal ID and the timestamp is done. After this the HMAC value is calculated. AuC recalculates the HMAC value and decrypt the message. After this the AuC encrypt the message with B's key and pass the message on to B along with a newly created HMAC value.

B recalculates the HMAC value and unpacks the message.

Now B creates a response message with a calculated point using B's private key along with the personal ID of B and a timestamp. The AuC repeats the decryption-encryption of the message from B and pass the message on to A.

The common share between A and B is $QK_A\ K_B$.

Table 1 Operation times.

Operation	Time [mS]
AES Encryption	0.014
AES Decryption	0.015
HMAC Hashcode	0.003
ECC Point calculation	105.0

4.2 Performance Analysis

It is well known that symmetric algorithms are faster than asymmetric algorithms. In another word the point calculation in ECC is slower than the AES encryption/decryption and calculation of the HMAC value.

As it can be seen above the asymmetric algorithm is only calculated on end-points (participants A and B). Furthermore, the participants A and B have to calculate AES and HMAC. The AuC calculates only the symmetric algorithms and therefore the data transport between the A and AuC and between B and AuC is fast.

The Routed SPDH algorithm will in the same way benefit from a fast algorithm; still only the end-point participants have to calculate the asymmetric algorithm so load on the shared AuCs is limited.

The speed in the system has been tested using an Intel® Core™ i3 CPU @ 2.27 GHz. In this performance test, the network time is not included.

The test results are shown in Table 1. These values will in real systems first of all depend on the calculation power of the participants A and B. Second, the load on the AuC will influence on calculation time. Third, the network load will also influence on calculation time.

5 Verification

5.1 Verification of the SPDH Algorithms

Both the SPDH algorithm and the Routed SPDH algorithm which is an extended version of SPDH algorithm have been verified.

The SPDH algorithms have been tested using ProVerif [10]. ProVerif is an automatic cryptographic protocol verifier, which is able to handle cryptographic primitives and also for instance public key cryptography. Especially Diffie–Hellman key agreements and hash functions can be documented using this tool.

The algorithms have been tested for passive attacks and active attacks. In the passive attack the attacker listen to the communication (eavesdropping).

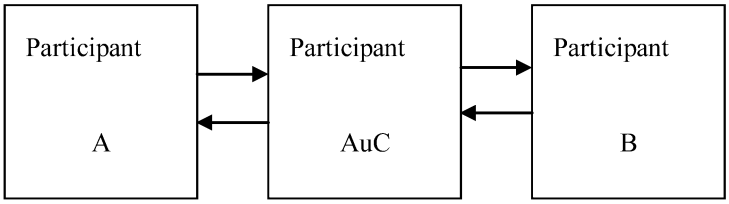

Figure 4 Participants in SPDH.

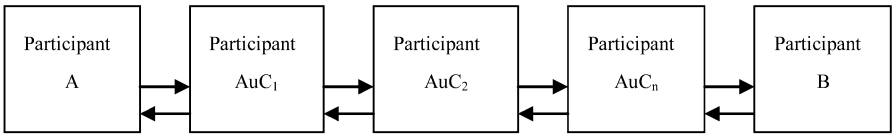

Figure 5 Participants in Routed SPDH.

In active attacks, as for instance Man-in-the-Middle attacks the attacker tries to alter data.

For the SPDH algorithm the setup in ProVerif is as shown in Figure 4.

For the Routed SPDH algorithm the setup in ProVerif is as shown in Figure 5.

It is well known that there is a vulnerability in the Diffie–Hellman algorithm, where the attacker can perform a Man-in-the-Middle attack simply by intercepting the communication initiated by A and B. In this attack type the attacker, E, intercepts the public key from A. Then E transmits E's own public key to B. B then transmits his public key to E without knowing that is E and not A, E also intercept the key to B.

No mechanism in the Diffie–Hellman protects against replay attacks. A replay attack is to done by retransmit valid data on the network that have been captured previously [11]. By using a timestamp retransmitted data can be discarded.

5.2 Verification Results

The results from the ProVerif test are presented in Table 2.

5.3 Limitations

AuCs must be reliable and trusted.

Table 2 ProVerif test results.

Algorithm	**Eavesdropping** (Passive)	**Replay attack** (Active)	**Man-In-The-Middle attack** (Active)
Diffie-Hellman algorithm	Secure	Vulnerable	Vulnerable
SPDH algorithm	Secure	Secure	Secure
Routed SPDH algorithm	Secure	Secure	Secure

6 Conclusion

A new and more secure variant of the original Diffie–Hellman algorithm has been introduced.

The Secure Plain Diffie–Hellman algorithm has the advantage of the original Diffie–Hellman algorithm, which is a fast two-pass communication algorithm. The secure Plain Diffie-Hellman algorithm is based on Elliptic Curve Cryptography and uses standard implementations of AES and HMAC to enhance security.

The Secure Plain Diffie–Hellman algorithm is protected against Man-in-the-Middle attacks, eavesdropping and replay attacks.

The proposed Secure Plain Diffie–Hellman algorithm can be used in numerous systems where a trusted central unit is present. This could for instance be in the mobile telephone system, in wireless systems, in secure sensor systems or military field units.

The Secure Plain Diffie–Hellman algorithm is especially well suited for mobile communication systems where the need is to achieve an endpoint-to-endpoint secure communication. But also in situations where ad-hoc network can be created the Secure Plain Diffie–Hellman algorithm can be used to provide endpoint-to-endpoint secure communication.

The Secure Plain Diffie–Hellman algorithm has also been presented in a routed version. The routed version is well suited in mobile communication systems where endpoint-to-endpoint users are using different providers or are roaming.

The performance test has shown that the use of an Authentication Center (AuC) will not influence substantially on the general performance since the operations on the AuC are only symmetric operations and thereby fast.

References

[1] Hankerson et al. Guide to Elliptic Curve Cryptography. Springer, 2004.
[2] D. Boneh. Twenty years of attacks on the RSA cryptosystem. Notices of the American Mathematical Society (AMS), 46(2):203–213, 1999.
[3] F. Blake (Ed.). Advances in Elliptic Curve Cryptography. Cambridge University Press, 2005.
[4] Mario Cagalj, Srdjan Capkun, and Jean-Pierre Hubaux. Key agreement in peer-to-peer wireless networks. IEEE (Special Issue on Cryptography and Security). bibitem5. RFC 5246, 2008.
[5] Hai Huang and Zhenfu Cao. An ID-based authenticated key exchange protocol based on bilinear Diffie–Hellman problem. Department of Computer Science and Engineering, Shanghai Jiaotong University, ASIACCS, 2009.
[6] Jooyoung Lee and Je Hong Park. Authenticated key exchange secure under the computational Diffie–Hellman assumption. The Attached Institute of Electronics and Telecommunications Research Institute, Korea, IACR, 2008.
[7] W. Trappe and L.C. Washington. Introduction to Cryptography with Coding Theory (second edition). Pearson, 2006.
[8] RFC 2104.
[9] http://www.proverif.ens.fr/.
[10] Priyanka Goyal, Sahil Batra, and Ajit Singh. A literature review of security attack in mobile ad-hoc networks. International Journal of Computer Applications, 9(12):11–15, November 2010.

Biographies

Henrik Tange received the B.Eng (export engineer) from the Copenhagen University College of Engineering in 1999 and the M.Sc. in Communication Network specializing in Security from Aalborg University in 2009. Since 2009 he has been a PhD student at Aalborg University. Since 2000 he has been teaching at Copenhagen University College of Engineering.

Birger Andersen is a professor at Copenhagen University College of Engineering, Denmark, and director of Center for Wireless Systems and Applications (CWSA) related. He received his M.Sc. in Computer Science in 1988 from University of Copenhagen, Denmark, and his Ph.D. in Computer Science in 1992 from University of Copenhagen. He was an assistant professor at University of Copenhagen, a visiting professor at Universität Kaiserslautern, Germany, and an associate professor at Aalborg University. Later he joined the IT Department of Copenhagen Business School, Denmark, and finally Copenhagen University College of Engineering. He is currently involved in research in wireless systems with a focus on security.