

## CERTIFICATION OF DL/EC KEYS

Benjamin Arazi  
LPK  
arazi@ee.bgu.ac.il

### ABSTRACT

It is shown that the explicit certification of public keys in customary DL/EC (Discrete-Log/Elliptic-Curve) applications, ranging from digital signatures of the DSA type to key-agreements of the DH type, can be abolished.

This facilitates highly efficient implementations in terms of the total number of exponentiations needed to be executed, the ability of having parallel processing, and communication overhead.

At the fundamental level it is shown how to integrate the processing of the public key of the trusted third party (needed, by definition, in establishing the validity of static public values submitted by a user) and the dynamic processing associated with the actual cryptographic process. This reduces, by a factor of at least 2, the processing time when compared to standard signature and key-agreement techniques, while further reducing communication overhead.

It is then shown how the performance of the introduced key-agreement techniques is further enhanced, by utilizing a principle termed "you are OK if I am OK". Here, the processing of the public key of Alice's trusted third party is not performed by Bob *after* he receives the values submitted by Alice, as customarily done. Instead, Bob refers to the said public key *prior* to his communication with Alice (utilizing the realistic observation under which Bob is supposed to know in any case the public key of Alice's trusted third party regardless of his communication with Alice). Here, if Bob is assured that *his* secret and public values are valid then he is subsequently assured that the public values submitted by Alice are valid as well.

## *Notations and terminology*

DL/EC (Discrete-Log/Elliptic-Curve) cryptographic applications relate to operations over a finite group of points in which the discrete log problem applies.

A group-point is denoted in **bold**.

$s*\mathbf{P}$  is a group-point obtained by multiplying the group-point  $\mathbf{P}$  by the scalar  $s$ . This scalar is considered an exponent. When operating over a group of integers modulo an agreed  $q$ , the notation  $s*\mathbf{P}$  means  $\mathbf{P}^s \bmod q$ , and  $s*\mathbf{P} + t*\mathbf{Q}$  means  $\mathbf{P}^s * \mathbf{Q}^t \bmod q$ .

$\mathbf{G}$  - the generating group-point, joint to all users that use the services of a system controlled by a certain trusted third party. Exponents are calculated modulo the order of  $\mathbf{G}$ .

$\text{Log}\mathbf{P}$  - the scalar  $k$  such that  $\mathbf{P} = k*\mathbf{G}$ .

$d$  - the secret key of the trusted third party.

(For simpler explanations, it is assumed that Alice and Bob use the services of the same trusted third party. Extensions to the general case, where these parties use the services of different trusted third parties, naturally follow.)

$\mathbf{R} = d*\mathbf{G}$  - the public key of the trusted third party.

$x_A, x_B$  - the secret key of Alice, Bob.

$\mathbf{Y}_A = x_A*\mathbf{G}$ ,  $\mathbf{Y}_B = x_B*\mathbf{G}$  - the public key of Alice, Bob.

$\mathbf{P}_{UA}$ ,  $\mathbf{P}_{UB}$  - the public *value* of Alice, Bob. (In the implementations presented in this document, the users submit the said public value, and not the public *key*  $\mathbf{Y}_A$ ,  $\mathbf{Y}_B$ .)

$IT_A, IT_B$  - a representative of the identification details or the attributes of Alice, Bob.

$H(v, \mathbf{W})$  - a transformation, known to all, that converts a scalar  $v$  and a group-point  $\mathbf{W}$  into a scalar. ( $H$  is not necessarily a hash. Features of  $H$  are discussed later.)

# 1. Background

## 1.1. DL/EC signature techniques

As a representative of DL/EC signature techniques, where Alice signs the message  $m$ , we describe the DSA [1].

Let  $x_A$  denote Alice's secret key. Let  $\mathbf{G}$  be a generating group-point known to all users. Let  $\mathbf{Y}_A$  denote Alice's public key, where  $\mathbf{Y}_A = x_A * \mathbf{G}$ .

Signature: Alice selects a one-time key pair  $k$  and  $\mathbf{V} = k * \mathbf{G}$ , represents  $\mathbf{V}$  by an integer  $p$  and calculates  $q = k^{-1} * (m + x_A * p)$  (where integer calculations are performed modulo the order of  $\mathbf{G}$ ). The signature on  $m$  is the pair  $\{p, q\}$ .

Alice then submits  $m, p, q, \mathbf{Y}_A$ .

Signature verification: The verifier, Bob, calculates  $t = q^{-1}$ ,  $v = m * t$  and  $w = p * t$ . He then calculates  $\mathbf{P} = v * \mathbf{G} + w * \mathbf{Y}_i$ , represents  $\mathbf{P}$  by an integer  $r$  (in the same way that  $p$  represents  $\mathbf{V}$ ). The signature is determined to be valid if  $r = p$ .

### *The certification principle*

Let  $IT_A$  denote Alice's identification details or claimed attributes. When Alice submits to Bob her public key  $\mathbf{Y}_A$ , Bob must be given an assurance that  $\mathbf{Y}_A$  is associated with  $IT_A$ . Here, Alice also submits to Bob a certificate  $CR_A$ , which is the signature of a trusted third party on the association between  $\mathbf{Y}_A$  and  $IT_A$ .

A complete DSA process therefore involves two signature verifications conducted by Bob. The first involves certificate verification. Here, the signed message is a value which combines  $IT_A$  and  $\mathbf{Y}_A$ . The signature is  $CR_A$ , generated by the trusted third party, whose private key is  $d$ . The public key used in the certificate verification is the value  $\mathbf{R} = d * \mathbf{G}$ , published by the trusted third party. The second signature verification involves the actual process of verifying the authenticity of the dynamic message  $m$ .

Signature generation involves one exponentiation operation of the form  $b * A$ . Signature verification involves an operation of the form  $b * A + d * C$ , performed twice (including certificate verification). The latter operation can amount to less than two individual exponentiations when using some 'speedup' methods [2].

## 1.2. DL/EC key-agreement techniques

DL/EC key-agreements are based on the DH (Diffie-Hellman) method [3] or its variants. Here, Alice and Bob exchange some public values, and by using values received from their counterpart and their own secret keys they end up with a value, common to the two of them, and which only they are supposed to know. This common value then acts as their generated session key.

A basic DH key-agreement takes the following steps:

- Alice sends to Bob the values  $Y_A$ ,  $IT_A$  and  $CR_A$ , defined before. Symmetrically, Bob sends to Alice:  $Y_B$ ,  $IT_B$  and  $CR_B$ .
- Alice and Bob establish the validity of  $Y_B$ ,  $Y_A$ , based on the certificate  $CR_B$  and  $CR_A$  they respectively received and by referring to the public key of the trusted third party.
- Alice and Bob respectively generate the session key  $K_{AB} = X_A * Y_B$  and  $K_{BA} = X_B * Y_A$ .

The mathematical operations performed by each user amount to one operation of the form  $b * A + d * C$  (for certificate verification) and one more exponentiation (for session key generation).

The above technique concerns a fixed-key-agreement, wherein Alice and Bob always end up with the same session key whenever they wish to generate such a key.

An ephemeral DH key-agreement, wherein Alice and Bob generate a different session key whenever they communicate, based on a different random value each of them generates in each communication session, takes the following possible steps:

- Alice generates a random  $p_{VA}$ , calculates the ephemeral value  $\mathbf{EVA} = p_{VA} * \mathbf{G}$  and uses  $x_A$  in order to generate a signature  $S_A$  on  $\mathbf{EVA}$ . She then sends to Bob:  $\mathbf{YA}$ ,  $IT_A$ ,  $CR_A$ ,  $\mathbf{EVA}$  and  $S_A$ .

Symmetrically, Bob sends to Alice:  $\mathbf{YB}$ ,  $IT_B$ ,  $CR_B$ ,  $\mathbf{EVB}$  and  $S_B$ .

- Alice and Bob establish the validity of  $\mathbf{YB}$ ,  $\mathbf{YA}$ , based, respectively, on the certificate  $CR_B$  and  $CR_A$  and by referring to the public key of the trusted third party.
- Alice and Bob establish the validity of the received ephemeral values  $\mathbf{EVB}$  and  $\mathbf{EVA}$ , based on the signatures  $S_B$  and  $S_A$  and by referring to the public keys  $\mathbf{YB}$  and  $\mathbf{YA}$  (whose validity was established in the preceding step).
- Alice and Bob respectively generate  $K_{AB} = p_{VA} * \mathbf{EVB}$  and  $K_{BA} = p_{VB} * \mathbf{EVA}$ .

The mathematical operations performed by each user, when implementing the above procedure, amount to two operations of the form  $b * \mathbf{A} + d * \mathbf{C}$  (one for certificate verification and one for signature verification) and three more exponentiations (for ephemeral value generation, for signature generation and for key generation).

A further ephemeral key-agreement method, is the MQV [4]. The following is one variation of this method:

- Alice generates a random  $p_{VA}$ , calculates  $\mathbf{EVA} = p_{VA} * \mathbf{G}$  and sends to Bob:  $\mathbf{YA}$ ,  $IT_A$ ,  $CR_A$  and  $\mathbf{EVA}$ . Symmetrically, Bob sends to Alice:  $\mathbf{YB}$ ,  $IT_B$ ,  $CR_B$  and  $\mathbf{EVB}$ .
- Alice and Bob establish the validity of  $\mathbf{YB}$ ,  $\mathbf{YA}$ .
- Alice and Bob respectively generate  $K_{AB} = (p_{VA} + T(\mathbf{EVA}) * x_A) * (\mathbf{EVB} + T(\mathbf{EVB}) * \mathbf{YB})$  and  $K_{BA} = (p_{VB} + T(\mathbf{EVB}) * x_B) * (\mathbf{EVA} + T(\mathbf{EVA}) * \mathbf{YA})$ , where  $T$  is a transformation that converts the group-point  $\mathbf{EVA}$  and  $\mathbf{EVB}$  into a scalar.

The mathematical operations performed by each user, when implementing the above, amount to one operation of the form  $b * \mathbf{A} + d * \mathbf{C}$  and three more exponentiations.

## 2. Description of the cryptographic technique

The various aspects of the presented techniques, described next, do not include security considerations. These are treated in detail in Part 4 of this document. For simpler explanations, we treat the case where Alice and Bob use the services of the same trusted third party. Extensions to the general case simply follow. Here, whenever a user makes a reference to the public key of a trusted third party, this reference is being made to the key of the trusted third party of his counterpart.

### 2.1. Generating user's secret and public values

The group-points  $\mathbf{G}$  and  $\mathbf{R}$ , and the scalar  $d$ , have already been defined as the generating group-point, the public key of the trusted third party, and the secret key of that party, where  $\mathbf{R} = d * \mathbf{G}$ . We further use a transformation  $H(v, \mathbf{W})$ , known to all, that converts a scalar  $v$  and a group-point  $\mathbf{W}$  into a scalar.

While Alice's public key, in the presented technique, is still  $\mathbf{Y}_A = x_A * \mathbf{G}$ , for a secret key  $x_A$ , *the public reference value submitted by Alice is a group-point  $\mathbf{P}_{UA}$ .*

The secret key  $x_A$  and the public value  $\mathbf{P}_{UA}$  of Alice are generated as follows:

- Alice generates a random  $h_A$  and submits  $\mathbf{V}_A = h_A * \mathbf{G}$  and  $IT_A$  to the trusted third party;
- The trusted third party generates a random  $k_A$ , he then calculates  $k_A * \mathbf{G}$ ,  $\mathbf{P}_{UA} = \mathbf{V}_A + k_A * \mathbf{G}$  and  $p_A = H(IT_A, \mathbf{P}_{UA}) * k_A + d$ ;
- The trusted third party issues the values  $p_A$  and  $\mathbf{P}_{UA}$  to Alice;
- Alice generates her secret key  $x_A = p_A + H(IT_A, \mathbf{P}_{UA}) * h_A$ .  
That is:  $x_A = H(IT_A, \mathbf{P}_{UA}) * (k_A + h_A) + d$ .

It is noted that  $\mathbf{Y}_A = x_A * \mathbf{G} = H(IT_A, \mathbf{P}_{UA}) * \mathbf{P}_{UA} + \mathbf{R}$ . *It is further noted that the trusted third party does not know the value of Alice's secret key  $x_A$ , while no party knows  $\log \mathbf{P}_{UA}$ .* ( $\log \mathbf{P}_{UA} = h_A + k_A$ , where Alice and the trusted third party each knows only one addend.)

Alice can establish the validity of the values  $p_A$  and  $\mathbf{PUA}$  issued to her by checking whether  $p_A * \mathbf{G} = \mathbf{H}(\mathbf{IT}_A, \mathbf{PUA}) * (\mathbf{PUA} - \mathbf{V}_A) + \mathbf{R}$ .

## 2.2. DSA procedure without a separate submission of a public key and a certificate

In the presented technique, Alice signs a message  $m$  in the standard lines of the DSA. That is: Alice generates a one-time key pair  $k$  and  $\mathbf{V} = k * \mathbf{G}$ , represents  $\mathbf{V}$  by an integer  $p$  and calculates  $q = k^{-1} * (m + x_A * p)$ . The signature on  $m$  is the pair  $\{p, q\}$ .

Alice then submits  $m$ ,  $p$ ,  $q$ , her  $\mathbf{IT}_A$  and the public value  $\mathbf{PUA}$ . (Unlike a customary DSA, she does not submit her public key  $\mathbf{Y}_A = x_A * \mathbf{G}$  and she does not submit a certificate.)

Following the lines of the DSA, the verifier calculates  $t = q^{-1}$ ,  $v = m * t$  and  $w = p * t$ , and then  $u = \mathbf{H}(\mathbf{IT}_A, \mathbf{PUA}) * w$  and  $\mathbf{P} = v * \mathbf{G} + u * \mathbf{PUA} + w * \mathbf{R}$ . The value  $\mathbf{P}$  is then represented by  $r$  (in the same way the signer represented  $\mathbf{V}$  by  $p$ ) and it is checked whether  $r = p$ .

Note that  $\mathbf{P} = v * \mathbf{G} + u * \mathbf{PUA} + w * \mathbf{R} = v * \mathbf{G} + w * (\mathbf{H}(\mathbf{IT}_A, \mathbf{PUA}) * \mathbf{PUA} + \mathbf{R}) = v * \mathbf{G} + w * \mathbf{Y}_A$ , which is the expression used in the DSA.

It is observed that *the single equality  $r=p$  establishes the validity of the message  $m$  as well as the validity of the public key  $\mathbf{Y}_A$  (which was implicitly calculated by Bob).*

The presented signature verification technique, including the implicit certificate verification, is executed by one operation of the form  $b * \mathbf{A} + d * \mathbf{C} + f * \mathbf{E}$  which is equivalent to a single ElGamal signature verification.

## 2.3. A modified DH fixed-key-agreement

A DH fixed-key-agreement concerns the basic implementation in which specific users always generate the same joint session key based on fixed exchanged values.

We present the following DH fixed-key-agreement technique.

After exchanging  $IT_A$  and  $IT_B$ , and the public values  $P_{UA}$  and  $P_{UB}$ , Alice and Bob respectively generate the session key  $K_{AB} = x_A * (H(IT_B, P_{UB}) * P_{UB} + R)$  and

$$K_{BA} = x_B * (H(IT_A, P_{UA}) * P_{UA} + R).$$

A key-confirmation now follows. That is, the two users verify that they share an identical key by encrypting and decrypting a randomly selected value.

(The two keys equal, having the value  $x_A * x_B * G$ . This equality, in itself, does not prove yet that only the valid owners of  $P_{UA}$  and  $P_{UB}$  arrive at the value  $x_A * x_B * G$ . As shown later, the key-confirmation closes the certification loop.)

The presented technique is executed by two exponentiations (or by one operation of the form  $b * A + d * C$ , by first calculating  $x_A * H(IT_B, P_{UB})$  modulo the order of  $G$ ).

#### 2.4. A modified DH ephemeral-key-agreement

A DH ephemeral-key-agreement, as treated in Section 1.2, concerns an implementation in which two specific users generate a different session key whenever they communicate, based on an ephemeral value generated by each party.

We present the following DH ephemeral-key-agreement technique.

Alice generates a random  $p_{VA}$ , calculates the ephemeral value  $E_{VA} = p_{VA} * G$  and submits  $IT_A$ ,  $P_{UA}$  and  $E_{VA}$  to Bob.

Bob performs the symmetric operations.

Alice and Bob respectively generate the ephemeral session key

$$K_{AB} = p_{VA} * (H(IT_B, P_{UB}) * P_{UB} + R) + (x_A + p_{VA}) * E_{VB}$$

$$K_{BA} = p_{VB} * (H(IT_A, P_{UA}) * P_{UA} + R) + (x_B + p_{VB}) * E_{VA}$$

A key-confirmation now follows.

(The two keys equal the value  $p_{VA} * x_B * G + x_A * p_{VB} * G + p_{VA} * p_{VB} * G$ . It is noted that the keys still equal if  $(x_A + p_{VA}) * EV_B$  and  $(x_B + p_{VB}) * EV_A$  are respectively replaced by  $x_A * EV_B$  and  $x_B * EV_A$ . The reasoning behind the presented choice follows in Part 4.)

$K_{AB}$  can be expressed as  $p_{VA} * R + [p_{VA} * H(IT_B, PUB)] * PUB + (x_A + p_{VA}) * EV_B$ . That is,  $K_{AB}$  is calculated as an operation of the form  $b * A + d * C + f * E$ . Furthermore, the group-point  $PA = p_{VA} * R$  can be calculated by Alice off-line, together with  $EV_A$ . (i.e., the random  $p_{VA}$  and then  $EV_A$  and  $PA$  can be generated prior to the communication with Bob.)

When Alice receives the values  $PUB$  and  $EV_B$ , she can calculate  $[p_{VA} * H(IT_B, PUB)] * PUB + (x_A + p_{VA}) * EV_B$  and add  $PA$  to the result. This way, the complete process is executed by two off-line exponentiations (the generation of  $EV_A$  and  $PA$ ) and one on-line operation of the form  $b * A + d * C$ , instead of the operation  $b * A + d * C + f * E$ .

## 2.5. A modified MQV key-agreement

The static public keys used in an MQV key-agreement are submitted with a certificate. Since the certificate is a fixed value, it cannot provide in an implied way or any other way the dynamic certification needed for verifying the authenticity of the ephemeral value submitted by a user. This dynamic certification is essentially achieved by a key-confirmation. We show next how the MQV version of Section 1.2 can be modified such that the key-confirmation implicitly provides for certification of the static public keys  $Y_A$  and  $Y_B$ , on top of the implied certification of the ephemeral values  $EV_A$  and  $EV_B$ .

Alice calculates the scalar values  $p_A = p_{VA} + T(EV_A) * x_A$ ,  $q_A = p_A * T(EV_B)$  and  $r_A = q_A * H(IT_B, PUB)$ . Bob calculates, symmetrically, the values  $p_B$ ,  $q_B$  and  $r_B$ . The session key is then  $K_{AB} = p_A * EV_B + r_A * PUB + q_A * R$  and  $K_{BA} = p_B * EV_B + r_B * PUB + q_B * R$ . The calculation of the session key is followed by a key-confirmation.

A complete MQV key-agreement process is therefore performed here by one operation of

the form  $b \cdot \mathbf{A} + d \cdot \mathbf{C} + f \cdot \mathbf{E}$ .

To realize why  $K_{AB} = K_{BA}$  note that

$$\begin{aligned} K_{AB} &= (p_{VA} + T(\mathbf{EV}_A) \cdot x_A) \cdot (\mathbf{EV}_B + T(\mathbf{EV}_B) \cdot H(IT_B, \mathbf{PUB}) \cdot \mathbf{PUB} + T(\mathbf{EV}_B) \cdot \mathbf{R}) = \\ &= (p_{VA} + T(\mathbf{EV}_A) \cdot x_A) \cdot (\mathbf{EV}_B + T(\mathbf{EV}_B) \cdot (H(IT_B, \mathbf{PUB}) \cdot \mathbf{PUB} + \mathbf{R})) = \\ &= (p_{VA} + T(\mathbf{EV}_A) \cdot x_A) \cdot (\mathbf{EV}_B + T(\mathbf{EV}_B) \cdot \mathbf{Y}_B) = K_{BA} \end{aligned}$$

## 2.6. A further modification: the 'you are OK if I am OK' principle

*Further modifying the DH fixed-key-agreement of Section 2.3.*

The fixed session key generated by Alice and Bob according to the technique presented in Section 2.3 is, respectively,

$$K_{AB} = x_A \cdot (H(IT_B, \mathbf{PUB}) \cdot \mathbf{PUB} + \mathbf{R}) \quad \text{and} \quad K_{BA} = x_B \cdot (H(IT_A, \mathbf{PUA}) \cdot \mathbf{PUA} + \mathbf{R}).$$

These can be re-written as

$$K_{AB} = [x_A \cdot H(IT_B, \mathbf{PUB})] \cdot \mathbf{PUB} + x_A \cdot \mathbf{R} \quad \text{and} \quad K_{BA} = [x_B \cdot H(IT_A, \mathbf{PUA})] \cdot \mathbf{PUA} + x_B \cdot \mathbf{R}.$$

Note that  $x_A \cdot \mathbf{R}$  is a fixed group-point which can be pre-calculated and stored by Alice. After receiving  $IT_B$  and  $\mathbf{PUB}$  Alice can calculate  $[x_A \cdot H(IT_B, \mathbf{PUB})] \cdot \mathbf{PUB}$  and add the pre-calculated  $x_A \cdot \mathbf{R}$  to the result. Similar operations apply to Bob. The presented technique then facilitates the execution of a fixed-key agreement, including a mutual authentication of the participants, by a *single* exponentiation.

*Further modifying the DH ephemeral-key-agreement of Section 2.4.*

The ephemeral session key generated by Alice and Bob according to the technique presented in Section 2.4 is, respectively,

$$\begin{aligned} K_{AB} &= p_{VA} \cdot (H(IT_B, \mathbf{PUB}) \cdot \mathbf{PUB} + \mathbf{R}) + (x_A + p_{VA}) \cdot \mathbf{EV}_B \quad \text{and} \\ K_{BA} &= p_{VB} \cdot (H(IT_A, \mathbf{PUA}) \cdot \mathbf{PUA} + \mathbf{R}) + (x_B + p_{VB}) \cdot \mathbf{EVA} \end{aligned}$$

These values can be re-written as:

$$K_{AB} = [p_{VA} * H(IT_B, PUB)] * PUB + (x_A + p_{VA}) * (EV_B + R) - x_A * R \quad \text{and}$$

$$K_{BA} = [p_{VB} * H(IT_A, PUA)] * PUA + (x_B + p_{VB}) * (EV_A + R) - x_B * R$$

As indicated before, the fixed group-point  $x_A * R$  can be pre-calculated and stored by Alice.

When Alice receives  $IT_B$  and  $PUB$  she can calculate  $[p_{VA} * H(IT_B, PUB)] * PUB + (x_A + p_{VA}) * (EV_B + R)$  and subtract the pre-calculated  $x_A * R$  from the result. Similar operations are performed by Bob. The presented technique can then be executed by one operation of the form  $b * A + d * C$ , which is preceded by the calculation of  $EV_A = p_{VA} * G$ .

To further clarify the observations made next, let us assume that the value  $EV_B + R$ , which appears in the expression of  $K_{AB}$ , was calculated by Bob and not by Alice. That is, Bob transmits the value  $VB = EV_B + R$  to Alice, instead of transmitting  $EV_B$ . Alice then calculates  $K_{AB} = [p_{VA} * H(IT_B, PUB)] * PUB + (x_A + p_{VA}) * VB - x_A * R$ . Noting this expression, it is observed that *the reference to the public key R of the trusted third party was effected prior to the communication of Alice with Bob*. The same applies to the expression  $K_{AB} = [x_A * H(IT_B, PUB)] * PUB + x_A * R$  specified above for the fixed-key-agreement.

After a successful key-confirmation, Alice, who knows that her own personal keys  $x_A$  and  $PUA$  are valid (i.e., they were provided to her by a recognized trusted third party), is assured that Bob has also used valid personal keys. This introduces a principle termed as 'you are OK if I am OK', where *Alice can effect a key-agreement process with another party just by knowing that her own personal keys are valid*, and without referring during the key-agreement process to the public key of a trusted third party.

The details discussed above concern the case where both parties use the services of the same trusted third party, and therefore refer to the same public key  $R$ . If Alice and Bob use the services of different trusted third parties, the saving in computational complexity based on the 'you are OK if I am OK' principle is still achieved if a participant knows in advance the public key of the trusted third party of his counterpart. Realistically, this is the case in most practical circumstances, as Bob must know the public key of Alice's trusted

third party regardless of his communication with Alice, and vice versa.

### **3. Claimed attributes and advantages of the technique**

#### **3.1. Savings in computational efforts**

The savings in the computational efforts introduced by the presented techniques, when compared to customary DL/EC techniques, are summarized next in detail. These concern an overall saving in exponentiation operations, as well as parallelism which further expedites the process, where the traditional serial two operations associated with verifying a certificate and verifying the validity of a submitted dynamic value are combined into a single operation. The indicated figures were substantiated in the preceding sections.

The very significant advantages of the presented techniques, over customary techniques, are clear for the case where  $b*A + d*C$  or  $b*A + d*C + f*E$  are executed by individual exponentiations as well as the case where speedup methods are used.

#### *Verifying a signature of the DSA type*

Customary implementation: Two operations of the form  $b*A + d*C$ , one for certificate verification and one for actual signature verification.

Presented technique: One operation of the form  $b*A + d*C + f*E$ .

#### *DH fixed-key-agreement*

Customary implementation: One operation of the form  $b*A + d*C$  for certificate verification, and one exponentiation for session-key generation.

Presented technique: A single exponentiation (or two exponentiations when not using the “You are OK if I am OK” principle).

### *DH ephemeral-key-agreement*

Customary implementation: Three exponentiations (generating an ephemeral value, signing this value, and generating the session-key); two operations of the form  $b*A + d*C$  (verifying the certificate and the signature).

Presented technique: One exponentiation (generating the ephemeral value) and one operation of the form  $b*A + d*C$  (or one operation of the form  $b*A + d*C + f*E$  when not using the “You are OK if I am OK” principle).

### *MQV key-agreement*

Customary implementation: One exponentiation (generating the ephemeral value); two operations of the form  $b*A + d*C$  (for certificate verification and session-key generation).

Presented technique: One exponentiation and one operation of the form  $b*A + d*C + f*E$ .

## **3.2. Cutting down communication overhead**

The presented techniques facilitate the replacement of separately submitted public key (which is a group-point), and a certificate (which is a DSA signature, consisting of a pair of scalars), by a single submitted value (a group-point), whose size is that of the public key. This significantly cuts down communication overhead.

## **3.3. Enhancing implementation efficiency and cutting down management overhead**

Abolishing an explicit certification in key-agreement schemes saves a need to include a signature verification procedure in the execution package, enhancing implementation efficiency.

Furthermore, abolishing the need for generating, storing and submitting an explicit certificate significantly cuts down management overhead.

## 4. Security assessment and considerations

### 4.1. On the possibility of forging the user's keys defined in Section 2.1

#### 1. *Extracting the secret key of the trusted third party:*

The secret key of the trusted third party is  $d$ , while  $\mathbf{R} = d * \mathbf{G}$  is known. Users know multiple values of the form  $H(IT_A, (h_A + k_A) * \mathbf{G}) * k_A + d$ , for known  $IT_A$ ,  $h_A$  and  $(h_A + k_A) * \mathbf{G}$ . The trusted third party prevents the extraction of  $d$  by associating each user with a different, randomly generated,  $k_A$ . It is further noted that the system secret  $d$  is revealed if  $H(IT_A, (h_A + k_A) * \mathbf{G}) * k_A$  equals any fixed or known value (modulo the order of the generating group-point  $\mathbf{G}$ ). The trusted third party, who has control over  $H(IT_A, (h_A + k_A) * \mathbf{G}) * k_A$  by choosing the random  $k_A$ , should take care of this threat.

#### 2. *Preventing a "first party attack":*

A "first party attack" concerns the case where a user repudiates the validity of a cryptographic scenario in which he participated, claiming that he had "weak" secret keys which were mathematically discovered. The presented key issuing process inherently prevents such an attack, as the the trusted third party has control over the randomness of the user's secret key (while the trusted third party still does not know this key).

#### 3. *Generating values $IT_A$ , $x_A$ and $\mathbf{P}U_A$ having a valid interdependence:*

The validity of the presented technique depends on the inability of any party to generate values  $x_A$ ,  $IT_A$  and  $\mathbf{P}U_A$  such that the interdependence  $x_A * \mathbf{G} = H(IT_A, \mathbf{P}U_A) * \mathbf{P}U_A + \mathbf{R}$  holds. That is, no party beside Alice (as defined in Section 2.1) should be able to submit values that stand for  $IT_A$  and  $\mathbf{P}U_A$  such that he knows  $\log[H(IT_A, \mathbf{P}U_A) * \mathbf{P}U_A + \mathbf{R}]$ .

It is first observed that a forger cannot know  $\log \mathbf{P}U_A$  while he manufactured himself  $\mathbf{P}U_A$ ,  $x_A$  and  $IT_A$ , where  $x_A * \mathbf{G} = H(IT_A, \mathbf{P}U_A) * \mathbf{P}U_A + \mathbf{R}$ , since he would then be able to recover  $\log \mathbf{R}$ , thereby being able to perform a general log operation.

There are three possible approaches in trying to falsify  $x_A$ ,  $IT_A$  and  $\mathbf{PUA}$ , while not knowing  $\log \mathbf{PUA}$ , such that  $x_A * \mathbf{G} = H(IT_A, \mathbf{PUA}) * \mathbf{PUA} + \mathbf{R}$ :

1. Select a  $\mathbf{PUA}$  (whose log is not known) and a scalar  $x_A$ , and then recover a scalar  $v$  such that  $\mathbf{R} = x_A * \mathbf{G} - v * \mathbf{PUA}$ . This attempt would fail as it would involve a log operation, regardless of the fact that  $v$  should also equal  $H(IT_A, \mathbf{PUA})$ .
2. Select a  $\mathbf{PUA}$ , calculate  $H(IT_A, \mathbf{PUA})$  and then recover a scalar  $x_A$  such that  $x_A * \mathbf{G} = H(IT_A, \mathbf{PUA}) * \mathbf{PUA} + \mathbf{R}$ . This, again, would fail as it involves a log operation.
3. Select values  $v$  and  $x_A$  and then determine the value  $\mathbf{PUA} = v^{-1} * (x_A * \mathbf{G} - \mathbf{R})$ . Even if it is possible to recover a  $\mathbf{PUA}$  such that  $v$  acts as  $H(IT_A, \mathbf{PUA})$ ,  $\mathbf{PUA}$  has to simultaneously satisfy two independent constraints, which appears to be impossible to achieve.

Trying to force  $(IT_A, \mathbf{PUA}) * \mathbf{PUA} = 0$  (regardless of whether this is possible or not), will also not help, since the forger then has to produce an  $x_A$  such that  $x_A * \mathbf{G} = \mathbf{R}$ . That is, he has to perform a log operation.

It is therefore claimed that a user who illegally tries to present himself as Alice cannot generate values  $x_A$ ,  $IT_A$  and  $\mathbf{PUA}$  such that  $x_A * \mathbf{G} = H(IT_A, \mathbf{PUA}) * \mathbf{PUA} + \mathbf{R}$ .

A trial to generate valid values  $IT_f$ ,  $x_f$  and  $\mathbf{PU}_f$  out of given valid values  $IT_A$ ,  $x_A$  and  $\mathbf{PUA}$  is not different from the above failed trials for generating  $IT_A$ ,  $x_A$  and  $\mathbf{PUA}$  from scratch.

It should further be demanded that the sum of two valid keys will not yield a valid key. This demand is satisfied if the transformation  $H$  is non-linear.

#### *4. Falsifying the role of the trusted third party:*

As described in Section 2.1, Alice verifies the validity of the values  $p_A$  and  $\mathbf{PUA}$ , issued to her, by checking whether  $p_A * \mathbf{G} = h(IT_A, \mathbf{PUA}) * (\mathbf{PUA} - h_A * \mathbf{G}) + \mathbf{R}$ , where  $\mathbf{R}$  is the public key of the trusted third party. Falsifying the role of the trusted third party involves here an ability to produce values  $p_A$  and  $\mathbf{PUA}$  such that  $p_A = \log[h(IT_A, \mathbf{PUA}) * (\mathbf{PUA} - h_A * \mathbf{G}) + \mathbf{R}]$ .

This cannot be done in view of the above considerations.

#### 4.2. Summarizing the required features of H

The transformation  $H(IT_A, PUA)$  can be a simple xor operation between  $IT_A$  and any one-to-one scalar representative of  $PUA$ . (When operating over an elliptic curve, this representative can be the x coordinate of  $PUA$ .)

It should be noted that a user can here make changes in both  $IT_A$  and  $PUA$ , while keeping  $H(IT_A, PUA)$  unchanged. That is, it is not demanded that H should be collision-free. This is based on the observation that the change enforced into  $PUA$  (in order that  $H(IT_A, PUA)$  remains unchanged for invalid  $IT_A$ ) would necessitate a corresponding change in the secret key  $x_A = H(IT_A, PUA) * \log PUA + d$ . The forger, who changed  $IT_A$ , is then unable to come up with a correct new  $x_A$  due to the new  $\log PUA$ .

It is possible to use, in any case, the hash transformation which serves in general certification applications (where the certificate is the signature of the trusted party on a value  $H(IT_A, PUA)$  which strongly combines the user's identification details or attributes and his public key) while utilizing the significant advantages of the techniques presented in Part 3.

#### 4.3. On the security of the modified DSA technique of Section 2.2

The difference between the certificate-less DSA technique of Section 2.2 and a standard DSA lies in the fact that the public key  $Y_A$  of the signer is implicitly generated by the verifier, rather than being received with a certificate.

The operation  $P = v * G + w * Y_A$ , performed in the standard DSA procedure, is replaced with  $P = v * G + w * (H(IT_A, PUA) * PUA + R)$ , which is technically executed by the parallel operation  $P = v * G + [w * H(IT_A, PUA)] * PUA + w * R$ .

A forger does not know  $\log[H(IT_A, PUA) * PUA + R]$  in view of preceding considerations. Also, he cannot submit  $IT_A$  (pretending to be Alice) and his own forged public value  $PUF$

such that he knows  $\log[H(IT_A, \mathbf{PUf}) * \mathbf{PUf} + \mathbf{R}]$ . Therefore, the existence of the single equality  $r = p$ , which assures the verifier that the signer knows  $x_A = \log Y_A$  in the lines of the DSA, provides for two purposes. It establishes the validity of the signed message, based on the fundamental principle of the DSA, and it further establishes the signer's valid ownership of the public key  $Y_A$ .

The main observation made above is summarized as follows: Alice's ability to sign a message is based on her knowledge of the log of the public key she submits to Bob. In the presented technique, Alice's public key is calculated by Bob, whereas Alice can know the log of that public key only if she is the valid owner of that key. That is, her knowledge of the log of the said public key is not only a necessary condition for Alice's ability to sign the message, but it further guarantees her valid ownership of the key itself, whereas Alice's knowledge of the said log is established by Bob by a single check.

#### **4.4. On the security of the presented key-agreement techniques**

The session key generated by the fixed-key technique of Section 2.3 is

$$K_{AB} = x_A * (H(IT_B, \mathbf{PUB}) * \mathbf{PUB} + \mathbf{R}) ; K_{BA} = x_B * (H(IT_A, \mathbf{PUA}) * \mathbf{PUA} + \mathbf{R})$$

The session key generated by the ephemeral-key generation technique of Section 2.4 is

$$K_{AB} = p_{VA} * (H(IT_B, \mathbf{PUB}) * \mathbf{PUB} + \mathbf{R}) + (x_A + p_{VA}) * \mathbf{EVB};$$

$$K_{BA} = p_{VB} * (H(IT_A, \mathbf{PUA}) * \mathbf{PUA} + \mathbf{R}) + (x_B + p_{VB}) * \mathbf{EVA}$$

The security of the technique depends again on the inability of a forger to know  $\log[H(IT_A, \mathbf{PUA}) * \mathbf{PUA} + \mathbf{R}]$  or  $\log[H(IT_B, \mathbf{PUB}) * \mathbf{PUB} + \mathbf{R}]$  or to falsify values in a way that enables him to know any of the said logs.

Only the valid Alice and Bob, who use their secret keys  $x_A$  and  $x_B$ , would then end up with the same session key, a fact which is established by the key-confirmation.

Similar considerations apply to the modified MQV technique of Section 2.5.

The key-confirmation concerns a single check made by each participant, enabling him to implicitly verify that his counterpart knows the log of the public key  $Y_A$  or  $Y_B$ , whereas the calculation of the public key is implied in the process. Like the case with the DSA, treated in the preceding Section, Alice's knowledge of the log of the said public key, established by Bob by a single check, is not only a necessary condition for a successful DH key-agreement, but it further guarantees Alice's valid ownership of her claimed public key.

*Forward secrecy (concerning the ephemeral-key generation technique of Section 2.4)*

Forward secrecy concerns the prevention of the possibility that the disclosure of any static secret value reveals session keys previously generated by two communicating parties. The said static value can either be the secret key of the trusted third party or users' secret key or a secret value common to two or more specific communicating users.

It was shown that the generated ephemeral session key is  $K_{AB} = K_{BA} = p_{VA} * x_B * G + x_A * p_{VB} * G + p_{VA} * p_{VB} * G$ , where  $x_A * G$ ,  $x_B * G$ ,  $p_{VA} * G$  and  $p_{VB} * G$  are publicly known. A disclosure at any stage of  $x_A$  and  $x_B$  would not reveal the value of a previously generated  $K_{AB}$  or  $K_{BA}$  due to the addend  $p_{VA} * p_{VB} * G$ . This explains why it is preferred to use the expression  $K_{AB} = p_{VA} * (H(IT_B, PUB) * PUB + R) + (x_A + p_{VA}) * EV_B$  rather than the expression  $K_{AB} = p_{VA} * (H(IT_B, PUB) * PUB + R) + x_A * EV_B$  which can also yield a valid key generation.

There are no static values common to two specific communicating users, whose disclosure reveals the value of previously generated session keys, as all the addends in the expression  $p_{VA} * (H(IT_B, PUB) * PUB + R) + (x_A + p_{VA}) * EV_B = p_{VA} * H(IT_B, PU_j) * PUB + p_{VA} * R + x_A * p_{VB} * G + p_{VA} * p_{VB} * G$  are ephemeral. That is, a key does not consist of a static part, which is fixed to two specific communicating users, where this static part encrypts an ephemeral value, and where the later leakage of this static part would reveal the value of generated session keys.

## **5. Known limitations and disadvantages**

In the proposed technique, the keys of the trusted third party and users' keys are of the same size. This can be considered a disadvantage when compared to implementations in which explicit independent certification of users' static keys is performed, and where the keys of the trusted third party can be made larger than users' keys.

However, the described apparent disadvantage of the proposed technique is common to all systems offering implied key certification, including identity-based systems.

## **6. Intellectual property issues**

A patent application on the proposed technique has been filed. A letter of assurance of "reasonable and non-discriminatory" patent licensing will be provided if this technique is accepted as part of p1363a.

## **References**

- [1] Approval of Federal Information Processing Standards Publication 186, Digital Signature Standard (DSS), "Federal Register, v. 58, n. 96, 19 May 1994, pp. 26208-26211.
- [2] A. J. Menezes, P. C. van Oorschot and S. A. Vanstone, "*Handbook of Applied Cryptography*", Chapter 14, pp. 617-618, CRC Press, 1996.
- [3] W. Diffie and M. Hellman, "New directions in cryptography", *IEEE Trans. on Information Theory*, IT-22, 1976, pp. 644-654.
- [4] Law, A. Menezes, M. Qu, J. Solinas and S. Vanstone, "An efficient protocol for authenticated key agreement", Technical report CORR 98-05, Dept. of C&O, University of Waterloo, Canada, March 1998.