International Journal of Multimedia and Ubiquitous Engineering
Vol. 3, No. 3, July, 2008

Dynamic Multithreshold Signature without the Trusted Dealer

Jacek Pomykała
1. Department of Mathematics, Informatics and Mechanics
Warsaw University
Institute of Mathematics
pomykala@mimuw.edu.pl

Tomasz Warchoł
Department of Mathematics, Informatics and Mechanics
Warsaw University
t.warchol@students.mimuw.edu.pl

Abstract

In the article a new threshold signature for dynamic groups without the trusted dealer is presented. The scheme is based on the Boneh, Lynn and Shacham short signature from the Weil pairing. The scheme provides an efficient solution especially for the family of dynamic groups. We prove the security of the scheme in the random oracle model assuming the chosen target CDH problem.

1. Introduction

Threshold cryptography gives the ways to distribute trust throughout a group and increase the availability of cryptographic systems. The threshold cryptosystems (including the digital signatures) are traditionally based on the corresponding single-server system, together with the setting of the secret sharing protocol. Two major types of threshold cryptosystems deal with the group being the encryption group (threshold signature) or the decryption group (threshold decryption system). The dynamics of the corresponding group members and the flexibility in the threshold size *t* provides one of the important criteria for the application of the considered cryptosystem (cf. eg. [5] for the threshold decryption system). Here we shall be concerned with the threshold signature scheme of the type *(t, n)*, where the group has the cardinality *n* and at least *t* members have to sign the message in order to pass through the verification process.

The familiar way of realization of such *(t, n)*-schemes is based on the application of the Lagrange's interpolation formula - the protocol being invented by Shamir [11]. The group generates a random polynomial *f* over the finite field with *f(0)* being the group secret and *f(i)*,(*i* distinct than zero), being the shares of the players. We shall be interested in situation where the fraction t/n of signers should overcome the given value α while the number of the group members is varying. This makes the traditional approach rather impractical since the addition (or deleting) of the group members requires each time the generation of a new polynomial f.

Another interesting challenge for the threshold type schemes is to deal with the value *t/n* depending for instance on the message "priority". Then the varying threshold size (depending on *n* and the priority of *m*) implies again that the traditional approach is is not useful from the computational cost point of view.

In this paper we present an efficient solution for the dynamical groups with the variable threshold bound. In fact we consider the more general situation, when there is a family of dynamical groups instead of one group. We were able to handle this more general situation by joining the secret sharing idea together with the certification system device which allows to distinguish the members of distinct groups of the family. The efficiency of the system is due to the fact that the "active" signing is related only to the corresponding group members, leaving the others to accept their "decisions".

Due to elegant structure of GDH groups and the simplicity of the basic signature scheme the presented system is provably secure. The security is proved in the random oracle model, provided the number of corrupted players in the family of groups satisfies the suitable estimates (see Theorem 1) and the "chosen target CDH problem" is computationally hard.

2. Related work

Let $G_1 = (G_1, +)$ and $G_2 = (G_2, *)$ be finite groups of prime order q and $e: G_1 \times G_1 \to G_2$ be the map satisfying the following conditions:

1.there exists $P \in G_1$ such that $e(P,P) \neq 1$ (non-degeneracy)

2. $\forall_{P,Q,R \in G_1} \forall_{a,b \in \mathbb{Z}} e(aP, bQ) = e(P,Q)^{ab}$ (bilinearity)

3.there exists an efficient algorithm computing the value $e(P,Q)$ for any $P, Q \in G_1$ (computability)

The bilinear group structure (G_1, G_2, e) implies that the corresponding Decisional Diffie-Hellman problem (DDH) in G_1 is easy. Namely it is sufficient to check if the corresponding 4-tuple (P, aP, bP, cP) is Diffie-Hellman quadruple, i.e. satisfies the inequality: e(P, cP)= e(aP, bP). Alternatively we shall write it as $(P, aP, bP, cP) = DH$ in the sequel. In this paper we assume that G_1 is the suitable additive group of points of an elliptic curve E/F_p, G_2 is the multiplicative group of the finite field F_{p^2} and e is the corresponding Weil Pairing as defined in [1].

We call such G_1 the Gap Diffie-Hellman (GDH) group if moreover the corresponding Computational Diffie-Hellman problem (CDH) is hard in G_1 (and therefore also the Discrete Logarithm problem in G_2 ([7]). The first such an example was given in [6]. Boneh, Lynn and Shacham [2] have proposed a new signature scheme working in GDH groups. Boldyreva [3] has extended the above protocol for the threshold signature scheme. The concept of the threshold-flexible signature based on the RSA cryptosystem [10] has appeared in [8]. The threshold signature scheme for the dynamic group has been considered quite recently in [9] with the assumption that there exists a trusted Dealer and the Administrator of the system. The present work is a further development and improvement of the above paper. The improvement concerns in particular the following aspects:

1.avoids the participation of the Administrator and the trusted Dealer in the protocol

2.corresponds to more realistic scenario of dynamic groups with group members responsible for the generation and the distribution of secret shares

3.is more flexible in dynamical aspect (it handles the case of adding and deleting of the users as well)

The suitable changes in the computational cost related to the presented protocol follow directly from the application of the results of the paper [4].

3. System and the communication model

Let the positive integers $l \in \mathbb{N}$ and $\theta > 1$ be the fixed parameters. We consider the triple (Γ, α, β), where:

1.$\Gamma = \cup \Gamma_i$ and $K \leq |\Gamma_i| < \theta K$, for some positive integer K and any $\theta > 1$, where $\forall_{i \neq j} \Gamma_i \neq \Gamma_j$

2. $\alpha : \{1,2,\ldots,l\} \rightarrow [0,1]$, $\alpha(i) = \alpha_i$ is the threshold level in the group Γ_i, which means that at least $\lceil \alpha_i |\Gamma_i| \rceil$ members of Γ_i should sign the given message,

3. $\beta : \{1,2,\ldots,l\} \rightarrow [0,1]$, $\beta(i) = \beta_i$ is the corruption level in the group Γ_i, which means that at least $\lfloor \beta_i |\Gamma_i| \rfloor$ players can be corrupted in Γ_i.

Here $\lceil z \rceil$ denotes the smallest integer greater or equal to z and $\lfloor z \rfloor$ denotes the biggest integer smaller or equal to z.

The system consists of two parties: the group Γ, which consists of the users belonging to the family of l groups $F_\Gamma = \{\Gamma_1, \Gamma_2, \ldots, \Gamma_l\}$, connected by the secure channels, and connected with them by the secure channels the group public server (GPS) where all the public data (of participated parties) can be sent and read.

3.1 The action

We are given $(l, K, \theta, \alpha, \beta)$.

1.System creator generates the bilinear group structure $\langle G \rangle = (G_1, G_2, e, P)$ with the corresponding cryptographic keys, where $P \neq 0$ is a random point of G_1.

2.The group members execute the distribution algorithm (see below) to generate the secret shares of the corresponding certification secret $s = s_{cert}$, which will be used to sign certificates. After that each member has one share for signing certificates.

3.Group members execute the distribution algorithm to generate the signing secret shares. At the end each member has θ distinct shares of a secret $s = s_{sig}$. Both secrets s_{cert} and s_{sig} can be reconstructed on the basis of at least *d + 1* shares.

4.Certificate signing phase: each member asks his group for the signature of his own certificate. Other members use their signing secret shares to sign this certificate. If at least $\alpha |G_i|$ members apply the correct shares, then the correct signature can be computed. All signed certificates are published in GPS.

5.In case of joining a new member he must first obtain a free share from a group member. Then he asks the group for the signature. Group checks correctness of data (validates the share, checks if the share was not used before) and creates the certificate for the new member signed by at least $\alpha |G_i|$ group members. The signed certificate is published in GPS.

6.Deleting members is done by introducing time-limit certificates.

7.Let *m* be the message to be signed by the chosen group $\Gamma_i \in F_\Gamma$ with the corresponding threshold level α_m , which may depend on the message *m* to be signed.
8.Any user from Γ_i checks if the condition $\alpha_m(i) \geqslant \alpha_i$ is satisfied, then he generates the partial signature and publishes it in GPS.
9.If the number of valid signature shares from Γ_i is at least $\alpha_i |\Gamma_i|$, then the members from Γ compute the remaining partial signatures and publish them on the GPS.
10.If the number of valid signature shares is at least *d+1* and at least $\alpha_m(i)|\Gamma_i|$ in Γ_i then the members of Γ_i compute and publish on GPS the complete signature.
11.One can check the correctness of signature using the verification algorithm.

4. The protocol

The Dynamic Multi-Threshold Signature Scheme (DMTSS) is the 6-tuple: DMTSS=(Generate, Register, Sign, Check, Combine, Verify).

To initialize the system the creator starts from the Generate algorithm. The input is the security parameter *k*. The output is the description of the bilinear group structure $(G_{1,} G_{2,} e)$, together with nonzero $P \in G_1$, and hash functions *Q*, *H*, *R*.

Algorithm 1. *Generate(1^k)*
$\langle G \rangle \leftarrow (G_{1,} G_{2,} e, P)$
$Q: \{0,1\}^* \rightarrow G_1$
$H: \{0,1\}^* \rightarrow G_1$
$R: \{0,1\}^* \rightarrow G_1$
$return(\langle G \rangle, P, Q, H, R)$

In the next steps we will use the distribution algorithm. This is an algorithm described in [4], and it's goal is to generate some secret and distribute securely the shares of it among the group members, so that they could reconstruct it when signing the message. Though the algorithm described there works for the discrete-log based cryptosystems, it can be easily converted to the bilinear pairing based cryptosystems.

To be more precise the group members first perform distribution algorithm to generate signing secret $s = s_{cert} = a_0$ with a corresponding polynomial $f = a_0 + \ldots a_d x^d$ of degree *d* and the public key $P_{cert} = sP$. The same distribution algorithm is then executed to obtain the signing group secret s_{sig} with the corresponding shares for every member. Each member executes it as θ personalities, to generate θ shares - one for himself and the others for the new members in the future. After that each member has θ pairs *(x, y)*, where *y = g(x)* for some generated polynomial $g = b_0 + \ldots b_d x^d$ with a group secret $b_0 = s_{sig}$.

The following algorithm is executed to create a certificate for the user of given identity *ID*. The user receives a share (x_{ID}, y_{ID}) and computes $P_{ID} = Q(ID)$ and $P_{usr} = y_{ID} P_{ID}$. Algorithm takes 4-tuple $(ID, x_{ID}, P_{usr}, group)$ as input and returns the certificate of the given user, which states that the given identity appertains to the suitable group.

Algorithm 2. *Register* $(ID, x_{ID}, P_{usr}, group)$
$P_{ID} \leftarrow Q(ID)$
$w_{ID} \leftarrow (P_{ID}, P_{usr}, group, x_{ID})$
$\tau_{ID} \leftarrow sR(w_{ID})$
$cert_{ID} = (w_{ID}, \tau_{ID})$
$return\, cert_{ID}$

Multiplying by *s* here is done by the group. Each member multiplies using its secret share, and then results are combined to obtain the complete signature.

The certificate should be validated before signed by the group. In the first step any user of the given group Γ_i checks if the certificate is correct, i.e. if $(R(w_{ID}), \tau_{ID}, P, P_{cert})$ is DH quadruple, where $w_{ID} = (P_{ID}, P_{usr}, group, X_{ID})$. In the second it is checked if none of already accepted certificates has the same value $x = x_{ID}$. Third step is performed only in case of joining the new member. We have already the group Γ_i with the validated list of certificates which has to check if the pair *(x, y)* received by the new user is an interpolation point of the polynomial *g*. In this connection the group Γ_i generates

the message $m = H(ID, group)$ and creates two signatures of it: one using the trusted quorum of Γ (with at least $\lceil \alpha_i | \Gamma_i | \rceil$ valid partial signatures of members from Γ_i) together with the signature share of the added user and the other of the quorum without the added user. If both signatures are the same the trusted quorum can easily declare the certificate as validated. Otherwise the group declares the deception of the added user or the user who has given the share (each of the above cases can be easily distinguished).

The following algorithm has as input the description *M* of message to be signed (message *m*, destination group and threshold level), and the user's secret share with the corresponding certificate. It returns the partial signature of the type: (message description, signature share, certificate). It is executed by any group member.

Algorithm 3. *Sign* $(M, y_{ID}, cert_{ID})$
$parse\, M\, as (m, group, \alpha_m)$
$\sigma_{ID}(m) \leftarrow y_{ID} H(m \| group \| \alpha_m)$
$return (M, \sigma_{ID}, cert_{ID})$

The next algorithm validates the above partial signatures. Validation concerns the correctness of the signature share and the validity of the certificate as well.

Algorithm 4. *Check* $(M, \sigma_{ID}, cert_{ID})$
$parse\, cert_{ID} as (w_{id}, \tau_{ID})$
$parse\, w_{ID} as (P_{ID}, P_{usr}, gr, x_{ID})$
$parse\, M\, as (m, group, \alpha_m)$
$r_1 \leftarrow (P, P_{cert}, R(w_{ID}), \tau_{ID}) = DH\ ?$
$r_2 \leftarrow (H(m \| group \| \alpha_m), \sigma_{ID}, P_{ID}, P_{usr}) = DH\ ?$
$if\, (r_1 = 1 \wedge r_2 = 2)$
$then return 1$
$else return 0$

The following algorithm combines the (validated) signature shares to compute the final signature. Both the algorithms *Check* and *Combine* are preformed by all members of the signing group.

Algorithm 5. *Combine* $((cert_{ID_1}, \sigma_{ID_1}), \ldots, (cert_{ID_n}, \sigma_{ID_n}))$
$parse\ cert_{ID_i} as (w_{ID_i}, \tau_{ID_i})$
$parse\ w_{ID_i} as (P_{ID_i}, P_{usr_i}, gr_{ID_i}, x_{ID_i})$
$S \leftarrow \{x_{ID_i} : i \in \{1, \ldots, n\}\}$
$\lambda_i \leftarrow \prod_{x \in S \setminus \{x_{ID_i}\}} \frac{(-x)}{(x_{ID_i} - x)}$
$\sigma \leftarrow \sum_{i=1}^{n} \lambda_i \sigma_{ID_i}$
$return\ \sigma$

The last algorithm is executed by any user of the system (verifier). It has as input the pair (message description, signature) and as output acceptation or rejection according to whether the corresponding 4-tuple is the Diffie-Hellman quadruple.

Algorithm 6. *Verify* (M, σ)
$parse\ M as (m, group, \alpha_m)$
$return (P, P_{sig}, H(m \| group \| \alpha_m), \sigma) = DH\ ?$

5. Security analysis

At the beginning of the game we assume that the adversary corrupts the set of players $F_{corr} = F_1, F_2, \ldots, F_c$. By this we mean that he has an access to all the data of the corrupted players (i.e. Their secret keys, free shares, certificates etc.). Also he can ask any player for the signature of any chosen message. Under the above assumptions we will prove:

Lemma 1. *It is hard to compute the valid partial signature of the chosen message without the knowledge of the corresponding secret key.*

Lemma 2. *All signed certificates are valid.*

By the word 'hard' we mean that there exist a polynomial reduction of this problem to the chosen-target CDH problem. It's detailed definition can be found in [3]. Shortly, we have a group *G* of a prime order *p*, some random $\kappa \in Z_p$ and two oracles: *CHALL* returning random distinct points from *G*, and *CDH* which for given $P \in G$ returns $Q \in G$, satisfying $Q = \kappa P$. The assumption says that there is no polynomial-time algorithm, which can 'answer' *n* questions of *CHALL* oracle, using less than *n* queries to *CDH* oracle.

Now the argument runs as follows: assume that exists an adversary, that can break our protocol in some way. Then, we can construct a simulator which will be indistinguishable from a real group implementing algorithm for adversary, and which will prepare data in some special way. After the successful attack on our simulator we could use this prepared

data and forged data from adversary to break the CDH problem. However this would contradict the hardness of the CDH problem.

Coming into details we first observe that all the shares are valid, i.e. every pair *(x, y)* belongs to the generated polynomial *g*. The proof can be found in [4].

Assume that we are given a GDH group *G* and two corresponding oracles: *CHALL* and *CDH* with the secret value κ .

We consider the following simulator:

Oracle 1. $Q(ID)$
$if\ ID \notin Q_{set}$
$then Q_{set} = Q_{set} \cup \{ID\}$
$Q[ID] \leftarrow random(G_1)$
$return Q[ID]$

Oracle 2. $R(x)$
$if\ x \notin R_{set}$
$then R_{set} = R_{set} \cup \{x\}$
$R[x] \leftarrow random(G_1)$
$return R[x]$

Oracle 3. $H(m)$
$if\ m \notin H_{set}$
$then H_{set} = H_{set} \cup \{m\}$
$H[m] \leftarrow CHALL()$
$return H[m]$

Algorithm 7. *Init* $(F_1, F_2, .., F_c)$
$\forall_{i \in \{1,..,c\}} X_i \leftarrow x_{F_i}; Y_i \leftarrow y_{F_i}$
$\forall_{i \in \{1,..,d-c\}} X_{c+i} \leftarrow random(\mathbb{Z}_q); Y_{c+i} \leftarrow random(\mathbb{Z}_q)$
$s \leftarrow random(\mathbb{Z}_q)$

Algorithm 8. *Translate* (x_0, σ_0, P_0, x')
$X_0 \leftarrow x_0; S \leftarrow \{X_0, .., X_d\}$
$$\forall_{i \in \{0,..,d\}} \lambda_i \leftarrow \frac{\prod_{x \in S \setminus \{x_i\}} x' - x}{\prod_{x \in S \setminus \{x_i\}} X_i - x}$$
$$\sigma' \leftarrow \sum_{i \in \{1,..,d\}} \lambda_i Y_i P_0 + \lambda_0 \sigma_0$$

Oracle 4. *CERT* $(group)$
$ID \leftarrow random(\{0,1\}^*)$
$P_{ID} \leftarrow Q(ID)$
$x_{ID} \leftarrow random(\mathbb{Z}_q)$
$P_{usr} \leftarrow Translate(0, CDH(P_{ID}), P_{ID}, x_{ID})$

$w_{ID} \leftarrow (P_{ID}, P_{usr}, group, x_{ID})$
$\tau_{ID} \leftarrow sR(w_{ID})$
$CERT_{set} = CERT_{set} \cup (R(w_{ID}), \tau_{ID})$
$return(w_{ID}, \tau_{ID})$

Oracle 5. $SIG(M, ID)$
$\sigma' \leftarrow CDH(H(M))$
$\sigma_{ID} \leftarrow Translate(0, \sigma', P, x_{ID})$
$return \sigma_{ID}$

First let us remark that by the Lagrange's polynomial interpolation formula the polynomial *f* of degree *d* is uniquely determined by the *d+1* points applying the following interpolation formula:

$f(x) = \sum_{x_i \in S} \lambda_{x_i} f(x_i)$, where $\lambda_{x_i} = \frac{\prod_{x_j \in S \setminus \{x_i\}} x - x_j}{\prod_{x_j \in S \setminus \{x_i\}} x_i - x_j}$, for any *x* and any set $S = \{x_0, x_1, .., x_d\}$.

This property is used in the simulator: the polynomial *f* is computed indirectly by its values in the set of points: *c* points from the certificates of corrupted players, *d-c* points generated randomly by the algorithm *Init*, and secret κ which is assumed to assign the value $f(0) = a_0$. Similarly the above interpolation formula is applied in the algorithm *Translate* which has as input the 4-tuple (x_0, σ_0, P_0, x'). Here for any fixed x_0, σ_0 is the suitable partial signature corresponding to the secret value attached to x_0 of the message with the hash equal to P_0 and x' is any element of $\mathbb{Z}_q$. The output of the algorithm is the partial signature of the same message but corresponding to the secret attached to x'.

In the simulator the oracle *H* transports the 'queries' of *CHALL*. The algorithm *Init* initializes the simulator generating the points needed for the computation of the interpolation polynomial and the signing secret key. The oracle *CERT* returns the certificate of the group member. All the parameters are chosen randomly with the uniform distribution. For the generation of keys we apply the *Translate* algorithm with the missing point $(0, \kappa)$. The secret point for signing is randomly chosen in *Init* algorithm. The oracle *SIG* returns the signature of a chosen player under a chosen message. Here the *CDH* oracle is used in order to get the missing interpolation point. *CERT* and *SIG* oracles provide all the information that the adversary can get from the system.

By Lagrange's interpolation all the corresponding values have the uniform distribution which is in accord to the generation of the system parameters. Therefore the simulator is undistinguishable from the adversary's system. Moreover the probability of generation of any collisions by the oracles *Q*, *R* or *H* is negligible.

Now we are ready to pass to the proper reduction proof.

Let us assume that there exists a polynomial-time algorithm *Cheat* that for a given message generates the false partial

signature. We consider first the pessimistic case when the adversary corrupts d players, hence he is missing only one partial signature to sign a chosen message.

Then we can construct the false environment for the adversary, in which he will forge the corresponding partial signature. Then from the generated data we will break the chosen-target CDH problem. Let us consider the following algorithm:

Algorithm 9. *Break* (F)
$Q_{set} \leftarrow \emptyset$; $H_{set} \leftarrow \emptyset$; $CERT_{set} \leftarrow \emptyset$
$init(F)$
$m \leftarrow random(\{0,1\}^*)$
$group \leftarrow random(\{1,\ldots,l\})$
$\alpha_m \leftarrow random([\alpha(group),1])$
$M \leftarrow (m, group, \alpha_m)$
$share \leftarrow Cheat(\langle G \rangle, M, Q, H, R, CERT, SIG, F)$
$parse\ share\ as(M, \sigma, cert)$

If the algorithm *Cheat* has generated the valid signature of the message *M* then the following events had to occur:

There was a query to the *H* oracle for the hash of the message *M* , but there was no query (if the algorithm has forged the signature) for the signature of *M* for any player from Γ . Since the oracle *H* transports the queries of the oracle *CHALL*, the above pair $(H(M),\sigma)$ breaks the chosen-target CDH problem. Concluding if the polynomial-time algorithm *Cheat* forges (with non-negligible probability) the signature of *M*, then the polynomial-time algorithm *Break* breaks the chosen-target CDH problem with the secret value κ (with non-negligible probability).

The considered case is pessimistic in the sense that if the adversary corrupts smaller number of players he has less information to be used in the attack on the system.

More precisely let us assume that the adversary corrupts *r−1* players and therefore computes the corresponding *r−1* partial signatures $(1 \le r-1 \le d)$.

To show that the problem of computation of the *r*-th partial signature is not easier than the already considered one, we simulate the system generating *d−(r−1)* partial signatures and transmitting them in an encrypted form to the adversary (so that he could not learn any information about their values). The adversary's algorithm breaking the corresponding *r*-th partial signature is then easily transformated to the algorithm breaking the *d+1*-th partial signature which we have proved to be a hard problem. To complete the argument it remains to observe that the probability that the *r*-th partial signature computed by the adversary is distinct from any of the encrypted partial signatures is non-negligible. This completes the proof of Lemma 1.

To prove Lemma 2 we need to observe that certificate is valid only if at least $\alpha_i|\Gamma_i|$ members of Γ_i have accepted it. If $\beta_i < \alpha_i$, then there is at least one honest player among them. Therefore 1) certificate is correct, 2) the corresponding share is correct, 3) there are no distinct certificates with the same value of *x* (since otherwise the suitable honest user wouldn't accept it). Hence the certificate is correct and therefore Lemma 2 is proved.

Now we are ready to prove the final result with the explicit security conditions.

Theorem 1. Assume that the triple (Γ, α, β) is as in the system description. If the following conditions:

1. $\forall_i \beta_i < min(\frac{1}{2,}\alpha_i)$
2. $\sum_i \beta_i < l - \frac{2}{K} - \theta \sum_i min(\frac{1}{2,}\alpha_i)$

are satisfied then the Dynamic Multi-Threshold Signature Scheme is secure in the random oracle model provided the chosen-target CDH assumption holds true and $d = \lceil K\theta \sum_i min(\frac{1}{2,}\alpha_i) \rceil + 1$.

Proof. By (1) it follows that $\beta_i |\Gamma_i| < \alpha_i |\Gamma_i|$, so there are not enough corrupted players to generate sufficiently many partial signatures, and by Lemma 1 they cannot forge them. Moreover the corrupted players are not able to stop the honest ones from generating of valid signatures since from (2) we have:

$K \sum_i \beta_i - Kl < -2 - \theta K \sum_i min(\frac{1}{2,}\alpha_i)$, hence

$$\sum_i (1-\beta_i)|\Gamma_i| \geq \sum_i (1-\beta_i) K > 2 + \theta K \sum_i min(\frac{1}{2,}\alpha_i) \geq 1 \lceil K\theta \sum_i min(\frac{1}{2,}\alpha_i) \rceil = d$$

which implies that the honest players are able to generate the required number *d+1* partial signatures. By Lemma 2 the corrupted players can not forge the certificates, so the published partial signatures are valid. This completes the proof of Theorem 1.

6. References

[1] Identity-based encryption from the Weil Pairing, D. Boneh, M. Franklin, SIAM J. of Computing, Vol. 32, No. 3, 2003

[2] Short signatures from the Weil pairing, D. Boneh, B. Lynn, H. Shacham, Advances in Cryptology - ASIACRYPT '01, LNCS Vol. 2248

[3] Efficient Threshold Signature, Multisignature and Blind Signature Schemes Based on the Gap-Diffie-Hellman-group signature scheme, A. Boldyreva, LNCS 2567, 2003

[4] Secure Distributed Key Generation for Discrete-Log Based Cryptosystems, R. Gennaro, S. Jarecki, H. Krawczyk, T. Rabin, Eurocrypt 1999

[5] Dynamic threshold cryptosystems: a new scheme in group oriented cryptography H. Ghodosi, J. Pieprzyk, R. Safavi Naini, Proceedings of PRAGOCRYPT 96

[6] A one-round protocol for tripartite Diffie-Hellman, A. Joux, ANTS-IV conference, LNCS 1838, 2000

[7] Reducing elliptic curves logarithms to logarithms in a finite field, A. Menezes, T. Okamoto, S. Vanstone, IEEE Transaction on Information Theory, Vol. 39, 1993

[8] Multi-threshold signature, B. Nakielski, J.A. Pomykała, J.M. Pomykała, Journal of Telecomunications and Information Technology 1 (2008), pp. 51-55

[9] Threshold signatures in dynamic groups, J. Pomykała, T. Warchoł, Proceedings of Future Generation Communication and Networking , December 6-8, 2007, Jeju-Island, IEEE Computer Society, pp. 32-37

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

[11] How to share a secret, A. Shamir, Communications of the ACM, 22:612-613, 1979

8. Information about authors

1. **dr. hab. Jacek Pomykała**, born in 1958. PHD thesis in 1986 in Number Theory at *Department of Mathematics, Informatics and Mechanics Warsaw University.* Habilitation in 1997 in Analytic Number Theory and Automorphic L-functions at Institute of Mathematics of Polish Academy of Sciences. Since 1998 interested in Computational Number Theory and Cryptology. The author of over 20 publications in international journals in number theory and cryptology. The director of the two grants of KBN in Arithmetics of Elliptic Curves and Distributions of zeros of automorphic L-functions respectively. Coordinator of the intrenational exchange

program in Number Theory between IMPAN (Poland) and CNRS (France) in 1996-98. Invited speaker in many international conferences and long term visitor in France (Paris) and Germany (Frankfurt am Main) . Currently coordinator of the cooperation program in Cryptology between *Department of Mathematics, Informatics and Mechanics of Warsaw University* and Warsaw Technical University.

2. **Tomasz Warchoł**, born in 1984. Student of the 5-th year graduate study in Mathematics and 3-rd year study in Computer Science at the *Department of Mathematics, Informatics and Mechanics of Warsaw University.* Interested subjects: Algebra, Number Theory, Cryptography and Artificial Intelligence. The master degree thesis is related to the application of the bilinear pairings in the threshold cryptography (supervisor: dr hab. Jacek Pomykała).