Key Agreement for Heterogeneous Mobile Ad-Hoc Groups*

Mark Manulis
Horst-Görtz Institute, Ruhr-University of Bochum
IC4/157, D-44801 Bochum, Germany
mark.manulis@rub.de

Abstract

In this paper we propose an efficient key agreement protocol suite for heterogeneous mobile ad-hoc groups, whose members use mobile devices with different performance limitations, e.g., laptops, PDAs, and mobile phones. Absence of a trusted central authority in ad-hoc groups requires contributory computation of the group key by interacting members. We introduce a performance ratio parameter to quantify the performance of a mobile device. Our protocols are based on elliptic curve cryptography (ECC) to achieve better computation efficiency and are proven secure.

1. Introduction

Consider a spontaneously built group of people who wish to establish secure mobile ad-hoc communication using their mobile devices, i.e., it should be guaranteed that only current group member should be able to obtain any secret information sent inside the group. It is desired to add new and delete current participants without security compromisation. Sets of participants and their devices are unpredictable, i.e., participants may be using laptops, PDAs and mobile phones. Examples for such *spontaneous dynamic heterogeneous groups* are workshop meetings at conferences, meetings for ad-hoc elections or auctions. The main goal is to allow each participant to take part in this secure ad-hoc group communication independent of the performance of its mobile device. The task of securing the communication reduces to the establishment of a shared secret key among all participants, and its update after dynamic group changes under consideration of different performance limitations of involved devices and of dynamic and fault-prone nature of ad-hoc communication.

* This is a full version of the paper appeared in 11th International Conference on Parallel and Distributed Systems - Workshops (ICPADS'05), pp. 290–294, © IEEE Computer Society 2005

1.1. Related Work

Several group key management protocols have been proposed for mobile ad-hoc group communication. In [1] Asokan *et. al.* propose a password authentication based key agreement protocol for small ad-hoc groups those members are on the same location (i.e., in one room). They assume that all members share a secret password. Obviously, this is not the case for spontaneously built ad-hoc groups considered here. Their protocol does not handle dynamic events and is less-efficient if the group size is not a power of two. The protocol of Besson *et. al.* [7] provides efficient mutual authentication and group key agreement for low-power mobile devices and supports dynamic changes, but requires a wireless infrastructure with some powerful trusted server (base station) that performes heavy computations. Such trusted authority is usually not available in described scenarios. There is a number of so-called *contributory group key agreement* (CGKA) protocols, like Burmester-Desmedt [3], CLIQUES [17], STR [10] and TGDH [11], that were originally proposed for local- or wide-area Networks. These protocols have similar trust relationship between communication participants as in ad-hoc groups (the group key is computed as a function of member's personal contributions). Spontaneity of group formation requires authentication over digital signatures with certificates issued by a publicly known certification authority (CA), that, however, is not actively involved in the computation of the group key. We assume that each mobile device obtains its certificate before it participates in the protocol. Original CGKA protocols have to be optimized for mobile ad-hoc networks, because they were originally designed for higher performance networks and devices. In this paper we optimize computation, communication and memory complexity of the most communication efficient STR protocol [10] with respect to the requirements of heterogeneous mobile ad-hoc group communication. Our general intention is to achieve that the more powerful a device is the higher computation, communication and memory costs it has to bear.

2. Heterogeneous Mobile Ad-Hoc Groups

2.1. Model

Mobile devices involved in heterogeneous mobile ad-hoc group communication have different performance capabilities. In order to distinguish them according to their performance we quantify the performance ratio of a mobile device using performance benchmarking. There is some ongoing work on the benchmarking of low-power devices, like [6]. We remark that it is possible to develope special benchmarks to measure performance of cryptographic protocols. For the remainder of this paper we assume that some generic *benchmarking function* f exists, which takes into account the hardware parameters of a mobile device, such as CPU clocks, memory capacity and battery power consumption and performs some network and cryptographic application specific operations to output a value $\mu \in \mathbb{R}$, called a *performance ratio* of a mobile device.

Definition 1 *Let $\mathcal{M}$ be a set of n mobile devices, f a generic performance benchmarking function, and μ_i a performance ratio value computed by f on device $M_i \in \mathcal{M}$. The permutation $P = (M_1, \ldots, M_n)$ is a performance ratio order of mobile devices if for any $M_i, M_{i+1} \in \mathcal{M}$ holds that $\mu_i \geq \mu_{i+1}$. Device M_i is called more powerful than M_j if $\mu_i > \mu_j$, less powerful if $\mu_i < \mu_j$, and equally powerful if $\mu_i = \mu_j$.*

2.2. Security Requirements

Key agreement protocols for heterogeneous ad-hoc groups should satisfy the following security requirements [11]: *computational group key secrecy* (for a passive adversary it must be computationally infeasible to discover any secret group key), *decisional group key secrecy* (for a passive adversary it must be computationally infeasible to distinguish any bits of the secret group key from random bits), *forward secrecy* (for any passive adversary being in possession of a subset of old group keys must not be able to discover any subsequent group key), *backward secrecy* (for any passive adversary being in possession of a subset of contiguous group keys must not be able to discover any preceding group key), and *key independence* (for any passive adversary being in possession of any subset of group keys must not be able to discover any other group key). Obviously, key independence is achieved whenether forward and backward secrecy are provided.

The protocols have also to take into account the common nature of ad-hoc communication: *absence of central authority* (the computation of the group key must be contributory, e.g., every participant should provide own contribution to the computation of the group key, such that these contributions can be verified by other participants), and *dynamics* (the group key management protocol must handle dynamic group changes, like joins, leave, merge and partition, without any risks for the group key secrecy).

Additionally, we specify two requirements for heterogeneous ad-hoc groups: *cost fairness* (computation, communication and memory costs of the key agreement protocol must be distributed between mobile devices non-uniformly, e.g., under consideration of their performance ratios), and *performance honesty* (no participant must be able to cheat on the performance ratio of its device, e.g., to pretend that it has a smaller performance ratio than it really does in order to save own costs during the protocol run).

Remark 1 *Performance honesty is a subcase of the problem of stimulating the cooperation between participants of ad-hoc communication in order to reduce their "*selfishness*" [4]. One approach solution is to use tamper resistant hardware components that cannot be modified by the user. These components must provide an authentic non-modifiable performance ratio of the device. Another solution is based on so-called "*incentive-based*" approaches that discourage selfish behaviour by making cooperation more attractive [8]. For cooperation between participants of an ad-hoc communication we refer to [5] and [14].* For the remainder of this paper we assume that every participant submits authentic performance ratio of its device.

3. μSTR-H protocol suite

In this section we describe a CGKA protocol suite for heterogeneous mobile ad-hoc groups, called μSTR-H that results from optimization of communication-efficient STR protocols [10].

3.1. Preliminaries

μSTR-H protocol suite consists of five protocols: setup, join, leave, merge and partition, and allows participants to agree on a secret group key and maintain it upon dynamic group changes. Consider a group of n members, denoted $M_1, \ldots M_n$ wishing to agree on a secret group key. We assume having public and reliable broadcast communication channel shared by all participants. Every member has its own private/public key pair $(skey_i, pkey_i)$ and certificate:

$$Cert_i = (ID_i, pkey_i, Sig_{\mathrm{CA}}(ID_i, pkey_i)),$$

where ID_i uniquely identifies M_i, and $Sig_{CA}(ID_i, pkey_i)$ is the signature of CA that binds member's identity to its public key. Before a member sends a message to the group it signs it using $skey_i$, such that every receiver is able to verify the signature using $pkey_i$.

Let E be an elliptic curve over a finite field $\mathbb{F}_q$, such that $\mathbb{F}_q$ is either prime (q is a prime) or binary ($q = 2^m$ and $m \in \mathbb{N}$) field. $E(\mathbb{F}_q)$ denotes a commutative group of points in E. Let $G \in E(\mathbb{F}_q)$ be a point with high prime order t that devides $q-1$. G generates a multiplicative (cyclic) subgroup

of $E(\mathbb{F}_q)$ denoted $<G>=\{O, G, 2G, \ldots, (t-1)G\}$, where O is the point of infinity. We remark that all computations in our protocols are done in $<G>$. Some protocols require to map a point in E to an integer in the range $[1, \ldots, q-1]$. In order to map a point P to an integer it is sufficient to map its x-coordinate (denoted $(P)_x$), since y-coordinate can be easily computed using the equation of E. We suggest to use function $map : E(\mathbb{F}_q) \rightarrow \mathbb{N}$ from [16, Sec. 2.3.9].

Definition 2 *Let $E(\mathbb{F}_q)$ be a group of points in an elliptic curve E over a finite field $\mathbb{F}_q$, and point $P \in E(\mathbb{F}_q)$. The point-to-integer mapping function $map : E(\mathbb{F}_q) \rightarrow \mathbb{N}$ is defined as*

$$map(P) = \begin{cases} (P)_x, & \text{for } q = p \text{ and prime } p \\ \sum_{i=0}^{m-1} 2^i a_i, & \text{for } q = 2^m, m \in \mathbb{N} \\ & \text{where } (P)_x = (a_{m-1} \ldots a_1 a_0) \end{cases}$$

For μSTR-H protocols every member M_i selects session random $r_i \in_{\mathcal{R}} \{1, \ldots, t-1\}$, and computes its blinded version $R_i = r_i G$. For each secret key k_i there exists a corresponding public key $K_i = k_i G$. Public values R_i and K_i computed as scalar-point multiplications are points in E. Every $k_i = r_i k_{i-1} G$, $i > 1$ is computed using tree-based Diffie-Hellman key exchange method [11] in two different ways:

$$k_i = map(r_i K_{i-1}) \qquad \text{or} \qquad k_i = map(k_{i-1} R_i)$$

Since k_i has to be an integer in order to compute k_{i+1}, and values $(r_i K_{i-1})$ and $(k_{i-1} R_i)$ are points in E, the point-to-integer mapping function map is used.

3.2. Protocols

In all protocols of this section authentication is done over digital signatures using members' certificates. We suggest to use ECDSA ([15], [16]) since its signature size is much more smaller than that of DSA or RSA without any loss of security. We stress that every message must be signed by the sender and verified by the receiver, and omit the indication of the signing and verifying processes in the following description. Members of the group are indexed according to the performance ratio order $P = (M_1, \ldots, M_n)$ that they update in every μSTR-H protocol. It must be possible to find the position i and the performance ratio μ_i of any member M_i from P. Every member M_i saves two lists: $\mathbf{R}_i$ for blinded session randoms, and $\mathbf{k}_i$ for secret keys. $\mathbf{R}_i$ consists of $(R_{i+1}, \ldots, R_n)$, and $\mathbf{k}_i$ consists of $(k_i, \ldots, k_n)$. Every M_i saves also own r_i and R_i, and if $i > 1$ saves K_{i-1} (note $k_1 = r_1$ and $K_1 = R_1$). μSTR-H protocols define the role of a sponsor M_s to handle dynamic events. The role is temporary and can be assigned to different members depending on the event and current P. M_s reduces communication overhead by performing some operations on behalf of the group. We stress that M_s is not a trusted central authority, because its messages can be verified by other members.

Protocol *Setup***:**

- M_i selects r_i, computes R_i, and broadcasts $(R_i, \mu_i, Cert_i)$.
- M_i computes performance ratio order $P = (M_1, \ldots, M_n)$, finds own index i and saves $\mathbf{R}_i = (R_{i+1}, \ldots, R_n)$. Additionally, M_1 computes $\mathbf{k}_1 = (k_2, \ldots, k_n)$, and broadcasts $(K_2, \ldots, K_{n-1})$.
- M_i computes $\mathbf{k}_i = (k_i, \ldots, k_n)$, and saves K_{i-1}.

Protocol *Join***:** In order to fulfill cost fairness requirement new member M_j is inserted in P according to its μ_j.

- New member M_j selects r_j, computes R_j, and broadcasts $(R_j, \mu_j, Cert_j)$.
- M_i updates P with M_j, finds index j of the new member, renumbers all members M_i $(i > j)$ to M_{i+1}, and if $i < j$ adds R_j to $\mathbf{R}_i$. Additionally, the sponsor M_s selects new r_s, computes R_s, recomputes $\mathbf{k}_s = (k_s, \ldots, k_{n+1})$, and broadcasts $(P, R_s, (R_{j+1}, \ldots, R_{n+1}), (K_s, \ldots, K_n))$.
- New member M_j saves P, K_s, $\mathbf{R}_j = (R_{j+1}, \ldots, R_{n+1})$, finds own index j, and computes $\mathbf{k}_j = (k_j, \ldots, k_{n+1})$. M_i $(i < s)$ updates R_s in $\mathbf{R}_i$, and recomputes $(k_s, \ldots, k_{n+1})$ in $\mathbf{k}_i$. M_i $(i > j)$ updates K_{i-1}, and recomputes $\mathbf{k}_i$.

M_s is the highest-numbered member below the position j of the new member. If $j = 1$ then the sponsor is M_2.

Protocol *Leave***:** Assume, member M_d leaves the group.

- M_i deletes M_d from P, if $i < d$ also R_d from $\mathbf{R}_i$, and k_d from $\mathbf{k}_i$, and renumbers all members M_i $(i > d)$ to M_{i-1}. Additionally, the sponsor M_s selects new r_s, computes R_s, recomputes $\mathbf{k}_s = (k_s, \ldots, k_{n-1})$, and broadcasts $(P, R_s, (K_s, \ldots, K_{n-2}))$.
- M_i $(i < s)$ updates R_s and recomputes $(k_s, \ldots, k_{n-1})$ in $\mathbf{k}_i$. M_i $(i > s)$ updates K_{i-1} and recomputes $\mathbf{k}_i$.

M_s is the highest-numbered member below the position d of the leaving member. If $d = 1$ then the sponsor is M_2.

Protocol *Merge***:** Two groups, G′ of size n' and G″ of size n'', are merging to a common group G. Resulting performance ratio order P is computed by merging of P' and P''. (Affiliation to G′ (G″) is denoted by ′ (″) in the superscript.)

- M'_1 and M''_1 broadcast $(P', (R'_1, \ldots, R'_{n'}), (Cert'_1, \ldots, Cert'_{n'}))$ and $(P'', (R''_1, \ldots, R''_{n''}), (Cert''_1, \ldots, Cert''_{n''}))$, respectively.
- Every member M_i merges P' and P'' to P, renumbers all members according to P, finds own position i, and updates $\mathbf{R}_i = (R_{i+1}, \ldots, R_{n'+n''})$. Additionally, the sponsor M_s selects new r_s, computes R_s, recomputes $\mathbf{k}_s$, and broadcasts $(R_s, (K_s, \ldots, K_{n'+n''-1}))$.
- M_i $(i < s)$ updates R_s in $\mathbf{R}_i$, and recomputes $(k_s, \ldots, k_{n'+n''})$ in $\mathbf{k}_i$. M_i $(i > s)$ updates K_{i-1} and recomputes $\mathbf{k}_i = (k_i, \ldots, k_{n'+n''})$.

M_s is the highest-indexed member in P below the least-indexed member M_j whose position j changed after P' and P'' had been merged. $j > 1$ holds always, because either M'_1 or M''_1 becomes M_1 in G.

Protocol *Partition*: Assume, a subgroup G′ leaves group G of size n. The number of survived members in G is v. (Affiliation to G′ is denoted by ′ in the superscript.)

- M_i deletes all M'_j from P, if $i < j$ also R'_j from $\mathbf{R}_i$ and k'_j from $\mathbf{k}_i$, and renumbers all survived members M_i accordingly. Additionally, the sponsor M_s selects new r_s, computes R_s, recomputes $\mathbf{k}_s = (k_s, \ldots, k_{n-v})$, and broadcasts $(P, R_s, (K_s, \ldots, K_{n-v-1}))$.
- M_i $(i < s)$ updates R_s and recomputes $(k_s, \ldots, k_{n-v})$ in $\mathbf{k}_i$. M_i $(i > s)$ updates K_{i-1} and recomputes $\mathbf{k}_i$.

M_s is the highest-indexed member below position j of the least-indexed leaving member. If $j = 1$ then the sponsor is the least-indexed survived member.

3.3. Complexity

The communication, computation and memory complexities of μSTR-H and STR protocols are given in Table 1. Costs of STR protocols are only given if they differ from those of μSTR-H. Size of P is negligible compared to lists of secret keys ($\mathbf{k}_i$) and blinded session randoms ($\mathbf{R}_i$), and is therefore omitted in the analysis. The total size of sent messages for the handling of the dynamic changes in μSTR-H could be decreased by a factor 2 on the average and depends now on the sponsor's position s that ranges between 1 and n. Similarly, the total size of saved data per member's device has been reduced and is in the range between 4 and $2n$ depending on member's position in P. In original STR join and merge new members get next possible highest indices, e.g., if a new member joins to a group of n members $M_1, \ldots, M_n$, then its becomes M_{n+1}. This allows to keep computation costs of STR join constant. In μSTR-H members have to be added in P preserving the order of their performance ratios, thus computation costs may vary. Similarly in case of μSTR-H merge. Computation costs of original STR protocols are given in modular exponentiations in a cyclic group $\mathbb{Z}_p^*$, whereas the costs of μSTR-H protocols in scalar-point multiplications. Switching to ECC brings additional computation efficiency and memory size reduction in practice ($|q| = 157$ bits in contrast to $|p| = 1024$ bits if $\mathbb{Z}_p^*$ is used [12]). Obviously, μSTR-H protocols fulfil cost fairness requirement for heterogeneous groups, because costs of a member depend on its position in P. Thus, μSTR-H protocol suite distributes costs non-uniformly.

3.3.1. Further Optimizations

- Reduced computation costs by precomputing the (r, R)-pairs together with corresponding digital signatures. Whenever a device has to change its pair it makes a random selection from the precomputed set. Precomputing saves one multiplication in each protocol, but is a trade-off between computation and memory costs.
- If a mobile device performs ECC operations in hardware then we suggest to use binary finite fields $\mathbb{F}_{2^m}$ since operations in these fields are performed efficienter in hardware than in prime fields $\mathbb{F}_p$ [18].

3.4. Security

In this section we discuss the security of μSTR-H protocols with respect to the requirements of Section 2.2. μSTR-H has reduced computation, communication and memory costs compared to STR and is suitable for heterogeneous mobile ad-hoc groups. The computation process of the group key still relies on the tree-based Diffie-Hellman key exchange method as in STR, except for the difference that mathematical operations are peformed in a subgroup of points $<G>$ of an elliptic curve E over a finite field $\mathbb{F}_q$ as described in Section 3.1, and not in a cyclic group $\mathbb{Z}_p^*$. We show that security of μSTR-H protocols benefits from the security of STR protocols as proven in [10] and [11]. The *computational group key secrecy* of STR protocols relies on the hardness of Computational Diffie-Hellman (CDH) problem, that has also been proven hard in $< G >$ [13]. The *decisional group key secrecy* of STR protocols relies on the hardness of Decisional Diffie-Hellman (DDH) problem [2], that has been proven hard in $< G >$ for certain kinds of elliptic curves (non-supersingular and non-trace-2 elliptic curves [9]). Thus, adversary A can neither compute nor distinguish the group key knowing only the public keys and blinded session randoms (note communication broadcast channel is public). Therefore, a group key can only be discovered if at least one secret value, either any r_i or k_i is known to A. Due to the hardness of Discrete Logarithm (DL) problem (its ECC counterpart is believed to be even more difficult to solve [13]) adversary is not able to reveal these values from their public values R_i and K_i. In case of *backward secrecy* we show that any A being a joining member is not able to obtain any of the previous used group keys. Assume, A becomes a new member of the group at position a in P. As a new member A is able to compute all secret keys k_i $(a \leq i \leq n)$. The sponsor of the additive event changes own r_s and causes the change of all k_i, $s \leq i \leq n$. Since $s < a$ A can only compute changed secret keys, and is therefore not able to compute the previously used group key. Thus, backward secrecy is provided. Analogously, for *forward secrecy* we have to show that any A being a leaving member at position a in P is not able to obtain any subsequently used group key. A knows all secret keys k_i $(a \leq i \leq n)$ that are valid during its group membership. However, the sponsor of the subtractive event changes own r_s and causes the change of all k_i, $s \leq i \leq n$. Since $s < a$ all secret keys that A knows are changed, and therefore it is not able to compute the subsequent group key. Thus, forward secrecy is provided. As combination of backward and forward secrecy we follow that μSTR-H protocols provide *key independence*. Updated group keys are independent due to a random change of sponsor's contribution.

Table 1: Computation, Communication and Memory Costs of μSTR-H and STR Protocols

	Communication			Computation	Memory
	Rounds	Messages	Message size in $\|q\|$ ($\|p\|$) bits	SP-Multiplications (Mod-Exponentiations)	Saved data size in $\|q\|$ ($\|p\|$) bits
S	2	$n+1$	$2n-2$ $(2n-1)$	$i=1$: $2n-1$ $i>1$: $n-i+2$	$i=1$: $2n$ $(3n-2)$ $i>1$: $2n-2i+4$ $(3n-i)$
J	1	2	$2n-2s+3$ $(2n)$	$i<s$: $n-s+2$ (2) $i=s$: $2n-2s+4$ (4) $i>s$: $n-i+2$ (1)	
L	1	1	$n-s$ $(2n-4)$	$i<s$: $n-s$ $i=s$: $2n-2s$ $i>s$: $n-i$	
M	2	3	$2n'+2n''-s+1$ $(4n'+4n''-6)$	$i<s$: $n'+n''-s+1$ $(n''+1)$ $i=s$: $2n'+2n''-2s+2$ $i>s$: $n'+n''-i+1$	
P	1	1	$n-v-s+1$ $(2n-2v'-2)$	$i<s$: $n-v-s+1$ $i=s$: $2n-2v-2s+2$ $i>s$: $n-v-i+1$	

Remarks: S- setup, J - join, L - leave, M - merge, P - partition, $|q|$ - length of q in F_q, $|p|$ - length of p in $\mathbb{Z}_p^*$, i (s) - member's (sponsor's) position in updated P, n - initial group size, n' (n'') - size of larger (smaller) merging group, v - number of leaving members, () - costs of original STR protocol if they differ from μSTR-H

4. Conclusion

In this paper we have proposed a new group key agreement for dynamic heterogeneous mobile ad-hoc groups on the basis of communication efficient CGKA protocol STR [10]. We have introduced *performance ratio* parameter that allows to distinguish between performances of mobile devices and specified additional requirements, such as *cost fairness* and *performance honesty*.

Acknowledgements

The author is thankful to Ahmad-Reza Sadeghi, Andre Adelsbach and Jörg Schwenk for their comments on this topic. This work is supported by the European Commission through IST-2002-507932 ECRYPT.

References

[1] N. Asokan and P. Ginzboorg. Key-agreement in ad-hoc networks. *Computer Communications*, 23(17):1627–1637, 2000.

[2] D. Boneh. The decision diffie-hellman problem. In *ANTS-III: Proceedings of the Third International Symposium on Algorithmic Number Theory*, pages 48–63. Springer-Verlag, 1998.

[3] M. Burmester and Y. Desmedt. A secure and efficient conference key distribution system. In *Advances in Cryptology (EUROCRYPT '94),Lecture Notes in Computer Science*, volume 950, pages 275–286. Springer-Verlag Berlin, May 1994.

[4] L. Buttyan and J.-P. Hubaux. Stimulating cooperation in self-organizing mobile ad hoc networks. *Mobile Networks and Applications*, 8(5):579–592, 2003.

[5] J. Crowcroft, R. Gibbens, F. Kelly, and S. Ostring. Modeling cooperation in mobile ad hoc networks: A formal description of selfishness. In *Proceedings of Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'03)*, 2003.

[6] Embedded Microprocessor Benchmark Consortium (EEMBC). `http://www.eembc.org`.

[7] A. E. Emmanuel Bresson, Olivier Chevassut and D. Pointcheval. Mutual authentication and group key agreement for low-power mobile devices. In *Proceedings of the 5th IFIP-TC6 International Conference on Mobile and Wireless Communications Networks (october 27 - 29, 2003, Singapore)*, pages 59–62. World Scientific Publishing, 2003.

[8] E. Huang, J. Crowcroft, and I. Wassell. Rethinking incentives for mobile ad hoc networks. In *PINS '04: Proceedings of the ACM SIGCOMM workshop on Practice and theory of incentives in networked systems*, pages 191–196. ACM Press, 2004.

[9] A. Joux and K. Nguyen. Separating decision diffie-hellman from diffie-hellman in cryptographic groups. Cryptology ePrint Archive, Report 2001/003, 2001. `http://eprint.iacr.org/`.

[10] Y. Kim, A. Perrig, and G. Tsudik. Communication-efficient group key agreement. In *Information Systems Security, Proc. of the 17th International Information Security Conference, IFIP SEC'01*, 2001.

[11] Y. Kim, A. Perrig, and G. Tsudik. Tree-based group key agreement. *ACM Transactions on Information and System Security*, 7(1):60–96, 2004.

[12] A. K. Lenstra and E. R. Verheul. Selecting cryptographic key sizes. *Journal of Cryptology: the journal of the International Association for Cryptologic Research*, 14(4):255–293, 2001.

[13] U. M. Maurer and S. Wolf. The Diffie-Hellman protocol. *Designs, Codes and Cryptography*, 19:147–171, 2000.

[14] P. Michiardi and R. Molva. A game theoretical approach to evaluate cooperation enforcement mechanisms in mobile ad hoc networks. In *Proceedings of Modeling and Optimization in Mobile, Ad Hoc, and Wireless Networks (WiOpt'03)*, 2003.

[15] National Institute Standards and Technology (NIST). Federal information processing standard 186-2: Digital signature standard, `http://www.nist.gov`, October 2001.

[16] Standards for Efficient Cryptography Group (SEC). Sec 1: Elliptic curve cryptography, `http://www.sec.org`, September 2000.

[17] M. Steiner, G. Tsudik, and M. Waidner. Key agreement in dynamic peer groups. *IEEE Transactions on Parallel and Distributed Systems*, 11(8), 2000.

[18] T. Wollinger, J. Pelzl, V. Wittelsberger, C. Paar, G. Saldamli, and C. K. Koc. Elliptic and hyperelliptic curves on embedded μp. *Trans. on Embedded Computing Sys.*, 3(3):509–533, 2004.