Simple and Efficient Threshold Cryptosystem from the Gap Diffie-Hellman Group

Joonsang Baek
Monash University
Frankston, VIC 3199, Australia
Email: joonsang.baek@infotech.monash.edu.au

Yuliang Zheng
UNC Charlotte
Charlotte, NC 28223, USA
Email: yzheng@uncc.edu

***Abstract*— In this paper, we construct a new threshold cryptosystem from the Gap Diffie-Hellman (GDH) group. The proposed scheme enjoys all the most important properties that a *robust* and *practical* threshold cryptosystem should possess, that is, it is non-interactive, computationally efficient and provably secure against adaptive chosen ciphertext attacks. In addition, thanks to the elegant structure of the GDH group, the proposed threshold cryptosystem has shorter decryption shares as well as ciphertexts when compared with other schemes proposed in the literature.**

I. Introduction

A. Threshold Cryptosystems

Although there have been several proposals on threshold cryptosystems[1], most of these proposed solutions are complex and inefficient in terms of computation and communication overheads. As a result, it remains a challenge to design a simple yet efficient threshold cryptosystem that is provably secure against adaptive chosen ciphertext attacks (CCA).

In this paper, we take up the challenge with the aim of further advancing this line of research. We achieve our goal, namely to design a simple, secure and efficient threshold cryptosystem, by taking advantage of an emerging technique for using a group of points on certain elliptic curves which have a special property.

B. Related work

In the research into secure threshold cryptosystem, perhaps the most significant progress was made by Shoup and Gennaro [14] who came up with a formal security notion for threshold cryptosystems, proposed two practical schemes and proved the security of their schemes against CCA in the random oracle model [4]. Although the schemes by Shoup and Gennaro were very efficient when compared to those proposed so far, the use of non-interactive zero-knowledge (NIZK) proofs of membership in their schemes contributed in a negative way to the computational costs of encryption as well as the expansion of ciphertexts. While our present work can be viewed as a novel application of Shoup and Gennaro's techniques, we set apart from Shoup and Gennaro's work by constructing an efficient and provably secure threshold scheme without invoking NIZK proofs of membership.

Recently, in [6], Boneh and Franklin mentioned that the Private Key Generator (PKG) in their identity-based encryption scheme can be distributed using the techniques of threshold cryptography, which only holds in the identity-based setting and hence is different from our scheme which is for the normal public key setting. Another difference is that our scheme is publicly checkable whereas the schemes in [6] are not. This will be discussed more in the next section. Nevertheless, we remark that they did mention about exploiting the easiness of solving the DDH problem in the group on which their identity-based encryption scheme is based, which plays a central role in our threshold cryptosystem.

Other related works include Boneh et al.'s short signature scheme [7], Lysyanskaya's unique signature scheme [11] and more recently, Bolyreva's construction of various signature schemes based on the Boneh et al.'s short signature scheme, all of which are based on the property of the GDH group. However, until the present work, a threshold cryptosystem on the GDH group has not yet emerged.

II. Publicly Checkable Cryptosystem from the GDH Group

A. The GDH Group

The Decisional Diffie-Hellman (DDH) problem is defined as a problem where, given $(P, aP, bP, cP) \in \mathcal{G}$ where $\mathcal{G} = \langle P \rangle$ is a group of prime order q and a, b, c are uniformly chosen at random from $\mathbb{Z}_q^*$,

[1]One might use the term "threshold decryption", instead.

one is asked to decide whether $c = ab$. Note that throughout this paper, an operation in $\mathcal{G}$ is denoted by "addition (+)". A closely related problem is the Computational Diffie-Hellman (CDH) problem where one is asked to compute abP, given (P, aP, bP).

The group $\mathcal{G}$ is called a Gap Diffie-Hellman group if there exists an efficient (polynomial-time in $|q|$) algorithm for solving the DDH problem, but not for solving the CDH problem. The GDH group can be constructed using a bilinear map on supersingular elliptic curves as follows [9].

Let p be a prime. Let E be an elliptic curve over $\mathbb{F}_{p^n}$ for some positive integer n. Let $E[q]$ be the q-torsion subgroup of E, that is, $E[q] = \{P \in E | qP = \mathcal{O}\}$ where q is a prime. The Weil (or Tate) paring is a map $e : E[q] \times E[q] \to \mathbb{F}^*_{p^{n\alpha}}$ for the least positive integer α called a "multiplier" such that $q | p^{n\alpha} - 1$ with the following properties.

- Identity: For all $R \in E[q]$, $e(R, R) = 1$.
- Bilinear: For all $R_1, R_2 \in E[q]$ and $a, b \in \mathbb{Z}$, $e(aR_1, bR_2) = e(R_1, R_2)^{ab}$.
- Non-degenerate: If for $R \in E[q]$, $e(R, R') = 1$ for $R' \in E[q]$ then $R = \mathcal{O}$.
- Computable: For all $R_1, R_2 \in E[q]$, the pairing $e(R_1, R_2)$ is efficiently computable.

Suppose that q divides $E(\mathbb{F}_{p^n})$ with a small cofactor. Suppose also that we have a non-$\mathbb{F}_{p^n}$-rational map $\phi : E \to E$. Then $\mathcal{G} \stackrel{\text{def}}{=} E(\mathbb{F}_{p^n})[q]$ is a group where a non-degenerate and efficiently computable bilinear map $\hat{e} : \mathcal{G} \times \mathcal{G} \to \mathbb{F}^*_{p^{n\alpha}}$ exists, which is called the "admissible" bilinear map [6]. Note that the bilinear map $\hat{e}$ is defined by $\hat{e}(P, Q) = e(P, \phi(Q))$.

The group $\mathcal{G}$ constructed above satisfies the property of a GDH group since we can determine whether a given tuple (P, aP, bP, cP) is a Diffie-Hellman one by checking whether $\hat{e}(P, cP) = \hat{e}(aP, bP)$, whereby solving the DDH problem.

B. Discussions on Publicly Checkable Cryptosystem

Validity check of ciphertexts is necessary in designing public key crytosystems to ensure chosen ciphertext security. However, in many public key cryptosystems, such check can be done only if a verifier (or a receiver) knows a private key. There are only a few schemes that are known to be publicly checkable, e.g., [1], [12], [14]. As observed by Lim and Lee [10], publicly checkable cryptosystems are particularly useful in designing threshold cryptosystems. The main reason is that in a threshold cryptosystem an attacker has decryption shares as additional information, as well as decryption of chosen plaintexts[2].

In the schemes in [1], [12], [14], the "public checkability" is achieved by using NIZK proofs (of language membership [14] or of knowledge [1], [12]) on ciphertexts. A downside of using ZKIP is that such proofs make the schemes more complex, computationally less efficient and more importantly, longer ciphertexts.

In what follows we present a publicly checkable cryptosystem. This efficient cryptosystem will be a building block for our threshold cryptosystem that does not rely on a NIZK proof, largely thanks to the special property of the GDH group on which the scheme is based.

C. Description of the Scheme $\mathcal{PCCG}$

Let $\mathcal{G}$be a GDH group with order q and a generator P. q and P are shared between a sender and a receiver. Assume that the receiver has published a public key $Y = xP$ where x is a private key chosen uniformly at random from $\mathbb{Z}^*_q$.

The sender encrypts a message $m \in \{0,1\}^l$ by creating a ciphertext

$$C = (U, V, W) = (rP, \mathsf{G}(rY) \oplus m, r\mathsf{H}(U, V))$$

where $\mathsf{G} : \mathcal{G} \to \{0,1\}^l$ and $\mathsf{H} : \mathcal{G} \times \{0,1\}^l \to \mathcal{G}$ are two one-way hash functions modelled as random oracles. Upon receiving the ciphertext C, the receiver decrypts C by computing $m = \mathsf{G}(xU) \oplus V$ *after* checking if $\hat{e}(P, W) = \hat{e}(U, H)$. If C does not pass this test, the receiver simply rejects C.

The scheme presented above is very simple – it is a simple ElGamal encryption scheme followed by a tag produced by signing on the ElGamal ciphertext using Boneh et al. [7]'s short signature scheme. Note also that the above scheme is publicly checkable *without* using a NIZK proof.

It can be shown that the scheme $\mathcal{PCCG}$ is secure against CCA in the random oracle model, relative to the CDH problem on the group $\mathcal{G}$. We skip the proof due to the limit in space.

III. Threshold Cryptosystem From the GDH Group

We start with a review of the basic (but informal) definition of a (t, n)-threshold cryptosystem. It follows largely Reference [14].

In a threshold cryptosystem, we assume the existence of a trusted dealer who runs a key generation algorithm denoted by K to output a public key, a

[2]Readers are referred to [10] and [14] for more detailed explanations on this.

verification key and distributes a share of private keys to each decryption server.

Given the public key, a sender encrypts a plaintext by running an encryption algorithm E.

Given a ciphertext, a receiver requests the decryption servers to generate each decryption share using algorithm denoted by D. The receiver can check the validity of the shares by running a decryption share verification algorithm denoted by V. When the receiver collects valid decryption shares from at least t servers, the ciphertext can be decrypted by running a share combining algorithm SC.

A. Description of the Proposed Scheme $\mathcal{TCG}$

Now, we describe our threshold cryptosystem from the GDH group, called $\mathcal{TCG} = (\mathsf{K}, \mathsf{E}, \mathsf{D}, \mathsf{V}, \mathsf{SC})$. Note that in the key generation described below, the Shamir's secrete sharing technique [13] is used.

Assume that the GDH group $\mathcal{G}$ with order q and its generator P is shared among all the parties. We need two hash functions $\mathsf{G} : \mathcal{G} \to \{0,1\}^l$ and $\mathsf{H} : \mathcal{G} \times \{0,1\}^l \to \mathcal{G}$ which are modelled as random oracles.

- $\mathsf{K}(k, n, t)$: given a security parameter k, the number n of decryption servers and a threshold parameter t, this algorithm picks $a_0, a_1, \ldots, a_{t-1}$ uniformly at random from $\mathbb{Z}_q^*$ and defines a polynomial $Poly(X) = \sum_{j=0}^{t-1} a_j X^j$. Then, for $0 \le i \le n$, it sets $x_i = Poly(i) \in \mathbb{Z}_q^*$ and computes $Y_i = x_i P$. Without loss of generality, it defines $x \stackrel{\text{def}}{=} a_0 = Poly(0)$ and $Y \stackrel{\text{def}}{=} Y_0 = xP$. Finally, it outputs a public key $pk = Y$, verification key $vk = (pk, Y_1, Y_2, \ldots, Y_n)$ and a private key $\mathbf{sk} = \{sk_i\} = \{(pk, i, x_i)\}$ for $1 \le i \le n$.
- $\mathsf{E}_{pk}(m)$: given a plaintext message $m \in \{0,1\}^n$, and a random value uniformly chosen from $\mathbb{Z}_q^*$, this algorithm computes $U = rP$, $V = \mathsf{G}(rY) \oplus m$ and $W = r\mathsf{H}(U, V)$ and outputs a ciphertext $C = (U, V, W)$.
- $\mathsf{D}_{sk_i}(C)$: given a ciphertext C, this algorithm first computes $H = \mathsf{H}(U, V)$ and checks if $\hat{e}(P, W) = \hat{e}(U, H)$. If this test holds, it computes $U_i = x_i U$ and outputs $D_i = (i, U_i)$. Otherwise, it returns $(i, \text{“?”})$.
- $\mathsf{V}_{vk}(C, D_i)$: this algorithm first computes $H = \mathsf{H}(U, V)$ and then checks if $\hat{e}(P, W) = \hat{e}(U, H)$. If this tests holds then it does the following:
  - If D_i is of the form $(i, \text{“?”})$ output “*invalid*”.
  - Else do the following:
    - Parse D_i as (i, U_i).
    - Check if $\hat{e}(P, U_i) = \hat{e}(U, Y_i)$.
    - If the test above holds, output “*valid*”, else output “*invalid*”.

  Otherwise, do the following:
  - If D_i is of the form $(i, \text{“?”})$, output “*valid*”, else output “*invalid*”.
- $\mathsf{SC}_{vk}(C, \{D_i\}_{i \in \Phi})$ where Φ has cardinality t: this algorithm computes $H = \mathsf{H}(U, V)$. If $\hat{e}(P, W) = \hat{e}(U, H)$ then it computes $m = \mathsf{G}(\sum_{i \in \Phi} \lambda_{0i}^{\Phi} U_i) \oplus V$ and outputs m. Otherwise, it outputs “?”[3]. Here, λ_{0i}^{Φ} denotes the Lagrange coefficient.

B. Security Analysis of the Scheme $\mathcal{TCG}$

First we formally define a security notion for threshold cryptosystem against CCA. We are interested the so-called semantic security notion against CCA for threshold cryptosystem, which we call “THD-IND-CCA”, given in [14].

Definition 1 (THD-IND-CCA): Consider an attacker $\mathsf{A}^{\mathsf{CCA}}$ in the following experiment which consists of several stages.

Corrupt: $\mathsf{A}^{\mathsf{CCA}}$ corrupts a fixed subset of $t-1$ servers.
Setup: The key generation algorithm on input a security parameter k is run. The private keys of the corrupted servers, the public key and the verification key (all of which are output by the key generation algorithm) are given to $\mathsf{A}^{\mathsf{CCA}}$. However, the private keys of uncorrupted servers are kept secret from $\mathsf{A}^{\mathsf{CCA}}$.
Phase 1: $\mathsf{A}^{\mathsf{CCA}}$ adaptively interacts with the uncorrupted decryption servers, submitting ciphertexts and obtaining decryption shares.
Challenge: $\mathsf{A}^{\mathsf{CCA}}$ chooses two equal length plaintexts (m_0, m_1). If these are given to the encryption algorithm then it chooses $b \in \{0,1\}$ at random and returns a target ciphertext $C^* = \mathsf{E}_{pk}(m_b)$ to $\mathsf{A}^{\mathsf{CCA}}$.
Phase 2: $\mathsf{A}^{\mathsf{CCA}}$ adaptively interacts with the uncorrupted decryption servers, submitting ciphertexts and obtaining decryption shares. However, the target ciphertext C^* is not allowed to query to the decryption servers.
Guess: $\mathsf{A}^{\mathsf{CCA}}$ outputs a guess $b' \in \{0,1\}$.

We define the attacker $\mathsf{A}^{\mathsf{CCA}}$'s success probability by $\mathbf{Succ}^{\mathrm{THD-IND-CCA}}(\mathsf{A}^{\mathsf{CCA}}) = 2\Pr[b' = b] - 1$. The probability is over the random bits used by the experiment and the attacker. We denote by $\mathbf{Succ}^{\mathrm{THD-IND-CCA}}(t_{\mathrm{CCA}}, q_D)$ the maximal success probability $\mathbf{Succ}^{\mathrm{THD-IND-CCA}}(\mathsf{A}^{\mathsf{CCA}})$ over all attackers whose running time and number of queries to the decryption share generation oracles are bounded by t_{CCA} and q_D, respectively.

[3]In this case, all the decryption shares are of the form $(i, \text{“?”})$ due to the validity test of C in the decryption share generation/verification algorithms run before.

Next we define formally the CDH problem on the GDH group $\mathcal{G}$.

Definition 2 (CDH): Let $\mathcal{G}$ denote the GDH group of prime order q as defined above. Let P be a generator of $\mathcal{G}$. Consider a probabilistic polynomial time attacker $\mathsf{A}^{\mathsf{CDH}}$ that tries to compute abP, given $(P, aP, bP) \in \mathcal{G}$ for $a, b, c \in \mathbb{Z}_q^*$. We define the attacker $\mathsf{A}^{\mathsf{CDH}}$'s success by the probability $\mathbf{Succ}^{\mathrm{CDH}}(\mathsf{A}^{\mathsf{CDH}}) = \Pr[\mathsf{A}^{\mathsf{CDH}} \text{ outputs } abP]$. We denote by $\mathbf{Succ}^{\mathrm{CDH}}(t_{\mathrm{CDH}})$ the maximal success probability $\mathbf{Succ}^{\mathrm{CDH}}(\mathsf{A}^{\mathsf{CDH}})$ over all attackers whose running time is bounded by t_{CDH}.

The following theorem is central to this paper, which basically says that our threshold cryptosystem $\mathcal{TCG}$ is THD-IND-CCA secure. Due to lack of space, we only provide a sketch of the proof here. More detailed analysis can be found in the full version [2].

Theorem 1: The (t, n)-threshold cryptosystem from the GDH group $\mathcal{TCG}$ is secure against CCA in the random oracle model, relative to the CDH problem. More precisely,

$$\frac{1}{2}\mathbf{Succ}^{\mathrm{THD-IND-CCA}}_{\mathcal{TCG}}(t_{\mathrm{CCA}}, q_G, q_H, q_D) \leq \mathbf{Succ}^{\mathrm{CDH}}(t_{\mathrm{CDH}}) + \frac{q_D + q_H q_D}{2^k},$$

where t_{CDH}=$t_{\mathrm{CCA}} + nO(k^3)q_G + q_H O(k^3) + q_G q_H q_D O(k^3)$. In addition, q_G, q_H and q_D denote the number of queries made by $\mathsf{A}^{\mathsf{CCA}}$ to the random oracles G and H and the decryption oracles (servers), respectively.

Proof: [Sketch] The basic idea of the proof is to construct an algorithm $\mathsf{A}^{\mathsf{CDH}}$ for solving the CDH problem by invoking an attack algorithm $\mathsf{A}^{\mathsf{CCA}}$ that defeats THD-IND-CCA of the scheme $\mathcal{TCG}$.

Below, we describe how $\mathsf{A}^{\mathsf{CDH}}$ simulates $\mathsf{A}^{\mathsf{CCA}}$'s view's in the real attack described in Definition 1. Suppose that $(\mathcal{G}, q, P, aP, bP)$ for $a, b \in_R \mathbb{Z}_q^*$ is given to $\mathsf{A}^{\mathsf{CDH}}$. The aim of this attacker is to find out abP, a Diffie-Hellman key of aP and bP.

Preparation: When $\mathsf{A}^{\mathsf{CCA}}$ corrupts a subset of $t-1$ servers where t is a threshold parameter, $\mathsf{A}^{\mathsf{CDH}}$ assumes the servers $P_1, P_2, \ldots, P_{t-1}$ have been corrupted without loss of generality. Let $S = \{0, 1, \ldots, t-1\}$. Then, $\mathsf{A}^{\mathsf{CDH}}$ chooses $x_1, x_2, \ldots, x_{t-1}$ uniformly at random from $\mathbb{Z}_q^*$ and sets $Y \stackrel{\mathrm{def}}{=} aP$ as a public key. Then it computes $Y_i = \lambda_{i0}^S Y + \sum_{j=1}^{t-1} x_j \lambda_{ij}^S P$ for $t \leq i \leq n$.

Simulation of the random oracle G: Whenever the random oracle G is queried at some point during the attack, $\mathsf{A}^{\mathsf{CDH}}$ checks if the value of the random oracle has been already defined at the given point. If there exists one, $\mathsf{A}^{\mathsf{CDH}}$ returns it to $\mathsf{A}^{\mathsf{CCA}}$ as an answer, otherwise it chooses a random value from appropriate space and returns the random value as answer to the query. Note that $\mathsf{A}^{\mathsf{CDH}}$ keeps records of all the "query-answer" pairs.

Simulation of the random oracle H: The random oracle H can be simulated in the same way as the random oracle G. However, this time, if there is no entry for the given query, $\mathsf{A}^{\mathsf{CDH}}$ picks s uniformly at random from $\mathbb{Z}_q^*$, computes sY and returns it as an answer. Note that $\mathsf{A}^{\mathsf{CDH}}$ also keeps records of all the "query-answer" pairs as well as the value $s \in \mathbb{Z}_q^*$.

Simulation of a target ciphertext: Suppose that $\mathsf{A}^{\mathsf{CCA}}$ submits two plaintext-messages (m_0, m_1) to the encryption oracle. On receiving (m_0, m_1), $\mathsf{A}^{\mathsf{CDH}}$ generates a random string V^* from the space $\{0,1\}^l$. (That is, it ignores the messages that $\mathsf{A}^{\mathsf{CCA}}$ submitted.) Now $\mathsf{A}^{\mathsf{CDH}}$ sets $U^* = bP$ and defines the value of $H^* \stackrel{\mathrm{def}}{=} \mathsf{H}(U^*, V^*)$ as s^*P for random $s^* \in \mathbb{Z}_q^*$. Then it sets $W^* = s^*U^*$. Finally it returns $C^* = (U^*, V^*, W^*)$ to $\mathsf{A}^{\mathsf{CCA}}$ as a target ciphertext. Note here that $(P, U^*, H^*, W^*) = (P, bP, s^*P, bs^*P)$ is a legitimate Diffie-Hellman tuple. Therefore, as long as $\mathsf{A}^{\mathsf{CCA}}$ does not query the random oracle H at the point abP, the simulation is perfect. However if such an event happens $\mathsf{A}^{\mathsf{CDH}}$ solves the CDH problem, so $\mathsf{A}^{\mathsf{CDH}}$ just simulates $\mathsf{A}^{\mathsf{CCA}}$'s view up to this event.

Simulation of the uncorrupted decryption servers: Recall that whenever $\mathsf{A}^{\mathsf{CCA}}$ queries H at a point (U, V) which is different from (U^*, V^*), $\mathsf{A}^{\mathsf{CDH}}$ picks s uniformly at random from $\mathbb{Z}_q^*$ and sets $H \stackrel{\mathrm{def}}{=} \mathsf{H}(U, V) = sY$. Now suppose that $\mathsf{A}^{\mathsf{CCA}}$ submits a decryption query $C = (U, V, W) \neq C^*$ to one of the uncorrupted decryption server P_i.

We claim that $(U, V) \neq (U^*, V^*)$. We prove this by a contradiction. Suppose that $(U, V) = (U^*, V^*)$. Then we have $U = U^*$ and $V = V^*$. Since H is well-defined, we have $\mathsf{H}(U, V)(\stackrel{\mathrm{def}}{=} H) = \mathsf{H}(U^*, V^*)(\stackrel{\mathrm{def}}{=} H^*)$. But, we also have $W = W^*$, since (P, U^*, H^*, W^*) is a Diffie-Hellman tuple by the construction above and the Diffie-Hellman key W^* (of U^* and H^*) is unique and hence $W = W^*$ by the assumption that $U = U^*$ and $H = H^*$. However this contradicts the fact that $C \neq C^*$. Therefore, we conclude that $(U, V) \neq (U^*, V^*)$.

Now, we demonstrate how $\mathsf{A}^{\mathsf{CDH}}$ can simulate decryption share generation upon decryption query $C = (U, V, W)$ made by $\mathsf{A}^{\mathsf{CDH}}$. Assume that $\mathsf{A}^{\mathsf{CCA}}$ has already made a query (U, V) to the random oracle H, so we have $H \stackrel{\mathrm{def}}{=} \mathsf{H}(U, V) = sY$. It is important to notice that $\mathsf{A}^{\mathsf{CDH}}$ knows the value $s \in \mathbb{Z}_q^*$. If C is legitimate, (P, U, H, W) is a Diffie-Hellman tuple, which passes the test checking whether $\hat{e}(P, W) = \hat{e}(U, H)$. In this case,

we have $(P, U, H, W) = (P, rP, sY, s(rY)) = (P, rP, sxP, srxP)$. Since $r, s, x \in \mathbb{Z}_q^*$, it is clear that above tuple is a Diffie-Hellman one. Now $\mathsf{A}^{\mathsf{CDH}}$ computes $(1/t)W = (1/s)srP = rY$. That is, $\mathsf{A}^{\mathsf{CDH}}$ has obtained the Diffie-Hellman key of U and Y without knowing r and x. Finally, $\mathsf{A}^{\mathsf{CDH}}$ can simulate the output (decryption share) of each uncorrupted server P_i, where $t \leq i \leq n$, by computing

$$\begin{aligned} U_i &= (\lambda_{i0}^S/s)W + \sum_{j=1}^{t-1} x_j \lambda_{ij}^S U = r\lambda_{i0}^S Y + \sum_{j=1}^{t-1} x_j \lambda_{ij}^S rP \\ &= r\lambda_{i0}^S Y + (rY_i - r\lambda_{i0}^S Y) = rY_i. \end{aligned}$$

Since $\mathsf{A}^{\mathsf{CDH}}$ has $U_i = rY_i$, it can determine whether (P, U, Y_i, U_i) is a Diffie-Hellman tuple by checking whether $\hat{e}(P, U_i) = \hat{e}(U, Y_i)$. ■

IV. Implementation Issues

To implement our threshold cryptosystem, the supersingular curves over a finite field with characteristic 3, which was used to construct the Boneh et al.'s short signature [7], might be the best choice in terms of computation and communication overheads. In these curves, e.g., an elliptic curve group whose size is 151 bits and provides a signature of length 154 bits with security comparable to that provided by the discrete logarithm problem in an 923-bit finite field. Accordingly, the size of a ciphertext and a decryption share of our scheme is much smaller than that of elliptic curve versions of the threshold cryptosystems proposed in [14] and [8].

In our scheme, we need to compute the hash function H which maps an arbitrary string to the group element. This hash function can be constructed in a similar way to the algorithm "*MapToGroup*" described in [7]. Also, we need several pairing computations. It was known that pairing computation, even the Tate pairing which is more efficient than the Weil pairing, is far more expensive than point multiplication,. Fortunately, significant improvements on pairing computation have been made quite recently by Barreto et al. [3]. Their results indicate that when the Tate pairing is used, the verification time of the Boneh et al.'s short signature scheme has been done nearly 55 times faster than all previously known methods. Hence, when the Barreto et al.'s pairing computation method is adopted, our threshold cryptosystem is expected to be preferable over other schemes in terms of computational and communication overheads.

V. Conclusion

In this paper, we have constructed a threshold cryptosystem from the GDH group. Our scheme is not only simple due to the elegant structure of the GDH group, but also enjoys all the important security requirements of a *robust* threshold cryptosystem.

References

[1] M. Abe, "Securing 'Encryption + Proof of Knowledge' in the Random Oracle Model", Proceedings of Topics in Cryptology - CT-RSA 2002, Vol. 2271 of LNCS, Springer-Verlag 2002, pages 277–289.

[2] J. Baek and Y. Zheng, "Simple and Efficient Threshold Cryptosystem from the Gap Diffie-Hellman Group", Full version, Available at `http://phd.pscit.monash.edu.au/joonsang`.

[3] P. Barreto, H. Kim, B. Lynn and M. Scott, "Efficient Algorithms for Pairing-Based Cryptosystems", Advances in Cryptology - Proceedings of CRYPTO 2002, Vol. 2442 of LNCS, Springer-Verlag 2002, pages 354–369.

[4] M. Bellare and P. Rogaway, "Random Oracles are Practical: A Paradigm for Designing Efficient Protocols", Proceedings of First ACM Conference on Computer and Communications Security 1993, pages 62–73.

[5] A. Boldyreva, "Efficient Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-group Signature Scheme", Proceedings of Public Key Cryptography 2003 (PKC 2003), Vol. 2567 of LNCS, Springer-Verlag 2003, pages 31–46.

[6] D. Boneh and M. Franklin, "Identity-Based Encryption from the Weil Pairing", Advances in Cryptology - Proceedings of CRYPTO 2001, Vol. 2139 of LNCS, Springer-Verlag 2001, pages 213–229.

[7] D. Boneh, B. Lynn and H. Shacham, "Short Signatures from the Weil Pairing", Advances in Cryptology - Proceedings of ASIACRYPT 2001, Vol. 2248 of LNCS, Springer-Verlag 2001, pages 566–582.

[8] P. Fouque and D. Pointcheval, "Threshold Cryptosystems Secure Chosen-Ciphertext Attacks", Advances in Cryptology - Proceedings of ASIACRYPT 2001, Vol. 2248 of LNCS, Springer-Verlag 2001, pages 351–368.

[9] A. Joux, "The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems", Proceedings of Algorithmic Number Theory Symposium (ANTS-V) 2002, Vol. 2369 of LNCS, Springer-Verlag 2002, pages 20–32.

[10] C. Lim and P. Lee, "Another Method for Attaining Security Against Adaptively Chosen Ciphertext Attack", Advances in Cryptology - Proceedings of CRYPTO '93, Vol. 773 of LNCS, Springer-Verlag 1993, pages 410–434.

[11] A. Lysyanskaya, "Unique signatures and verifiable random functions from the DH-DDH separation", Advances in Cryptology - Proceedings of CRYPTO 2002, Vol. 2242 of LNCS, Springer-Verlag 2002, pages 597–612.

[12] M. Naor and M. Yung, "Public-key Cryptosystems Provably Secure against Chosen Ciphertext Attacks", Proceedings of the 22nd Annual ACM Symposiumm on the Theory of Computing STOC, 1990, pages 427–437.

[13] A. Shamir, "How to Share a Secret", Communications of the ACM, Vol. 22, 1979, pages 612–613.

[14] V. Shoup and R. Gennaro, "Securing Threshold Cryptosystems against Chosen Ciphertext Attack", Advances in Cryptology - Proceedings of EUROCRYPT '98, Vol. 1403 of LNCS, Springer-Verlag 1998, pages 1–16.