A New ID-based Signature with Batch Verification

Jung Hee Cheon[1], Yongdae Kim[2] and Hyo Jin Yoon[1]

[1] School of Mathematical Sciences,
Seoul National University, Korea
{jhcheon, jin25}@math.snu.ac.kr
[2] Department of Computer Science
University of Minnesota - Twin Cities, USA
kyd@cs.umn.edu

Abstract. An identity (ID)-based signature scheme allows any pair of users to communicate securely and to verify each other's signatures without exchanging public key certificates. We have several ID-based signatures based on the discrete logarithm problem. While they have an advantage that the system secret can be shared by several parties through threshold schemes, they have a critical disadvantage in efficiency. To enhance the efficiency of verification, we propose a new ID-based signature scheme that allows batch verification of multiple signatures. The verification cost of the proposed signature scheme for k signatures is almost constant with minimal security loss and when a new signature by a different signer is added to the batch verification, the additional cost is almost a half of that of a single signature. We prove that the proposed signature scheme is secure against existential forgery under adaptively chosen message and ID attack in the random oracle model and show why other ID-based signature schemes are hard to achieve these properties.

Key words : ID-based signatures, Batch verifications

1 Introduction

In 1984, Shamir proposed a new model for public key cryptography, called identity (ID)-based encryption and signature schemes, to simplify key management procedures of certificate-based public key infrastructures (PKIs) [Sha84]. Since then, several ID-based encryption and signature schemes have been proposed based on integer factorization problem [DG86,Tan87,TI89,MY91]. While ID-based signature has advantages for key management and key recovery, it has a disadvantage that the signer's key is shared with the private key generator [FS86,FFS88]. This problem can be alleviated in the signatures based on the discrete logarithm problem (DLP) since the secret key can be shared by several parties through threshold schemes. There have been several ID-based signatures with these properties using pairings in elliptic curves [Hess02,Pat02,CC03].

In spite of several advantages of ID-based signatures schemes based on pairings, they suffer some restriction on applications due to efficiency problem: Their signature verifications are ten times or one hundred times slower than that of DSS or RSA [BKLS02].

This problem may be critical in some applications such as electronic commerce or banking service in which one server has to verify many signatures simultaneously. In order to enhance the efficiency of verification process, we consider batch verification of several signatures. Unfortunately, it appears that previous signatures such as [Hess02,Pat02,CC03] are not secure for batch verification of signatures signed by different users. In this paper, we propose a new signature scheme which allows secure batch verifications. Using the new scheme, we can reduce the signature size into almost a half and efficiently verify multiple signatures. The verification cost of k signatures by a single signer is one signature verification plus k elliptic curve addition and k hashing. When a new signature by a different signer is added, additional verification cost is almost a half of that of ordinary verification of a single signature.

We prove that the proposed signature scheme is secure against existential forgery for a chosen ID under adaptively chosen message and ID attack in the random oracle model. More precisely, we can show that if there is an attacker who can forge a set of signatures to pass batch verification, then the computational Diffie-Hellman problem (CDHP) is solved. Note that we do not require that one signature in the set should have a signer with a fixed ID as in non-ID-based model. The proof relies on the forking lemma and the property of the proposed scheme whose random part is removable by the simulator. To obtain a solution of a given CDHP from the forged signatures, we have to get rid of the all commitments simultaneously by the oracle replay. However, batch verification contains several random commitments and the usual forking lemma can remove only one commitment. That is the reason why similar security proof fails for other ID-based signature schemes and the Schnorr scheme based on the DLP. Further, we show that they are not secure for batch verification.

Batch verification was devised to improve the efficiency of verification process for multiple signatures and has been studied by many researchers. The homomorphic property of the RSA signature scheme admits a weak batch verification called *screening* [Fiat89,BGR98]. Screening means that a signature passed the batch verification is an already signed one by the legitimate signer at least one time at the past. In the DLP case, most efforts have been devoted to simultaneous verifications of modular exponentiations [NMVR96,MN96,BGR98,BP00]. This method is independent of specific signature schemes, but the efficiency gain is not so much from the sum of individual verifications when the security loss goes to zero. In 2003, Boneh *et al.* proposed aggregate signatures (BGLS scheme) from bilinear maps in which multiple signatures can be aggregated into one signature [BGLS03]. In the BGLS scheme, the verification cost of n signatures is almost constant when signed by a single signer and a half of n verifications when signed by different signers. Our signature scheme is the first ID-based signature which admits efficient batch verifications. The gain of batch verifications is almost the same with the BGLS scheme. While the BGLS compresses n signatures into one, the proposed one could compress only a half of signatures.

The rest of the paper is organized as follows: In Section 2, we introduce hard problems which our scheme relies on. In Section 3, we present a new ID-based signature scheme and its enhanced verification with rigorous security proof. We also discuss why other ID-based signatures and the Schnorr signature fail to provide secure batch verification. In Section 4, we analyze the efficiency of our scheme and its batch verification. We conclude in Section 5.

2 Preliminary

2.1 Bilinear Maps

Consider an additive cyclic group G of prime order ℓ and a cyclic multiplicative group V. Let $e\colon G \times G \to V$ be a map which satisfies the following properties.

1. **Bilinear** For any $aP, bP \in G$, $e(aP, bP) = e(P, P)^{ab}$.
2. **Non-degenerate** If $e(P, Q) = 1_V$ for all P (or Q) in G, then Q (or P) is the identity of G, respectively.
3. **Efficient** There exists an efficient algorithm to compute the map.

We call such a bilinear map as an admissible bilinear pairing.

The Weil pairing and Tate pairing in elliptic curve give good implementations of the admissible bilinear pairing. Let E be an elliptic curve over $\mathbb{F}_q$ where $q = p^n$ and p is a prime. For a prime ℓ and an ℓ torsion subgroup $E[\ell]$ of E, we define a Weil pairing $e : E[\ell] \times E[\ell] \to \mathbb{F}_{q^\alpha}^*$ for suitable α. Now let $G = E(\mathbb{F}_q)[\ell]$ and define a map $\widehat{e}\colon G \times G \to \mathbb{F}_{q^\alpha}^*$, where $\widehat{e}(P, Q) = e(P, \phi(Q))$ and ϕ is an automorphism over G. Then $\widehat{e}$ is an efficiently computable non-degenerate bilinear map. The Tate pairing has similar properties and is more efficient than the Weil pairing. For the details, refer to [BLS01].

2.2 Some Problems

Let G be a cyclic group of prime order ℓ and P a generator of G.

1. The decisional Diffie-Hellman Problem (DDHP) is to decide whether $c = ab$ in $\mathbb{Z}/\ell\mathbb{Z}$ for given $P, aP, bP, cP \in G$. If so, (P, aP, bP, cP) is called a valid Diffie-Hellman (DH) tuple.
2. The computation Diffie-Hellman Problem (CDHP) is to compute abP for given $P, aP, bP \in G$.

Now we define a gap Diffie-Hellman (GDH) group.

Definition 1 *A group G is a gap Diffie-Hellman (GDH) group if the decisional Diffie-Hellman problem in G can be efficiently computable and there exists no algorithm which can solve the computational Diffie-Hellman problem in G with non-negligible probability within polynomial time.*

If we have an admissible bilinear pairing e in G, we can solve the DDHP in G efficiently as follows:

$$(P, aP, bP, cP) \text{ is a valid DH tuple} \Leftrightarrow e(aP, bP) = e(P, cP).$$

Hence an elliptic curve becomes an instance of a GDH group if the Weil (or the Tate) pairing is efficiently computable and the CDHP is sufficiently hard on the curve.

3 A New ID-based Signature

From now on, we assume that G is a GDH group generated by P, whose order is a large prime ℓ.

3.1 An ID-based Signature

This scheme consists of four algorithms: *Setup, Extract, Signing* and *Verification.*

Setup Given a GDH group G and its generator P, pick a random $s \in \mathbb{Z}/\ell\mathbb{Z}$ and set $P_{pub} = sP$. Choose two hash functions $H_1 : \{0,1\}^* \times G \to (\mathbb{Z}/\ell\mathbb{Z})^*$ and $H_2 : \{0,1\}^* \to G^*$. The system parameter is (P, P_{pub}, H_1, H_2). The master key is s.

Extract Given an identity ID, the algorithm computes $Q_{\mathrm{ID}} = H_2(\mathrm{ID})$ and $D_{\mathrm{ID}} = sH_2(\mathrm{ID})$ and outputs D_{ID} as a private key of the identity ID corresponding to $Q_{\mathrm{ID}} = H_2(\mathrm{ID})$.

Signing Given a secret key D_{ID} and a message m, pick a random number $r \in \mathbb{Z}/\ell\mathbb{Z}$ and output a signature $\sigma = (U, V)$ where $U = rP$, $h = H_1(m, U)$, and $V = rQ_{\mathrm{ID}} + hD_{\mathrm{ID}}$.

Verification Given a signature $\sigma = (U, V)$ of a message m for an identity ID, compute $h = H_1(m, U)$. The signature is accepted if and only if $(P, Q_{\mathrm{ID}}, U + hP_{pub}, V)$ is a valid Diffie-Hellman tuple.

3.2 Security Proof

Our ID-based signature scheme contains a random value in its signature. We cannot directly reduce the security of our scheme to the hardness of the CDLP because of such a random value. To remove the random value in the forged signature produced by a forger $\mathcal{F}$, we use the oracle replay method and the forking lemma [PS00] as in [CC03].

At first, we can reduce the adaptively chosen ID attack to the *given* ID attack by the following lemma.

Lemma 1 ([CC03, Lemma 1]). *If there is a forger $\mathcal{F}_0$ for an existential forgery under adaptively chosen message and ID attack to our scheme within time bound T_0 with probability ϵ_0, then there is a forger $\mathcal{F}$ for an existential forgery under an adaptively chosen message and given ID attack within time bound $T \le T_0$ with the probability*

$\epsilon \le \epsilon_0(1-\frac{1}{\ell})/q_{H_2}$, where q_{H_2} is the maximum number of queries to H_2 asked by $\mathcal{F}_0$ and ℓ is a security parameter. In addition, the number of queries to hash functions, **Extract** *and* **Signing** *asked by $\mathcal{F}_0$ are the same as those of $\mathcal{F}$.*

Theorem 1. *Let $\mathcal{F}_0$ be a forger which performs, within a time bound T_0, an existential forgery under an adaptively chosen message and* ID *attack against our* ID*-based scheme with probability ϵ_0 in random oracle model. The forger $\mathcal{F}_0$ can ask queries to the oracles H_1, H_2,* **Extract** *and* **Signing** *at most q_{H_1}, q_{H_2}, q_E, and q_S-times, respectively. Assume that $\epsilon_0 \ge (10(q_S+1)(q_S+q_{H_1})q_{H_2})/(\ell-1)$, then the CDHP can be solved with probability $\ge$ 1/9 and within running time $\le (23q_{H_1}q_{H_2}T_0)/\left(\epsilon_0\left(1-\frac{1}{\ell}\right)\right)$ where ℓ is a security parameter.*

Proof. Using the Lemma 1, we can reduce the forger $\mathcal{F}_0$ to $\mathcal{F}$ an adaptively chosen message and given ID attack within time bound $T \le T_0$ with the probability $\epsilon \le \epsilon_0(1-\frac{1}{\ell})/q_{H_2}$. We construct an algorithm $\mathcal{C}$ using $\mathcal{F}$ to solve the CDHP. We assume that P, aP, and bP are given. Since the forger $\mathcal{F}$ is an adaptively chosen message attacker, he can access to the hash oracles, the extraction oracle, and the signing oracle, and ask at most q_{H_1}, q_{H_2}, q_E, and q_S queries for each oracles respectively. The algorithm $\mathcal{C}$ simulates a real signer to get a valid signature from the forger $\mathcal{F}$. If $\mathcal{C}$ does not fail this simulation, he gets a valid signature, and using the oracle replaying technique he can solve the CDHP.

We may assume the forger is well-behaved in the following sense: A forger $\mathcal{F}$ makes a **Extract** query for an ID only if an H_2 query has been made before for the ID. Also **Signing** query is made for a message m only if a H_1 queries has been made before for the m.

Then the algorithm $\mathcal{C}$ puts $P_{pub} = aP$ and performs the following game with the forger $\mathcal{F}$ for a fixed identity ID as follows:

ID-Hash Query When $\mathcal{F}$ makes an ID-hash query ID_i, $\mathcal{C}$ gives to $\mathcal{F}$ an answer $H_2(\mathrm{ID}_i) = bP$ if $\mathrm{ID}_i = \mathrm{ID}$ and $H_2(\mathrm{ID}_i) = x_iP$ for $x_i \in_R \mathbb{Z}/\ell$ otherwise.

Extract Query When $\mathcal{F}$ makes an extract query for ID_{i_k}, $\mathcal{C}$ gives $x_{i_k}P_{pub} = x_{i_k}(aP)$ as the secret key corresponding to $H_2(\mathrm{ID}_{i_k})$ for an identity ID_{i_k}. Note that $\mathcal{F}$ must not ask the secret key corresponding to the $bP = H_2(\mathrm{ID})$.

Message-Hash Query $\mathcal{F}$ makes q_H message-hash queries. For the j-th hash query $\mathcal{Q}_j$, $\mathcal{C}$ chooses a random value $h_j \in \mathbb{Z}/\ell$ and gives to $\mathcal{F}$ as the hash value of $\mathcal{Q}_j$ for $j = 1, \cdots q_{H_1}$ and stores them as $H_1(\mathcal{Q}_j) = h_j$.

Signing Query If $\mathcal{F}$ asks the signature on m_{j_t} of ID_{i_t}, $\mathcal{C}$ chooses a random value $r_t \in \mathbb{Z}/\ell$ responses

$$Sign(\mathrm{ID}_{i_t}, m_{j_t}) = (\mathrm{ID}_{i_t}, m_{j_t}, U_t, h_t, V_t),$$

where $U_t = r_t P - h_t P_{pub}$ and $V_t = r_t(x_{i_t} P)$ for $t = 1, \cdots, q_S$. Since $(P, H_2(\mathrm{ID}_{i_t}), U_t + h_t P_{pub}, V_t)$ is a valid Diffie-Hellman tuple, these signatures pass the verification algorithm.

If the simulation does not fail, the forger $\mathcal{F}$ outputs a valid signature $(\mathrm{ID}, m, U, h, V)$ with probability ϵ. After a replay of the forger $\mathcal{F}$, apply the forking lemma in [PS00]. Then $\mathcal{C}$ obtains two valid signatures $\sigma = (\mathrm{ID}, m, U, h, V)$ and $\sigma' = (\mathrm{ID}, m, U, h', V')$ such that $h \neq h'$ with probability $\geq 1/9$ within the time $23 q_{H_1} T/\epsilon$. $\mathcal{C}$ can easily obtain the value abP from

$$\frac{(hD_{\mathrm{ID}} - h'D_{\mathrm{ID}})}{h - h'} = D_{\mathrm{ID}} = abP.$$

By the forking lemma [PS00] and the Lemma 1, we obtain the result of this theorem. □

3.3 Enhancing Signature Verification

We construct an efficient batch verification for k signatures. We denote by (ID, m, U, V) a signature (U, V) for a message m by a signer with an identity ID.

Aggregation Given k signatures $(\mathrm{ID}_1, m_1, U_1, V_1), \ldots, (\mathrm{ID}_k, m_k, U_k, V_k)$ compute $V = \sum_{i=1}^{k} V_i$ and output an aggregate signature

$$\sigma = (\mathrm{ID}_1, \ldots, \mathrm{ID}_k, m_1, \ldots, m_k, U_1, \ldots, U_k, V).$$

Aggregate Verification Given an aggregate signature σ as above, compute $Q_i = H_2(\mathrm{ID}_i)$ and $h_i = H_1(m_i, U_i)$ for all $i = 1, \cdots, k$. The aggregate signature is accepted if and only if

$$e(P, V) = \prod_{i=1}^{n} e\left(Q_i, U_i + h_i P_{pub}\right).$$

3.4 Security Proof of Aggregate Verification

Now we discuss the security of our aggregate verification. Boneh *et al.* suggested the aggregate chosen key model [BGLS03] for the security of aggregate signatures, in which a forger performs an existential forgery under an adaptively chosen-message attack in the random oracle model. In this model, a forger is given a target public key for which a forged signature should be made. While each secret key of users is chosen independently in the traditional public key system, all secret keys of users are mutually related in ID-based system. In fact, they are produced from one secret key of the whole system. Hence in ID-based setting it is reasonable to give not an specific ID but a system parameter to a forger. More precisely, a forger succeeds if he can produce a set of k signatures which pass the aggregate verification. We call this type of forger *a k-aggregate forger of a chosen* ID. On the other hand, if a forger produces a set of k signatures one of which has the signer with the given ID, then this type of forger is called *a k-aggregate forger of a given* ID.

Lemma 2. *If there is a k-aggregate forger $\mathcal{F}_0$ of a chosen* ID *under an adaptively chosen message and ID attack to our scheme within time bound T_0 with probability ϵ_0, then there is a k-aggregate forger $\mathcal{F}$ of a given* ID *under an adaptively chosen message and ID attack within time bound $T \leq T_0$ with the probability $\epsilon \leq \epsilon_0 \left(1 - \frac{k}{\ell}\right)\left(\frac{k}{q_{H_2}+k}\right)$, where q_{H_2} is the maximum number of queries to H_2 asked by $\mathcal{F}_0$, ℓ is a security parameter and k is the maximum number of signatures to be aggregated. In addition, the number of queries to hash functions,* **Extract** *and* **Signing** *asked by $\mathcal{F}_0$ are the same as those of $\mathcal{F}$.*

Proof. We assume, without loss of generality, a k-aggregate forger $\mathcal{F}_0$ has an extract queries for any ID at most once. We consider an algorithm $\mathcal{F}$ that performs the following simulation:

Setup $\mathcal{F}$ chooses a random number $r \in \{1, \cdots, q_{H_1}\}$. Let ID_i be the $\mathcal{F}_0$'s i-th H_2-query and $\mathrm{ID}'_i = ID$ if $i = r$ and $\mathrm{ID}'_i = \mathrm{ID}_i$ otherwise. Let $H'_2(ID_i) = H_2(ID'_i)$, **Extract**$'(ID_i)$=**Extract**(ID'_i) and **Signing**$'(ID_i, m_i)$=**Signing**(ID'_i, m_i)

Queries If $\mathcal{F}_0$ makes the H_1, H_2 hash queries and **Extract**, **Signing** queries, then $\mathcal{F}$ computes H_1, H'_2, **Extract**$'$and **Signing**$'$ as above and answers the results.

If the simulation does not fail, $\mathcal{F}_0$ outputs a tuple $(\mathrm{ID}^1_{out}, \cdots, \mathrm{ID}^k_{out}, m_1, \cdots, m_k, \sigma)$ where σ is the aggregation of the k signatures with probability ϵ_0. Finally, if $\mathrm{ID}^i_{out} = ID$ for some $i = 1, \cdots, k$ and $\mathrm{ID}^j_{out} \neq \mathrm{ID}$ for all $j \neq i$ and $(\mathrm{ID}^1_{out}, \cdots, \mathrm{ID}^{i-1}_{out}, \mathrm{ID}, \mathrm{ID}^{i+1}_{out}, \cdots, \mathrm{ID}^k_{out}, m_1, \cdots, m_k, \sigma)$ is a valid tuple, then $\mathcal{F}$ outputs $(\mathrm{ID}^1_{out}, \cdots, \mathrm{ID}^{i-1}_{out}, \mathrm{ID}, \mathrm{ID}^{i+1}_{out}, \cdots, \mathrm{ID}^k_{out}, m_1, \cdots, m_k, \sigma)$. Otherwise the simulation fails.

Since the output distributions of H'_2, **Extract**$'$, **Signing**$'$-queries are not distinguishable those of original ones, we know

$$\Pr[(\mathrm{ID}^1_{out}, \cdots, \mathrm{ID}^k_{out}, m_1, \cdots, m_k, \sigma) \text{ is valid }] \geq \epsilon.$$

Since we consider the hash functions as the random oracles, we obtain the following result.

$$\begin{aligned}&\Pr[\mathrm{ID}^j_{out} = ID_i \text{ for some } j = 1, \cdots, k \text{ and } i = 1, \cdots q_{H_2} \\ &\qquad \mid (\mathrm{ID}^1_{out}, \cdots, \mathrm{ID}^k_{out}, m_1, \cdots, m_k, \sigma) \text{ is valid}] \geq \left(1 - \frac{1}{\ell}\right)^k \geq 1 - \frac{k}{\ell}\end{aligned}$$

Furthermore since the randomness of r, we have the following inequality.

$$\Pr[\mathrm{ID}_{out}^{i} = ID_r \text{ for some } i = 1, \cdots, k \text{ and } \mathrm{ID}_{out}^{j} \neq \mathrm{ID} \text{ for some } j = 1, \cdots, i-1, \\ i+1, \cdots, k \mid \mathrm{ID}_{out}^{j} = ID_i \text{ for some } j = 1, \cdots, k \text{ and } i = 1, \cdots q_{H_2}] \\ \geq \frac{{}_{q_{H_2}-1}\mathrm{H}_{\mathrm{k}-1}}{{}_{\mathrm{q}_{\mathrm{H}_2}}\mathrm{H}_{\mathrm{k}}} \geq \frac{k(q_{H_2}-1)}{(q_{h_2}+k-1)(q_{H_2}+k-2)} \geq \frac{k}{2(q_{H_2}+k)}$$

Finally, summarizing these, we get the following result as desired

$$\Pr[\mathrm{ID}_{out}^{i} = ID_r = ID \text{ for some } i = 1, \cdots, k \text{ and } \mathrm{ID}_{out}^{j} \neq \mathrm{ID} \text{ for some } \\ j = 1, \cdots, i-1, i+1, \cdots, k \text{ and } (\mathrm{ID}_{out}^{1}, \cdots, \mathrm{ID}_{out}^{k}, m_1, \cdots, m_k, \sigma) \\ \text{is valid}] \geq \epsilon \cdot \left(1 - \frac{1}{\ell}\right) \cdot \frac{k}{2(q_{H_2}+k)}. \square$$

Now in the random oracle model we show that if there exists a k-aggregate forger $\mathcal{F}$ of given ID under an adaptively chosen message and ID attack, then there exists an algorithm $\mathcal{C}$ which can solve the CDHP. The forger $\mathcal{F}$ performs the following game:

Setup The k-aggregate forger $\mathcal{F}$ is given an ID_0.
Queries $\mathcal{F}$ adaptively asks the hash values of his chosen IDs (including ID_0), the secret keys by his chosen ID.
Response $\mathcal{F}$ outputs a signature aggregation

$$\sigma = (\mathrm{ID}_1, \ldots, \mathrm{ID}_k, m_1, \ldots, m_k, U_1, \ldots, U_k, V),$$

which passes an aggregate verification. Here one of ID_i should be equal to ID_0 and then m_i has not been asked to the signature oracle for ID_i.

Lemma 3. *Let $\mathcal{F}$ be a k-aggregate forger which succeeds the above game within a time bound T with probability ϵ in the random oracle model. We denote by q_{H_1}, q_{H_2}, q_E, and q_S the maximum number of queries that $\mathcal{F}_0$ can ask to the oracles H_1, H_2,* **Extract***, and* **Signing** *oracles, respectively. If $\epsilon \geq (10(q_S+1)(q_S+q_{H_1}))/\ell$, then the CDHP can be solved with probability $\geq 1/9$ and within running time $\leq (23 q_{H_1} T)/\epsilon$.*

Proof. We construct an algorithm $\mathcal{C}$ using the k-aggregate forger $\mathcal{F}$ to solve the CDHP. We assume that P, aP, bP are given as the CDHP instances. The algorithm $\mathcal{C}$ simulates a real signer to get a valid signature from $\mathcal{F}$. If $\mathcal{C}$ does not fail this simulation, he gets a valid signature and using the general oracle replaying technique, it can solve the CDHP. In Setup, the algorithm $\mathcal{C}$ fixes a target identity ID_0, and put $P_{pub} = aP$.

Note that ID**-Hash Query, Extract Query, Message-Hash Query**, and **Signing Query** are the same as the single signature case. After the queries, if the simulation does not fail, the forger $\mathcal{F}$ outputs a signature aggregation

$$\sigma = (\{\mathrm{ID}_1, \mathrm{ID}_2, \cdots, \mathrm{ID}_n\}, \{m_1, \cdots, m_n\}, \{U_1, U_2, \cdots, U_n\}, \{h_1, \cdots, h_n\}, V)$$

where $n \leq k$ and one of ID_i is equal to ID_0.

Now $\mathcal{C}$ replays the oracles and obtains another valid signature σ'

$$\sigma' = (\{\mathrm{ID}'_1, \mathrm{ID}'_2, \cdots, \mathrm{ID}'_{n'}\}, \{m'_1, \cdots, m'_{n'}\}, \{U'_1, U'_2 \cdots, U'_{n'}\}, \{h'_1, \cdots, h'_{n'}\}, V')$$

where $n' \leq k$. By the forking lemma, the replay succeeds with the probability $\geq 1/9$ and the running time $\leq (23q_{H_2}T)/\epsilon$. Note that we may assume $h_1 \neq h'_1$ since the probability of collision of two random numbers is negligible. Since $\mathcal{F}$ performs an attack for ID_0, both of σ and σ' must contain a signature for the ID_0. Further since the random commitment r is fixed before the hash queries of a message, the corresponding random commitment of σ must be the same with that of σ by the forking lemma. That is, we have $\mathrm{ID}_i = \mathrm{ID}'_j = \mathrm{ID}_0$ and $U_i = U'_j$ for some $i \in \{1, \cdots, n\}$ and $j \in \{1, \cdots, n'\}$. Remark that according to the **Extract Query**, $\mathcal{C}$ knows the discrete log of the secret keys except that of ID_0. Hence from

$$V = \sum_{i=1}^{n} V_i = \sum_{i=1}^{n} (r_i Q_i + h_i D_i) = \sum_{i=1}^{n} \{x_i(r_i P) + h_i D_i\},$$
$$V' = \sum_{i=1}^{n'} V'_i = \sum_{i=1}^{n'} \{x'_i(r'_i P) + h'_i D'_i\},$$

we obtain two equations

$$\alpha = r_1 Q_1 + h_1 D_1, \quad \beta = r_1 Q_1 + h'_1 D_1.$$

Finally from the equation $\alpha - \beta = h_1 D_1 - h'_1 D_1$, we obtain $abP = D_1 = (\alpha - \beta)(h_1 - h'_1)^{-1}P$ as desired. The total running time is bounded by the running time of the forking lemma. □

¿From the Lemma 2 and Lemma 3, we obtain the following result.

Theorem 2 *Let $\mathcal{F}_0$ be a k-aggregate forger for an existential forgery under an adaptively chosen message and ID attack to our scheme within a time bound T_0 with probability ϵ_0. We denote by q_{H_1}, q_{H_2}, q_E, and q_S the maximum number of queries that $\mathcal{F}_0$ can ask to the oracles H_1, H_2,* **Extract**, *and* **Signing** *oracles respectively. If $\epsilon_0 \geq (10(q_S + 1)(q_S + q_{H_1})(q_{H_2} + k)q_{H_2})/k(\ell - k)$, then the CDHP can be solved with probability $\geq$ 1/9 and within running time $\leq (23q_{H_1}(q_{H_2} + k)T_0)/\left(\epsilon_0 k \left(1 - \frac{k}{\ell}\right)\right)$.*

3.5 What if we use other ID-based Signatures

We may consider other signature schemes based on the DLP for enhanced verifications. In this subsection, however, we show that the batch verification is not secure for some ID-based signature schemes (CC scheme [CC03], Hess scheme [Hess02], and Paterson scheme [Pat02]) and Schnorr signature [Sch89].

First, we consider the CC scheme.

Setup, Extract, Signing Same as the original CC scheme.

Aggregation Given users' identities $\mathrm{ID}_1, \cdots, \mathrm{ID}_k$, compute $Q_i = H_1(\mathrm{ID}_i)$ for all $i = 1, \cdots, k$. Given messages $m_1, \cdots, m_k$ (The messages needs not be distinct.), signatures $\sigma_i = (U_i, V_i)$ where $U_i = r_i Q_i$, $V_i = (r_i + h_i)D_i$ and $h_i = H_1(m_i, U_i)$, compute $V = \sum_{i=1}^{k} V_i$ and output $\sigma = (U_1, \cdots, U_k, V)$ as an aggregate signature.

Aggregate Verification Given an aggregate signature $\sigma = (U_1, \cdots, U_k, V)$ of messages $m_1, \cdots, m_k$ for users' identities $\mathrm{ID}_1, \cdots, \mathrm{ID}_k$, compute $Q_i = H_2(\mathrm{ID}_i)$ and $h_i = H_1(m_i, U_i)$ for all $i = 1, \cdots, k$. The aggregate signature is accepted if and only if

$$e(P, V) = e\left(P_{pub}, \textstyle\sum_{i=1}^{k} U_i + \sum_{i=1}^{k} h_i Q_i\right).$$

We will show that the aggregate verification of the CC signatures is not secure: We consider an aggregate forger which performs the following attack. Let ID_1 be an identity of a user $\mathcal{U}_1$ and ID_2 an identity of an aggregate forger $\mathcal{F}$. We may assume that $\mathcal{F}$ has access to the **ID-hash** oracle, so gets the public keys Q_1, Q_2 corresponding to ID_1 and ID_2 respectively. Now $\mathcal{F}$ selects two random values r_1, r'_2 and messages m_1, m_2, compute $U_1 = r_1 Q_1$, $h_1 = H_1(m_1, U_1)$ and

$$U_2 = r'_2 Q_2 - h_1 Q_1 - r_1 Q_1.$$

Finally, $\mathcal{F}$ computes $h_2 = H_2(m_2, U_2)$ and $V = (r'_2 + h_2)D_2$, and outputs a forged aggregate signature

$$\sigma = (U_1, U_2, V).$$

Though $\mathcal{F}$ does not know the discrete log r_2 of U_2, this forged aggregate signature passes the verification algorithm:

$$\begin{aligned} e(P_{pub}, U_1 + h_1 Q_1 + U_2 + h_2 Q_2) &= e(P, r_1 D_1 + h_1 D_1 + r'_2 D_2 - h_1 D_1 - r_1 D_1 + h_2 D_2) \\ &= e(P, r'_2 D_2 + h_2 D_2) \\ &= e(P, V). \end{aligned}$$

That is, the forged signature σ can be regarded as an aggregate signature on m_1 and m_2.

Remark 3 *We may consider an aggregate verification of the Hess scheme. In the original Hess scheme, we must compute a hash value and compare it with some value to verify. But a hash function does not have any homomorphic property, thus we cannot use directly the original Hess scheme for aggregate verification. Hence we slightly modify the scheme to get an aggregation of two signatures:* $\sigma = (U_1, U_2, V)$[1] *where* $h_i = H_1(m_i, U_i)$, $U_i = e(P, R_i)$, $V = \sum_{i=1}^{k} V_i$ *and* $V_i = h_i D_i + R_i$ $(i = 1, 2)$. *Similarly to the CC scheme, let* $U_2 = e(P_{pub}, -h_1 Q_1) \cdot e(P, R_2') = e(P, -h_1 D_1 + R_2')$ *where a random point* R_2 *is the same role as* $U_2 = r_2 Q_{ID}$ *in CC scheme, then the forged signature* σ *passes the aggregate verification process:*

$$e(P, V) = e\left(P_{pub}, h_1 Q_1 + h_2 Q_2\right) \cdot U.$$

Remark 4 *For the Paterson signature, the original scheme is* (R, S) *where* $R = kP$ *and* $S = k^{-1}(H_2(M)P + H_3(R)D_{\mathrm{ID}})$ *according to [Pat02]. The verification is to check the equality* $e(R, S) = e(P, P)^{H_2(M)} \cdot e(P_{pub}, Q_{\mathrm{ID}})^{H_3(R)}$. *In this equation,* R *and* S *are located the same side and the random value which is distinct for each user is multiplied both* P *and* D_{ID}, *so it cannot be aggregated.*

Remark 5 *In the Schnorr signature case, similarly to CC scheme, we can construct a forged aggregate signature* $\sigma = (r_1, r_2, s)$ *where* $r_i = g^{K_i}$, $s = s_1 + s_2$, $s_i = K_i + e_i x_i$ *and* $e_i = h(m_i, r_i)$ $(i = 1, 2)$ *following the notation in [PS00]. Verify* $g^s = r_1 \cdot r_2 \cdot y_1^{e_1} \cdot y_2^{e_2}$. *But if we let* $r_2 = r_2' - x_1 e_1$, *then the forged aggregate signature becomes* $(g^{K_1}, g^{K_2}, r_1 + r_2' + e_2 x_2)$, *which passes the aggregate verification process.*

4 Efficiency

In this section, we discuss the efficiency of our scheme and its batch verification. Here we assume that we use as a GDH group an elliptic curve with an admissible Tate pairing. First, we note that our ID-based signature has the similar efficiency in signing and a single verification with the previous ID-based signatures [Boyen03] as in Table 1.

Batch verification enhances efficiency of verification especially for signatures signed by a single signer: Given k signatures $(U_1, V_1), \ldots, (U_k, V_k)$ for messages $m_1, \cdots, m_k$ issued by a signer with an identity ID, compute $Q = H_2(\mathrm{ID})$ and $h_i = H_1(m_i, U_i)$ for all $i = 1, \cdots, k$. The k signatures are accepted if and only if

$$e\left(P, \sum_{i=1}^{k} V_i\right) = e\left(Q, \left(\sum_{i=1}^{k} U_i\right) + \left(\sum_{i=1}^{k} h_i\right) P_{pub}\right).$$

It requires two pairing computations, one scalar multiplication, k elliptic curve additions, and $k + 1$ hashes. Since elliptic curve additions and hashes are far more efficient

[1] Its individual signature (U_i, V_i) is verified as $e(P, V_i) = e(P_{pub} h_i Q_i) \cdots U_i$ where $i = 1, 2$.

Table 1. Comparison of Efficiency for ID-based Signatures

	Schemes	Pairing	Point Mul on G	Exp in V	Hash
Signing	Paterson [Pat02]	0	4	0	1
	Hess [Hess02]	1	2	1	1
	CC [CC03]	0	2	0	1
	Ours	0	3	0	1
Verification	Paterson [Pat02]	2	0	2	1
	Hess [Hess02]	2	0	1	1
	CC [CC03]	2	1	0	1
	Ours	2	1	0	1

than pairing computations, we can say that the batch verification is almost constant for the number of signatures by a single signer.

When a signature by a different signer is added to batch verification, one pairing and one scalar multiplication are added. Since a pairing computation is almost ten times slower than a scalar multiplication and the others are trivial [BKLS02], the additional verification cost is almost a half of that of a single signature.

When we verify signatures, we need only $\sum_{i=1}^{k} V_k$ rather than individual V_i', so we can half signature sizes when using batch verification. Note that U_i's cannot be aggregated into one element since each of them is used as an input of a hash function.

5 Conclusion and Open problem

In this paper, we proposed a new ID-based signature scheme admitting secure and efficient batch verification. When we add one signature by the same signer to batch verification, the additional cost is only a hash plus one point addition in an elliptic curve. One can, therefore, verify many signatures at the cost of almost one signature verification. If a signature by a different signer is added, the additional cost is a half of the single verification.

Aggregated Signature is a generalized version of Batch Signature, where many signatures for different messages signed by different signers are aggregated into one signature and verified by one equation. We may extend the notion of aggregate signatures to (ϵ, δ) aggregate (or batch) signatures where ϵ and δ are compression ratio for signature size and verification cost (i.e. Number of expensive cryptographic operations such as modular exponentiations or Bilinear maps). For example, the BGLS scheme is a $(1/k, 1)$ aggregate signature, and $(1/k, 1/k)$ aggregate signature is optimal where k signatures are aggregated. In this sense, our signature gives $(1/2, 1/2)$ aggregate signature and $(1/2, 1/k + \tau)$ batch signature for very small constant τ. It is an open problem to find $(1/k, 1/k)$ aggregate signatures or $(1/k, 1/k)$ ID-based batch signatures.

References

[BK02] P. Barreto and H. Kim, Fast Hashing onto Elliptic Curves over Fields of Characteristic 3, *Available from http://eprint.iacr.org*, 2002.

[BKLS02] P. Barreto, H. Kim, B. Lynn and M. Scott, Efficient Algorithms for Pairing- Based Cryptosystcms, *Advances in Cryptology - Crypto 2002*, LNCS, Vol. 2442, pp. 354–368, Springer-Verlag, 2002.

[BGR98] M. Bellare, J. Garay, and T. Rabin. Fast batch verification for modular exponentiation and digital signatures. *Advances in Cryptology - Eurocrypt'98*, LNCS Vol. 1403, pp. 236–250, Sringer-Verlag, 1998.

[BGLS03] D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. *Advances in Cryptology - Eurocrypt 2003*, LNCS Vol. 2656, pp. 416–432, Springer-Verlag, 2003.

[BLS01] D. Boneh, B. Lynn, and H. Shacham. Short signature from the Weil pairing. *Advances in Cryptology - Asiacrypt 2001*, LNCS Vol. 2248, pp. 514–531, Springer-Verlag, 2001. The extended version is available at `http://crypto.stanford.edu/~dabo/abstracts/weilsigs.html`.

[BP00] C. Boyd and C. Pavlovski. Attacking and repairing batch verification schemes. *Advances in Cryptology - Asiacrypt 2000*, LNCS, Vol. 1976, pp. 58–71, Springer-Verlag, 2000.

[Boyen03] X. Boyen. Multipurpose Identity-Based Signcryption - A Swiss Army Knife for Identity-Based Cryptography. *A*dvances in Cryptology - Crypto 2003, LNCS, Vol. 2729, pp. 383 – 399, Springer-Verlag, 2003.

[CC03] J. Cha and J. Cheon. An ID-based signature from gap-Diffie-Hellman groups. *Public Key Cryptography - PKC 2003*, LNCS Vol. 2567, pp. 18–30, Springer-Verlag, 2003.

[DG86] Y. Desmedt and J. Quisquater. Public-key Systems based on the Difficulty of Tampering. *A*dvances in Cryptology - Crypto '86, LNCS, Vol. 263, pp.111–117, Springer-Verlag, 1987.

[FFS88] U. Feige, A. Fiat, and A. Shamir. Zero-knowledge Proofs of Identity. *J. Cryptology*, Vol. 1, pp. 77–94, 1988.

[Fiat89] A. Fiat. Batch RSA. *J. Cryptology*, Vol. 10, No. 2, pp. 75–88, Springer-Verlag, 1997. A preliminary version appeared in *Advances in Cryptology - Crypto'89*, LNCS Vol. 435, pp, 175–185, Springer-Verlag, 1989.

[FS86] A. Fiat and A. Shamir. How to prove youself: Practical Solutions to Identification and Signature Problems. *A*dvances in Cryptology - Crypto '86, LNCS, Vol. 263, pp. 186–194, Springer-Verlag, 1987.

[Hess02] F. Hess. Efficient Identity Based Signature Schemes Based on Pairings. *Selected Areas in Cryptography - SAC 2002*, LNCS, Vol. 2595, pp. 310–324, Springer-Verlag, 2002.

[HHM00] D. Hankerson, J. Hernandez, and A. Menezes. Software Implementation of Elliptic Curve Cryptography Over Binary Fields. *Proc. of CHES 2000*, LNCS, Vol. 1965, pp. 1–24, Springer-Verlag, 2000.

[Joux00] A. Joux. A One Round Protocol for Tripartite Diffie-Hellman. *Algorithmic Number Theory Symposium, ANTS-IV*, LNCS, Vol. 1838, pp. 385–394, Springer-Verlag, 2000.

[LL94] C.H. Lim and P.J. Lee. More flexible exponentiation with precomputation. *Advences in cryptology - Crypto'94*, LNCS Vol. 839, pp. 95–107, Springer-Verlag, 1994.

[MN96] D. M'Raithi and D. Naccache. Batch Exponentiation - A Fast DLP based Signature Generattion Strategy. *ACM Conference on Computer and Communications Security*, pp. 58–61, ACM, 1996.

[MY91] U. Maurer and Y. Yacobi. Non-interective public-key cryptography. *Advances in Cryptology - Eurocrypto '91*, LNCS, Vol. 547, pp. 458–460, Springer-Verlag, 1992.

[NMVR96] D. Naccache, D. M'Raithi, S. Vaudenay, and D. Raphaeli. Can D.S.A be improved? Complexity trade-offs with the Digital Signature Standard. *Advances in Cryptology - Eurocrypt '94*, LNCS, Vol. 950, pp. 77–85, Springer-Verlag, 1994.

[Pat02] K. Paterson. ID-based Signatures from Pairings on Elliptic Curves. *Available from* *http://eprint.iacr.org*, 2002.

[PS00] D. Pointcheval and J. Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, Vol. 13, No. 3, pp. 361–396, 2000. A preliminary version has appeared in *Advances in Cryptology - Eurocrypt '96*, LNCS, Vol. 1070, pp. 387–398, Springer-Verlag, 1996.

[Sha84] A. Shamir. Identity-base Cryptosystems and Signature Schemes. *Advances in Cryptology - Crypto '84*, LNCS, Vol. 196, pp. 47–53, Springer-Verlag, 1985.

[Sch89] C. Schnorr. Efficient Identification and Signatures for Smart Cards. *Advances in Cryptology - Crypto '89*, LNCS, Vol. 435, pp. 239–252, Springer-Verlag, 1989.

[Tan87] H. Tanaka. A Realization Scheme for the Identity-based Cryptosystem. *Advances in Cryptology - Crypto '87*, LNCS, Vol. 293, pp. 340–349, Springer-Verlag, 1987.

[TI89] S. Tsuji and T. Itoh. An ID-based Cryptosystem based on the Discrete Logarithm Problem. *IEEE Journal of Selected Areas in Communications*, Vol. 7, pp. 467–473, 1989.