Boneh *et al.*'s k-Element Aggregate Extraction Assumption Is Equivalent to The Diffie-Hellman Assumption

Jean-Sebastien Coron and David Naccache

Gemplus Card International
34, rue Guynemer, Issy-les-Moulineaux, F-92447, France
{jean-sebastien.coron,david.naccache}@gemplus.com

Abstract. In Eurocrypt 2003, Boneh *et al.* presented a novel cryptographic primitive called *aggregate signatures*. An aggregate signature scheme is a digital signature that supports aggregation: *i.e.* given k signatures on k distinct messages from k different users it is possible to aggregate all these signatures into a single short signature.

Applying the above concept to verifiably encrypted signatures, *Boneh et al.* introduced a new complexity assumption called *the k-Element Aggregate Extraction Problem*.

In this paper we show that the k-Element Aggregate Extraction Problem is nothing but a Computational Diffie-Hellman Problem in disguise.

Key-words: aggregate signatures, Diffie-Hellman problem, complexity assumption.

1 Introduction

In Eurocrypt 2003, Boneh, Gentry, Lynn and Shacham [2] introduced the concept of *aggregate signatures*. An aggregate signature scheme is a digital signature that supports aggregation: given k signatures on k distinct messages from k different users it is possible to aggregate all these signatures into a single short signature. This useful primitive allows to drastically reduce the size of public-key certificates, thereby saving storage and transmission bandwidth.

Applying the previous construction to verifiably encrypted signatures, *Boneh et al.* introduced in [2] a new complexity assumption called the *k-Element Aggregate Extraction Problem* (hereafter k-EAEP). In this paper we will prove that k-EAEP is equivalent to the Computational Diffie Hellman assumption (CDH).

This paper is structured as follows: section 2 recalls Boneh *et al.*'s setting, section 3 contains [2, 3]'s definition of the k-EAEP and section 4 concludes the paper by proving the equivalence between k-EAEP and CDH.

Key generation	
	Pick random $x \xleftarrow{R} \mathbb{Z}_p$
	Compute $v \leftarrow g_1^x$
Public :	$v \in G_1$
Private :	$x \in \mathbb{Z}_p$
Signature	
	Hash the message $M \in \{0,1\}^*$ into $h \leftarrow h(M) \in G_2$
	Compute the signature $\sigma \leftarrow h^x \in G_2$
Verification of σ (with respect to v and M)	
	Compute $h \leftarrow h(M)$
	Check that $e(g_1, \sigma) = e(v, h)$

Fig. 1. Boneh, Lynn, Shacham Signatures

2 Verifiable Encrypted Signatures via Aggregation

We will adopt [2, 3]'s notations and settings, namely:

- G_1 and G_2 are two multiplicative cyclic groups of prime order p;
- g_1 is a generator of G_1 and g_2 is a generator of G_2;
- ψ is a computable isomorphism from G_1 to G_2 with $\psi(g_1) = g_2$;
- e is a computable bilinear map $e : G_1 \times G_2 \to G_T$ where G_T is multiplicative and of order p. The map e is:
  - Bilinear: for all $u \in G_1, v \in G_2$ and $a, b \in \mathbb{Z}$, $e(u^a, v^b) = e(u, v)^{ab}$
  - Non-degenerate: $e(g_1, g_2) \neq 1$
- $h : \{0,1\}^* \to G_2$ is a hash function.

2.1 Boneh-Lynn-Shacham Signatures

Figure 1 briefly recalls Boneh, Lynn and Shacham's signature scheme [1], upon which the aggregate signatures schemes of [2, 3] are based.

2.2 Aggregate Signatures

Consider now a set of k users using Figure 1's scheme (each user having a different key pair bearing an index i) and signing different messages M_i. Aggregation consists in combining the resulting k signatures $\{\sigma_1, \ldots, \sigma_k\}$ into one aggregate signature σ. This is done by simply computing:

$$\sigma \leftarrow \prod_{i=1}^{k} \sigma_i$$

Aggregate verification is very simple and consists in checking that the M_i are mutually distinct and ensuring that:

$$e(g_1, \sigma) = \prod_{i=1}^{k} e(v_i, h_i) \qquad \text{where} \quad h_i = h(M_i)$$

This holds because:

$$e(g_1, \sigma) = e(g_1, \prod_{i=1}^{k} h_i^{x_i}) = \prod_{i=1}^{k} e(g_1, h_i)^{x_i} = \prod_{i=1}^{k} e(g_1^{x_i}, h_i) = \prod_{i=1}^{k} e(v_i, h_i)$$

2.3 Verifiably Encrypted Signatures via Aggregation

As explained in [2, 3], verifiably encrypted signatures are used in contexts where Alice wants to show Bob that she has signed a message but does not want Bob to possess her signature on that message. Alice can achieve this by encrypting her signature using the public key of a trusted third party (*adjudicator*, hereafter Carol), and send the resulting ciphertext to Bob along with a proof that she has given him a valid encryption of her signature. Bob can verify that Alice has signed the message but cannot deduce any information about her signature. Later in the protocol, Bob can either obtain the signature from Alice or resort to the good offices of Carol who can reveal Alice's signature.

To turn the aggregate signature scheme into a verifiably encrypted signature scheme, [2, 3] proceed as follows:

- Alice wishes to create a verifiably encrypted signature that Bob will verify, Carol being the adjudicator. Alice and Carol's keys are generated as if they were standard signers participating in the aggregate signature protocol described in the previous subsection.
- Alice creates a signature σ on M under her public key. She then forges a signature σ' on some random message M' under Carol's public key (we refer the reader to [2, 3] for more details on the manner in which this existential forgery is produced). She then combines σ and σ' obtaining the aggregate ω. The verifiably encrypted signature is $\{\omega, M'\}$.
- Bob validates Alice's verifiably encrypted signature $\{\omega, M'\}$ on M by checking that ω is a valid aggregate signature by Alice on M and by Carol on M'.
- Carol adjudicates, given a verifiably encrypted signature $\{\omega, M'\}$ on M by Alice, by computing the signature σ' on M' and removing σ' from the aggregate thereby revealing Alice's signature σ.

3 The k-Element Aggregate Extraction Problem

As is clear, the security of Boneh *et al.*'s verifiable encrypted signature scheme depends on the assumption that given an aggregate signature of k signatures (here $k = 2$) it is difficult to extract from it the individual signatures (namely: Alice's signature on M). This is formally proved in theorem 3 of [2, 3].

Considering the bilinear aggregate signature scheme on G_1 and G_2, Boneh *et al.* assume that it is difficult to recover the individual signatures σ_i given the aggregate σ, the public-keys and the message digests. Actually, [2, 3] assume that it is difficult to recover any aggregate σ' of any proper set of the signatures and term this the k-Element Aggregate Extraction Problem (hereafter k-EAEP).

More formally, this assumption is defined in [2, 3] as follows: Let G_1 and G_2 be two multiplicative cyclic groups of prime order p, with respective generators g_1 and g_2, a computable isomorphism $\psi : G_1 \to G_2$ such that $g_2 = \psi(g_1)$, and a computable bilinear map $e : G_1 \times G_2 \to G_T$.

Consider a k-user aggregate in this setting. Each user has a private key $x_i \in \mathbb{Z}_p$ and a public key $v_i = g_1^{x_i} \in G_1$. Each user selects a distinct message $M_i \in \{0,1\}^*$ whose digest is $h_i \in G_2$ and creates a signature $\sigma_i = h_i^{x_i} \in G_2$. Finally, the signatures are aggregated yielding:

$$\sigma = \prod_{i=1}^{k} \sigma_i \in G_2$$

Let I be the set $\{1, \ldots, k\}$. Each public-key v_i can be expressed as $g_1^{x_i}$, each digest h_i as $g_2^{y_i}$, each signature σ_i as $g_2^{x_i y_i}$ and the aggregate signature σ as g_2^z where:

$$z = \sum_{i \in I} x_i y_i$$

Definition 1 (k-EAEP). *The k-Element Aggregate Extraction Problem is the following: given the group elements $g_1^{x_1}, \ldots, g_1^{x_k}$, $g_2^{y_1}, \ldots, g_2^{y_k}$ and $g_2^{\sum_{i \in I} x_i \cdot y_i}$, output (σ', I') such that $I' \subsetneq I$ and $\sigma' = g_2^{\sum_{i \in I'} x_i \cdot y_i}$.*

The advantage of an algorithm $\mathcal{E}$ in solving the k-EAEP is defined as:

$$\mathsf{Adv}\ k\text{-}\mathsf{Extr}_{\mathcal{E}} \stackrel{\text{def}}{=} \Pr \begin{bmatrix} (I' \subsetneq I) \wedge (\sigma' = g_2^{\sum_{i \in I'} x_i y_i}) \ : \\ x_1, \ldots, x_k, y_1, \ldots, y_k \stackrel{R}{\leftarrow} \mathbb{Z}_p, \sigma \leftarrow g_2^{\sum_{i \in I} x_i y_i}, \\ (\sigma', I') \stackrel{R}{\leftarrow} \mathcal{E}(g_1^{x_1}, \ldots, g_1^{x_k}, g_2^{y_1}, \ldots, g_2^{y_k}, \sigma) \end{bmatrix}$$

wherein the probability is taken over the choices of all x_i and y_i and the coin tosses of $\mathcal{E}$.

In the following, we define the hardness of the k-EAEP. For simplicity, we use the asymptotic setting instead of the concrete setting of [2].

Definition 2. *The k-Element Aggregate Extraction Problem is said to be hard if no probabilistic polynomial-time algorithm can solve it with non-negligible advantage.*

[2, 3] is particularly concerned with the case $k = 2$ where the aggregate extraction problem boils down to the following:

Definition 3 (2-EAEP). *Given g_1^a, g_1^b, g_2^u, g_2^v and g_2^{au+bv}, output g_2^{au}.*

We refer the reader to [3] for more details on the manner in which this assumption is used in proving the security of the verifiable encrypted signature scheme.

4 *k*-EAEP is equivalent to the Computational co-Diffie-Hellman problem

The Computational co-Diffie-Hellman problem (hereafter co-CDH) is a natural generalization to two groups G_1 and G_2 of the standard Computational Diffie-Hellman problem; it is defined as follows [2]:

Definition 4 (co-CDH). *Given $g_1, g_1^a \in G_1$ and $h \in G_2$, output $h^a \in G_2$.*

The advantage of an algorithm $\mathcal{A}$ in solving co-CDH in groups G_1 and G_2 is:

$$\mathsf{Adv\ co\text{-}CDH}_{\mathcal{A}} \stackrel{\text{def}}{=} \Pr\left[\mathcal{A}(g_1, g_1^a, h) = h^a : a \stackrel{R}{\leftarrow} \mathbb{Z}_p, h \stackrel{R}{\leftarrow} G_2\right]$$

The probability is taken over the choice of a, h and $\mathcal{A}$'s coin tosses. Note that when $G_1 = G_2$, this problem reduces to the standard CDH problem.

Definition 5. *The Computational co-Diffie-Hellman problem in groups G_1 and G_2 is said to be hard if no probabilistic polynomial-time algorithm can solve it with non-negligible advantage.*

The following theorem shows that the k-Element Aggregate Extraction Problem is equivalent to the Computational co-Diffie-Hellman problem.

Theorem 1. *The k-Element Aggregate Extraction Problem is hard if and only if the Computational co-Diffie-Hellman problem is hard.*

Proof. It is straightforward to show that an algorithm $\mathcal{A}$ solving co-CDH can be used to solve the k-EAEP. Namely, given the instance $g_1^{x_1}, \ldots, g_1^{x_k}$, $g_2^{y_1}, \ldots, g_2^{y_k}$ and $g_2^{\sum_{i \in I} x_i \cdot y_i}$, using $\mathcal{A}$ we obtain $\sigma' = g_2^{x_1 y_1}$ from $g_1, g_1^{x_1}, g_2^{y_1}$. This gives $(\{1\}, \sigma')$ as a solution to the k-EAEP.

For the converse, we start with $k = 2$, *i.e.* an algorithm solving the 2-EAEP and show how to generalize the method to arbitrary k. Letting g_1, g_1^a, g_2^u be a given instance of co-CDH, we must compute $g_2^{a \cdot u}$ using an algorithm $\mathcal{A}$ solving the 2-EAEP. We generate $x \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and $y \stackrel{R}{\leftarrow} \mathbb{Z}_p$; one can see that:

$$(g_1^a, g_1^{a+x}, g_2^{-u}, g_2^{u+y}, g_2^{a \cdot y + u \cdot x + x \cdot y})$$

is a valid random instance of the 2-EAEP. The instance is valid because:

$$-a \cdot u + (a + x) \cdot (u + y) = a \cdot y + u \cdot x + x \cdot y$$

The instance is a random one because g_1^{a+x} and g_2^{u+y} are uniformly distributed in G_1 and G_2. Moreover, the instance can be computed directly from g_2^u and $g_2^a = \psi(g_1^a)$. Therefore, given as input this instance, the algorithm $\mathcal{A}$ outputs $g_2^{-a \cdot u}$, from which we compute $g_2^{a \cdot u}$ and solve the co-CDH problem.

More generally, for $k > 2$, we generate $x_2, \ldots, x_k, y_2, \ldots, y_k \xleftarrow{R} \mathbb{Z}_p$; then we generate the following instance of the k-EAEP:

$$(g_1^a, g_1^{a+x_2}, \ldots, g_1^{a+x_k}, g_2^{-(k-1)u}, g_2^{u+y_2}, \ldots, g_2^{u+y_k}, g_2^z)$$

where

$$z = \sum_{i=2}^{k} a \cdot y_i + x_i \cdot (u + y_i)$$

As previously, this is a valid random instance of the k-EAEP, which can be computed from g_2^u and $g_2^a = \psi(g_1^a)$. Therefore, given this instance as input, an algorithm $\mathcal{A}$ solving k-EAEP outputs (I', σ'). We assume that $1 \in I'$, otherwise we can take $I'' \leftarrow I \setminus I'$ and $\sigma'' \leftarrow g_2^z/\sigma'$. Letting $\sigma' = g_2^{z'}$ and $k' = |I'| < k$, we have:

$$z' = -(k-1) \cdot a \cdot u + \sum_{i \in I', i > 1} (a + x_i)(u + y_i)$$

$$z' = a \cdot u \cdot (k' - k) + \sum_{i \in I', i > 1} a \cdot y_i + x_i \cdot (u + y_i)$$

Therefore we can compute:

$$g_2^{a \cdot u} = \left(\sigma' \cdot \prod_{i \in I', i > 1} (g_2^a)^{-y_i} (g_2^u)^{-x_i} g_2^{-x_i y_i} \right)^{\frac{1}{k'-k}}$$

which is the solution of the co-CDH instance.

Therefore, given a polynomial time probabilistic algorithm solving the k-EAEP with non-negligible advantage, we obtain a polynomial time probabilistic algorithm solving co-CDH with non-negligible advantage, and conversely, with a tight reduction in both directions. □

5 Conclusion

In this paper we showed that the k-element Aggregate Extraction Problem introduced by Boneh, Gentry, Lynn and Shacham in [2, 3] is equivalent to the Computational Diffie Hellman Problem.

By shedding light on the connection between Boneh *et al.*'s verifiable encrypted signature scheme and the well-researched Computational Diffie-Hellman Problem, we show that [2, 3] features, not only attractive computational requirements and short signature size, but also strong security assurances.

References

1. D. Boneh, B. Lynn and H. Shacham, *Short Signatures From the Weil Pairing*, Proceedings of ASIACRYPT' 2001, Lecture Notes in Computer Science vol. 2248, Springer-Verlag, pp. 514-532, 2001.
2. D. Boneh, C. Gentry, B. Lynn and H. Shacham, *Aggregate and Verifiably Encrypted Signatures from Bilinear Maps*, Advances in Cryptology - EUROCRYPT' 2003 Proceedings, Lecture Notes in Computer Science vol. 2656, E. Biham ed., Springer-Verlag, pp. 416-432, 2003.
3. D. Boneh, C. Gentry, B. Lynn and H. Shacham, *Aggregate and Verifiably Encrypted Signatures from Bilinear Maps*, Cryptology ePrint Archive, Report 2002/175, 2002, `http://eprint.iacr.org/`.