

Attribute-Based Broadcast Encryption Scheme Made Efficient

David Lubicz
Thomas Sirvent



1 Context

- Broadcast encryption
- Efficiency of standard schemes
- Attributes in broadcast encryption
- A previous construction

2 Our Scheme

- Principle of the scheme
- In practice ?
- Full description of the scheme
- Efficiency and security

1 Context

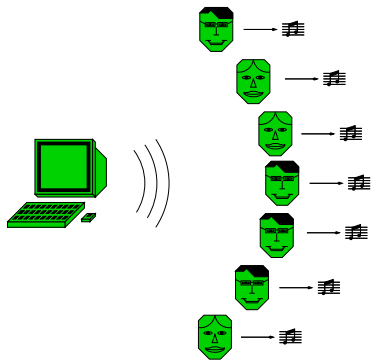
- Broadcast encryption
- Efficiency of standard schemes
- Attributes in broadcast encryption
- A previous construction

2 Our Scheme

- Principle of the scheme
- In practice ?
- Full description of the scheme
- Efficiency and security

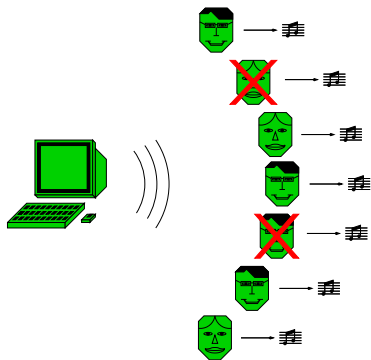
Broadcast

A **broadcaster** intends to send securely and efficiently the **same message** to a **large number of receivers**:



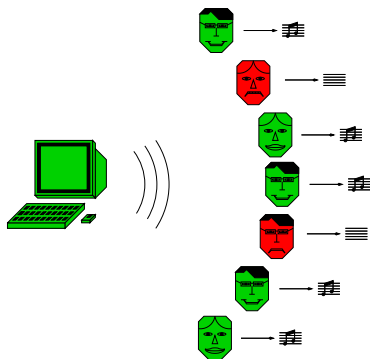
Revocation

In the case of a revocation, some users are **removed** from the set of receivers (for example when their decryption keys are compromised):



Permanent revocation

In the case of a **permanent revocation**, the decryption keys are updated. The revoked users are not able anymore to obtain the messages sent by the broadcaster:



Temporary revocation

When permanent revocations are used ([stateful schemes](#)),

- receivers must remain [online](#),
- receivers must [store](#) and use new decryption keys.

To avoid these limitations, it is possible to use [stateless schemes](#), where revocations are [temporary](#):

- in the encryption process, the broadcaster chooses the set of receivers,
- only members of this set may decrypt the message.

Efficiency of standard schemes

In stateful schemes (like LKH),

- a common decryption key is known by all receivers,
- a specific structure allows **join** and **revoke** operations.
→ Join and revoke operations require large bandwidth.

▶ Good schemes for sets of receivers with **rare modifications**.

In stateless schemes (like CS or SD),

- join and revoke operations do not exist,
- a specific structure allows **encryption**.
→ Ciphertexts require large bandwidth.

▶ Good schemes for a **small number** of revoked users.

Efficiency of standard schemes

In stateful schemes (like LKH),

- a common decryption key is known by all receivers,
- a specific structure allows **join** and **revoke** operations.
→ Join and revoke operations require large bandwidth.

▶ Good schemes for sets of receivers with **rare modifications**.

In stateless schemes (like CS or SD),

- join and revoke operations do not exist,
- a specific structure allows **encryption**.
→ Ciphertexts require large bandwidth.

▶ Good schemes for a **small number** of revoked users.

Attributes and users

In practical applications of broadcast, users have **attributes** that can be used to describe efficiently the set of receivers.

Id	Name	Subscription	Expiry Date	Location
1	Alice	Movies	Jan 2009	Europe
2	Bob	News / Sports	Aug 2009	Africa
3	Charlie	Entertainment	May 2008	Asia
4	Dave	News / Movies	Jun 2009	Asia
5	Eve	Entertainment / Sports	Jan 2008	Africa

Is it possible to send efficiently a movie in June 2008 ?

- Evolution of the set of receivers : **fast**
- Number of revoked users: **large**
- Number of users: **large**
- But a **specific structure** !

Build a broadcast encryption scheme such that:

- when the set of receivers is defined by attributes, the efficiency **depends only on the number of attributes used**,
- any set of receivers has a “reasonable” efficiency ?

A previous construction

An attribute-based broadcast scheme comes from:

- [SW05]: use of a single attribute,
- [GPSW06]: use of several attributes,
- [BSW07]: policy defined by the ciphertext,
- [OSW07]: non-monotonic policy (use of NOT).

The policy is defined by:

- **threshold functions** (including AND and OR functions),
- **NOT functions**.

Efficiency:

- ++ Ciphertexts have a **linear size in the number of attributes**,
- Decryption requires a **linear number of pairing computations**.

A previous construction

An attribute-based broadcast scheme comes from:

- [SW05]: use of a single attribute,
- [GPSW06]: use of several attributes,
- [BSW07]: policy defined by the ciphertext,
- [OSW07]: non-monotonic policy (use of NOT).

The policy is defined by:

- **threshold functions** (including AND and OR functions),
- **NOT functions**.

Efficiency:

++ Ciphertexts have a **linear size in the number of attributes**,

-- Decryption requires a **linear number of pairing computations**.

A previous construction

An attribute-based broadcast scheme comes from:

- [SW05]: use of a single attribute,
- [GPSW06]: use of several attributes,
- [BSW07]: policy defined by the ciphertext,
- [OSW07]: non-monotonic policy (use of NOT).

The policy is defined by:

- **threshold functions** (including AND and OR functions),
- **NOT functions**.

Efficiency:

- ++ Ciphertexts have a **linear size in the number of attributes**,
- Decryption requires a **linear number of pairing computations**.

1 Context

- Broadcast encryption
- Efficiency of standard schemes
- Attributes in broadcast encryption
- A previous construction

2 Our Scheme

- Principle of the scheme
- In practice ?
- Full description of the scheme
- Efficiency and security

Principle of the scheme (1)

Each attribute is associated with an element $\mu_i \in \mathbb{Z}/p\mathbb{Z}$.

User u :

- $\Omega(u)$ is its set of attributes,
- dk_u is its decryption key.

$$dk_u \longleftrightarrow \prod_{\mu \in \Omega(u)} (X - \mu).$$

Encryption: a header hdr and a key K are built from

- Ω^R : the set of revoked attributes,
- Ω^N : the set of needed attributes.

$$K \longleftrightarrow \prod_{\mu \in \Omega^N} (X - \mu) \qquad \text{hdr} \longleftrightarrow \prod_{\mu \in \Omega^R \cup \Omega^N} (X - \mu).$$

Principle of the scheme (1)

Each attribute is associated with an element $\mu_i \in \mathbb{Z}/p\mathbb{Z}$.

User u :

- $\Omega(u)$ is its set of attributes,
- dk_u is its decryption key.

$$dk_u \longleftrightarrow \prod_{\mu \in \Omega(u)} (X - \mu).$$

Encryption: a header hdr and a key K are built from

- Ω^R : the set of revoked attributes,
- Ω^N : the set of needed attributes.

$$K \longleftrightarrow \prod_{\mu \in \Omega^N} (X - \mu) \qquad \text{hdr} \longleftrightarrow \prod_{\mu \in \Omega^R \cup \Omega^N} (X - \mu).$$

Principle of the scheme (1)

Each attribute is associated with an element $\mu_i \in \mathbb{Z}/p\mathbb{Z}$.

User u :

- $\Omega(u)$ is its set of attributes,
- dk_u is its decryption key.

$$dk_u \longleftrightarrow \prod_{\mu \in \Omega(u)} (X - \mu).$$

Encryption: a header hdr and a key K are built from

- Ω^R : the set of revoked attributes,
- Ω^N : the set of needed attributes.

$$K \longleftrightarrow \prod_{\mu \in \Omega^N} (X - \mu) \qquad \text{hdr} \longleftrightarrow \prod_{\mu \in \Omega^R \cup \Omega^N} (X - \mu).$$

Principle of the scheme (2)

Decryption: compute the **GCD** (greatest common divisor) of the decryption key dk_u and the header hdr to obtain the key K .

$$\text{GCD}(dk_u, \text{hdr}) \iff \prod_{\mu \in \Omega(u) \cap (\Omega^R \cup \Omega^N)} (X - \mu).$$

Is it accurate ?

$$\begin{aligned} & \text{GCD}(dk_u, \text{hdr}) = K \\ \iff & \Omega(u) \cap (\Omega^R \cup \Omega^N) = \Omega^N \\ \iff & \Omega(u) \cap \Omega^R = \emptyset \text{ and } \Omega^N \subset \Omega(u). \end{aligned}$$

Principle of the scheme (2)

Decryption: compute the **GCD** (greatest common divisor) of the decryption key dk_u and the header hdr to obtain the key K .

$$\text{GCD}(dk_u, \text{hdr}) \iff \prod_{\mu \in \Omega(u) \cap (\Omega^R \cup \Omega^N)} (X - \mu).$$

Is it accurate ?

$$\begin{aligned} & \text{GCD}(dk_u, \text{hdr}) = K \\ \iff & \quad \Omega(u) \cap (\Omega^R \cup \Omega^N) = \Omega^N \\ \iff & \quad \Omega(u) \cap \Omega^R = \emptyset \quad \text{and} \quad \Omega^N \subset \Omega(u). \end{aligned}$$

In Practice ?

In practical applications, it is not possible to use polynomials.

For all polynomial P , we use $P(\alpha) g_1$ instead of P , where:

- g_1 is a public generator of a group G_1 in which the DLP is hard,
- α is a secret value.

The “GCD” is computed using extended Euclide’s algorithm.

We need a **non-degenerate pairing**, i.e. a map $e : G_1 \times G_1 \rightarrow G_2$ where:

- (G_1, g_1) and (G_2, g_2) are two cyclic groups of same prime order p ,
- $e(g_1, g_1) = g_2$,
- e is bilinear.

The group laws in G_1 and G_2 are noted additively: $e(a g_1, b g_1) = a b g_2$.

In Practice ?

In practical applications, it is not possible to use polynomials.

For all polynomial P , we use $P(\alpha) g_1$ instead of P , where:

- g_1 is a public generator of a group G_1 in which the DLP is hard,
- α is a secret value.

The “GCD” is computed using extended Euclide’s algorithm.

We need a **non-degenerate pairing**, i.e. a map $e : G_1 \times G_1 \rightarrow G_2$ where:

- (G_1, g_1) and (G_2, g_2) are two cyclic groups of same prime order p ,
- $e(g_1, g_1) = g_2$,
- e is bilinear.

The group laws in G_1 and G_2 are noted additively: $e(a g_1, b g_1) = a b g_2$.

In Practice ?

In practical applications, it is not possible to use polynomials.

For all polynomial P , we use $P(\alpha) g_1$ instead of P , where:

- g_1 is a public generator of a group G_1 in which the DLP is hard,
- α is a secret value.

The “GCD” is computed using extended Euclide’s algorithm.

We need a **non-degenerate pairing**, i.e. a map $e : G_1 \times G_1 \rightarrow G_2$ where:

- (G_1, g_1) and (G_2, g_2) are two cyclic groups of same prime order p ,
- $e(g_1, g_1) = g_2$,
- e is bilinear.

The group laws in G_1 and G_2 are noted additively: $e(a g_1, b g_1) = a b g_2$.

Protection against some attacks

Attack 1 : Attributes in headers can be modified.

(Linear combinations of different headers)

↔ **Randomized headers** (using z).

Attack 2 : Attributes in decryption keys can be modified.

(Linear combinations of different decryption keys)

↔ **Randomized decryption keys** (using s_U).

Attack 3 : Other computations (than “GCD”) can be performed.

(Pairing computations on some specific pairs of group elements)

↔ **New parameters** (γ and δ).

Protection against some attacks

Attack 1 : Attributes in headers can be modified.

(Linear combinations of different headers)

↔ **Randomized headers** (using z).

Attack 2 : Attributes in decryption keys can be modified.

(Linear combinations of different decryption keys)

↔ **Randomized decryption keys** (using s_u).

Attack 3 : Other computations (than “GCD”) can be performed.

(Pairing computations on some specific pairs of group elements)

↔ **New parameters** (γ and δ).

Protection against some attacks

Attack 1 : Attributes in headers can be modified.

(Linear combinations of different headers)

↔ **Randomized headers** (using z).

Attack 2 : Attributes in decryption keys can be modified.

(Linear combinations of different decryption keys)

↔ **Randomized decryption keys** (using s_u).

Attack 3 : Other computations (than “GCD”) can be performed.

(Pairing computations on some specific pairs of group elements)

↔ **New parameters** (γ and δ).

Full scheme - key generation

- We randomly choose a secret 4-uple $(\alpha, \beta, \gamma, \delta) \in ((\mathbb{Z}/p\mathbb{Z})^*)^4$,
- Each user u is associated with a secret $s_u \in (\mathbb{Z}/p\mathbb{Z})$,
- Each attribute is associated with a public $\mu_i \in (\mathbb{Z}/p\mathbb{Z}) \setminus \{\alpha\}$.

$$\text{EK} = \left(g_1, \beta \gamma \delta g_1, (\mu_i, \alpha^i g_1, \alpha^i \gamma g_1, \alpha^i \delta g_1)_{0 \leq i < l} \right).$$

$$\text{dk}_u = \left(\Omega(u), (\beta + s_u) \delta g_1, \gamma s_u \Pi(u) g_1, (\alpha^i \gamma \delta s_u g_1)_{0 \leq i < l(u)} \right),$$

$$\text{where } \begin{cases} \Omega(u) = \{\mu_i \in (\mathbb{Z}/p\mathbb{Z}) / \mu_i \text{ attribute of } u\}, \\ l(u) = |\Omega(u)| \text{ is the number of attributes of } u, \\ \Pi(u) = \prod_{\mu \in \Omega(u)} (\alpha - \mu). \end{cases}$$

Full scheme - encryption

- Let Ω^N be the set of needed attributes.
- Soit $\Omega^R \neq \emptyset$ be the set of revoked attributes.
- A user u is valid for these sets if: $\Omega^N \subset \Omega(u)$ and $\Omega^R \cap \Omega(u) = \emptyset$.

The encryption for these sets (Ω^N, Ω^R) gives :

$$\text{hdr} = \left(\Omega^N, \Omega^R, z \prod^{NR} g_1, \gamma z \prod^N g_1, (\alpha^i \delta z g_1)_{0 \leq i < I^R} \right),$$

$$K = \beta \gamma \delta z \prod^N g_2.$$

where

$$\begin{cases} z \text{ is randomly chosen in } (\mathbb{Z}/p\mathbb{Z})^*, \\ I^R = |\Omega^R|, \\ \prod^N = \prod_{\mu \in \Omega^N} (\alpha - \mu), \\ \prod^{NR} = \prod^N \prod_{\mu \in \Omega^R} (\alpha - \mu). \end{cases}$$

Full scheme - decryption

The decryption is based on a decryption key dk_u and a header hdr :

$$\begin{cases} dk_u = (\Omega(u), dk_1, dk_2, dk_{3,0}, \dots, dk_{3,l(u)-1}), \\ hdr = (\Omega^N, \Omega^R, hdr_1, hdr_2, hdr_{3,0}, \dots, hdr_{3,l^R-1}). \end{cases}$$

If u is a valid receiver, extended Euclide's algorithm gives two polynomials

$V(X) = \sum_{i=0}^{l(u)-1} v_i X^i$ and $W(X) = \sum_{i=0}^{l^R-1} w_i X^i$ such that:

$$V(X) \prod_{\mu \in (\Omega^N \cup \Omega^R)} (X - \mu) + W(X) \prod_{\mu \in \Omega(u)} (X - \mu) = \prod_{\mu \in \Omega^N} (X - \mu).$$

The key is obtained by:

$$e(dk_1, hdr_2) - e\left(\sum_{i=0}^{l(u)-1} v_i dk_{3,i}, hdr_1\right) - e\left(dk_2, \sum_{i=0}^{l^R-1} w_i hdr_{3,i}\right).$$

$$V(\alpha)\Pi^{NR} + W(\alpha)\Pi(u) = \Pi^N.$$

$$V(\alpha) = \sum_{i=0}^{l(u)-1} v_i \alpha^i$$

$$W(\alpha) = \sum_{i=0}^{l^R-1} w_i \alpha^i$$

$$dk_1 = (\beta + s_u) \delta g_1$$

$$hdr_1 = z \Pi^{NR} g_1$$

$$dk_2 = \gamma s_u \Pi(u) g_1$$

$$hdr_2 = \gamma z \Pi^N g_1$$

$$dk_{3,i} = \alpha^i \gamma \delta s_u g_1$$

$$hdr_{3,i} = \alpha^i \delta z g_1$$

$$e(dk_1, hdr_2) = e\left(\sum_{i=0}^{l(u)-1} v_i dk_{3,i}, hdr_1\right) = e\left(dk_2, \sum_{i=0}^{l^R-1} w_i hdr_{3,i}\right)$$

$$\hookrightarrow K = \beta \gamma \delta z \Pi^N g_2.$$

Efficiency of this scheme

Size of ciphertexts : linear in $|\Omega^N| + |\Omega^R|$.

Computations :

- Decryption : 3 pairing computations,
- Encryption : 1 pairing computation (none if we extend EK).

Size of keys :

- EK linear in l ,
- dk_u linear in $l(u)$.

Efficiency of this scheme

Size of ciphertexts : linear in $|\Omega^N| + |\Omega^R|$.

Computations :

- Decryption : 3 pairing computations,
- Encryption : 1 pairing computation (none if we extend EK).

Size of keys :

- EK linear in l ,
- dk_U linear in $l(u)$.

Efficiency of this scheme

Size of ciphertexts : linear in $|\Omega^N| + |\Omega^R|$.

Computations :

- Decryption : 3 pairing computations,
- Encryption : 1 pairing computation (none if we extend EK).

Size of keys :

- EK linear in l ,
- dk_u linear in $l(u)$.

Security and other features

The security has been proved in the **generic model of groups with pairings**.

Some features:

- EK can be **strongly reduced** in some cases,
- **new users** can join easily,
- for any set of receivers, **at least as efficient as the SD scheme**.

Threshold functions are however not available in this scheme.

The security has been proved in the **generic model of groups with pairings**.

Some features:

- EK can be **strongly reduced** in some cases,
- **new users** can join easily,
- for any set of receivers, **at least as efficient as the SD scheme**.

Threshold functions are however not available in this scheme.

The security has been proved in the **generic model of groups with pairings**.

Some features:

- EK can be **strongly reduced** in some cases,
- **new users** can join easily,
- for any set of receivers, **at least as efficient as the SD scheme**.

Threshold functions are however not available in this scheme.

▶ Thank you for your attention! ◀