Permanent Revocation Systems

by

Dan Brownstein, Niv Gilboa and **Shlomi Dolev**

Technical Report #17-02

May 25, 2017

The Lynne and William Frankel Center for Computer Science,

Department of Computer Science, Ben-Gurion University,

Be'er Sheva, Israel.

Permanent Revocation Systems

Dan Brownstein[1], Niv Gilboa[2], and Shlomi Dolev[1]

[1] Dept. of Computer Science, Ben-Gurion University of the Negev, Israel, dolev,danbr@cs.bgu.ac.il,
[2] Dept. of Communication Systems Engineering, Ben-Gurion University of the Negev, Israel gilboan@bgu.ac.il

Abstract. We present a public-key permanent revocation system. We define a stateful scheme in which an authorized entity can perform permanent revocations of groups of users by sending revocation messages which are used for state updating of non-revoked users. The complexity of a permanent revocation of r users is $O(r)$ operations for creating the revocation message and updating the state of a user and the size of a revocation message is $O(r)$ group elements. The ciphertext and keys in our scheme consist of constant number of groups elements. Our scheme is based on 3-linear maps and is secured in the generic group model.

1 Introduction

In broadcast encryption (BE) a single broadcaster can broadcast encrypted messages to a group of users (sometimes called receivers) such that any member of an authorized subset of users can decrypt the message. We denote the universe n users as $\mathcal{U}$, the subset of authorized users as S and the subset of non-authorized users as $R = \mathcal{U}/S$ where $|R| = r$. Since Fiat and Naor [FN93] initiated a formal study of BE, quite a few schemes were proposed each with different characteristics regarding functionality and security.

A main challenge in constructing BE schemes is achieving *Collusion resistance*- ensuring that even when unauthorized users collude, it is infeasible to decrypt the message. The first line of works used a combinatorial approach which yields schemes that are secure against collusions of up to t users, for some parameter t (also known as threshold collusion resistance) [FN93,GSW00]. Naor et al. [NNL01] presented a tree based scheme which is secured against any arbitrary set of revoked users with ciphertext size of $O(r)$ elements and private keys of size $O(\log^2 n)$. Their scheme was later improved by [HS02,GST04] to achieve a ciphertext size of $O(\log n)$. In these schemes, the same set of keys is used for both encryption and decryption meaning that only a trusted user can encrypt messages. Dodis et al. [DF02] show how to transform [NNL01,HS02] into public-key schemes where untrusted users are allowed to broadcast information. Several constructions [KD98,NP10,YJCK04] achieve schemes without a bound on the number of revoked users by using secret sharing.

The scheme of Gentry et al. [BGW05] has ciphertext and private key of constant size but the size of the public key is linear in n. Delerablée et al.

[DPP07] presents 3 constructions. The public key constructions in [DPP07] also suffer from public parameters with size linear in n. Their third construction achieves public parameters and ciphertexts of constant sizes, however, it is not a public key scheme.

The first adaptively secure scheme is due to Gentry and Waters [GW09]. The size of the keys in this scheme grows linearly with n. Lewko et al. [LSW10] employs the dual system encryption [Wat09] to achieve an adaptively secure revocation system that has ciphertext size overhead $O(r)$ and the size of public and private keys is constant. Boneh and Silverberg [BS03] construct a scheme with constant keys and ciphertexts from n-linear maps.

In temporary revocation, the broadcaster encrypts the data with respect to a set S of authorized users or a set R of revoked users. The overhead of the revocation shows in the computation complexity of the encryption and decryption algorithms and the size of the ciphertext. On the other hand, in permanently revocation, a revoked user should not be able to decrypt any transmissions following his revocation, regardless of the designated recipients of the message. Because permanent revocation enforces redistribution of users' keys, it is sometimes called re-keying. Wallner et al. [Age99] and Wong et al. [WGL00] consider tree-based re-keying private key schemes which after improvement of Canetti et al. [CGI+99] achieve private keys of size $\log n + 1$, encryption key of size $2n - 1$ and revocation communication of size $\log n$. Later, Canetti at al. [CMN99] generalized these schemes and showed that for user with private keys of size $b + 1$, the revocation message size is $O(bn^{1/b}b)$. The encryption key size multiplied by the revocation message size is $O(n)$. A specific parameterization of their scheme yields revocation messages of size $O(\log n)$, private keys size $O(\log n)$ and encryption key of size $O(n/\log n)$.

A Stateless receiver in BE is a receiver that is not capable of recording the past history of transmissions and changing its state accordingly. Instead, its operation must be based on the current transmission and its initial conguration. Stateful receivers are receivers that are not stateless. Stateless schemes [NNL01,DF02,BGW05] are preferable in the sense that receivers do not have to be continuously on-line to correctly decrypt a transmission. Traditionally, BE schemes were one-way and therefore schemes had to be either stateless or dependent on repeated transmissions of critical data. However, broadcast models in which the receivers can open a two-way channel to the broadcaster are becoming more prevalent, e.g., IPTV and Over-The-Top broadcasting.

Permanent revocation necessarily implies modification of the keys. Stateful schemes open new avenues to achieve permanent revocation efficiently by basing decryption on a revolving state and not enabling revoked users to correctly update a state as proposed by Dolev et al. [DGK12]. In their scheme, they use a secret counter variable $CTR \in \mathbb{Z}_p$ in the master secret key. Then, each user i is assigned a state E_i which is updated after every successful decryption. Their encryption algorithm uses CTR in order to mask plaintexts such that only users with up to date state can successfully decrypt. With this technique they were able to transfer temporary revocation schemes into full revocation schemes by

utilizing states and with almost no overhead. A shortcome of this technique is that a secret information (CTR) is needed for encryption, hence, it is not a public key scheme. Also, their scheme relies on a model in which revoked users have no access to decrypted ciphertexts. That is, it is not secured against collusions in which authorized user shares a plaintext with an unauthorized one.

Contribution. We formally define permanent revocation systems and their security definition. We then propose a public stateful scheme which supports permanent revocation based on 3-linear mapping and is secure in the generic group model. Our scheme enjoys small (constant size) keys and efficient procedures $O(1)$ operations for encryption / decryption and $O(r)$ operations for revocation / state update where r is the number of revoked users.

Organization In Section 2 we give the definition of a permanent revocation scheme followed by the security definition. We then provide the definition of 3-linear maps. In section 3 we provide the construction of our scheme which is followed by security proof in section 4.

2 Preliminaries

2.1 Permanent Revocation System

Permanent revocation encryption scheme consists of six algorithms: Setup, KeyGen, Revoke, UpdateState, Encrypt and Decrypt.

$\mathsf{Setup}(\lambda)$. The setup algorithm takes as input the security parameter λ and outputs public parameters PP and a master secret key MSK.

$\mathsf{KeyGen}(MSK, ID)$. The key generation algorithm takes as input the master secret key MSK, an identity ID and outputs a secret key SK_{ID}. Each key has a boolean property $SK_{ID}.\mathsf{revoked}$ which is set by default to false.

$\mathsf{Revoke}(S, PP, MSK)$. The revocation algorithm takes as input the master secret key MSK, the public parameters PP and a set S of identities to revoke, updates the master secret and public parameters and outputs a state update message SUM which is then broadcast to all users.

UpdateState(SK_{ID}, SUM, ID). The state update algorithm takes as input the user's secret key SK_{ID}, the state update message SUM and the user's identity ID. The algorithm updates the user's secret key. If ID is in the set of revoked users that corresponds to SUM, the algorithm sets $SK_{ID}.\mathsf{revoked} = \mathsf{true}$.

Encrypt(PP, M). The encryption algorithm takes as input the public parameters PP and a message M. The algorithm outputs a ciphertext CT.

Decrypt(SK_{ID}, CT, PP). The decryption algorithm takes as input a secret key, SK_{ID}, a ciphertext CT and the public parameters PP. If $SK_{ID}.\mathsf{revoked} = \mathsf{true}$,

the algorithm outputs $\perp$. Otherwise it outputs the message M associated with CT.

We require that our system be correct, namely, it must satisfy the following property:

Correctness. For all messages M, sets of identities $S_1 \ldots, S_n$ and all $ID \notin \bigcup_{i=1}^{n} S_i$, if $(PP, MSK) \leftarrow Setup(\lambda)$, $SK_{ID} \leftarrow KeyGen(MSK, ID)$, $SUM_1 \leftarrow Revoke(S_1, PP, MSK)$, $\ldots$, $SUM_n \leftarrow Revoke(S_n, PP, MSK)$ and the state of user ID was updated with calls to $UpdateState(SK_{ID}, SUM_1, ID)$, $\ldots$, $UpdateState(SK_{ID}, SUM_n, ID)$ then if $CT \leftarrow Encrypt(PP, M)$ then $Decrypt(SK_{ID}, CT, PP) = M$.

Security Definition Adaptive security of a permanent revocation scheme is defined as a game between a challenger and an attack algorithm $\mathcal{A}$ with the following phases:

Setup. The challenger runs the *Setup* algorithm with security parameter λ to obtain the PP and the MSK. It maintains a set Q initialized to the empty set and then sends the PP to $\mathcal{A}$.

Key Query and Revocation. In this phase $\mathcal{A}$ adaptively issues secret key and revocation queries. For every private key query for identity ID, the challenger adds ID to Q, runs the *KeyGen* algorithm and sends $\mathcal{A}$ the corresponding secret key. For every revocation query for set S of Identities, the challenger removes each identity $ID \in S$ from Q, runs the *Revoke* algorithm, updates the MSK and the PP and sends $\mathcal{A}$ the new PP and the corresponding state update messages SUM.

Challenge. $\mathcal{A}$ sends the challenger two messages M_1, M_2. In case $Q \neq \emptyset$ the challenger sends $\perp$ to $\mathcal{A}$ and aborts. Otherwise, the challenger flips a random coin $b \in \{0, 1\}$, runs the *Encrypt* algorithm to obtain an encryption of M_b and sends it to $\mathcal{A}$.

Guess. $\mathcal{A}$ outputs a guess $b' \in \{0, 1\}$ and wins if $b = b'$.

The advantage $\mathcal{A}$ has in the security game for a permanent revocation scheme with security parameter λ is defined as

$$Adv_{\mathcal{A},\lambda} = \left| Pr[\mathcal{A} \text{ wins}] - \frac{1}{2} \right|$$

A permanent revocation scheme is adaptively secure if for all poly-time algorithms $\mathcal{A}$ we have that $Adv_{\mathcal{A},\lambda} = negl(\lambda)$.

2.2 3-Linear maps

For groups $\mathbb{G}, \mathbb{G}_T$ of the same prime order p, a 3-linear map $e : \mathbb{G} \times \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ satisfies:

1. Multilinearity. For every $g_1, g_2, g_3 \in \mathbb{G}$ and $\alpha \in \mathbb{Z}_p$ it holds that $e(g_1^\alpha, g_2, g_3) = e(g_1, g_2^\alpha, g_3) = e(g_1, g_2, g_3^\alpha) = e(g_1, g_2, g_3)^\alpha$
2. Non-degeneracy. If $g_1, g_2, g_3 \in \mathbb{G}$ are all generators of $\mathbb{G}$ then $e(g_1, g_2, g_3)$ is a generator of $\mathbb{G}$.

We call $\mathbb{G}$ a (symmetric) 3-linear group and $\mathbb{G}_T$ the target group.

3 The scheme over a 3-linear mapping

In this section we present our scheme which is based on techniques from Lewko et al. [LSW10]. *Setup*(λ). The algorithm input is a security parameter λ. The setup algorithm chooses a 3-linear group $\mathbb{G}$ of prime order p for $p > 2^\lambda$. It then chooses random generators $g, w, h \in \mathbb{G}$ and random exponents $s, \alpha, b, x_{sb}, x_{\alpha s1}, x_{\alpha s2}, \widehat{ID}, \hat{x} \in \mathbb{Z}_p$. The outputs of the algorithm are the public parameters:

$$PP = (g^{\hat{x}}, g^{x_{\alpha s1} x_{\alpha s2} \hat{x}}, g^{s x_{sb} \hat{x}}, g^{\frac{b}{x_{sb}}}, g^{\frac{s}{x_{\alpha s1}}}, w^{\frac{\alpha}{x_{\alpha s2}}} \in \mathbb{G})$$

and the master secret key:

$$MSK = (s, \alpha, b, x_{sb}, x_{\alpha s1}, x_{\alpha s2}, \widehat{ID}, \hat{x} \in \mathbb{Z}_p, h, w \in \mathbb{G})$$

KeyGen(MSK,ID) Given a user identity $ID \in \mathbb{Z}_p$ and the master secret key MSK, the algorithm chooses random $t \in \mathbb{Z}_p$. It then sets

$$D_1 = w^{\frac{\alpha + t}{b}}$$
$$D_2 = g^t, D_3 = (w^{ID} h)^t$$
$$S_1 = g^{\frac{s}{\widehat{ID} - \widehat{ID}}}, S_2 = (w^{\widehat{ID}} h)^{\frac{s}{\widehat{ID} - \widehat{ID}}}, \text{revoked} = \text{false}$$

The output of the algorithm is $SK_{ID} = \{D_1, D_2, D_3, S_1, S_2\}$.

Revoke(S, PP, MSK). Given a set S of r identities to revoke, the algorithm chooses random $s'_1, \ldots, s'_r \in \mathbb{Z}_p$, sets $s' = s'_1 + \ldots + s'_r \bmod p$ and then for each identity $ID_i \in S$ it sets:

$$S_{1,ID_i} = g^{s'_i}, S_{2,ID_i} = (w^{ID_i} h)^{s'_i}, S_{3,ID_i} = ID_i$$

The algorithm then:

1. Updates the master secret key by replacing s with $s + s' \bmod p$.
2. Updates the public parameters by replacing $g^{s x_{sb} \hat{x}}, g^{\frac{s}{x_{\alpha s1}}}$ with $g^{(s+s') x_{sb} \hat{x}}, g^{\frac{(s+s')}{x_{\alpha s1}}}$ respectively.
3. Broadcasts the state update message $SUM = \{S_{1,ID_i}, S_{2,ID_i}, S_{3,ID_i}\}_{i=1}^r$.

UpdateState(SK_{ID}, SUM, ID). Given a state update message SUM the algorithm updates the secret key SK_{ID}. It first checks if $ID \in \bigcup_{i=1}^{r} S_{3,ID_i}$ and if so it sets $SK_{ID}.\mathsf{revoked} = \mathsf{true}$. Otherwise, it calculates

$$S_1' = \prod_{i=1}^{r} S_{1,ID_i}^{\frac{1}{ID - S_{3,ID_i}}} = \prod_{i=1}^{r} g^{\frac{s_i'}{ID - ID_i}}$$
$$S_2' = \prod_{i=1}^{r} S_{2,ID_i}^{\frac{1}{ID - S_{3,ID_i}}} = \prod_{i=1}^{r} (w^{ID_i} h)^{\frac{s_i'}{ID - ID_i}}$$

It then updates SK_{ID} by setting:

$$S_1 = S_1 \cdot S_1'$$
$$S_2 = S_2 \cdot S_2'$$

Encrypt(PP, M). The encryption algorithm takes as input the public parameters PP and a message $M \in \mathbb{G}_T$. The algorithm first chooses a random exponent $x \in \mathbb{Z}_p$. Next, the algorithm sets:

$$C_0 = M \cdot e(g^{\frac{s}{x_{\alpha s1}}}, w^{\frac{\alpha}{x_{\alpha s2}}}, (g^{x_{\alpha s1} x_{\alpha s2} \hat{x}})^x),$$
$$C_1 = (g^{\hat{x}})^x, C_2 = (g^{s x_{sb} \hat{x}})^x$$

The output of the algorithm is $CT = \{C_0, C_1, C_2\}$.

Decrypt(SK_{ID}, CT, PP). The algorithm is given a secret key SK_{ID}, a ciphertext CT and the public parameters PP. First, if $SK_{ID}.\mathsf{revoked} = \mathsf{true}$ the algorithm outputs $\perp$. Otherwise the algorithm calculates:

$$\begin{aligned}
A &= \frac{e(S_1, D_3, C_1)}{e(S_2, D_2, C_1)} \\
&= \frac{e(g^{\frac{s}{ID - \widehat{ID}}} \cdot \prod_{i=1}^{r} g^{\frac{s_i'}{ID - ID_i}}, (w^{ID} h)^t, (g^{\hat{x}})^x)}{e((w^{\widehat{ID}} h)^{\frac{s}{ID - \widehat{ID}}} \cdot \prod_{i=1}^{r} (w^{ID_i} h)^{\frac{s_i'}{ID - ID_i}}, g^t, (g^{\hat{x}})^x)} \\
&= \frac{e(g^{\frac{s}{ID - \widehat{ID}}}, (w^{ID} h)^t, (g^{\hat{x}})^x) \cdot \prod_{i=1}^{r} e(g^{\frac{s_i'}{ID - ID_i}}, (w^{ID} h)^t, (g^{\hat{x}})^x)}{e((w^{\widehat{ID}} h)^{\frac{s}{ID - \widehat{ID}}}, g^t, (g^{\hat{x}})^x) \cdot \prod_{i=1}^{r} e((w^{ID_i} h)^{\frac{s_i'}{ID - ID_i}}, g^t, (g^{\hat{x}})^x)} \\
&= \frac{e(g, g, w)^{\frac{s ID t \hat{x} x}{ID - \widehat{ID}}}}{e(g, g, w)^{\frac{s \widehat{ID} t \hat{x} x}{ID - \widehat{ID}}}} \prod_{i=1}^{r} \frac{e(g, g, w)^{\frac{s_i' ID t \hat{x} x}{ID - ID_i}}}{e(g, g, w)^{\frac{s_i' ID_i t \hat{x} x}{ID - ID_i}}} \\
&= e(g, g, w)^{st\hat{x}x} \prod_{i=1}^{r} e(g, g, w)^{s_i t \hat{x} x} = e(g, g, w)^{(s + s') t \hat{x} x}
\end{aligned}$$

It then calculates

$$
\begin{aligned}
M &= \frac{C_0 \cdot A}{e(C_2, g^{\frac{b}{x_{sb}}}, D_1)} \\
&= \frac{M \cdot e(g^{\frac{(s+s')}{x_{\alpha s1}}}, w^{\frac{\alpha}{x_{\alpha s2}}}, (g^{x_{\alpha s1} x_{\alpha s2} \hat{x}})^x) \cdot e(g,g,w)^{(s+s')tx}}{e((g^{(s+s')x_{sb}\hat{x}})^x, g^{\frac{b}{x_{sb}}}, w^{\frac{\alpha+t}{b}})} \\
&= \frac{M \cdot e(g,g,w)^{(s+s')\alpha\hat{x}x} \cdot e(g,g,w)^{(s+s')t\hat{x}x}}{e(g,g,w)^{(s+s')\hat{x}x\alpha+(s+s')\hat{x}xt}} = M
\end{aligned}
$$

4 Security

In this section we define a problem based on our scheme, extend the lower bound in generic bilinear Groups of Boneh et al. [BBG05] to k-linear groups and prove that our problem holds in the generic group model.

4.1 Permanent revocation problem definition

We now state the security of our scheme as an assumption, this translation uses the following lemma.

Notation 1 *Let* Requests *be a string representing requests of an adversary of the form* $\{IDR, RR\}^*$ *where* IDR *is a single identity and* RR *is a set of identities. We denote by* $\mathcal{A}View_{Requests}$ *the view of an adversary* $\mathcal{A}$ *during the* Key Query and Revocation *phase of the security game where* Requests *represents the requests made by* $\mathcal{A}$.

Lemma 1. *For every adversary* $\mathcal{A}$ *with a requests string* Requests, *there exist an adversary* $\mathcal{A}'$ *and a string* Requests' *of the form* $\{IDR||SURR\}^*$ *where* $IDR||SURR$ *is an identity followed by a singleton of a single identity s.t.* $\mathcal{A}'$ *can generate a view indistinguishable to* $\mathcal{A}View_{Requests}$ *using* $\mathcal{A}'View_{Requests'}$.

Proof. We prove the lemma in two steps. First we show that breaking each revocation request for a set of identities, RR into a sequence of single user revocation request, $\{SURR\}^{|RR|}$ yields a view which contains the original view thus moving from $\{IDR, RR\}^*$ to $\{IDR, SURR\}^*$. Let RR be a set of r identities $ID_1, \ldots, ID_r$ for revocation. After sending the revocation request to the challenger, the adversary learns

$$S_{1,ID_i} = g^{s'_i}, S_{2,ID_i} = (w^{ID_i} h)^{s'_i}, S_{3,ID_i} = ID_i$$

for each $ID_i \in RR$ and the new public parameters

$$g^{(s+s')x_{sb}\hat{x}}, g^{\frac{(s+s')}{x_{\alpha s1}}}$$

where $s' = \sum_{i=1}^{r} s'_i$. Instead of sending one revocation request, $\mathcal{A}'$ requests r revocations, one for each $ID_i \in RR$ and learns for $i = 1$ to r

$$S_{1,ID_i} = g^{s'_i}, S_{2,ID_i} = (w^{ID_i} h)^{s'_i}, S_{3,ID_i} = ID_i$$

and r pairs of the new public parameters, for $i = 1$ to r

$$g^{(s+\sum_{k=1}^{i} s'_k) x_{sb} \hat{x}}, g^{\frac{(s+\sum_{k=1}^{i} s'_k)}{x_{\alpha s1}}}$$

We then show that $\mathcal{A}'$ is able to add a key query request between every two consecutive revocation requests and vice versa thus moving from $\{IDR, SURR\}^*$ to $\{IDR||SURR\}^*$. We first observe that revocations requests do not depend on key query requests, thus between any two consecutive revocation requests, $\mathcal{A}'$ can add a key query request and ignore the key returned from the challenger. Next, between any two consecutive key query requests, $\mathcal{A}'$ can add an empty revocation request which does not change it's view.

The security of a permanent revocation scheme is defined as a game between a challenger and an attack algorithm $\mathcal{A}$. Using lemma 1 we can assume that during the keys generation and revocations requests phase, the adversary sends a polynomial pairs of requests, each in the form of a key query followed by a single user revocation query. We can thus state the security of our scheme as the next problem.

Permanent revocation problem definition. We define the permanent revocation problem as follows. Choose a 3-linear group $\mathbb{G}$ of prime order $p > 2^\lambda$ for a security parameter λ. Next choose random $g, h, w \in \mathbb{G}$ and random $s'_0, s'_1, \ldots, s'_q, t_1, \ldots, t_q, \alpha, b, x_{sb}, x_{\alpha s1}, x_{\alpha s2}, \widehat{ID}, \hat{x} \in \mathbb{Z}_p$. Suppose an adversary is given $\overrightarrow{X} =$

$$\begin{aligned}
& g^{\hat{x}}, g^{x_{\alpha s1} x_{\alpha s2} \hat{x}}, g^{\frac{b}{x_{sb}}}, w^{\frac{\alpha}{x_{\alpha s2}}} \\
\forall_{i \in [1,q]} \; & w^{\frac{\alpha + t_i}{b}}, g^{t_i}, (w^{ID_i} h)^{t_i}, g^{\frac{\sum_{k=0}^{i-1} s'_k}{ID_i - \widehat{ID}}}, (w^{\widehat{ID}} h)^{\frac{\sum_{k=0}^{i-1} s'_k}{ID_i - \widehat{ID}}} \\
\forall_{j \in [1,q]} \; & g^{s'_j}, (w^{ID'_j} h)^{s'_j}, ID'_j \\
\forall_{j \in [0,q]} \; & g^{(\sum_{k=0}^{j} s'_k) x_{sb} \hat{x}}, g^{\frac{\sum_{k=0}^{j} s'_k}{x_{\alpha s1}}} \\
& (g^{\hat{x}})^x, (g^{(\sum_{j=0}^{q} s'_j) x_{sb} \hat{x}})^x
\end{aligned}$$

where $\bigcup_{i=1}^{q} ID_i = \bigcup_{j=1}^{q} ID'_j$. Then it must be hard to distinguish

$$T = e(g^{\frac{(\sum_{i=0}^{q} s'_i)}{x_{\alpha s1}}}, w^{\frac{\alpha}{x_{\alpha s2}}}, (g^{x_{\alpha s1} x_{\alpha s2} \hat{x}})^x) = e(g, g, w)^{(\sum_{i=0}^{q} s'_i) \alpha \hat{x} x}$$

from a random element in $\mathbb{G}_T$. An algorithm $\mathbb{A}$ that outputs $z \in \{0,1\}$ has advantage ϵ in solving decision permanent revocation in $\mathbb{G}$ if

$$|Pr[\mathcal{A}(\overrightarrow{X}, T = e(g,g,w)^{(\sum_{i=0}^{q} s'_i)\alpha\hat{x}x})] - Pr[\mathcal{A}(\overrightarrow{X}, T = R)]| \geq \epsilon$$

We say that the decision permanent revocation assumption holds if no polytime algorithm has a non-negligible advantage in solving the decision permanent revocation problem. We next give a proof that the assumption holds in the generic group model.

4.2 Lower Bound in Generic k-linear groups

Boneh et al. [BBG05] gave complexity lower bound for decisional problems in the generic Bilinear groups that was later geneleraized by Katz et al. [KSW13]. Extending the result of [BBG05] to generic k-linear groups is straight forward and is given here for completeness. Using their terminology, a decisional bilinear problem is a (P, Q, f) problem where P and Q are the sets of a polynomials that are given in the k-linear group $\mathbb{G}$ and the target group $\mathbb{G}_T$ respectively and f is a polynomial given in the target group which the adversary needs to distinguish from a random polynomial.

In the generic group model [Sho97], a generic adversary $\mathcal{A}$ is not given the actual polynomials. Instead, it is given unique handles to each polynomial and an oracle access to the needed operations. In the k-linear settings, the groups' operations correspond to addition of polynomials from the same group and the k-linear mapping correspond to multiplication of k polynomials from the bilinear group (which result in polynomial in the target group).

Some notations:

We describe polynomials using formal variables denoted by capital letters. Let $P \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be an s-tuple of n-variate polynomials over $\mathbb{F}_p$. Write $P = (p_1, \ldots, p_s)$. We denote by $p_i(x_1, \ldots, x_n)$ the value of a polynomial p_i under the assignment $x_1, \ldots, x_n \in \mathbb{F}_p$. Similarly, we use $P(x_1, \ldots, x_n)$ to denote the tuple $(p_1(x_1, \ldots, x_n), \ldots, p_s(x_1, \ldots, x_n))$. For $x \in \mathbb{F}_p$, $h(x)$ denotes the handle for the element x. Similarly, we define $h(x_1, \ldots, x_n) = (h(x_1), \ldots, h(x_n))$.

For a polynomial $f \in \mathbb{F}_p[X_1, \ldots, X_n]$ we let d_f denote the total degree of f. For a set $P \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ we let $d_P = max\{d_f | f \in P\}$.

Definition 1. *Let $P, Q \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_p$. Write $P = (p_1, \ldots, p_s)$ and $Q = (q_1, \ldots, q_s)$. We say that a polynomial $f \in \mathbb{F}_p[X_1, \ldots, X_n]$ is dependent on the sets (P, Q) if there exist $s^k + s$ constants $\{a_{i_1,\ldots,i_k}\}_{i_1,\ldots,i_k=1}^{s}, \{b_j\}_{j=1}^{s}$ such that $f = \sum_{i_1,\ldots,i_k=1}^{s} (a_{i_1,\ldots,i_k} \prod_{i \in \{i_1,\ldots,i_k\}} p_i) + \sum_{j=1}^{s} b_j q_j$. We say that f is independent of (P, Q) if f is not dependent on (P, Q).*

Theorem 1. *Let $P, Q \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_p$. Let $f \in \mathbb{F}_p[X_1, \ldots, X_n]$. Let $\mathbb{G}$ be a generic k-linear group. Let*

$d = max\{k \cdot d_P, d_Q, d_f\}$. Let $x_1, \ldots, x_n, y \in \mathbb{F}_p, b \in \{0,1\}, t_b = f(x_1, \ldots, x_n)$ and $t_{1-b} = y$. If f is independent from (P,Q) then for any generic algorithm $\mathcal{D}$ that makes at most q queries to the oracles computing the groups operation in $\mathbb{G}, \mathbb{G}_T$ and the k-linear mapping e we have that:

$$|Pr[\mathcal{D}(p, h(P(x_1, \ldots, x_n)), h(Q(x_1, \ldots, x_n)), h(t_0), h(t_1)) = b] - \frac{1}{2}| \leq \frac{(q+2s+2)^2 \cdot d}{2p}$$

Corollary 1. *Let $P, Q \in \mathbb{F}_p[X_1, \ldots, X_n]^s$ be two s-tuples of n-variate polynomials over $\mathbb{F}_p$ and let $f \in \mathbb{F}_p[X_1, \ldots, X_n]$. Led $d = max\{kd_P, d_Q, d_f\}$. If f is independent of (P,Q) then any $\mathcal{A}$ that has advantage $1/2$ in solving the decision (P,Q,f)-Diffie-Hellman Problem in a generic k-linear group $\mathbb{G}$ must take time at least $\Omega(\sqrt{p/d} - s)$.*

High level description of the proof of [BBG05] is as follows. Let $\mathcal{S}$ be an algorithm which simulates the generic group oracle for adversary $\mathcal{A}$.

In a perfect simulation, S instantiate all of the polynomials (P,Q,f) by choosing random values $(x_1, \ldots, x_n)$ for each of the formal variables $(X_1, \ldots, X_n)$ and sends the handles to $\mathcal{A}$ which then issues a sequence of group and mapping operations and is given in return the appropriate handles. Finally, $\mathcal{A}$ outputs a bit b'. We note that two elements get different handles only if they are evaluated to different values under the assignment.

Now, consider a simulation where S does not start by choosing random assignment for the polynomials, but instead keeps track of the formal polynomials themselves. In this simulation, $\mathcal{A}$ can only distinguish between t_b and t_{1-b} if it is able to generate a formal polynomial that would be symbolically equivalent to some previously-generated polynomial for one of these polynomials but not the other. Doing so contradicts f being independent of (P,Q).

In this simulation two polynomials gets different handles if they are equal as formal polynomials. This only introduces a difference in case it happens during the course of the simulation that two different formal polynomials would take on the same value. That is, if there exist p_i, p_j s.t. $(p_i - p_j)(x_1, \ldots, x_n) = 0$ but $p_i \neq p_j$. For any particular pair of polynomials, the probability that this occurs is bounded by d/p. Since there are $\binom{q+2s+2}{2}$ such pairs, by the union bound we get that the statistical difference between these simulations is $O(\frac{(q+2s+2)^2 \cdot d}{2p})$.

Permanent revocation problem is generically secure. In order to show that the permanent revocation problem is generically secure we first argue that it is an instance of the (P,Q,f) Diffie-Hellman problem. Let w_g, h_g be the discrete log of w, h in base g, that is $g^{w_g} = w, g^{w_h} = h$. Let P be a sequence of all the exponents of elements in $\mathbb{G}$ in the permanent revocation problem and $f = (\sum_{i=0}^{q} s'_i)\alpha\hat{x}xw_g$. The polynomials in P contain inverses of $x_{\alpha s1}, x_{\alpha s2}, b, ID_1 - \widehat{ID}, \ldots, ID_q - \widehat{ID}$ and therefore do not fit the generic proof template of [BBG05]. Stating our assumption in terms of a generator g' such that $g = (g')^{x_{\alpha s1} x_{\alpha s2} b \prod_{i \in [1,q]} (ID_i - \widehat{ID})}$

eliminates these inverses and therefore P is a sequence of polynomials in g'. Furthermore, if the max degree is defined by $d = \max\{3d_P, d_f\}$ then d is linear in q.

$$
\begin{aligned}
& g^{\hat{x}}, g^{x_{\alpha s1}x_{\alpha s2}\hat{x}}, g^{\frac{b}{x_{sb}}}, w^{\frac{\alpha}{x_{\alpha s2}}} \\
\forall_{i\in[1,q]}\ & w^{\frac{\alpha+t_i}{b}}, g^{t_i}, (w^{ID_i}h)^{t_i}, g^{\frac{\sum_{k=0}^{i-1} s'_k}{ID_i - \widehat{ID}}}, (w^{\widehat{ID}}h)^{\frac{\sum_{k=0}^{i-1} s'_k}{ID_i - \widehat{ID}}} \\
\forall_{j\in[1,q]}\ & g^{s'_j}, (w^{ID'_j}h)^{s'_j}, ID'_j \\
\forall_{j\in[0,q]}\ & g^{(\sum_{k=0}^{j} s'_k)x_{sb}\hat{x}}, g^{\frac{\sum_{k=0}^{j} s'_k}{x_{\alpha s1}}} \\
& (g^{\hat{x}})^x, (g^{(\sum_{j=0}^{q} s'_j)x_{sb}\hat{x}})^x
\end{aligned}
$$

It follows from corollary 1 that proving the generic security of the permanent revocation problem is implied by proving that f is independent of (P).

We now show independence by elimination. First, note that $f = (\sum_{i=0}^{q} s'_i)\alpha\hat{x}xw_g$ contains the multiplication αx. The only polynomials that involve α are

$$\frac{\alpha}{x_{\alpha s2}}w_g,\ \forall_{i\in[1,q]}\frac{\alpha+t_i}{b}w_g$$

It is not possible to use $\frac{\alpha}{x_{\alpha s2}}w_g$ since the $x_{\alpha s2}$ denominator enforces multiplication by $x_{\alpha s1}x_{\alpha s2}\hat{x}$ and then another multiplication by one of $\frac{\sum_{k=0}^{j} s'_k}{x_{\alpha s1}}$ for $j \in [0,q]$ which yields a polynomial in the target group that does not involve x.

The b denominator in $\frac{\alpha+t_i}{b}w_g$ enforces a multiplication by $\frac{b}{x_{sb}}$ which in turn enforces a multiplication by one of $(\sum_{k=0}^{j} s'_k)x_{sb}\hat{x}$ for $j \in [0,q]$ or $(\sum_{j=0}^{q} s'_j)x_{sb}\hat{x}x$. As before, it is not possible to use one of $(\sum_{k=0}^{j} s'_k)x_{sb}\hat{x}$ for $j \in [0,q]$ since it yields a polynomial in the target group that does not involve x.

Following this train of thoughts, in order to construct a formal polynomial that would be symbolically equivalent to f, one must begin with

$$
\begin{aligned}
\frac{\alpha+t_i}{b}w_g \cdot \frac{b}{x_{sb}} \cdot (\sum_{j=0}^{q} s'_j)x_{sb}\hat{x}x & \\
&= (\alpha+t_i)w_g(\sum_{j=0}^{q} s'_j)\hat{x}x \\
&= f + t_i w_g(\sum_{j=0}^{q} s'_j)\hat{x}x
\end{aligned}
$$

We are thus left with showing independence between $t_i w_g(\sum_{j=0}^{q} s'_j)\hat{x}x$ and P. We note that $t_i w_g(\sum_{j=0}^{q} s'_j)\hat{x}x$ contains the variable x. The only polynomials that involve x are

$$\hat{x}x, (\sum_{j=0}^{q} s'_j)x_{sb}\hat{x}x$$

It is not possible to use $(\sum_{j=0}^{q} s'_j)x_{sb}\hat{x}x$ because it contains x_{sb} (while backtracking the multiplication done so far cancels this term, it also cancels f). We are left with constructing $t_i w_g(\sum_{j=0}^{q} s'_j)$ using one multiplication. We can also write it as $A + B$ where $A = t_i w_g(\sum_{j=0}^{i-1} s'_j)$ and $B = \sum_{j=i}^{q} t_i w_g s'_j$. Since only one multiplication is possible, we are left with the following polynomials:

$$\forall_{i\in[1,q]} \;\; t_i, (wID_i + h)t_i, \frac{\sum_{k=0}^{i-1} s'_k}{ID_i - \widehat{ID}}, (w\widehat{ID} + h)\frac{\sum_{k=0}^{i-1} s'_k}{ID_i - \widehat{ID}} \tag{1}$$

$$\forall_{j\in[1,q]} \;\; s'_j, (wID'_j + h)s'_j, ID'_j \tag{2}$$

We note that we can construct A with polynomials in (1). On the other hand, this does not apply to B since it involves s'_k values for $k = i$ to q which are not found in (1) for the same i. Finally, we show that for each i, there exists an s'_k where $k \in [1, q]$ such that it is not possible to construct $t_i w_g s'_k$. For each i there exists a j such that $ID_i = ID'_j$ and $j \geq i$. For these i and j, it is only possible to construct $t_i w_g s'_j$ by either multiplying t_i by $(wID'_j + h)s'_j$ or multiplying s'_j by $(wID_i + h)t_i$. Multiplying t_i by $(wID'_j + h)s'_j$ leads to a reminder of $t_i h s'_j$ which can only be canceled subtracting the multiplication of s'_j and $(wID_i+h)t_i$. However, this also cancels $t_i w_g s'_j$. Multiplying s'_j by $(wID_i+h)t_i$ similarly leads to a dead end.

5 The scheme over a GES

5.1 Ideal Graded Encoding Scheme

Following definitions are taken almost verbatim from [GGH12] and [GGH+16].

Ideal Graded Encoding Scheme Let $\mathcal{R}$ be a ring and $\mathbb{U}$ denote a universe set. An ideal graded encoding scheme for $\mathcal{R}, \mathbb{U}$ is a collection of elements $\{[\alpha]_S \subset \{0,1\}^* | \alpha \in \mathcal{R}, S \subseteq \mathbb{U}\}$ with the following properties:

1. There are binary operations '+' and '-' such that for all $\alpha_1, \alpha_2 \in \mathcal{R}$, all $S \subseteq \mathbb{U}$ and all $u_1 \in [\alpha_1]_S, u_2 \in [\alpha_2]_S$:

$$u_1 + u_2 \in [\alpha_1 + \alpha_2]_S \, , u_1 - u_2 \in [\alpha_1 - \alpha_2]_S$$

   where $\alpha_1 + \alpha_2$ and $\alpha_1 - \alpha_2$ are the addition and subtraction in $\mathcal{R}$.
2. There is a binary operation '$\cdot$' such that for all $\alpha_1, \alpha_2 \in \mathcal{R}$, all $S_1, S_2 \subseteq \mathbb{U}$ such that $S_1 \cap S_2 = \emptyset$ and all $u_1 \in [\alpha_1]_S, u_2 \in [\alpha_2]_S$:

$$u_1 \cdot u_2 \in [\alpha_1 \cdot \alpha_2]_{S_1 \cup S_2}$$

   where $\alpha_1 \cdot \alpha_2$ is multiplication in $\mathcal{R}$.

For an element $u \in [\alpha]_S$, we call α the value and S is the index of the element.

Ideal Graded Encoding Scheme procedures Ideal Graded Encoding scheme consists of the following algorithms:

InstGen$(1^\lambda, 1^\kappa) \to (p, prms, s_{GES})$. The instance generator takes as input the security parameter 1^λ and the multi-linearity parameter 1^κ and outputs a prime p of size at least 2^λ, public system parameters $prms$, and a secret parameter s_{GES}.

Encode$(p, prms, s_{GES}, S, a)$. The encoding algorithm takes as input the prime p, the public parameters $prms$, the secret parameter s_{GES}, and a pair (S, a) with $S \subseteq \{1, \ldots, \kappa\}$ and $a \in \mathcal{R}$. The algorithm outputs encoding of an element $v \in [a]_S$.

Add$(prms, v_1, v_2)$. The addition algorithm takes as input the public parameters $prms$ and encoding of two elements in the same index set $v_1 \in [a_1]_S, v_2 \in [a_2]_S$ and outputs the encoding of their added values in the same index set $v \in [a_1 + a_2]_S$. In case the elements' index sets are not equal, the algorithm outputs $\perp$. Similarly, *Subtract*$(prms, v_1, v_2) \to v \in [a_1 - a_2]_S$.

Multiplication$(prms, v_1, v_2)$. The multiplication algorithm takes as input the public parameters $prms$ and encoding of two elements $v_1 \in [a_1]_{S_1}, v_2 \in [a_2]_{S_2}$ and outputs the encoding of their multiplied values in the union of their index sets $v \in [a_1 \cdot a_2]_{S_1 \cup S_2}$. In case the elements' index sets are not disjoint, the algorithm outputs $\perp$.

isZero$(prms, v)$. The zero testing algorithm takes as input the public parameters $prms$ and encoding an element $v \in [a]_S$ and outputs 1 if $a = 0$ and $S = \{1, \ldots, \kappa\}$. Otherwise, it outputs 0.

Extract$(prms, v)$. For an element $v \in [a]_S$, if $S = \{1, \ldots, \kappa\}$ the extraction algorithm outputs a "canonical" and "random" representation in $\{0,1\}^\lambda$ of the value a from it's encoding:

1. For any $a \in \mathcal{R}$ and two $v_1, v_2 \in [a]_{\{1,\ldots,\kappa\}}$, Extract$(params, v_1)$=Extract$(params, v_2)$.
2. The distribution $\{Extract(params, v) : a \in \mathcal{R}, v \in [a]_{\{1,\ldots,\kappa\}}\}$ is nearly uniform over $\{0,1\}^\lambda$.

5.2 The scheme in ideal GES

The scheme in the ideal GES is the same as the scheme in 3-linear maps with two minor changes. First, since the encryption algorithm has no access to the GES secret parameter, it cannot encode messages as elements in the GES. Second, in contrast to 3-linear maps, since there are many target groups in GES, there is no need to "secret share" elements from levels > 1 into shares in level 1. Our scheme is intended to work in the hybrid encryption model. As such, messages in our scheme are random values that are used as keys in a symmetric encryption system for the actual payload using the GES extraction algorithm.

Setup(λ). The algorithm input is a security parameter λ. The setup algorithm sets $\kappa = 3$ and runs *InstGen*$(1^\lambda, 1^\kappa)$ to obtain$(p, prms, s_{GES})$.

The algorithm then chooses random elements $g, w, h, s, \alpha, b, \widehat{ID}, \hat{x} \in \mathbb{Z}_p$ and another 2λ random elements $\{e_i^b\}_{i=1,b\in\{0,1\}}^{\lambda} \in \mathbb{Z}_p$.

The output of the algorithm are the public parameters:

$$PP = (prms, [\hat{x}]_{\{3\}}, [bs\hat{x}]_{\{1,2\}}, [w\alpha s\hat{x}]_{\{1,2,3\}}, \{[e_i^b]_{\{1,2,3\}}\}_{i=1,b\in\{0,1\}}^{\lambda}$$

and the master secret key:

$$MSK = (p, s_{GES}, g, s, \alpha, b, \hat{x}, \widehat{ID}, h, w)$$

KeyGen(MSK,ID) Given a user identity $ID \in \mathbb{Z}_p$ and the master secret key MSK, the algorithm chooses random $t \in \mathbb{Z}_p$. It then sets

$$D_1 = [w\frac{\alpha + t}{b}]_{\{3\}}$$
$$D_2 = [t]_{\{1\}}, D_3 = ([(wID + h)t]_{\{1\}}$$
$$S_1 = [\frac{s}{ID - \widehat{ID}}]_{\{2\}}, S_2 = ([(w\widehat{ID} + h)\frac{s}{ID - \widehat{ID}}]_{\{2\}}$$

The output of the algorithm is $SK_{ID} = \{D_1, D_2, D_3, S_1, S_2\}$.

Revoke(S, PP, MSK). Given a set S of r identities to revoke the algorithm chooses random $s'_1, \ldots, s'_r \in \mathbb{Z}_p$, sets $s' = s'_1 + \ldots + s'_r$ and then for each identity $ID_i \in S$ it sets

$$S_{1,ID_i} = [s'_i]_{\{2\}}, S_{2,ID_i} = ([(wID_i + h)s'_i]_{\{2\}}$$

The algorithm then:

1. Updates the master secret key by replacing s with $s + s'$.
2. Update the public parameters by replacing $[bs\hat{x}]_{\{1,2\}}, [w\alpha(s+s')\hat{x}]_{\{1,2,3\}}$ with $[b(s + s')\hat{x}]_{\{1,2\}}, [w\alpha s\hat{x}]_{\{1,2,3\}}$ respectively.
3. Broadcast the state update message $SUM = \{S_{1,ID_i}, S_{2,ID_i}, S_{3,ID_i}\}_{i=1}^{r}$.

UpdateState(SK_{ID}, SUM, ID). Given a state update message SUM the algorithm updates the secret key SK_{ID}. If first calculates

$$S'_1 = \sum_{i=1}^{r} S_{1,ID_i}{}^{\frac{1}{ID - S_{3,ID_i}}} = \sum_{i=1}^{r} [s'_i]_{\{2\}}^{\frac{1}{ID - ID_i}}$$
$$S_2 = \sum_{i=1}^{r} S_{2,ID_i}{}^{\frac{1}{ID - S_{3,ID_i}}} = \sum_{i=1}^{r} ([(wID_i + h)s'_i]_{\{2\}}^{\frac{1}{ID - ID_i}}$$

It then updates SK_{ID} by setting:

$$S_1 = S_1 + S'_1$$
$$S_2 = S_2 + S'_2$$

Encrypt(PP, M). The encryption algorithm takes as input the public parameters PP and a random message $M \in \{0,1\}^\lambda$. The algorithm first sets $e = \sum_{i=1}^{\lambda} e_i^{m_i}$. The symmetric key for the payload encryption is derived from Extract(prms,e). It then chooses a random exponent $x \in \mathbb{Z}_p$ and sets:

$$C_0 = e + [w\alpha s\hat{x}]^x_{\{1,2,3\}}$$
$$C_1 = [\hat{x}]^x_{\{3\}}, C_2 = [bs\hat{x}]^x_{\{1,2\}}$$

The output of the algorithm is $CT = \{C_0, C_1, C_2\}$.

Decrypt(SK_{ID}, CT). The decryption algorithm takes as input a secret key SK_{ID} and a ciphertext CT. It first sets $A = Mult(S_1, D_3) - Mult(S_2, D_2) = [wst]_{\{1,2\}}$. Next, the algorithm sets $B = Mult(D_1, C_2) = [w\alpha s\hat{x}x + wts\hat{x}x]_{\{1,2,3\}}$. Finally, the algorithm outputs $C_0 - (B - Mult(A, C_1)) = e$.

6 Concluding remarks

In this work we gave a formal definition of permanent revocation systems and their security. We also proposed a public stateful scheme which supports permanent revocation based on 3-linear mapping and is secure in the generic group model. Our scheme enjoys small (constant size) keys and efficient procedures $O(1)$ operations for encryption / decryption and $O(r)$ operations for revocation / state update where r is the number of revoked users.

References

[Age99] Debby M. Wallner Eric J. Harder Ryan C. Agee. Key Management for Multicast: Issues and Architectures. RFC 2627, IEFT, June 1999.

[BBG05] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. *IACR Cryptology ePrint Archive*, 2005:15, 2005.

[BGW05] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO*, volume 3621 of *Lecture Notes in Computer Science*, pages 258–275. Springer, 2005.

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. *Contemporary Mathematics*, 324(1):71–90, 2003.

[CGI+99] Ran Canetti, Juan A. Garay, Gene Itkis, Daniele Micciancio, Moni Naor, and Benny Pinkas. Multicast security: A taxonomy and some efficient constructions. In *INFOCOM*, pages 708–716. IEEE, 1999.

[CMN99] Ran Canetti, Tal Malkin, and Kobbi Nissim. Efficient communication-storage tradeoffs for multicast encryption. In *EUROCRYPT*, volume 1592 of *Lecture Notes in Computer Science*, pages 459–474. Springer, 1999.

[DF02] Yevgeniy Dodis and Nelly Fazio. Public key broadcast encryption for stateless receivers. In *Digital Rights Management Workshop*, volume 2696 of *Lecture Notes in Computer Science*, pages 61–80. Springer, 2002.

[DGK12] Shlomi Dolev, Niv Gilboa, and Marina Kopeetsky. Permanent revocation in attribute based broadcast encryption. In *CyberSecurity*, pages 203–208. IEEE Computer Society, 2012.

[DPP07] Cécile Delerablée, Pascal Paillier, and David Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. In *Pairing*, volume 4575 of *Lecture Notes in Computer Science*, pages 39–59. Springer, 2007.

[FN93] Amos Fiat and Moni Naor. Broadcast encryption. In *CRYPTO*, volume 773 of *Lecture Notes in Computer Science*, pages 480–491. Springer, 1993.

[GGH12] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices and applications. *IACR Cryptology ePrint Archive*, 2012:610, 2012.

[GGH+16] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. *SIAM J. Comput.*, 45(3):882–929, 2016.

[GST04] Michael T. Goodrich, Jonathan Z. Sun, and Roberto Tamassia. Efficient tree-based revocation in groups of low-state devices. In *CRYPTO*, volume 3152 of *Lecture Notes in Computer Science*, pages 511–527. Springer, 2004.

[GSW00] Juan A. Garay, Jessica Staddon, and Avishai Wool. Long-lived broadcast encryption. In *CRYPTO*, volume 1880 of *Lecture Notes in Computer Science*, pages 333–352. Springer, 2000.

[GW09] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems (with short ciphertexts). In *EUROCRYPT*, volume 5479 of *Lecture Notes in Computer Science*, pages 171–188. Springer, 2009.

[HS02] Dani Halevy and Adi Shamir. The LSD broadcast encryption scheme. In *CRYPTO*, volume 2442 of *Lecture Notes in Computer Science*, pages 47–60. Springer, 2002.

[KD98] Kaoru Kurosawa and Yvo Desmedt. Optimum traitor tracing and asymmetric schemes. In *EUROCRYPT*, volume 1403 of *Lecture Notes in Computer Science*, pages 145–157. Springer, 1998.

[KSW13] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. *J. Cryptology*, 26(2):191–224, 2013.

[LSW10] Allison B. Lewko, Amit Sahai, and Brent Waters. Revocation systems with very small private keys. In *IEEE Symposium on Security and Privacy*, pages 273–285. IEEE Computer Society, 2010.

[NNL01] Dalit Naor, Moni Naor, and Jeffery Lotspiech. Revocation and tracing schemes for stateless receivers. In *CRYPTO*, volume 2139 of *Lecture Notes in Computer Science*, pages 41–62. Springer, 2001.

[NP10] Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. *Int. J. Inf. Sec.*, 9(6):411–424, 2010.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266. Springer, 1997.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In *CRYPTO*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636. Springer, 2009.

[WGL00] Chung Kei Wong, Mohamed G. Gouda, and Simon S. Lam. Secure group communications using key graphs. *IEEE/ACM Trans. Netw.*, 8(1):16–30, 2000.

[YJCK04] Eun Sun Yoo, Nam-Su Jho, Jung Hee Cheon, and Myung-Hwan Kim. Efficient broadcast encryption using multiple interpolation methods. In *ICISC*, volume 3506 of *Lecture Notes in Computer Science*, pages 87–103. Springer, 2004.