A Traitor Tracing Scheme Based on RSA for Fast Decryption

John Patrick McGregor, Yiqun Lisa Yin, and Ruby B. Lee

Princeton Architecture Laboratory for Multimedia and Security (PALMS)
Department of Electrical Engineering
Princeton University
{mcgregor,yyin,rblee}@princeton.edu

Abstract. We describe a fully k-resilient traitor tracing scheme that utilizes RSA as a secret-key rather than public-key cryptosystem. Traitor tracing schemes deter piracy in broadcast encryption systems by enabling the identification of authorized users known as traitors that contribute to unauthorized pirate decoders. In the proposed scheme, upon the confiscation of a pirate decoder created by a collusion of k or fewer authorized users, contributing traitors can be identified with certainty. Also, the scheme prevents innocent users from being framed as traitors. The proposed scheme improves upon the decryption efficiency of past traitor tracing proposals. Each authorized user needs to store only a single decryption key, and decryption primarily consists of a single modular exponentiation operation. In addition, unlike previous traitor tracing schemes, the proposed scheme employs the widely deployed RSA algorithm.

1 Introduction

Broadcast encryption is beneficial in scenarios where a content provider wishes to securely distribute the same information to many users or subscribers. The broadcast content is protected with encryption, and only legitimate users should possess the information (e.g., decryption keys) necessary to access the content. These keys can be embedded in software or in tamper-resistant hardware devices such as smart cards.

Current tamper-resistant hardware is vulnerable to a variety of attacks [1], however. Furthermore, truly tamper-resistant software, which includes programs that resist unauthorized tampering or inspection of code and data, has yet to be developed. Thus, authorized users can extract decryption keys from a legitimate software or hardware decoder. These users can then circumvent the security of the system by divulging the compromised decryption keys to unauthorized users. Alternatively, the authorized users may employ the compromised keys to generate new decryption keys for distribution to unauthorized users. The authorized users who illegally extract and distribute decryption keys are *traitors*, and the unauthorized users who unfairly obtain the keys are *pirates*. The illegal decoder software or hardware devices created by the traitors are *pirate decoders*.

J. Ioannidis, A. Keromytis, and M.Yung (Eds.): ACNS 2005, LNCS 3531, pp. 56–74, 2005.

Traitor tracing schemes, which are also called *traceability schemes*, protect keys by enabling the identification of the source of pirated broadcast decryption keys. In systems that incorporate a traitor tracing scheme, it is possible to identify one or more contributing traitors upon confiscation of a pirate decoder using a *traitor tracing algorithm*. For a traitor tracing algorithm to be valuable, the scheme must be *frameproof*. The frameproof property ensures that a collusion of traitors cannot create a pirate decoder that would implicate an innocent user as being a traitor.

Past traitor tracing proposals have focused on providing an extensive suite of security services while reducing encryption and network communication requirements. However, decryption is often slow: existing traceability schemes may require dozens of modular exponentiations or thousands of symmetric-key decryptions per user per broadcast secret. In this work, we introduce a new secret-key traitor tracing system that improves upon the decryption performance of existing proposals by enabling decryption to be performed with essentially a single modular exponentiation operation.

The paper is organized as follows. In Section 1, we discuss past work in traitor tracing research and the contributions of this paper. In Section 2, we present the system model and discuss the RSA encryption algorithm. In Section 3, we present the implementation of the new traceability scheme. We analyze the security of the scheme in Section 4, and we present the traitor tracing algorithms in Section 5. In Section 6, we analyze the performance and implementation costs of the scheme, and we conclude in Section 7.

1.1 Past Work

Fiat and Naor introduced broadcast encryption in [14]. In their scheme, there exists a set of n authorized users, and a content provider can dynamically specify a privileged subset (of size $\leq n$) of authorized users that can decrypt certain encrypted messages. A message can be securely broadcast to such a privileged subset unless a group of $k+1$ or more authorized users not belonging to the privileged subset collude to construct a pirate decoder to recover the message. The communication overhead, i.e., the factor increase in message size, is $O(k^2 \log^2 k \log n)$. Also, each user must store $O(k \log k \log n)$ decryption keys. Many improvements to this scheme have been presented, but few enable the identification of traitors that collude to distribute pirate decryption keys to unauthorized users.

To combat such piracy of decryption keys, Chor, Fiat and Naor introduced traitor tracing schemes in [7, 8]. These schemes are *k-resilient*, which means if k or fewer traitors contribute to the construction of the pirate decoder, at least one of those traitors can be identified. In the deterministic symmetric-key one-level scheme of [7, 8], the computation and communication costs depend on the total number of users, n, and on the largest tolerable collusion size, k. Each user must store $O(k^2 \log n)$ decryption keys, and each user must perform $O(k^2 \log n)$ operations to decrypt the content upon receipt of the broadcast transmission. The one-level scheme also increases the communication cost of broadcasting secret content by a factor of $O(k^4 \log n)$. The deterministic symmetric-key two-

level scheme of [7, 8] reduces the encryption complexity and communication overhead relative to the one-level scheme at the cost of increasing the decryption complexity and the number of decryption keys per user.

Pfitzmann introduced the concept of *asymmetric* traitor tracing in [26]. This feature allows a content provider to unambiguously convince a third party of a traitor's guilt. Previous proposals for traitor tracing were *symmetric*, which means the content provider shares all secret information with the set of authorized users. In a symmetric scheme, a dishonest content provider can frame an innocent authorized user as being a traitor by building an "unauthorized" decoder that contains a particular user's decryption key.

Public-key k-resilient traitor tracing schemes have also been introduced (e.g., [5, 19, 22, 36]). In such a scheme, publicly known encryption keys can be used to encrypt and subsequently transmit a secret to the entire set of authorized users. The authorized users then employ their respective private decryption keys to decode the transmission. The public-key scheme presented in [5] is symmetric, and the one described in [22] is asymmetric but requires a trusted third party. Asymmetric public-key traitor tracing schemes that do not require a trusted third party are described in [19, 36].

In some situations, a traitor may decrypt the broadcast information and then transmit the plaintext result to pirates rather than distribute a pirate decoder that contains valid decryption keys. Researchers have suggested combining digital fingerprinting and traitor tracing to prevent such piracy [15, 27, 29]. The systems discussed in [27] employ provably secure, robust digital watermark constructions presented in [6, 9]. More efficient and effective integrated fingerprinting and traceability schemes are described in [15, 29]. However, as illustrated by attacks on digital fingerprinting technologies, it can be difficult to design a practical fingerprinting scheme that a savvy attacker cannot thwart [10]. In this paper, we consider only those scenarios in which traitors do not command the resources necessary to distribute decrypted content efficiently; we assume that traitors can only distribute decryption keys.

Researchers have presented many other traceability schemes and traitor tracing algorithms that employ a rich variety of mathematical tools (e.g., [8, 13, 17, 20–22, 24, 25, 32, 34, 35]). For instance, Kurosawa and Desmedt describe a highly efficient k-resilient symmetric traceability scheme in [22]. Their scheme incurs a communication overhead of $O(k)$ and requires each user to store 1 decryption key and to perform $O(k)$ decryption operations per transmitted secret. Kiayias and Yung propose a public key traitor tracing scheme with "constant transmission overhead" [21]. However, in that scheme, the minimum size of the broadcast message may be impractical if protection against large collusions is desired. Researchers present systems that efficiently incorporate broadcast encryption and some degree of traceability in [13, 17, 24, 35]. For example, in [24], a highly efficient trace-and-revoke scheme is described that allows pirate decoders to be disabled upon confiscation of a pirate decoder without incurring significant re-keying costs. However, the scheme does not guarantee identification of the contributing traitors.

Table 1. Past work comparison (k is the maximum traitor collusion size, n is the total number of authorized users, and M is an RSA modulus).

Traitor Tracing Scheme	Communication Overhead	Decryption Complexity (Dominant Component) per User	Number of Decryption Keys per User
One-level [7, 8]	$O(k^4 \log n)$	$O(k^2 \log n)$ sym. decryptions	$O(k^2 \log n)$
Two-level [7, 8]	$O(k^3 \log^4 k \log(n/k))$	$O(k^2 \log^2 k \log n)$ sym. decryptions	$O(k^2 \log^2 k \log(n/k))$
Public-key [5, 19, 22, 36]	$O(k)$	$O(k)$ exponentiations	$O(1)$
Our Proposal	$O(\max(k \log n, k \log \log M / \log k))$	$\sim$1 exponentiation	1

Table 1 summarizes the performance and characteristics of certain traitor tracing schemes that can *identify* members of a traitor collusion of k or fewer traitors with certainty. We do not compare our results to trace-and-revoke schemes or probabilistic traceability schemes that do not guarantee traitor identification upon confiscation of a pirate decoder. In the table, n is the maximum number of authorized users, k is the maximum tolerable collusion size, and M is a typical value for an RSA modulus (e.g., $\sim 2^{1024}$). "Sym. decryptions" means symmetric-key decryption operations.

1.2 Our Contributions

We propose Traitor Tracing using RSA (TTR), a fully k-resilient traceability scheme based upon the RSA encryption algorithm. Although we employ RSA, TTR is not public-key: we apply RSA as a secret-key cryptosystem rather than as a public-key cryptosystem. This design choice enables many security features, including the prevention of common modulus attacks [11, 33]. Our system enables traceability against collusions of k or fewer traitors if the factoring problem is hard, and the encryption scheme is secure against known and chosen plaintext attacks if the RSA problem is hard. Furthermore, TTR prevents traitor collusions from framing innocent users.

We present both clear-box and black-box traitor tracing algorithms for TTR. The efficient clear-box algorithm can always identify at least one of the traitors in a collusion of size k or fewer. The efficient black-box algorithm can identify all of the contributing traitors in a collusion of size k or fewer, even when keys cannot be explicitly extracted from the pirate decoder, but only for a limited and special class of pirate decoders.

TTR improves decryption performance relative to past proposals at the cost of increasing the computation and transmission requirements of the content provider. As shown in Table 1, TTR requires only a single modular exponentiation operation and a relatively insignificant number of modular multiplication

operations to perform decryption upon receipt of a broadcast secret[1]. Though modular exponentiations are computationally more expensive than symmetric key encryptions, TTR still exhibits the highest decryption performance for realistic numbers of users and traitors. Furthermore, TTR only requires each authorized user to store a single decryption key, which may be only 256 bytes in size in realistic scenarios. The communication overhead and encryption complexity of TTR are $O(\max(k \log n, k \log \log M / \log k))$.

2 Preliminaries

2.1 System Model

The broadcast encryption system model used in this paper involves several entities:

- **Content Provider.** The content provider prepares, encrypts, and transmits broadcast messages.
- **Universe of Users.** Broadcast messages are transmitted to the universe of all authorized and unauthorized users, U.
- **Authorized Users.** Only the members of the set of authorized users, T, are provided with the information needed to decode broadcast messages. The maximum number of authorized users is n, so $T = \{t_1, t_2, \ldots, t_n\}$, and $T \subseteq U$.

We present an open traceability scheme in which the methods employed to perform encryption and decryption are public, but the keys used to perform these operations are private. The content provider does not reveal the secret encryption keys to the users, and authorized users (who are not traitors) do not reveal their personal decryption keys to other users.

The following six components typically comprise an open traceability scheme:

- **Provider Initialization.** A content provider generates initial values required to produce the broadcast encryption keys and the user decryption keys.
- **User Initialization.** An authorized user t_i is added to the set of authorized users, T, by requesting that the content provider generate and securely distribute a user decryption key to t_i.
- **Encryption.** The content provider encrypts a message one or more times using one or more secret encryption keys.
- **Transmission.** The content provider transmits the encrypted message to all users.
- **Decryption.** Upon receipt of an encrypted message from the content provider, each authorized user decrypts the secret using his respective decryption key.
- **Traitor Tracing Algorithms.** Upon confiscation of a pirate decoder, the content provider invokes a tracing algorithm to identify contributing traitors.

[1] $(1 + O(\log n / \log M))$ exponentiations are required by a TTR decryption, which is ~ 1 exponentiation for realistic values of n and M (see Section 6.4).

2.2 RSA

Since TTR is based upon the RSA encryption algorithm, we will briefly describe the operation of standard RSA. In RSA, security is related to the computational difficulty of factoring large integers [28]. Let M be the product of two large prime integers, p and q, where p and q are roughly the same size. We call M the RSA modulus. Now, find two integers e and d such that $ed \equiv 1 \bmod \phi(M)$, where ϕ is Euler's totient function: $\phi(M) = (p-1)(q-1)$. The integers e and d are called the encryption exponent and the decryption exponent, respectively. The pair $\langle M, e\rangle$ is the encryption key, and the pair $\langle M, d\rangle$ is the decryption key. Given a plaintext block $a \in \mathbf{Z}_M^*$, a sender can encrypt a to produce a ciphertext c using the public key by computing $c = a^e \bmod M$. The receiver can decrypt c using the private key by computing $a = c^d \bmod M$. There are many minor implementation details required to fortify protocols using RSA against potential attacks. Boneh summarizes several attacks on RSA implementations in [3].

Consider a very simple traitor tracing scheme based on RSA: the content provider generates a common modulus M and a key pair $\langle e_i, d_i\rangle$ for each user t_i. The provider keeps the encryption exponent e_i secret (i.e., unknown to t_i) and passes the decryption key $\langle M, d_i\rangle$ to t_i. To broadcast information to authorized users, the provider simply encrypts messages individually for each user. However, this scheme is very inefficient, since the number of encryption keys and the communication overhead are the same as the number of users, n.

We can improve upon the performance of this simple scheme. Using the multiplicative properties of RSA, we can generate ciphertexts with a few encryption keys that can be decrypted using many decryption keys. The general method, which employs techniques also used in RSA-based threshold cryptosystems (initiated in [12]), operates as follows. Given a plaintext block a and a modulus M, we generate L ciphertexts $C = \{c_1, c_2, \ldots, c_L\}$ encrypted using L different non-zero positive encryption exponents $e_1, e_2, \ldots, e_L$:

$$c_j = a^{e_j} \bmod M \tag{1}$$

Now, a user can multiply all L ciphertexts modulo M to obtain a "product ciphertext", c_{PROD}, that is equivalent to encrypting a one time using a single encryption exponent that is the sum of the L encryption exponents:

$$c_{PROD} = \prod_{i=1}^{L} c_i \bmod M = \left(\prod_{i=1}^{L} a^{e_i} \bmod M\right) \bmod M = a^{e_{SUM}} \bmod M, \tag{2}$$

where $e_{SUM} = \sum_{i=1}^{L} e_i$. A user could subsequently decrypt the c_{PROD} using a decryption exponent $d_{SUM} \equiv e_{SUM}^{-1} \bmod \phi(M)$.

In general, upon obtaining the L ciphertext blocks, a user can multiply (modulo M) a subset of the ciphertexts in C to obtain a different product ciphertext. This new product ciphertext could be decrypted using a new decryption exponent. Since there exist $2^L - 1$ nonempty subsets of C, there exist $2^L - 1$ product ciphertexts that can be decrypted with $2^L - 1$ decryption keys. Hence, we can generate ciphertexts that can be deciphered using up to $2^L - 1$ decryption keys by performing only L encryptions with L encryption keys.

3 A Traceability Scheme Using RSA

TTR takes advantage of the multiplicative properties of RSA to generate and support many decryption keys using relatively few encryption keys. TTR is parameterized by M, n, k, s, L, and α. M is the RSA modulus, and n is the number of authorized users in T. The parameter k represents the maximum tolerable traitor collusion size, and s is the security parameter of the scheme. For example, in a scenario with $k = 10$ and $s = 20$, any collusion of size at most 10 can produce a non-traceable key with probability at most 2^{-20}.

The parameters L and α are based on the values of M, n, k, and s. L represents the number of encryption exponents in the scheme, and the value of L is as follows:

$$L = O(\max(k \log n, k \log \log M / \log k)). \tag{3}$$

The parameter α equals the probability that an element in a user's decryption vector (described below) equals 1, and $\alpha = 1/k$. The rationale for the values of L and α will be presented in Section 4.4.

We now present the provider initialization, user initialization, encryption, transmission, and decryption components of the new scheme. The traitor tracing algorithms are presented in Section 5.

3.1 Provider Initialization

During provider initialization, the content provider creates the secret encryption keys and the information required to generate future user decryption keys. First, the content provider generates a RSA modulus $M = pq$, where p and q are both prime. For performance reasons, we require that p and q be safe primes, i.e., the integers $(p-1)/2$ and $(q-1)/2$ are also prime. Second, the content provider randomly generates a vector E of L unique encryption exponents for M. For each $e_j \in E$, $e_j \sim M$. The content provider keeps all of the encryption exponents and the values of p, q, and $\phi(M)$ secret.

We assume that the content provider's secrets are contained in a single device, but we note that the content provider is not required to be centralized. We could improve security by using RSA threshold techniques (initiated in [12]) to securely store the content provider's secrets and to securely perform key generation operations across multiple devices. Furthermore, the encryption exponents and operations can be distributed across multiple content provider devices using known RSA threshold techniques. Applying such techniques would require an attacker to compromise most or all of the devices in order to successfully expose E or $\phi(M)$.

3.2 User Initialization

When an authorized user t_i joins the system, the content provider generates and securely distributes a user decryption key DK_i to t_i as follows:

1. Randomly generate an L-dimensional Boolean vector $v^{(i)}$. Each element in the vector is set to 1 with probability α, and each element in the vector is set to 0 with probability $1 - \alpha$. In addition, $v^{(i)}$ must not consist of all zeroes, and no two $v^{(i)}$'s are the same. Repeat until $v^{(i)}$ is found where $\sum_{j=1}^{L} v_j^{(i)} e_j$ is relatively prime to $\phi(M)$.
2. Using the extended Euclidean algorithm, calculate a decryption exponent d_i such that:
$$d_i = \left(\sum_{j=1}^{L} v_j^{(i)} e_j \right)^{-1} \bmod \phi(M) \tag{4}$$
If d_i is probabilistically prime, proceed to the next step; otherwise, return to Step 1 and restart the process.
3. Distribute $DK_i = \langle v^{(i)}, d_i, M \rangle$ to t_i via a secure channel.

3.3 Encryption, Transmission, and Decryption

To encrypt a plaintext message P using this scheme, the content provider performs L RSA encryptions on P using each of the encryption exponents in E. The resulting ciphertext C is:

$$C = \langle P^{e_1} \bmod M, ..., P^{e_L} \bmod M \rangle. \tag{5}$$

To ensure semantic security, all plaintext messages P should be prepared using Optimal Asymmetric Encryption Padding (OAEP) [2, 4, 16, 31] or a similar method. OAEP is a provably secure mechanism for padding and encoding plaintext messages prior to RSA encryption.

The resulting ciphertext C is then broadcast to all users in U.

Upon receiving the ciphertext C, an authorized user t_i in T can decrypt the ciphertext using his decryption key $DK_i = \langle v^{(i)}, d_i, M \rangle$ as follows:

$$P = \left(\prod_{j=1}^{L} (c_j)^{v_j^{(i)}} \right)^{d_i} \bmod M, \text{ where } c_j = P^{e_j} \bmod M. \tag{6}$$

It is easy to see that decryption works by the definition of the decryption keys.

4 Security Analysis

We now analyze the security properties of TTR. Proof sketches are provided for each of the theorems introduced in this section, and detailed proofs of the theorems are presented in the full version of this paper. Also, throughout this section, the terms "polynomial number", "polynomial size", and "polynomial time" imply that a number or running time is bounded by a function that is polynomial in a reasonable security parameter, such as s or the number of bits in the chosen modulus M.

4.1 Resilience Against Known Attacks on RSA

All authorized users possess unique decryption keys based upon the same RSA modulus M. However, the system is *not* vulnerable to the common modulus attacks described in [11, 33].

We classify common modulus attacks into two types. In the first type of common modulus attack, if an adversary has knowledge of two (or more) RSA encryption exponents used to encrypt the same message, the adversary can recover the message using the two (or more) ciphertexts without requiring knowledge of $\phi(M)$ or any decryption exponents [33]. The proposed scheme defends against this attack by treating RSA as a *secret-key cryptosystem* rather than as a *public-key cryptosystem*. The encryption keys employed by the content provider are not revealed to the authorized users, and therefore neither a collusion of users nor a passive adversary can implement this attack.

The second type of common modulus attack operates as follows. If an adversary has knowledge of an encryption key and the corresponding decryption key for a given RSA modulus M, the adversary can factor the modulus using a probabilistic algorithm or can calculate the decryption key corresponding to any encryption key [11]. Neither unauthorized nor authorized users can realize such attacks in the proposed scheme, however, for they do not possess knowledge of both the encryption and decryption keys for the modulus M.

4.2 Security of TTR Encryption

We begin the security analysis with two theorems concerning the security of TTR encryption. Theorem 1 shows that it is difficult for a *passive* unauthorized user to decrypt a given ciphertext in the proposed encryption scheme. For certain applications, however, it may also be desirable for the scheme to be semantically secure against an *active* adversary that can prepare specific content (i.e., plaintext) for the content provider to encrypt. Theorem 2 demonstrates semantic security for TTR against chosen plaintext attacks.

Theorem 1. *Given an RSA modulus M and a polynomial number of known plaintext-ciphertext pairs, a passive adversary cannot decrypt a new ciphertext with non-negligible probability in TTR, assuming the intractability of the RSA problem.*

Proof Sketch. Assume that there exists a polynomial-time algorithm A such that a passive adversary can use algorithm A to decrypt a new ciphertext with non-negligible probability given a polynomial number of plaintext-ciphertext pairs. We can show how to construct a polynomial time algorithm B that finds a solution to the RSA problem by using A as a subroutine.

At a high level, given the inputs of an RSA modulus M, a random encryption exponent e, and a ciphertext $C = P^e \bmod M$, B operates as follows. B first randomly defines (but does not explicitly calculate) a set E of L encryption exponents that are based on a function of the input e. Using these definitions, B

computes a polynomial number, R, of plaintext-ciphertext pairs that are consistent with E and M. With these results, B prepares a new $(R+1)$th ciphertext that is based on the input C. B applies this new ciphertext with the R plaintext-ciphertext pairs as inputs to A, and then A returns a valid plaintext P that is the solution to the instance of the RSA problem. □

Theorem 2. *The TTR encryption scheme is semantically secure against chosen plaintext attacks, assuming the intractability of the RSA problem.*

Proof Sketch. Using similar ideas as presented in the proof of Theorem 1, we can show that the encryption scheme is semantically secure against chosen plaintext attacks (CPA) if RSA-OAEP is semantically secure against CPA. It is well known that RSA-OAEP is semantically secure against CPA assuming the intractability of the RSA problem [2, 4, 16, 31]. □

4.3 Security Against Traitor Collusions

We now discuss the security of TTR against traitor collusions of authorized users in T. We show that a collusion of up to k traitors cannot produce an "untraceable" decryption key, i.e., a key that does not implicate at least one member of the collusion as being a traitor. Furthermore, we show that for sufficient values of L, a collusion of up to k traitors cannot produce any key that would implicate an innocent user as being a traitor.

We define a "traceable key" as follows. Recall that each authorized user is issued a distinct key with a distinct decryption exponent. We say that a pirate key is traceable to an authorized user t_i if the decryption exponent associated with t_i divides one of the integer components of the pirate key. More formally stated, given a user t_i with a key $\langle v^{(i)}, d_i, M\rangle$, a pirate decryption key of the form $\langle v^*, d^*, M\rangle$ is traceable to the user t_i if d_i divides v_j^* for any $1 \leq j \leq L$ or d_i divides d^*. Thus, an "untraceable key" produced by a traitor collusion is a new key in which neither the decryption exponent nor any of the vector elements are divisible by any of the decryption exponents of the traitors' original keys.

We consider the following two classes of keys that may be produced by a traitor collusion:

- The first class includes new valid decryption keys of the standard form defined by the scheme.
- The second class includes new decryption keys that are *not* of the form as defined by the scheme but can be used to successfully decrypt ciphertext and obtain correct plaintext.

We will demonstrate security for TTR against keys from both of these classes. In particular, the vector $v^{(i)}$ generated by the collusion is not required to be a Boolean vector, and the decryption exponents generated by the collusion are not required to be large prime numbers. Instead, each vector entry can be any integer, and each decryption exponent can be any integer. We remark that this

is a much stronger result than only showing that the collusion cannot produce a decryption key of the standard form.

We need only to prove security against the strongest possible collusion, i.e., a collusion of k traitors with knowledge of k linearly independent decryption keys. If the k or fewer decryption vectors of the traitor collusion's decryption keys are not linearly independent, then there exists a feasible attack that may allow the collusion to factor M and defeat the scheme. Thus, the vectors must be constructed by the content provider such that any k vectors chosen at random will be linearly independent. As we will explain in Section 4.4, our choice for the value of L ensures that, with overwhelming probability, a collusion's decryption vectors are always linearly independent.

We now present two theorems on the security of TTR against traitor collusions. Theorem 3 states that collusions cannot create untraceable keys, and Theorem 4 states that collusions cannot frame innocent users as being traitors.

Theorem 3. *No collusion of k or fewer authorized users can create an untraceable decryption key with probability greater than 2^{-s} if $L \geq (k-1)(s + \log_2 k + \log_2 n)/(e \log_2 e)$, assuming the difficulty of factoring.*

Proof Sketch. Assume that there exists a polynomial-time memoryless algorithm A such that a collusion of k authorized users can employ algorithm A to create a new, untraceable decryption key with non-negligible probability given their k decryption keys. We show how to construct a polynomial time algorithm B that factors a given modulus M by using A as a subroutine.

At a high level, on input $M(= pq)$, algorithm B operates as follows. B begins by randomly generating k valid and unique decryption keys $DK_i = \langle v^{(i)}, d_i, M \rangle$ of the same form described in the scheme. These keys represent the keys of the traitor collusion. B then applies the k keys as inputs to A to obtain a new decryption key, DK_{NEW}. Next, the algorithm B generates an additional $(L-k)$ decryption keys of a special form. We can show that B can use the first k valid keys, DK_{NEW}, and the additional $(L-k)$ decryption keys to obtain a multiple of $\phi(M)$. If the multiple is non-zero, B can efficiently factor M [11]. If the multiple is zero, we can show that with overwhelming probability given an L of the required size, the key DK_{NEW} is traceable; at least one of the first k d_i's must divide one of the vector elements or the decryption exponent of DK_{NEW}. □

Theorem 4. *If the number of possible valid decryption keys exceeds 2^s, then the probability is exponentially small in s that a collusion of k authorized users can create a decryption key of size that is polynomial in s and that implicates an innocent user as a traitor.*

Proof Sketch. Assume that a collusion of up to k authorized users can use their respective decryption keys to create a new decryption key. In the proposed scheme, the selection of the $(n-k)$ innocent users' decryption vectors and exponents can be performed entirely independently of the selection of the k traitor keys. That is, the keys that represent the innocent authorized users may be any

$(n-k)$-subset chosen uniformly at random from the set of $(2^s/\log M - k)$ possible decryption keys that do not belong to the traitor collusion. Given these facts, we can show that the probability is exponentially small in s that the new decryption key will implicate one of the $(n-k)$ innocent authorized users. □

4.4 Choosing the Parameters

We now discuss the requirements for the values of the parameters α and L. As stated in Section 3, $\alpha = 1/k$. It may possible for this value to be increased or decreased; determining the optimal value of α is a subject for future work. We note that, as shown in the full version of this paper, some values of α will cause the scheme to be insecure. For example, setting α to $1/2$ prevents traceability in some scenarios.

The value of L depends on several factors. First, to satisfy several of the theorems stated above, we require the following lower bound for the value of L:

$$L \geq (k-1)\,(s + \log_2 k + \log_2 n)\,/\,(e \log_2 e) \tag{7}$$

Second, we must ensure that there are enough possible distinct decryption keys to accommodate all of the users in the system. In addition, to satisfy the frameproofing theorem, we must ensure that the number of possible distinct decryption keys exceeds 2^s. Since the expected number of 1's in a vector of length L is $L\alpha$, the number of possible vectors is roughly $\binom{L}{L\alpha} = \binom{L}{L/k}$. However, only a subset of these vectors will correspond to a prime (and therefore valid) decryption exponent. Considering that the probability that a large exponent is prime is approximately $1/\log M$, we have $\binom{L}{L/k} \geq n \log M$, and $\binom{L}{L/k} \geq 2^s \log M$, where log is the natural logarithm. Using Stirling's approximation, a simple calculation shows that $L \geq (k(s + \log_2 n + \log_2 \log M))/\log_2 ek$ is sufficient.

Third, we must ensure that any k vectors produced by the scheme are linearly independent. Otherwise, a traitor collusion of size k or fewer may be able to factor M. Hence, to maintain security, L should be large enough such that, with overwhelming probability, a set of k randomly generated Boolean vectors of length L are linearly independent. In [18], it is shown that the probability of linear independence is at least $1 - O((1-\epsilon)^L)$ for some $\epsilon > 0$ if $k \leq L$. Thus, the lower bound for L cited above is sufficient, as that bound requires $L > s$, and therefore the probability of linear dependence will be exponentially small in the security parameter s.

Hence, we have the following expression for L:

$$L \geq \max\left(\frac{(k-1)\,(s + \log_2 k + \log_2 n)}{e \log_2 e}, \frac{ks + k\log_2 n + k \log_2 \log M}{\log_2 ek}\right) \tag{8}$$

If the security parameter s is treated as a constant, the size of L and the communication overhead of the scheme is $O(\max(k \log n, k \log\log M/\log k))$. In most scenarios, $\log n > \log\log M$, so the communication overhead would be $O(k \log n)$.

Finally, we note that the system can be implemented *without* requiring decryption vectors to be prime. We choose to require all d_i's to be prime in order to simplify the proofs of security.

5 Traitor Tracing Algorithms

Upon confiscation of a pirate decoder device, the content provider invokes traitor tracing algorithms to identify the authorized users that contributed to the construction of the device. First, we explore the "clear-box" case, where it is possible to explicitly extract the representations of all the keys embedded in the pirate decoder. Using the clear-box tracing algorithm, we can always efficiently identify at least one of the traitors who contributed to a pirate decoder. Second, we present a limited "black-box" tracing algorithm. In this case, we cannot extract keys from the pirate decoder, but we can apply inputs to the decoder and observe the resulting outputs. Unlike the clear-box algorithm, the black-box algorithm enables the tracing of keys only in special cases.

5.1 A Clear-Box Tracing Algorithm

We assume that a pirate decoder contains easily recognizable representations of one or more valid decryption keys; these keys are employed by the decoder to perform all message decryptions. As shown in Section 4, a traitor collusion can generate new decryption keys only of a certain form. That is, assuming we choose appropriate values for L, α, and s, traitors cannot create untraceable keys (Theorem 3) or create traceable keys that implicate innocent users as traitors (Theorem 4). Hence, we can use the keys in a pirate decoder to identify contributing traitors.

The clear-box tracing algorithm simply compares components of the decryption keys within the pirate device to all existing user decryption keys. The algorithm proceeds as follows:

1. Let $\langle v^*, d^*, M \rangle$ be a pirate key extracted from a pirate decoder, where $v^* = \{v_1^*, ..., v_L^*\}$. For $1 \leq i \leq n$, repeat the following for each authorized user t_i (whose decryption exponent is d_i):
   (a) If d_i divides v_j^* for any $1 \leq j \leq L$ or d_i divides d^*, then user t_i is a traitor.

We now present a theorem stating that, without framing innocent users, this clear-box algorithm can identify at least one of the traitors that colluded to build the pirate decoder.

Theorem 5. *Given a pirate decryption key generated by a collusion of at most k traitors using their respective decryption keys, then with probability exceeding $1 - 2^{-s}$, the clear-box traitor tracing algorithm can identify at least one traitor in the collusion without implicating any innocent users.*

Proof Sketch. As shown by Theorem 3, no group of traitors of size fewer than $k+1$ can generate a new decryption key with non-negligible probability that does not implicate at least one of the colluding traitors using the clear-box traitor tracing algorithm. Furthermore, as shown by Theorem 4, no collusion of traitors of size fewer than $k+1$ can generate a new decryption key with non-negligible probability that implicates an innocent user. □

Upon discovering the presence of one or more traitors using the proposed scheme, the content provider must re-issue decryption keys to the set of authorized users. We can address this issue by constructing a protocol that distributes new decryption keys to individual, legitimate users at fixed intervals, similar to that described in [30].

5.2 A Limited Black-Box Tracing Algorithm

For the black-box algorithm, we wish to achieve the same goals as desired for the clear-box tracing algorithm, i.e., the identification of at least one contributing traitor and no false implications of guilt. We achieve these goals for a limited class of pirate decoders: *limited-ability pirate decoders*. We define a limited-ability pirate decoder to be a device that contains k or fewer decryption keys that are identical to k or fewer keys issued by the content provider; any one and only one of these keys can be used to perform a single decryption for a given broadcast message. Restricting the pirate device model to limited-ability pirate decoders is reasonable in several practical situations. In a smart card-based decoder or a mass-produced ASIC decoder, storage space may be available only for a single decryption key from a single traitor. Also, it may not be feasible for a pirate decoder to perform multiple decryptions per broadcast message and maintain adequate throughput.

In the black-box algorithm, we identify traitors by applying random data as ciphertext input to the pirate decoder. The decryption of the random data using a decryption key (issued by the content provider) will yield a different and predictable plaintext result for each distinct decryption key. Thus, we can infer which keys are stored in a limited-ability pirate decoder without performing explicit inspection of the pirate device's contents. The decryption key for authorized user t_i is $DK_i = \langle v^{(i)}, d_i, M \rangle$, and the black-box algorithm operates as follows:

1. Randomly generate a set C of L $\lceil \log_2 M \rceil$-bit values, $C = \{c_1, c_2, ..., c_L\}$.
2. Repeat for all i such that $1 \leq i \leq n_{CUR}$, where n_{CUR} is the current number of authorized users:
   (a) Randomly select an integer z such that $v_z^{(i)} = 1$.
   (b) Construct $C' = \{c'_1, c'_2, ..., c'_L\}$ such that $c'_j = c_j$ for $j = z$, and $c'_j = 1$ for $j \neq z$.
   (c) Apply C' to the pirate decoder to obtain decrypted result P.
   (d) Compute $P_{TEST} = \left(\prod_{j=1}^{L} (c'_j)^{v_j^{(i)}} \right)^{d_i} \bmod M$. If P_{TEST} equals P, user t_i is a traitor.

When ciphertext input is applied to a limited-ability pirate decoder, the device chooses one of its keys and employs that key to perform the decryption operation. As a result, the black-box tracing algorithm may identify only one of the many keys stored in the device. To find all traitors with high probability, one can simply repeat the black-box tracing algorithm a number of times that is a multiple of k, e.g., $10k$ times, assuming the decryption keys are chosen at random by the pirate decoder.

Without framing innocent users, the black-box algorithm can identify at least one of the traitors that colluded to build a limited-ability pirate decoder. The straightforward proof of this statement is given in the full version of the paper.

6 Performance Analysis

This section investigates the computation and storage costs of TTR.

6.1 Provider Initialization Costs

The computation and storage costs of the provider initialization procedure described in Section 3.1 are as follows. The one-time computation required to generate M and E includes $O(\log^2 M)$ $(\log M)$-bit probabilistic primality tests such as Miller-Rabin (a summary of which can be found in [23]) and $O(L + \log^2 M)$ $(\log M)$-bit random number generations. For convenience of user initialization and encryption of broadcast messages, it is prudent for the content provider to store E, M, and $\phi(M)$. Thus, the expected storage requirement for the content provider is at minimum $L\lceil \log_2 M \rceil + 2\lceil \log_2 M \rceil$ bits.

6.2 User Initialization Costs

The computation and storage costs of the 3-step user initialization procedure described in Section 3.2 are as follows. Since the probability of the summation in Step 1 being relatively prime to $\phi(M)$ is $1/2$, and since the probability that a possible decryption exponent is prime is approximately $1/\log M$, Step 1 will be executed fewer than $2 \log M$ times on average, and Step 2 will be executed fewer than $\log M$ times on average. Hence, the total time required by Steps 1 and 2 to generate a prime decryption key is dominated by $O(\log M)$ $(\log M)$-bit probabilistic primality tests and $O(\log M)$ L-bit random number generations. Following key generation, the costs required to securely distribute the user decryption key in Step 3 highly depend on the method that is chosen to secure the channel.

A user decryption key, which consists of a Boolean vector v, a prime decryption exponent d, and a modulus M, requires at most $L + 2\lceil \log_2 M \rceil$ bits of storage. This equates to a decryption key size of approximately 256 bytes in realistic scenarios when using a 1024-bit modulus. The content provider also needs to store a copy of each issued decryption key to avoid issuing the same decryption key to two different users.

Table 2. Decryption computation cost for $2^{-s} = 2^{-80}$.

Number of Users (n)	Number of exponentiations			
	$k = 2$	$k = 10$	$k = 100$	$k = 1000$
2	1.006	-	-	-
10	1.006	1.012	-	-
100	1.007	1.013	1.015	-
1000	1.007	1.013	1.015	1.016
1 million	1.008	1.015	1.017	1.018
1 billion	1.010	1.016	1.019	1.019

6.3 Encryption and Transmission Costs

To encrypt a broadcast message in TTR as described in Section 3.3, the content provider must perform L multiple-precision modular exponentiations using L different encryption exponents. We note that we can reduce the bit length of the encryption exponents to improve the speed of the encryption operations without compromising security. Since one encrypted block is transferred for each encryption exponent, the communication overhead is $O(L)$. Though L may range in the hundreds, in many broadcast encryption systems, the transmission costs may apply only to a small portion (e.g., the header) of broadcast messages.

6.4 Decryption Costs

To decrypt a broadcast message in TTR as described in Section 3.3, an authorized user must perform αL multiple-precision modular multiplications and a single multiple-precision modular exponentiation. For reasonable values of M, n, and k, the $O(\alpha L) = O(\max(\log n, \log \log M / \log k))$ modular multiplications require much less computation than the single modular exponentiation.

Table 2 lists the computation required to perform decryption for various sizes of n and k when $2^{-s} = 2^{-80}$. The values in the table are normalized to a single random 1024-bit modular exponentiation. For example, a value of 1.015 indicates that the decryption operation requires 1.5% more computation than an average 1024-bit modular exponentiation. Some table cells do not have entries because the maximum collusion size k cannot exceed the number of users. If $k = 10$, $2^{-s} = 2^{-80}$, and n equals one million users, then L is 237. The number of modular multiplications required to obtain the product ciphertext is therefore $L\alpha - 1 = 237/10 - 1 \approx 23$. If the size of the RSA modulus is 1024 bits, the exponentiation requires 1535 modular multiplications on average [23]. Hence, in this case, generating product ciphertext requires only 1.48% of the computation involved in decryption. As shown in Table 2, the cost of generating the product ciphertext never exceeds 2% of the overall decryption computation.

In practice, a 1024-bit modular exponentiation can be 1000 times slower per decrypted bit than a 128-bit symmetric key decryption operation [23]. However, for realistic numbers of authorized users and traitors, the new scheme still exhibits the highest decryption performance among the past proposals listed in

Table 1. When k is 10 or greater, TTR outperforms the schemes of [8]. In a realistic implementation where n is one million and k is 10, the decryption speedup of TTR over the schemes of [8] exceeds 7.86x.

6.5 Tracing Algorithm Costs

The clear-box tracing algorithm runs in polynomial time. The algorithm requires at most $O(nL)$ integer division operations to identify a traitor. We note that the actual computational complexity of the algorithm is a function of n_{CUR}, which is the current number of authorized users, rather than a function of n, which is the maximum number of authorized users. This is an important distinction, as the values of n_{CUR} and n can differ by orders of magnitude in practice.

The limited black-box tracing algorithm runs in polynomial time. The values of P_{TEST} for each user can be precomputed (at user initialization time) and can be stored in a hash table. Using this precomputed hash table, the expected computation required by the black-box tracing algorithm for single-key decoders is 1 modular exponentiation and an insignificant number (i.e., $O(\alpha L)$) of modular multiplications. In the multiple-key case, we repeat the algorithm $O(k)$ times, so the computation would be $O(k)$ modular exponentiations and $O(k\alpha L)$ modular multiplications.

7 Conclusion

We presented TTR, a fully k-resilient traitor tracing scheme based on RSA that improves upon the decryption efficiency of past traitor tracing proposals. The scheme employs a single RSA modulus that is shared by multiple users, and we realize our security goals by applying RSA as a secret-key cryptosystem rather than a public-key cryptosystem. TTR is also frameproof against collusions of k or fewer traitors, and the scheme enables black-box traitor tracing in certain scenarios. In future work, we will investigate parameter optimizations and extensions to this scheme to enable higher performance and other security features. Furthermore, we will explore constructions for generalizing TTR and other traceability schemes based on similar cryptographic primitives.

Acknowledgements

The authors thank Scott Contini, Jeremy Horwitz, Joe Kilian, and Benny Pinkas for their suggestions and pointers regarding the security analysis. The authors also thank the anonymous reviewers for their helpful comments.

References

1. R. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, John Wiley and Sons, Inc., New York, 2001.
2. M. Bellare and P. Rogaway, "Optimal Asymmetric Encryption," *Proc. of EUROCRYPT '94*, Springer-Verlag LNCS, vol. 950, pp. 92–111, 1995.

3. D. Boneh, "Twenty Years of Attacks on the RSA Cryptosystem," *Notices of the American Mathematical Society*, vol. 46, no. 2, pp. 203–213, 1999.
4. D. Boneh, "Simplified OAEP for the RSA and Rabin Functions," *Proc. of CRYPTO '01*, Springer-Verlag LNCS, vol. 2139, pp. 275–291, 2001.
5. D. Boneh and M. Franklin, "An Efficient Public Key Traitor Tracing Scheme," *Proc. of CRYPTO '99*, Springer-Verlag LNCS, vol. 1666, pp. 338–353, 1999.
6. D. Boneh and J. Shaw, "Collusion-Secure Fingerprinting for Digital Data," *Proc. of CRYPTO '95*, Springer-Verlag LNCS, vol. 963, pp. 452–465, 1995.
7. B. Chor, A. Fiat, and M. Naor, "Tracing Traitors," *Proc. of CRYPTO '94*, Springer-Verlag LNCS, vol. 839, pp. 257–270, 1994.
8. B. Chor, A. Fiat, M. Naor, and B. Pinkas, "Tracing Traitors," *IEEE Transactions on Information Theory*, vol. 46, no. 3, pp. 893–910, May 2000.
9. I. J. Cox, J. Kilian, T. Leighton, and T. Sharmoon, "A Secure Robust Watermark for Multimedia," *Proceedings of the First Information Hiding Workshop – IHW '96*, Springer-Verlag LNCS, vol. 1174, pp. 185–206, 1996.
10. S. A. Craver, J. P. McGregor, M. Wu, B. Liu, A. Stubblefield, B. Swartzlander, D. S. Wallach, D. Dean, and E. W. Felten, "Reading Between the Lines: Lessons from the SDMI Challenge," Princeton Univ. CS Technical Report TR-657-02, 2002.
11. J. M. DeLaurentis, "A Further Weakness in the Common Modulus Protocol for the RSA Cryptoalgorithm," *Cryptologia*, vol. 8, no. 3, pp. 253–259, 1984.
12. Y. Desmedt and Y. Frankel, "Shared Generation of Authenticators and Signatures," *Proc. of CRYPTO '91*, Springer-Verlag LNCS, vol. 576, pp. 457–469, 1991.
13. Y. Dodis, N. Fazio, A. Kiayias, and M, Yung, "Fully Scalable Public-Key Traitor Tracing," *Proceedings of Principles of Distributed Computing (PODC-2003)*, July 2003.
14. A. Fiat and M. Naor, "Broadcast Encryption," *Proc. of CRYPTO '93*, Springer-Verlag LNCS, vol. 773, pp. 480–491, 1993.
15. A. Fiat and T. Tassa, "Dynamic Traitor Tracing," *Proc. of CRYPTO '99*, Springer-Verlag LNCS, vol. 1666, pp. 354–371, 1999.
16. E. Fujisaki, T. Okamoto, D. Pointcheval, and J. Stern, "RSA-OAEP Is Secure under the RSA Assumption," *Proc. of CRYPTO '01*, Springer-Verlag LNCS, vol. 2139, pp. 260–274, 2001.
17. E. Gafni, J. Staddon, and Y. L. Yin, "Efficient Methods for Integrating Traceability and Broadcast Encryption," *Proc. of CRYPTO '99*, Springer-Verlag LNCS, vol. 1666, pp. 372–387, 1999.
18. J. Kahn, J. Komlos, and E. Szemeredi, "On the Probability that a Random $+/-1$ Matrix Is Singular," *Journal of the AMS*, vol. 8, no. 1, pp. 223–240, Jan. 1995.
19. A. Kiayias and M. Yung, "Breaking and Repairing Asymmetric Public-Key Traitor Tracing," *Proceedings of the ACM Workshop on Digital Rights Management*, 2002.
20. A. Kiayias and M. Yung, "Self Protecting Pirates and Black-box Traitor Tracing," *Proc. of CRYPTO '01*, Springer-Verlag LNCS, vol. 2139, pp. 63–79, 2001.
21. A. Kiayias and M. Yung, "Traitor Tracing with a Constant Transmission Rate," *Proc. of EUROCRYPT '02*, Springer-Verlag LNCS, vol. 2332, pp. 450–465, 2002.
22. K. Kurosawa and Y. Desmedt, "Optimum Traitor Tracing and Asymmetric Schemes," *Proc. of EUROCRYPT '98*, Springer-Verlag LNCS, vol. 1403, pp. 145–157, 1998.
23. A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, LLC, Boca Raton, FL, 1997.
24. D. Naor, M. Naor, and J. Lotspiech, "Revocation and Tracing Schemes for Stateless Receivers," *Proc. of CRYPTO '01*, Springer-Verlag LNCS, vol. 2139, pp. 41–62, 2001.

25. M. Naor and B. Pinkas, "Threshold Traitor Tracing," *Proc. of CRYPTO '98*, Springer-Verlag LNCS, vol. 1462, pp. 502–517, 1998.
26. B. Pfitzmann, "Trials of Traced Traitors," *Proceedings of the First Information Hiding Workshop – IHW '96*, Springer-Verlag LNCS, vol. 1174, pp. 49–64, 1996.
27. B. Pfitzmann and M. Waidner, "Asymmetric Fingerprinting for Larger Collusions," *Proceedings of the ACM Conference on Computer and Communications Security*, pp. 145–157, 1997.
28. R. L. Rivest, A. Shamir, and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, February 1978.
29. R. Safavi-Naini and Y. Yang, "Sequential Traitor Tracing," *Proc. of CRYPTO '00*, Springer-Verlag LNCS, vol. 1880, pp. 316–332, 2000.
30. S. Setia, S. Koussih, S. Jajodia, and E. Harder, "Kronos: A Scalable Group Rekeying Approach for Secure Multicast," *Proceedings of the 2000 IEEE Symposium on Security and Privacy*, pp. 215–228, 2000.
31. V. Shoup, "OAEP Reconsidered," *Proc. of CRYPTO '01*, Springer-Verlag LNCS, vol. 2139, pp. 239–259, 2001.
32. A. Silverberg, J. Staddon, and J. Walker, "Efficient Traitor Tracing Algorithms using List Decoding," *Proc. of ASIACRYPT '01*, Springer-Verlag LNCS, vol. 2248, pp. 175–192, 2001.
33. G. J. Simmons, "A Weak Privacy Protocol Using the RSA Cryptosystem," *Cryptologia*, vol. 7, no. 2, pp. 180–182, 1993.
34. D. Stinson and R. Wei, "Combinatorial Properties and Constructions of Traceability Schemes and Frameproof Codes," *SIAM Journal on Discrete Mathematics*, vol. 11, no. 1, 1998.
35. D. Stinson and R. Wei, "Key Preassigned Traceability Schemes for Broadcast Encryption," *Proceedings of Selected Areas in Cryptology – SAC '98*, Springer-Verlag LNCS, vol. 1556, pp. 144–156, 1999.
36. Y. Watanabe, G. Hanaoka and H. Imai, "Efficient Asymmetric Public-Key Traitor Tracing without Trusted Agents," *Progress in Cryptology – CT-RSA 2001*, Springer-Verlag LNCS, vol. 2020, pp. 392–407, 2001.