Practical Pay TV Schemes

Arvind Narayanan[1], C. Pandu Rangan[1]*, and Kwangjo Kim[2]

[1] Department of Computer Science and Engineering,
Indian Institute of Technology, Madras,
Chennai-600036, INDIA.
arvindn@meenakshi.iitm.ernet.in,rangan@iitm.ernet.in
[2] IRIS(International Center for Information Security)
ICU(Information and Communications University)
58-4, Hwaam-Dong, Yusong-gu, Taejon, 305-732, South Korea.
kkj@icu.ac.kr

Abstract. We propose an efficient and robust *Pay TV* scheme for the case when there are a number of streams, as opposed to just one. In our model, the broadcast is divided into *billing periods*; during each billing period the entitlement of the users does not change. We achieve *full flexibility* with only a constant factor data redundancy. Our scheme has very little secure memory requirements and does not require the users' secure keys to be changed once they have been written into the secure memory. There is also no upper limit on the number of subscribers. We extend this scheme to have the *cracker identification* property: If a collusion of less than t users crack their *set-top terminals* and produce a new decryption key, the *exact set* of crackers can be efficiently identified with high probability. This property is similar to but different from the *traitor tracing* schemes of Chor et al [5].

Keywords Broadcast Encryption, Pay TV, Set-top terminal, Cracker Identification.

1 Introduction

A *Pay TV* system consists of a set of *services* or *streams* $S_1, S_2 ..., S_m$ and a set of *users* $U_1, U_2, ..., U_n$. Each user is *subscribed* or *entitled* to some subset of the set of services. The *Broadcaster* encrypts the content of the streams and sends them over the *Broadcast Channel*. Every user is provided with a *Set-top terminal (STT)*, which receives the broadcasted content. A typical STT holds a very limited number of decryption keys in secure, tamper-proof and non-volatile memory. Users can change their entitlement by subscribing to or unsubscribing from a service.

In modeling the system properties, we will stick as closely as possible to actual practice. Accordingly, we assume that the channel is insecure but authenticated.

* Financial support from Ministry of Information Technology,Government of Korea for this author is acknowledged

That is, Eve can receive all messages that are broadcast but cannot herself insert, delete or alter any messages. An important assumption we have made implicitly is that each message sent on the channel belongs to one of a small number of streams, instead of being associated with an arbitrary subset of the users.

One obvious requirement of a Pay TV system is that Alice should be able to decrypt exactly those streams for which she has subscribed. Another important requirement is *full flexibility*: by this we mean that the scheme must impose no restrictions on the set of services to which Alice can subscribe (for instance, by offering all-or-nothing *packages* of services).

We note that the use of an STT containing a secure chip along with tamper-proof memory for storing the users' keys is unavoidable: if we merely give Alice's decryption keys to her, there is nothing to prevent her from sharing them with Oscar who can subsequently view all the streams that Alice is entitled to. It may be argued that this problem is insoluble anyway: Oscar could simply download content off Alice even when STTs are used. There is a crucial difference, however: while sharing of keys is a one time transaction, sharing of content needs to be done continuously. The latter is much easier to detect and is costly for Oscar. In fact, Oscar might find it more economical to subscribe with the broadcaster (Bob) directly. Content sharing is sufficiently difficult in practice to be a non-issue [5].

The STT needs to keep its decryption keys in non-volatile, secure and tamper proof memory. Because of these stringent requirements, the amount of secure memory available is very limited, typically a few kilobytes. In addition, the STT also has a (large) store of non-volatile memory which is neither secure nor tamper-proof.

The tamper proof nature of STTs should not be taken for granted; indeed, STTs have been extensively cracked in practice [9]. A very desirable property to have is that if Alice's STT is cracked, the cracker should not be able to decrypt any streams to which Alice is not currently subscribed.

Clearly, the STT is vulnerable to known-plaintext attack; however, it is infeasible for an attacker to mount a chosen plaintext attack. Further, an adversary can feed in ciphertext and observe the plaintext without requiring broadcast of the ciphertext; thus the STT must also be secure against *chosen-ciphertext* attacks.

Some schemes that have been used in STTs, the *bit-vector scheme* and the *block-by-block scheme* are discussed in [12], in which it is pointed out that these fare poorly both in terms of security and in terms of flexibility and efficient use of bandwidth.

1.1 Related Work

Broadcast encryption was introduced by Fiat and Naor in [7]. They consider the problem of securely broadcasting a secret to a *dynamically changing* subset of the users. Extensions to this work are found in [2] and [3]. The trade-off between transmission length in broadcast encryption and the number of keys per user is studied in [8].

Pay TV is a specific application of broadcast encryption in which the set to which the secret is to be broadcast is associated with one of a small number of services. Therefore, focusing on Pay TV allows one to construct more efficient protocols than by adopting a generic broadcast encryption protocol to Pay TV. Another difference is that in the case of Pay TV, legitimate users may volunteer to help users in the non-privileged set learn the secret. Therefore techniques such as tamper-proof decoding boxes and traitor tracing are used.

Pay TV schemes have been proposed by [12] and [10]. The former work describes schemes which allow the broadcaster to offer a hierarchy of packages to the users. The latter has focused on unsubscription process being totally transparent to the users. To achieve this they use techniques that associate multiple decryption keys with one encryption key.

Schemes for controlling the number of users who are not entitled to the broadcast but can still receive it can be found in [1].

Some of the literature deals with single-stream schemes. We note that such a scheme can not be trivially extended to an m-stream case since this would involve an m-fold increase in the users' secure as well as non-secure keys.

Traitor tracing has been studied in [5] and [11]. These schemes work by distributing some subset of a master set of keys to every authorized user; if a pirate constructs a decoding box using information from a threshold number of users, by observing the behavior of the box on various inputs (and *without* examining its internals), it is possible to identify at least one user who co-operated with the pirate. These protocols need to be combined with a broadcast encryption protocol to obtain a Pay TV scheme.

More recently Boneh and Franklin ([4]) have proposed a deterministic public-key traitor tracing scheme which identifies all traitors. Their scheme has a computational complexity of $O(k)$ modular exponentiations. This constitutes a significant improvement in efficiency over previous traitor tracing schemes.

1.2 Contributions

In this paper we consider the broadcast to be divided into *billing periods*. All subscription and unsubscription happens only at the end of each billing period. This closely models actual practice. It has the further advantage that the revocation problem no longer exists: the key re-transmission associated with unsubscription gets amortized over the set of users. Our model differs from that of the broadcast encryption schemes in [7] because we do not require each message to be sent to an arbitrary subset of the users.

In this model we present an RSA based system in which the message expansion is by a constant factor, independent of the number of users or services. Only a constant number of keys need to be stored in the secure memory. Further, these keys need never be changed once they have been written. Each user is also given a set of non-secret keys; these alone are changed by subscription/unsubscription. Adding or removing a service can be done at any time with no overhead. Subscription can also occur at any time without involving other users.

Our scheme is also very robust. If Alice cracks her STT, she can still decrypt only the streams to which she is entitled. Therefore, the only harm that she can do is to publish information that would allow everyone to decrypt the streams that Alice is entitled to; the information she publishes, however, can be directly traced to her. Also, no collusion of crackers can compromise system security or decrypt more streams than they are together entitled to.

We consider the problem of a collusion of crackers producing a new decryption key. We extend our protocol in such a way that no collusion of crackers smaller than a given threshold can untraceably publish enough information to decode the broadcast. To achieve this, we make use of the following construction:

Construct n $(t+1)$*-tuples of integers such that given any linear combination of any* $s < t$ *of the tuples, it is possible to efficiently identify (with high probability) the exact set of tuples that were used.*

Our scheme shares certain similarities with that of [4] (which is based on the hardness of the discrete log problem). In both schemes the computational complexity and the message expansion factor are $O(t)$, where t is minimum number of STTs to be cracked to circumvent the scheme. The main advantage of our scheme is that it works with multiple streams (it uses a common RSA modulus for all streams). Thus, the total size of the keys that each STT needs to store becomes $O(m+t)$, instead of $O(mt)$.

Finally we discuss several practical issues that arise in the implementation of our scheme.

2 The new Pay TV scheme

Our scheme is based on RSA. In the next subsection, we describe a basic protocol to present the essentials of our scheme in a simple setting. We demonstrate the security of this protocol against a single cracker and show how a collusion of three crackers is enough to compromise system security. In protocol P2, this weakness is removed; however, it is possible for a collusion of three crackers to untraceably publish information that would enable everyone to decode the broadcast. This motivates the cracker identification algorithm of section 4.

2.1 Basic protocol P1

We represent the entitlements of the users as a $m \times n$ matrix $Subsc$ where $Subsc[i, j]$ is 1 if U_j is entitled to S_i. Each user U_j has a set of secure keys e_{rj} and a set of non-secure keys, one for each service S_i that she is entitled to.

Our scheme consists algorithms *Setup*, *AddUser*, *AddStream*, *Broadcast*, *Receive*, *Subscribe*, and *Unsubscribe*.

Algorithm Setup1

Bob selects two large primes p and q and sets $N = pq$ as in RSA. He selects two decryption keys d_1, d_2 such that $0 \leq d_1, d_2 < \phi(N)$ and d_2 is coprime to $\phi(N)$. Bob makes N public while p, q, d_1 and d_2 are kept secret.

Algorithm AddStream1
To add a new service S_i, Bob randomly selects $g_i \in Z_N^*$ such that g_i has high order modulo N. $Subsc[i,j]$ is set to 0 for each j. g_i is kept secret.

Algorithm AddUser1
To add a new user U_j, Bob randomly chooses e_{1j}, $0 < e_{1j} < \phi(N)$ and sets e_{2j} to the smallest positive residue of $(1 - d_1 e_{1j}) d_2^{-1}$ modulo $\phi(N)$. U_j is given an STT whose secure key is the pair (e_{1j}, e_{2j}). The secure chip of the STT allows raising an arbitrary input to the power of e_{1j} or e_{2j} modulo N. [3] $Subsc[i,j]$ is set to 0 for each i.

Algorithm Subscribe1
When U_j subscribes to S_i, Bob gives her the non-secure key ${g_i}^{e_{1j}}$ and sets $Subsc[i,j]$ to 1.

Algorithm Unsubscribe1
When U_j unsubscribes from S_i, Bob sets $Subsc[i,j]$ to 0, picks a new value for g_i as in Algorithm AddStream1, and sends ${g_i}^{e_{1j}}$ to each user U_j having $Subsc[i,j] = 1$.

Algorithm Broadcast1
To broadcast a message M belonging to the stream S_i, Bob picks a random x coprime to $\phi(N)$ and sends the ciphertext $C = (x, C_1, C_2)$ where $C_1 = M^{d_1} g_i^x$ and $C_2 = M^{d_2}$ over the broadcast channel.

Algorithm Receive1
To decrypt the ciphertext $C = (x, C_1, C_2)$ belonging to the stream S_i, U_j computes $C_1^{e_{1j}} C_2^{e_{2j}} / (g_i^{e_{1j}})^x$.

It is easy to see that the Algorithm Receive1 works correctly: If $Subsc[i,j] = 0$, U_j does not know ${g_i}^{e_{1j}}$ and hence can not execute Receive1 for service S_i. If $Subsc[i,j] = 1$, $C_1^{e_{1j}} C_2^{e_{2j}} / (g_i^{e_{1j}})^x = (M^{d_1} g_i^x)^{e_{1j}} (M^{d_2})^{e_{2j}} / g_i^{x e_{1j}} = M^{d_1 e_{1j} + d_2 e_{2j}} = M$.

The last equality holds because e_{1j} and e_{2j} are chosen in such a way that $d_1 e_{1j} + d_2 e_{2j} \equiv 1 (mod\ \phi(N))$.

2.2 Security

We consider two types of adversaries: *Honest but curious* or *Semi-honest* users are willing to deviate from the protocol and share information with other users about their (non-secure) keys. However, they will not tamper with their STTs. Malicious users, in addition to this, will also actively crack their STTs and discover their secure keys. An important assumption regarding security is that the algorithms used are publicly specified and only the keys are secret, even though the algorithms to be executed by the users are implemented in a secure chip and hence not directly accessible.

It is obvious that a collusion of semi-honest users can not crack the system: ${g_i}^{e_{1j}}$ gives no information because e_{1j} are not known to the users. Also, a semi-honest user U_j can not decrypt a message M belonging to a channel i to which

[3] Henceforth, we omit the modulus N when it is obvious.

she is not entitled: She can compute $C_1^{e_{1j}}C_2^{e_{2j}} = Mg_i^{xe_{1j}}$, but can not obtain M from this as she does not know $g_i^{e_{1j}}$. Further U_j cannot get any information about a combination of two messages M_1 and M_2 in stream i: she can compute $M_1 g_i^{x_1 e_{1j}}$ and $M_2 g_i^{x_2 e_{1j}}$, but since x_1 and x_2 are different she does not get any information about M_1/M_2.

Security against a single cracker: If U_j cracks her STT, she knows e_{1j} and e_{2j} in the equation $d_1 e_{1j} + d_2 e_{2j} \equiv 1 \ (mod\ \phi(N))$. However, this gives her no new information since she does not know $\phi(N)$. U_j can publish the triple $(e_{1j}, e_{2j}, g_i^{e_{1j}})$ enabling everyone to decrypt stream S_i, but she can not do this without revealing her identity to the broadcaster. Publishing g_i would enable everyone possessing an STT to decrypt stream S_i, but she can not get g_i from $g_i^{e_{1j}}$ since g_i has high order.

2.3 Weaknesses

Protocol P1 is not secure against the collusion of three crackers. To see this, we eliminate d_1 and d_2 from the equations $d_1 e_{11} + d_2 e_{21} \equiv d_1 e_{12} + d_2 e_{22} \equiv d_1 e_{13} + d_2 e_{23} (mod\ \phi(N))$ to get

$$(e_{12}e_{21} - e_{11}e_{22})(e_{12} - e_{13}) - (e_{12}e_{23} - e_{13}e_{22})(e_{12} - e_{11}) \equiv 0 \ (mod\ \phi(N))$$

If U_1, U_2 and U_3 all crack their STTs, then they can compute the left hand side of the above congruence. The left hand side is a multiple of $\phi(N)$. The chance that this multiple is 0 is negligible, since e_{1j} were randomly chosen. Note that if Oscar knows some (nonzero) multiple of $\phi(N)$, he can invert modulo $\phi(N)$, and hence the system is irreparably broken.

A further weakness is as follows: If U_1 and U_2 crack their STTs, there is a non-negligible probability that e_{11} and e_{12} are coprime. If that is the case, there are integers u_1 and u_2 such that $u_1 e_{11} + u_2 e_{12} = 1$. Therefore, $g_i = (g_i^{e_{11}})^{u_1}(g_i^{e_{12}})^{u_2}$ can be obtained for all i having $Subsc[i, j_1] = Subsc[i, j_2] = 1$.

3 A cracker proof protocol

In this section we extend protocol P1 such that system security can not be compromised by cracking STTs.

3.1 Protocol P2

Algorithm Setup2

Bob chooses N, p, q as in Algorithm Setup1. Let l be the number of bits of N. Bob selects a random l bit number R such that $R\phi(N)+1$ has a divisor d of roughly l bits in length. Bob also selects $2l$ bit numbers d_1, d_2, d_3 which are divisible by d such that $\frac{d_2}{d}$ and $\frac{d_3}{d}$ are coprime.

Algorithm AddUser2

Bob chooses e_{1j} randomly such that it is even. e_{2j} and e_{3j} are chosen in order that $\sum_{r=1}^{3} e_{rj} d_r = R\phi(N)+1$. (This is possible because $\frac{d_2}{d}$ and $\frac{d_3}{d}$ are coprime.) U_j's secure keys are e_{1j}, e_{2j}, e_{3j}.

Algorithm Broadcast2

To broadcast a message M belonging to the stream S_i, Bob picks a random x coprime to $\phi(N)$ and sends the ciphertext $C = (x, C_1, C_2, C_3)$ where $C_1 = M^{d_1} g_i^x$, $C_2 = M^{d_2}$, $C_3 = M^{d_3}$.

Algorithm Receive2

To decrypt the ciphertext $C = (x, C_1, C_2, C_3)$ belonging to the stream S_i, U_j computes
$(\prod_{r=1}^{3} C_r^{e_{rj}})/(g_i^{e_{1j}})^x$.

The other algorithms are the same as the corresponding algorithms for Protocol P1.

Algorithm Receive2 is correct because of the identity $(\prod_{r=1}^{3} C_r^{e_{rj}})/g_i^{xe_{1j}} = M^{\sum_{r=1}^{3} e_{rj} d_r} = M^{R\phi(N)+1} = M$.

3.2 Security

The results pertaining to semi-honest users proved for the previous protocol carry over to this one, too. We proceed to show how this protocol is secure against a collusion of crackers. We need to show that a collusion of crackers can not break the system, i.e, find $\phi(N)$ or allow semi-honest users to decrypt streams to which they are not entitled, i.e, find g_i.

Claim. *No collusion of crackers can discover (a multiple of) $\phi(N)$.*

If $U_1, U_2, \ldots U_s$ crack their STTs, then they know the values of e_{rj} in the equation $\sum_{r=1}^{3} d_r e_{r1} = \sum_{r=1}^{3} d_r e_{r2} = \ldots \sum_{r=1}^{3} d_r e_{rs}$. This equation is not a congruence modulo $\phi(N)$ but instead an exact equality. This fact is the key to the security of the protocol. For any integer x, the tuple $(x\frac{d_1}{d}, x\frac{d_2}{d}, x\frac{d_3}{d})$ in place of (d_1, d_2, d_3) also satisfies the above equation. Therefore, while a collusion of crackers might be able to find $(\frac{d_1}{d}, \frac{d_2}{d}, \frac{d_3}{d})$, and hence $\frac{R\phi(N)+1}{d}$, finding $R\phi(N)+1$ itself would involve guessing d and is hence infeasible. [4] □

As a consequence, an attack similar to the one in section 2.3 would fail because the resulting multiple of $\phi(N)$ would in fact be 0.

Claim. *No collusion of crackers can find g_i.*

Suppose that $U_{j_1}, U_{j_2}, \ldots U_{j_s}$ all crack their STTs, and are all subscribed to the service S_i. It is clear that they can compute g_i^e for exactly those e which are an integral linear combination of the $e_{j_1}, e_{j_2}, \ldots e_{j_s}$. Since all the e_j are even, no odd power of g_i can be computed. In particular, g_i cannot be found. Note that publishing g_i^2 will enable any cracker to decrypt S_i. However, knowledge of g_i^2 is of no use to an honest user U_j because she does not know e_{1j}. □

---

[4] For guessing d to be infeasible, it is enough for d to be a k-bit number, where k is a security parameter. However, this will not work if k is smaller than $\frac{l}{2}$: If an attacker knows $\frac{R\phi(N)+1}{d}$, then $\frac{R\phi(N)+1}{Nd}$ gives her a very good approximation to $\frac{R}{d}$ since the first $\frac{l}{2}$ bits of N and $\phi(N)$ are the same. This approximation is sufficient to find both R and d since R and d are coprime.

3.3 Weakness

Cracking 3 STTs gives 2 homogeneous linear equations in d_1, d_2, d_3. From this one can obtain the ratios $d_i/d_j (1 \leq i, j \leq 3)$. Once these are known, it is easy to enumerate *all* tuples (e_1, e_2, e_3) such that $\sum_{r=1}^{3} d_r e_r = R\phi(N) + 1$. In addition, if g_i^2 is known $g_i^{e_1}$ is known as well. Publishing $(e_1, e_2, e_3, g_i^{e_1})$ allows *anyone* to decrypt stream S_i, not just users possessing an STT. Further, this information is not traceable to any of the cracked STTs.

On the other hand, if only 2 STTs are cracked, not all possible tuples (e_1, e_2, e_3) can be generated, and thus the two cracked STTs can be identified (with high probability).[5] If we generalize the above protocol so that there are α secret keys, $d_1, d_2, \ldots d_\alpha$, and $\beta < \alpha$ STTs are cracked, then the crackers can be identified. However, the naive way of doing this has a time complexity of $\theta(n^\beta)$ where n is the number of users. What we need to do is to use e_{rj} of some special form such that the cracker identification algorithm becomes efficient.

4 Cracker identification

Our technique for cracker identification is based on the following construction:

Construct n $(t+1)$-tuples of integers $\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_n}$ such that given any linear combination of any $s < t$ of the vectors, it is possible to efficiently identify (with high probability) the exact set of vectors that were used.

In the following, k is a security parameter and δ is a tolerance bound such that the probability of our algorithm producing incorrect output is less than δ.

Vector generation algorithm

- *Constraint generation:* First we generate $et \, log \frac{n}{\delta}$ constraints, in $h = e \, log \frac{n}{\delta}$ groups of t constraints each.[6] The constraint $\gamma = (\mu_0, \mu_1, \mu_2, \ldots \mu_t, P)$ represents the equation $\sum_{i=0}^{t} \mu_i x_i \equiv 0 (mod\ P)$
  For each i, $1 \leq i \leq h$, we generate the i^{th} group of constraints as follows: Select a random k bit prime P_i and a matrix $\mathbf{M_i}$ with t rows and $t+1$ columns whose entries are uniformly selected modulo P_i. This gives t linear constraints on the (column) vector $\mathbf{x}$, one corresponding to each row of $\mathbf{M_i}$ in the equation $\mathbf{M_i x} \equiv \mathbf{0} (mod\ P_i)$.
- To generate a new vector $\mathbf{x} = (x_0, x_1, x_2, \ldots x_t)$, we select each of the constraints with probability $1 - \frac{1}{t}$; $\mathbf{x}$ is constructed so as to satisfy all the selected constraints.
  For each i, $1 \leq i \leq h$, let $r \leq t$ be the number of constraints in the i^{th} group of constraints that are selected. Fix $x_0, x_1, \ldots x_{t-r}$ randomly modulo P_i and solve (uniquely) for the remaining r x_j's modulo P_i using the r linear equations. (Since the coefficients of the linear equations were randomly generated and P_i is large, the probability that there is no solution is negligible.)

[5] The probability that the same tuple (e_1, e_2, e_3) can be generated from 2 different pairs of cracked STTs is negligibly small.

[6] Here *log* refers to the natural logarithm.

Thus the values of $x_0, x_1, \ldots x_t$ modulo each of the primes P_i are obtained; these are combined using the Chinese Remainder Theorem to get $\mathbf{x} = (x_0, x_1, x_2, \ldots x_t)$.

Identification algorithm

Input: A vector $\mathbf{y}$ that is a linear combination of some $s < t$ of the $\mathbf{x_j}$.

- Set the "suspect set" S to $\{\mathbf{x_1}, \mathbf{x_2}, \ldots \mathbf{x_n}\}$.
- For each constraint γ that is satisfied by $\mathbf{y}$
  - For each $\mathbf{x_j} \in S$
    - If $\mathbf{x_j}$ does not satisfy γ, remove it from S.
- Output S.

Correctness

In the following we refer to the vectors used in the given linear combination as "guilty" and the other vectors as "innocent".

We will prove that at the end of the identification algorithm, the probability that any $\mathbf{x_j} \in S$ is innocent is negligible.

We observe that if all the guilty vectors satisfy a constraint γ, then so does $\mathbf{y}$ (since the constraints are linear, and $\mathbf{y}$ is a linear combination of the guilty vectors). Conversely, suppose that at least one of the guilty vectors $\mathbf{x} = (x_0, x_1, x_2, \ldots x_t)$ does not satisfy $\gamma = (\mu_0, \mu_1, \mu_2, \ldots \mu_t, P)$. Then $\sum_{i=0}^{t} \mu_i x_i$ is uniformly distributed modulo P and therefore $\sum_{i=0}^{t} \mu_i y_i$ is also uniformly distributed modulo P. The probability that this value is zero is negligibly small, since P is large, and hence $\mathbf{y}$ does not satisfy γ with very high probability.

Let s be the number of guilty vectors and let $\mathbf{x}$ be one of the vectors that is innocent. For each constraint γ, the probability that all of the guilty vectors satisfy γ is $(1-\frac{1}{t})^s$ and the probability that $\mathbf{x}$ does not satisfy γ is $\frac{1}{t}$. Therefore the probability that $\mathbf{x}$ is eliminated by γ is $\frac{1}{t}(1-\frac{1}{t})^s \geq \frac{1}{t}(1-\frac{1}{t})^{t-1} > \frac{1}{et}$. There are $et\ log\frac{n}{\delta}$ constraints; the probability that all of them fail to eliminate $\mathbf{x}$ is $(1-\frac{1}{et})^{et\ log\frac{n}{\delta}} \approx \frac{\delta}{n}$. Hence, the probability that at least one of the innocent vectors is declared guilty is at most $1-(1-\frac{\delta}{n})^n < \delta$, establishing the claim.

We are now ready to describe Protocol P3.

4.1 Protocol P3

In the following, let t be a threshold such that we want at least t STTs to be cracked in order for the crackers to be unidentifiable and δ the error probability we are willing to tolerate.

Algorithm Setup3

Bob selects $N = pq, d_1, d_2, d_3, R$ as in Algorithm Setup2. He further selects $d_r \in_R \{1, 2, ..., \phi(N)\}$ for $4 \leq r \leq 4+t$. Bob also runs the constraint generation part of the vector generation algorithm described above to obtain $et\ log(n/\delta)$ constraints.

Algorithm AddUser3

U_j's secure key is a $(t+4)$-tuple $(e_1, e_2, \ldots e_{t+4})$. Of these, e_1 is selected randomly so that it is even. $e_4, e_5, \ldots e_t$ are generated by executing the vector generation algorithm above, using the constraints generated in Algorithm Setup3. Finally e_{2j} and e_{3j} are chosen so that the equation $\sum_{r=1}^{t+4} e_{rj} d_r = R\phi(N) + 1$ is satisfied.

The other algorithms are essentially the same as in Protocol P2, the difference being that the ciphertext that is broadcast is a $(t+4)$-tuple instead of a 3 tuple.

4.2 Security

The proof of system security under a collusion of arbitrary many crackers is similar to the proof for protocol P2, and we omit it. The cracker identification property is a direct consequence of the correctness of the identification algorithm above: as long as less than t STTs are cracked, the only way to generate a new decryption vector is as a rational linear combination of the cracked vectors (with the sum of the coefficients being unity). If $(e_1, e_2, \ldots e_{t+4})$ is the new vector, then by running the identification algorithm with $(e_4, e_5, \ldots e_{t+4})$ as input, we can identify the cracked STTs with high probability.

We note that the identification algorithm depends on certain linear combinations being uniformly distributed modulo P_i. Therefore, to ensure security it is necessary for the primes P_i and the matrices $\mathbf{M_i}$ to be kept secret by the broadcaster.

4.3 Size of the users' secure keys

It is essential that choosing $\mathbf{e} = (e_4, e_5, \ldots e_{t+4})$ satisfying all the constraints does not increase its size unacceptably. To estimate the size, we recall that in the Chinese Remainder Theorem, the solution is unique modulo the product of the individual moduli. Therefore, the size of the solution is roughly the sum of the sizes of the moduli. In the constraint generation algorithm, satisfying each of the $e\ log\frac{n}{\delta}$ groups of constraints fixes each component of the vector modulo a k bit prime; therefore each e_r has $ek\ log\frac{n}{\delta}$ bits. Importantly, this number is independent of t and only weakly dependent on n. For typical values of n, δ and k: 10^6, 10^{-3} and 128 respectively, we find that this number is about 900 bytes, which is reasonable.

5 Issues in practice

In this section, we discuss several issues pertaining to the implementation of a Pay TV system using our scheme.

In a Pay TV system, subscription and unsubscription can happen only at the end of a fixed time unit called the *billing period*. This is very important because it amortizes the communication overhead of the unsubscription algorithm. At the start of each billing period, the broadcaster simply sends to each user her non-secure keys for each service that she is entitled to. The billing period is typically a week or a month.

We have assumed all messages to belong to some service. In practice, a service is either a “channel”, or a single pay-per-view event (such as movies). We might have to use a smaller billing period for the latter category of service than for the former. We also note that unsubscription is not meaningful for pay-per-view events.

The *power-on latency* is the maximum time between Alice switching on her STT and the start of decoding of broadcast. The broadcast is divided into *sessions*, each of which lasts for a time equal to the power-on latency. During each session, the broadcast is encrypted using some *symmetric key* algorithm; the protocols we have proposed are used only for the broadcast of these symmetric keys. This achieves the following:

– Content sharing between users remains unfeasible.
– The bandwidth inefficiency of broadcast protocols is amortized.
– Costly public key operations need to be performed only once per session.

5.1 Key distribution

The users' non-secret keys can be transmitted in-band with the broadcast. However, Alice's STT must be switched on during key broadcast in order for her to be able to receive her keys. In the following we give several techniques to address this problem.

1. **Physical transfer**. A simple approach is to physically transfer to the user at the start of each billing period a removable chip containing the user's non-secure keys that can be plugged into and read by the STT.
2. **Uplink**. If an STT has uplink capability [6], it would be easy to request a key from the broadcast center whenever one is required. For example, the STT sends i to the broadcast center whenever the user switches channels to service i, upon which the center broadcasts that user's non-secure key for service i.
3. **Synchronization**. The broadcaster transmits all keys of all users exactly once (say) every day. Then the STT can easily be equipped with a clock that “wakes up” the STT once every day and downloads the users' keys. Of course the STT's clock needs to be high precision and synchronized with the broadcaster's clock.
4. **Caching**. Caching of the user's non-secure keys by the STT can be used in combination with other strategies. The STT caches the non-secure keys in its non-volatile memory even when switched off.

6 Conclusion

We have considered a model for Pay TV in which the target set of users is not arbitrary but is associated with one of a small number of services or streams.

We have also assumed that the users' decoding boxes are provided with a secure chip and that entitlements of users are fixed for each billing period. Under these assumptions, we are able to achieve full flexibility with a constant data redundancy and a constant number of secure keys per user. We believe that this model is the most appropriate one in practice.

Our cracker identification algorithm is motivated by the need to prevent a collusion of crackers from untraceably publishing a decryption key. It would be interesting to see if our construction has other applications apart from Pay TV. We suggest a value for t of between 20 and 50 to realize a robust system (this value is sufficient because of the use of tamper proof decoding boxes). There appears to be scope for further work in our this algorithm - in particular, it might be possible to reduce the message expansion factor. Another possibility is to construct a scheme under our model in which it is impossible for crackers to construct a new decryption key.

Acknowledgements. The authors would like to thank K. Srinathan for useful discussions and suggestions.

References

1. M. Abdalla, Y. Shavitt, and A. Wool. Towards making broadcast encryption practical. In *Financial Cryptography*, volume 1648 of *LNCS*, pages 140–157, 1999.
2. C. Blundo and A. Cresti. Space requirements for broadcast encryption. In *Advances in Cryptology, Eurocrypt '94*, volume 950 of *LNCS*, pages 287–298, 1994.
3. C. Blundo, L. A. Frotta Mattos, and D. R. Stinson. Trade-offs between communication and storage in unconditionally secure schemes for broadcast encryption and interactive key distribution. In *Advances in Cryptography, CRYPTO '96*, LNCS, pages 387–400, 1996.
4. D. Boneh and M. Franklin. An efficient public key traitor tracing scheme. In *Proc. 19th International Advances in Cryptology Conference – CRYPTO '99*, pages 338–353, 1999.
5. B. Chor, A. Fiat, and M. Naor. Tracing traitors. In *Advances in cryptology - CRYPTO'94*, volume 839 of *LNCS*, pages 257–270, 1994.
6. J. L. Cohen, M. H. Etzel, D. W. Faucher, and D. N. Heer. Security for broadband digital networks. In *Communications Technology*, pages 58–69, 1995.
7. A. Fiat and M. Naor. Broadcast encryption. In *Advances in Cryptology, CRYPTO '93*, volume 773 of *LNCS*, 1994.
8. M. Luby and J. Staddon. Combinatorial bounds for broadcast encryption. In *Advances in Cryptology, Eurocrypt '98*, volume 1403 of *LNCS*, pages 512–526, 1998.
9. J. McCormac. European scrambling systems 5. *Waterford University Press, Waterford, Ireland*, 1996.
10. Y. Mu and V. Varadharajan. Robust and secure broadcast. In *Progress In Cryptology - Indocrypt 2001*, volume 2247 of *LNCS*, pages 223–231, 2001.
11. M. Naor and B. Pinkas. Threshold traitor tracing. In *Advances in cryptology - CRYPTO'98*, volume 1462 of *Lecture Notes in Computer Science (LNCS)*, pages 502–517, 1998.
12. A. Wool. Key management for encrypted broadcast. In *5th ACM conference on Computer and Communications Security*, pages 7–16, 1998.