Proc. European Symposium on Research in Computer Security (ESORICS), LNCS v. 4734, p. 311-326, Springer, 2007.

Improved Anonymous Timed-Release Encryption

K. Chalkias, D. Hristu-Varsakelis, and G. Stephanides

Computational Systems and Software Engineering Laboratory,
Department of Applied Informatics,
University of Macedonia,
Thessaloniki, Greece
chalkias@java.uom.gr, dcv@uom.gr, steph@uom.gr

Abstract. We revisit the problem of "sending information into the future" by proposing an anonymous, non-interactive, server-based Timed-Release Encryption (TRE) protocol. We improve upon recent approaches by Blake and Chan, Hwang et al., and Cathalo et al., by reducing the number of bilinear pairings that users must compute, and by enabling additional pre-computations. Our solution compares favorably with existing schemes in terms of computational efficiency, communication cost and memory requirements, and is secure in the random oracle model.

Key words: timed-release encryption, bilinear pairings, pre-computations, multiple receivers

1 Introduction

Timed-Release Encryption (TRE) is a special field of cryptography that studies the problem of "sending information into the future", i.e., encrypting a message so that it cannot be decrypted by anyone, including the designated recipients, until a future time chosen by the sender. This problem was originally posed in [22] and then explored further in [27].

There are numerous applications in distributed computing and networks that require TRE, such as sealed-bid auctions in which one seeks to provide assurance that bids cannot be opened by anyone (including the auction board) before the end of the bidding period [27], payment schedules, and key escrow. Other examples include the release of important documents (e.g., memoirs, wills, press articles) [27]; e-voting which requires delayed opening of votes [26]; internet programming contests, where participating teams cannot access the challenge problem before the beginning of the contest [3]; delayed verification of signed documents, such as lottery [29] and check cashing, contract signing [14], and verification of online card game results [13].

Solutions to the TRE problem follow one of two basic techniques. The first is based on so-called time-lock puzzles [24, 27, 1], [8, 18, 19], where the receiver must perform non-stop, non-parallelizable computation in order to recover a message. Although this approach does not involve a trusted third party, it puts immense computational overhead on the receiver, it makes encryption dependent on the receiver's CPU speed, and does not guarantee that the message will be retrieved at a precise moment in the future.

To sidestep these problems, a second approach developed, based on the use of trusted time-servers. The server-based approach relieves the receiver from performing non-stop computation, and can specify the decryption time with precision. The trade-off is the required interaction between the (trusted) server and the users. To ensure security, scalability and anonymity, a time-server should have as little interaction as possible with the users. Ideally, this server should not be involved in the encryption or decryption process, and should only provide a common time reference by periodically releasing unforgeable, time-embedded information, which will be used to decrypt timed-release ciphertexts. The vast majority of the early attempts at TRE did not satisfy this last requirement.

In the early nineties, [22] proposed a system where the server is a trusted escrow agent, storing messages and releasing them to the designated recipients at specified times. That approach did not provide anonymity, and the server knew the content of the message and its release time. Another approach, combining symmetric and asymmetric encryption, was proposed by [27]; it required active interaction between senders and server, and thus guaranteed anonymity only for receivers. To provide sender anonymity, [16] proposed a solution in which interaction was needed between the server and the receiver only. In that scheme, the receiver's anonymity is compromised because server and receiver must engage in a conditional oblivious transfer protocol.

Recently, there have been attempts to use bilinear pairing-based schemes for TRE. The work in [7] mentioned TRE as one of the possible applications of Identity Based Encryption (IBE), and [25] implemented that idea. Although IBE is certificate-less, their scheme was not scalable, because the server must generate and transmit to each receiver a unique secret key, corresponding to a specific time instant. Other TRE approaches allow the recovery of past time-specific trapdoors from a current trapdoor. Among them are the protocol in [6] which uses the tree-like structure of [9] backwards, and [13] which uses a hash chain for the construction of the trapdoors. In both cases, the root of the tree-like structure and the hash chain, respectively, correspond to the "last" time instant for which a trapdoor can be produced, which implies an upper bound on the lifetime of their systems.

The first attempt at scalable, server-passive, user-anonymous TRE was due to Blake and Chan [3], as recently as three years ago. The breakthrough of that pairing-based approach is that the server does not interact with either the sender or the receiver; its sole responsibility is to provide a common time reference by releasing time-specific *universal* (i.e., receiver-independent) trapdoors. In fact, the server need not even be aware of the existence of a sender or receiver; hence, user anonymity and message privacy are guaranteed. That work has formed the basis for the majority of modern TRE schemes [26]. Hwang, Yum and Lee [20] proposed a user-anonymous TRE scheme that had similarities with that of Blake and Chan but could also provide a pre-open capability, meaning that the sender can decide to allow "early" decryption by issuing to the receiver a secondary trapdoor (different from the one to be given later by the time server). Another efficient anonymous TRE scheme that can take advantage of pre-computations[1] — and forms the point of departure for this work — was proposed by Cathalo, Libert and Quisquater [10].

The contribution of this paper is to combine the desirable properties of existing TRE schemes in order to create a new, efficient, server-passive, provably-secure, pairing-based, user-Anonymous TRE protocol (termed AnTRE). A key advantage of our protocol is its simple public key format which enables pairing pre-computations and leads to significant computational savings. Recently-proposed TRE schemes either use a more complex public key format [3, 10], thus requiring pairing-based verification of users' public keys, or lack support for pre-computations [3, 20]. In terms of computational efficiency, our protocol is more than twice as fast as the best existing approaches when sending to *unknown* receivers, while also comparing favorably when sending the same information to multiple (more than two) receivers. Moreover, under our approach, the amount of data (public keys and pre-computed values) to be stored in the sender's machine is very small compared to that of other schemes [10]. In our scheme, as in [3], the time-server does not need to store any of the required trapdoors, since it can generate them on demand, using its own private key. Moreover, the time-server has a passive role and its sole responsibility is to periodically publish time-specific trapdoors, avoiding any interaction at all with either the sender or the receiver, thus providing user-anonymity.

The remainder of this paper is organized as follows. In Section 2 we define our model for anonymous TRE. In Section 3 we describe the proposed protocol and its security properties. Section 4 compares our protocol with three of the best-known TRE approaches in terms of computational efficiency and memory usage.

2 TRE Model

2.1 Modeling a user-anonymous TRE scheme

There are two types of entities involved in a general TRE scheme: a trusted time-server that periodically issues authenticated time-specific trapdoora, and users that act either as senders or as receivers. In this work, we assume that ciphertexts always contain information about their release-time. We will let $T \in \{0,1\}^\tau$, $\tau \in \mathbb{N}$ denote time. For instance, T could indicate the τ-bit string representation of a specific time instant (e.g. T = "10:00AM, October 10, 2007 GMT from the Denver Atomic Clock used in Global Positioning System (GPS)"). Based on these assumptions, an anonymous TRE scheme (AnTRE) consists of a quintuple of polynomial-time algorithms:

AnTRE.Setup: *It is run by the time-server; it takes as input a security parameter 1^k, and returns system parameters params that include the server's public key, s_{pub}, for which the corresponding private key, s_{pr}, is securely stored, to be used in the generation of all time-specific trapdoors.*

AnTRE.ReleaseT *is run by the time-server; given the server's private key s_{pr}, and*

[1] By pre-computation we mean that some of the calculations necessary to run a protocol can be performed off-line, prior to specifying a message or a receiver

a time $T \in \{0,1\}^\tau$, it returns a verifiable time trapdoor s_T.

AnTRE.KeyGen: *This is a key generation algorithm run by a user. Its inputs are a security parameter 1^k and the system parameters params; it returns a private/public key pair (u_{pr}, u_{pub}).*

AnTRE.Enc *is run by the sender of a message m. It takes as inputs the system parameters params, the message m, the release time $T \in \{0,1\}^\tau$, and the public keys of both receiver and time server (u_{pub} and s_{pub} respectively), and returns a ciphertext C that the recipient must be unable to decrypt before being given the trapdoor that is to be published by the server at a later time.*

AnTRE.Dec: *This is a decryption algorithm that takes as inputs a ciphertext (C,T), the system parameters params, a private key u_{pr}, and a time-specific trapdoor s_T, and returns a plaintext m or an error message.*

2.2 Adversarial Models

We distinguish between two kinds of adversaries. One is a so-called *outside* attacker that models a "curious" time-server (i.e., one that knows the time-specific trapdoor for any time) trying to decrypt a ciphertext he may have intercepted. This attacker is challenged on a random user's public key for which he is equipped with a decryption oracle. A second adversary is an *inside* attacker that models a malicious, "impatient" receiver, trying to decrypt a ciphertext before its designated release time. In that case, the adversary has knowledge of the receiver's private key, but does not have any information about the time-server's private key and the specific trapdoor that will be published at the appointed time. We assume that an inside attacker can freely choose the public key on which he is challenged in a "find-then-guess" game, to be made precise shortly. The adversary can also access a release-time oracle returning trapdoors for any time period, except the one for which the challenge ciphertext is computed. Furthermore, in a chosen-ciphertext scenario, he is given access to an oracle decrypting other ciphertexts than the challenge. In the AnTRE model, this adversary is called chosen-time period and ciphertext attacker (CTCA).

Definition 1 ([10]). *Let $\mathcal{A}$ be an outside adversary. An AnTRE scheme is said to be secure against chosen-ciphertext attacks (IND-CCA secure) if no polynomially bounded adversary $\mathcal{A}$ has a non-negligible advantage in the following game:*

1. A challenger, CH, takes a security parameter 1^k and runs AnTRE.Setup(1^k) and AnTRE.KeyGen to obtain a list of public parameters, params, and a key pair (u_{pr}, u_{pub}). The public key u_{pub}, params, and the server's private key, s_{pr}, are given to $\mathcal{A}$, while the private key, u_{pr}, is kept secret.

2. $\mathcal{A}$ has access to a decryption oracle, AnTRE.Decrypt(.), which given a ciphertext (C, T) and the time-specific trapdoor s_T (always computable by anyone who knows s_{pr}), returns the decryption of C using the private key u_{pr}. At some point, $\mathcal{A}$ outputs *two equal-length messages m_0, m_1 and a challenge time-period T^*. He gets a ciphertext $(C^*, T^*) = AnTRE.Encrypt(m_b, u_{pub}, params, T^*)$, for $b \xleftarrow{R} \{0,1\}$, computed under the public key u_{pub}.*

3. $\mathcal{A}$ issues a new sequence of queries but is prohibited from asking for the decryption of the challenge for the time period T^*. He eventually outputs a bit $b^{'}$, and wins if $b^{'} = b$. His advantage is $Adv^{IND-CCA}_{AnTRE,\mathcal{A}}(\mathcal{A}) := |Pr[b^{'} = b] - 1/2|$.

Definition 2 ([10]). *Let $\mathcal{A}$ be an inside adversary. An AnTRE scheme is said to be secure against chosen-time period and ciphertext attacks (IND-CTCA secure) if no polynomially bounded adversary, $\mathcal{A}$, has a non-negligible advantage in the following game:*

1. The challenger, CH, takes the security parameter 1^k and runs AnTRE.Setup(1^k) to return the resulting public parameters params to $\mathcal{A}$. The server's public key, s_{pub}, is given to $\mathcal{A}$, while the corresponding private key, s_{pr}, is kept secret.

2. $\mathcal{A}$ has access to a release-time oracle AnTRE.ReleaseT(.) returning trapdoors s_T for any time T. $\mathcal{A}$ is also given access to a decryption oracle, AnTRE.Dec(.), which given a ciphertext C and a receiver's public key, u_{pub}, provided by $\mathcal{A}$, computes the decryption of C using s_T, but without knowing the corresponding user's private key u_{pr}. At some moment, $\mathcal{A}$ outputs messages m_0, m_1, an arbitrary public key u^*_{pub}, and a time instant T^* that has not been submitted to the AnTRE.ReleaseT oracle. He receives the challenge (C^*, T^*) =AnTRE.Enc(m_b, u^*_{pub}, params, T^*), for a hidden bit $b \xleftarrow{R} \{0,1\}$.

3. $\mathcal{A}$ issues a new sequence of release-time queries for any time instant T^* and decryption queries for any ciphertext but the challenge (C^*, T^*), for the public key u^*_{pub}. He eventually outputs a bit $b^{'}$ and wins if $b^{'} = b$. His advantage is $Adv^{IND-CTCA}_{AnTRE,\mathcal{A}}(\mathcal{A}) := |Pr[b^{'} = b] - 1/2|$.

3 Proposed Protocol

In order to construct time-specific trapdoors, we will use the short signature scheme from [4] and [30]. This scheme was initially used in the selective-ID secure IBE in [5] which was proven to be secure without random oracles. In our case, the proposed TRE protocol detailed below is based on the anonymous TRE protocol in [10], the first to make use of such signature schemes for TRE purposes. Its security proofs hold in the random oracle model [2]. In the following, we describe the proposed protocol, named AnTRE. We will sometimes refer to AnTRE as the "full" version of our protocol, in order to distinguish it from its simpler, "basic" counterpart which is used in the security proofs and is included in Appendix A.

3.1 Preliminaries

For the purposes of this work, we will require an abelian, additive finite group $\mathbb{G}_1$, of prime order q, and an abelian multiplicative group, $\mathbb{G}_2$, of the same order. For example,

$\mathbb{G}_1$ may be the group of points on an elliptic curve. We will let P denote the generator of $\mathbb{G}_1$. Also, H_1, H_2, H_3, H_4 will be four secure hash functions, with $H_1 : \{0,1\}^\tau \mapsto \mathbb{Z}_q^*$, $H_2 : \{0,1\}^n \mapsto \{0,1\}^k$, $H_3 : \mathbb{G}_2 \mapsto \mathbb{Z}_q^*$, $H_4 : \mathbb{G}_1 \mapsto \{0,1\}^{n+k_0}$, where $n, k_0 \in \mathbb{N}$. Finally, $e : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ will be a bilinear pairing, defined below.

Definition 3. *Let $\mathbb{G}_1$ be an additive cyclic group of prime order q generated by P, and $\mathbb{G}_2$ be a multiplicative cyclic group of the same order. A map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ is called a bilinear pairing if it satisfies the following properties:*

– *Bilinearity: $\hat{e}(aV, bQ) = \hat{e}(bV, aQ) = \hat{e}(abV, Q) = \hat{e}(V, abQ) = \hat{e}(V, Q)^{ab}$ for all $V, Q \in \mathbb{G}_1$ and $a, b \in Z_q^*$.*
– *Non-degeneracy: there exist $V, Q \in \mathbb{G}_1$ such that $\hat{e}(V, Q) \neq 1$.*
– *Efficiency: there exists an efficient algorithm to compute the bilinear map.*

Admissible bilinear pairings can be constructed via the Weil and Tate pairings [21]. For a detailed description of pairings and conditions under which they can be applied to elliptic curve cryptography, see [21, 28].

3.2 Full Version of AnTRE

To send a message m that will be decrypted at (or after) a pre-defined time instant T, the following protocol is to be executed (see also [12] for tabular form):

AnTRE.Setup: given security parameters k and k_0, where k_0 is polynomial in k, the setup algorithm:
1. Outputs a k-bit prime number q, two groups $\mathbb{G}_1$, $\mathbb{G}_2$ of order q, an admissible bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ and an arbitrary generator $P \in \mathbb{G}_1$.
2. Chooses the cryptographic hash functions $H_1 : \{0,1\}^\tau \mapsto \mathbb{Z}_q^*$, $H_2 : \{0,1\}^{n+k_0+\tau} \mapsto \{0,1\}^{2k}$, $H_3 : \mathbb{G}_2 \mapsto \mathbb{Z}_q^*$, $H_4 : \mathbb{G}_1 \mapsto \{0,1\}^{n+k_0+2k}$ for some $n, \tau \in \mathbb{N}$. These functions will be treated as random oracles when it comes to security considerations.
3. Generates the time-server's private key, $s \xleftarrow{R} \mathbb{Z}_q^*$, and the corresponding public key, $S = sP \in \mathbb{G}_1^*$.
4. Chooses the message space $M = \{0,1\}^n$ and the ciphertext space $C = \mathbb{G}_1 \times \mathbb{G}_1 \times \{0,1\}^{n+k_0+2k+\tau}$.
The public parameters are $params := \{k, k_0, q, \mathbb{G}_1, \mathbb{G}_2, P, S, \hat{e}, H_1, H_2, H_3, H_4, n, M, C\}$.

AnTRE.ReleaseT: given a time instant $T \in \{0,1\}^\tau$, its hash value $t = H_1(T)$, and the server's private key s, it returns the time-specific trapdoor $s_T = (s+t)^{-1}P \in \mathbb{G}_1^*$.

AnTRE.KeyGen: given *params*, it chooses a private key $b \in \mathbb{Z}_q^*$ and produces receiver's public key $B = bP \in \mathbb{G}_1^*$.

AnTRE.Enc: to encrypt $m \in \{0,1\}^n$ using the time information $T \in \{0,1\}^\tau$ and the receiver's public key B, the sender executes the following:
1. Choose $x \xleftarrow{R} \{0,1\}^{k_0}$, compute $t = H_1(T) \in \mathbb{Z}_q^*$ and $h = H_2(m||x||T) \in \{0,1\}^{2k}$ and get $r_1, r_2 \in \mathbb{Z}_q^*$, where $r_1||r_2 = \overline{h}$, where $\overline{h}$ denotes the $2k$-bit integer value of h.

2. Compute $c_1 = r_1S + r_1tP \in \mathbb{G}_1^*$ and $c_2 = r_2P \in \mathbb{G}_1^*$.
3. Compute $d = H_3(\hat{e}(P,P)^{r_1}) \in \mathbb{Z}_q^*$.
4. Compute $K = H_4(dr_2B) \in \{0,1\}^{n+k_0+2k}$ and then $c_3 = (m||x||h) \oplus K \in \{0,1\}^{n+k_0+2k}$.
The ciphertext is $C := \langle c_1, c_2, c_3, T\rangle$.

AnTRE.Dec: given $C := \langle c_1, c_2, c_3, T\rangle$, the trapdoor s_T and his private key b, the recipient computes $d = H_3(\hat{e}(c_1, s_T)) \in \mathbb{G}_1$, and the session key $K = H_4(dbc_2) \in \{0,1\}^{n+k_0+2k}$. He is then able to retrieve the message as $m||x||h = K \oplus c_3$. To verify the message, he checks whether $H_2(m||x||T) = h$.

The following two theorems (proofs are included in Appendix B) concern the security properties of AnTRE. In particular, the proposed protocol is secure against IND-CTCA and IND-CCA attackers.

Theorem 1. *Assume that a polynomial-time IND-CTCA attacker has a non-negligible advantage $\varepsilon(k)$ against AnTRE when making q_{H_i} queries to random oracles H_i $\forall i \in \{1,2,3,4\}$ and q_T time server queries. Then the q-BDHI*[2] *problem can be solved (in polynomial time) with non-negligible probability.*

Theorem 2. *Assume that a polynomial-time IND-CCA attacker has a non-negligible advantage $\varepsilon(k)$ against AnTRE when making q_{H_i} queries to random oracles H_i $\forall i \in \{1,2,3,4\}$. Then the CDH* [3] *problem can be solved (in polynomial time) with non-negligible probability.*

4 Comparisons

In this section, we compare AnTRE with three of the best-known existing approaches to non-interactive server-based anonymous TRE: the BC-TRE scheme proposed by Blake and Chan [3], HYL-TRE proposed by Hwang, Yum and Lee [20, 15], and CLQ-TRE [4] proposed by Cathalo, Libert and Quisquater [10].

4.1 Computational Efficiency

Because some of the protocols mentioned previously allow for pre-computations under certain circumstances, we distinguish between three cases of anonymous TRE: i) message transmission to *unknown* receivers, ii) transmission to *known* receivers (in which case there is no need to verify their public keys), and iii) messages sent to multiple recipients with the same release-time.

For the purposes of calculating the computing time needed to run each protocol, we will let *Pa* denote the pairing operation, *Sm* scalar multiplication in $\mathbb{G}_1$, *PSm* parallel

[2] The q-Bilinear Diffie-Hellman Inversion Problem (q-BDHI) is: given $(Q, aQ, a^2Q, ..., a^qQ) \in \mathbb{G}_1^{q+1}$, compute $\hat{e}(Q,Q)^{a^{-1}} \in \mathbb{G}_2$.
[3] The Computational Diffie-Hellman Problem (CDH) is: given $Q \in \mathbb{G}_1$, aQ, bQ for some $a, b \in \mathbb{Z}_q^*$, compute $abQ \in \mathbb{G}_1$.
[4] We note that [10] uses multiplicative notation for the groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

scalar multiplication of the form $aP+bQ$ in $\mathbb{G}_1$, Ex exponentiation in $\mathbb{G}_2$, Mtp map-to-point hashing, and Inv inversion in $\mathbb{Z}_q$. To make a fair comparison, the cost of each operation will be related to that of an elliptic curve scalar multiplication (M). Table 1 summarizes the benchmarking results using the *MIRACL* open-source library [23], considering an order-q subgroup of a supersingular elliptic curve E over F_p, where p is a 512 bit prime and q is a 160 bit prime. Pairing values belong to a finite field of 1024 bits.

Table 1. Cost of basic operations in relation to that of an elliptic curve scalar multiplication.

Operation	**Notation**	**Cost**
Bilinear Pairing	Pa	$9M$
Parallel Scalar Multiplication in $\mathbb{G}_1$	PSm	$1.2M$
Scalar Multiplication in $\mathbb{G}_1$	Sm	$1M$
Exponentiation in $\mathbb{G}_2$	Ex	$1M$
Map-To-Point	Mtp	$0.7M$
Inversion in $\mathbb{Z}_q$	Inv	$0.4M$

If we assume that $\hat{e}(P,P)$ is computed in advance and included among the public parameters, then the encryption phase of AnTRE requires the following operations: 1 *PSm* to compute c_1, 1 *Sm* for c_2, 1 Ex for d, and 1 *Sm* to compute K. That is, no pairing computations are necessary at execution time, and the total cost of the encryption phase is equivalent to $4.2M$. In the decryption phase, the recipient must perform 1 *Pa* operation to calculate the point value d, and 1 *Sm* to produce K, thus the total decryption cost is approximately $10M$. Tables 2 and 3 summarize the comparisons of computational cost for the cases of *unknown* and *known* receivers, respectively.

Table 2. Computational cost comparison of BC-TRE, HYL-TRE, CLQ-TRE, and proposed AnTRE protocol (sending to *unknown* receivers).

Protocol	**Encryption**	**Decryption**	**Total**
BC-TRE	$3Pa+2Sm+1Mtp = 29.7M$	$1Pa+1Ex = 10M$	$39.7M$
HYL-TRE	$1Pa+1PSm+2Sm+1Mtp = 12.9M$	$2Pa+1Sm = 19M$	$31.9M$
CLQ-TRE	$2Pa+1PSm+1Ex = 20.2M$	$1Pa+1PSm+1Ex = 11.2M$	$31.4M$
Proposed	$1PSm+2Sm+1Ex = 4.2M$	$1Pa+1Ex = 10M$	$14.2M$

Remarks: We note that neither BC-TRE nor HYL-TRE support pairing pre-computations, because the sender must compute a pairing that depends on the release time. Moreover,

Table 3. Computational cost comparison of BC-TRE, HYL-TRE, CLQ-TRE, and proposed AnTRE protocol (sending to *known* receivers).

Protocol	**Encryption**	**Decryption**	**Total**
BC-TRE	$1Pa + 2Sm + 1Mtp \; = \; 11.7M$	$1Pa + 1Ex \; = \; 10M$	$21.7M$
HYL-TRE	$1Pa + 1PSm + 2Sm + 1Mtp \; = \; 12.9M$	$2Pa + 1Sm \; = \; 19M$	$31.9M$
CLQ-TRE	$1PSm + 1Ex \; = \; 2.2M$	$1Pa + 1PSm + 1Ex = 11.2M$	$13.4M$
Proposed	$1PSm + 2Sm + 1Ex \; = \; 4.2M$	$1Pa + 1Ex \; = \; 10M$	$14.2M$

these two protocols require a special hash function, *map-to-point*, which is needed for mapping strings onto cyclic groups, and which is much less efficient than plain hash functions [30, 10, 23]. Furthermore, unlike BC-TRE and CLQ-TRE, AnTRE retains the same efficiency whether or not the receivers are *known* entities. It is also worth mentioning that BC-TRE and CLQ-TRE use a slightly different public key format, with users' public key consisting of two points in $\mathbb{G}_1$ instead of one (as in conventional cryptographic schemes). This means that on the first use of any public key (for transmitting to an *unknown* receiver) the sender must verify the validity of this two-point public key, to ensure that the recipient will be able to decrypt the message. Such verification is not needed in our proposed protocol or in HYL-TRE[5].

Transmitting to multiple receivers AnTRE is practical for encrypting a message to multiple receivers with the same release-time (e.g., in an Internet programming contest). In our approach, the value of d (in AnTRE.Dec) can be calculated by anyone who knows the time-specific trapdoor s_T. Also, the session key, K, depends on d, c_2 (sent in the clear), and the recipient's private key, u_{pr}. Thus, if one wishes to use AnTRE to send a message to multiple receivers, he is able to use the same random values r_1 and r_2 for all of them. In that case, the computed session key K (and thus c_3 as well) will differ from receiver to receiver; the corresponding ciphertexts will be of the form $C < c_1, c_2, c_{3.1} ... c_{3.N}, T >$. Finally, because c_1, c_2 and d are computed only once, the total encryption cost per receiver will be $\frac{3.2M}{N} + 1M$, where N is the number of receivers. Compared to CLQ-TRE [10] suitably modified for multiple receivers (it costs approximately $2.2M$ for *known* receivers), our approach is more efficient, even in the special case of *known* receivers, if the number of designated recipients is greater than two. Although BC-TRE and HYL-TRE can be modified for improved efficiency when sending to multiple receivers, neither of them can avoid the pairing computation during the encryption process (to the best of our knowledge), and thus they appear to be less efficient compared to AnTRE and CLQ-TRE.

[5] When comparing the cost of implementations of the above approaches, we did not include the cost of a group membership test for the public keys. We note, however, that the schemes [3, 10] use two points in their public keys (ours uses only one) and would thus require some additional checking.

4.2 Communication Cost

In order to compare the communication complexity of the four TRE schemes, we must take into account the bit-length of both the transmitted public keys and the ciphertext. As we have mentioned in Section 4.1, in BC-TRE and CLQ-TRE the users' public keys consist of two elliptic curve points, i.e., $u_{pub} \in \mathbb{G}_1 \times \mathbb{G}_1$, while in the proposed AnTRE and HYL-TRE protocols $u_{pub} \in \mathbb{G}_1$. Consequently, if the recipient is an *unknown* entity, the cost to download the recipient's public key from a public database for BC-TRE and CLQ-TRE is twice that of the proposed AnTRE and HYL-TRE schemes.

The ciphertext space (including transmission of time information) of each scheme is:

- IND-CCA secure BC-TRE:[6] $C_{BC-TRE} = \mathbb{G}_1 \times \{0,1\}^{n+k_0+\tau}$.
- HYL-TRE: $C_{HYL-TRE} = \mathbb{G}_1 \times \mathbb{G}_1 \times \mathbb{G}_2 \times \{0,1\}^{n+k_0+\tau}$.
- CLQ-TRE: $C_{CLQ-TRE} = \mathbb{G}_1 \times \{0,1\}^{n+k_0+\tau}$.
- Proposed AnTRE: $C_{AnTRE} = \mathbb{G}_1 \times \mathbb{G}_1 \times \{0,1\}^{n+k_0+2k+\tau}$.

BC-TRE and CLQ-TRE have a smaller ciphertext space than AnTRE (about 1 elliptic curve point less), while the ciphertext space of HYL-TRE is the largest because of the pairing value (e.g., 1024 bits) that must be transmitted.

4.3 Storage Requirements

In settings where TRE needs to be executed in low-end, limited-memory computing systems (e.g., smartcards and other handheld computing devices), the memory/storage requirements of the protocol(s) to be used must be taken into account. As noted in Section 4.1, the user's public key space for AnTRE and HYL-TRE is a single elliptic curve point $u_{pub} \in \mathbb{G}_1$, while for the other two protocols it consists of two points, i.e., $u_{pub} \in \mathbb{G}_1 \times \mathbb{G}_1$. As a result, BC-TRE and CLQ-TRE require twice the memory to store the public keys for *known* receivers. Moreover, CLQ-TRE and AnTRE are the only two of the protocols considered here that enable pre-computations at a minor cost of storing $\hat{e}(P,P) \in \mathbb{G}_2$, leading to increased efficiency[7].

5 Conclusions

We have presented a new, server-based cryptographic scheme for anonymous timed-release encryption, and proved that is IND-CCA and IND-CTCA secure in the random oracle model. Our protocol requires no interaction between users and the server, whose sole responsibility is to publish time-specific trapdoors that correspond to specific time instants. We compared our approach with three of the best-known existing TRE schemes; the main advantage of the proposed protocol is its low computational cost and memory storage requirements. Other properties of the proposed scheme include scalability and practicality when sending a message to multiple receivers.

[6] As the basic BC-TRE has not been proven to be secure against IND-CCA attacks, it could be modified using the technique in [17].

[7] We did not include here the memory cost during real-time execution (volatile memory) because it depends on the implementation of the basic elliptic curve operations.

References

1. M. Bellare and S. Goldwasser, Encapsulated Key Escrow, *MIT Laboratory for Computer Science Technical Report 688*, 1996.
2. M. Bellare and P. Rogaway, Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols, in 1st *ACM Conf. on Computer and Communications Security*, pp. 62-73, 1993.
3. I. F. Blake and A. C.-F. Chan, Scalable, Server-Passive, User-Anonymous Timed Release Cryptography, in *25th IEEE Intl. Conf. on Distributed Computing Systems*, pp. 504-513, 2005.
4. D. Boneh and X. Boyen, Short Signatures Without Random Oracles, in *Advances in Cryptology - EUROCRYPT '04*, LNCS 3027, pp. 56-73, Springer Verlag, 2004.
5. D. Boneh and X. Boyen, Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles, in *Advances in Cryptology - EUROCRYPT '04*, LNCS 3027, pp. 223-238, Springer Verlag, 2004.
6. D. Boneh, X. Boyen, and E.-J. Goh, Hierarchical Identity Based Encryption with Constant Size Ciphertext, available at http://eprint.iacr.org/2005/015, 2005.
7. D. Boneh and M.Franklin, Identity Based Encryption From the Weil Pairing, in *Advances in Cryptology - CRYPTO '01*, LNCS 2139, pp. 213-229, Springer-Verlag, 2001.
8. D. Boneh and M. Naor, Timed Commitments and Applications, in *Advances in Cryptology - CRYPTO '00*, LNCS 1880, pp. 236-254, Springer-Verlag, 2000.
9. R. Canetti, S. Halevi, and J. Katz, A Forward Secure Public Key Encryption Scheme, in *Advances in Cryptology - EUROCRYPT '03*, LNCS 2656, pp. 254-271, Springer Verlag, 2003.
10. J. Cathalo, B. Libert, and J.-J. Quisquater, Efficient and Non-interactive Timed-Release Encryption, in 7th *Intl. Conf. on Information and Communications Security*, LNCS 3783, pp. 291-303, Springer-Verlag, 2005.
11. J. Cathalo, B. Libert, and J.-J. Quisquater, Unpublished Extended version of [10], *private communication*.
12. K. Chalkias, D. Hristu-Varsakelis and G. Stephanides, A Protocol for Improved Timed-Release Encryption, Technical Report, Computational Systems and Software Engineering Laboratory, Department of Applied Informatics, University of Macedonia, 2007. Available at: http://csse.uom.gr/eprints/58/01/AnTRE-full.pdf
13. K. Chalkias and G. Stephanides, Timed Release Cryptography from Bilinear Pairings Using Hash Chains, in 10th *IFIP Conf. on Communications and Multimedia Security*, LNCS 4237, pp. 130-140, Springer-Verlag, 2006.
14. I. Damgard, Practical and probably secure release of a secret and exchange of signatures, in *Advances in Cryptology - EUROCRYPT '93*, LNCS 765, pp. 200-217, Springer-Verlag, 1994.
15. A. W. Dent and Q. Tang, Revisiting the Security Model for Timed-Release Public-Key Encryption with Pre-Open Capability, available at http://eprint.iacr.org/2006/306.pdf, 2006.
16. G. D. Crescenzo, R. Ostrovsky, and S. Rajagopalan, Conditional Oblivious Transfer and Timed-Release Encryption, in *Advances in Cryptology - EUROCRYPT '99*, LNCS 1592, pp. 74-89, Springer-Verlag, 1999.
17. E. Fujisaki and T. Okamoto, How to Enhance the Security of Public-Key Encryption at Minimum Cost, in *PKC '99*, LNCS 1560, pp. 53-68. Springer Verlag, 1999.
18. J. Garay and M. Jakobsson, Timed Release of Standard Digital Signatures, in *Financial Cryptography '02*, LNCS 2357, pp. 168-182, Springer-Verlag, 2002.
19. J. Garay and C. Pomerance, Timed Fair Exchange of Standard Signatures, in *Financial Cryptography '03*, LNCS 2742, pp. 190-207, Springer-Verlag, 2003.
20. Y. H. Hwang, D. H. Yum, and P. J. Lee, Timed-Release Encryption with Pre-open Capability and its Application to Certified E-mail System, in 8th *Information Security Conf.*, LNCS 3650, pp. 344-358, Springer Verlag, 2005.

21. A. Joux, The Weil and Tate Pairings as Building Blocks for Public Key Cryptosystems (Survey), in *ANTS '02*, LNCS 2369, pp. 20-32, Springer Verlag, 2002.
22. T. May, Timed-Release Crypto, manuscript, available at http://www.hks.net.cpunks/cpunks-0/1560.html, 1993.
23. S. S. Ltd, Miracl - Multiprecision Integer and Rational Arithmetic C/C++ Library, (See: http://indigo.ie/ mscott/).
24. W. Mao, Timed Release Cryptography, in *Selected Areas in Cryptography 2001*, LNCS 2259, pp. 342-357, Springer-Verlag, 2001.
25. M. C. Mont, K. Harrison, and M. Sadler, The HP time vault service: Innovating the way confidential information is disclosed at the right time, in 12^{th} *Intl. World Wide Web Conf.*, pp. 160-169, ACM Press, 2003.
26. I. Osipkov, Y. Kim, and J.-H. Cheon, Timed-Release Public Key Based Authenticated Encryption, available at http://eprint.iacr.org/2004/231, 2004.
27. R. L. Rivest, A. Shamir, and D. A. Wagner, Time-Lock Puzzles and Timed-Release Crypto, *MIT Laboratory for Computer Science Technical Report 684*, 1996.
28. M. Stogbauer, Efficient Algorithms for Pairing-Based Cryptosystems, *Diploma Thesis: Darmstadt University of Technology, Dept. of Mathematics*, 2004.
29. P. F. Syverson, Weakly Secret Bit Commitment: Applications to Lotteries and Fair Exchange, in *11th IEEE Computer Security Foundations Workshop*, pp. 2-13, 1998.
30. F. Zhang, R. Safavi-Naini, and W. Susilo, An Efficient Signature Scheme from Bilinear Pairings and Its Applications, in 7^{th} *PKC '04*, LNCS 2947, pp. 277-290, Springer Verlag, 2004.

A Basic Version of the Protocol

A "basic" version of AnTRE, termed BasicAnTRE, will be useful when discussing the security of our protocol (Appendix B). The ReleaseT and KeyGen algorithms of BasicAnTRE are identical to those of AnTRE (Sec. 3.2). The Setup, Encryption and Decryption primitives are as follows:

AnTRE.Setup: given security parameters k and k_0, where k_0 is polynomial in k, the setup algorithm:
1. Outputs a k-bit prime number q, two groups $\mathbb{G}_1$, $\mathbb{G}_2$ of order q, an admissible bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ and an arbitrary generator $P \in \mathbb{G}_1$.
2. Chooses the following cryptographic hash functions: $H_1 : \{0,1\}^{\tau} \mapsto \mathbb{Z}_q^*$, $H_2 : \{0,1\}^n \mapsto \{0,1\}^{k_0}$, $H_3 : \mathbb{G}_2 \mapsto \mathbb{Z}_q^*$, $H_4 : \mathbb{G}_1 \mapsto \{0,1\}^{n+k_0}$ for some $n \in \mathbb{N}$. These functions will be treated as random oracles when it comes to security considerations.
3. Generates the time-server's private key $s \xleftarrow{R} \mathbb{Z}_q^*$ and the corresponding public key $S = sP \in \mathbb{G}_1^*$.
4. Chooses the message space to be $M = \{0,1\}^n$ and the ciphertext space is $C = \mathbb{G}_1 \times \mathbb{G}_1 \times \{0,1\}^{n+k_0}$.
The public parameters are $params := \{k, k_0, q, \mathbb{G}_1, \mathbb{G}_2, P, S, \hat{e}, H_1, H_2, H_3, H_4, n, M, C\}$.

AnTRE.Enc: to encrypt $m \in \{0,1\}^n$ using the time information $T \in \{0,1\}^{\tau}$ and the receiver's public key B, the sender executes the following:
1. Choose random $r_1, r_2 \in \mathbb{Z}_q^*$, compute $t = H_1(T) \in \mathbb{Z}_q^*$.
2. Compute $h = H_2(m)$.
3. Compute $c_1 = r_1 S + r_1 t P \in \mathbb{G}_1^*$ and $c_2 = r_2 P \in \mathbb{G}_1^*$.

4. Compute $d = H_3(\hat{e}(P,P)^{r_1}) \in \mathbb{Z}_q^*$.
5. Compute $K = H_4(dr_2B) \in \{0,1\}^{n+k_0}$ and then $c_3 = (m||h) \oplus K \in \{0,1\}^{n+k_0}$.
The ciphertext is $C := \langle c_1, c_2, c_3, T \rangle$.

AnTRE.Dec: given $C := \langle c_1, c_2, c_3, T \rangle$, the trapdoor s_T and his private key b, the recipient computes $d = H_3(\hat{e}(c_1, s_T)) \in \mathbb{G}_1$ and the session key $K = H_4(dr_2B) \in \{0,1\}^{n+k_0}$. Then, he is able to retrieve the message as $m||h = K \oplus c_3$. To verify the message, he checks whether $H_2(m) = h$.

B Security Proofs for AnTRE

Our proofs are in the random oracle model, and follow those of Theorems 2 and 3 of [11] (an extended version of [10]). We will first consider the security of BasicAnTRE, described in Appendix A. As in [10], AnTRE results from a variant of the first Fujisaki-Okamoto transform [17] applied to BasicAnTRE, by hashing the message m, a random number x, and a time T, and using the resulting bits in order to encrypt. This conversion is slightly different from the one in [17] because the hash function H_2 takes as an additional input the time T. Including T among the inputs is necessary (the encryption algorithm of BasicAnTRE is parameterized by T) and enables a knowledge extractor to simulate the behavior of a decryption oracle with the same probability as the plaintext extractor in the security proof of the Fujisaki-Okamoto conversion [17]. Our security proofs apply the modified version of Theorem 3 from [17], established in [11].

B.1 Proof of Theorem 1 - security against "impatient" recipients

We first show that BasicAnTRE is secure against chosen time and plaintext attacks (IND-CTPA)[8] [10], using a slightly modified form of the security proof in [11, 10] (similar to [4], [5]). Assuming an IND-CTPA attacker $\mathcal{A}$ which succeeds against BasicAnTRE with non-negligible probability, $\varepsilon(k)$, we will construct an algorithm $\mathcal{B}$ which takes as inputs $< P, \alpha P, \alpha^2 P, \alpha^3 P, ..., \alpha^q P >$, for some interger q, and computes $\hat{e}(P,P)^{\alpha^{-1}}$ with non-negligible probability.

Let q_T be the number of queries made by $\mathcal{A}$ to the time server. Without loss of generality, we will assume that $q_T = q_{H_1} - 1 = q - 1$ (if $q_T < q_{H_1} - 1$, then $\mathcal{B}$ can issue dummy queries to the time-server broadcast oracle for itself). Initially, $\mathcal{B}$ chooses $\ell \xleftarrow{R} \{1, ..., q\}$ and $a, b \xleftarrow{R} \mathbb{Z}_q^*$ and sets $I_\ell = ab \in \mathbb{Z}_p^*$. Then, he chooses $I_i \xleftarrow{R} \mathbb{Z}_q^*$ and computes $w_i = \frac{I_\ell - I_i}{a}$ $\forall\, i \in \{1, ..., q\} \setminus \{\ell\}$.

Next, $\mathcal{B}$ uses its input to compute a generator $Q \in \mathbb{G}_1$ and a server public key $s_{pub} = xQ$ for some $x \in \mathbb{Z}_q^*$, such that $\mathcal{B}$ can know all of the q_T pairs $(I_i, (I_i + x)^{-1} Q)$, $i \neq \ell$, as in [4]. He does this in the following manner. $\mathcal{B}$ expands the polynomial $f(z) = \prod_{i=0, i\neq\ell}^{q} (z + w_i) = \sum_{j=0}^{q-1} c_j z^j$, to find c_j's. Then, Q, $U \in \mathbb{G}_1$ are obtained as

$$Q = \prod_{j=0}^{q-1} (\alpha^j P) c_j = f(\alpha) P \in \mathbb{G}_1, \quad U = \prod_{j=1}^{q} (\alpha^j P) c_{j-1} = \alpha f(\alpha) P = \alpha Q.$$

[8] This type of adversary is defined similarly to IND-CTCA, but without access to a decryption oracle.

Similarly to [4], the q_T pairs $(w_i, Q_i = (w_i + \alpha)^{-1}Q)$ can then be obtained from expanding $f_i(z) = \frac{f(z)}{z+w_i} = \sum_{j=0}^{q-2} d_j z^j$ and computing

$$Q_i = \prod_{j=0}^{q-2}(\alpha^j P)d_j = f_i(\alpha)P = \frac{f(\alpha)}{a+w_i}P = (\alpha + w_i)^{-1}Q,\ \forall\, i \in \{1,...,q\}\backslash\{\ell\}.$$

Now, $\mathcal{B}$ chooses the time-server's public key to be $s_{pub} = -aU - I_\ell Q = (-a)\alpha Q - abQ = -a(\alpha + b)Q$. Thus, the server's private key (which is unknown to $\mathcal{B}$), is $x = -a(\alpha+b) = -a\alpha - I_\ell \in \mathbb{Z}_q^*$, and for all $i \in \{1,...,q\}\backslash\{\ell\}$ we have:

$$(I_i, -a^{-1}Q_i) = (I_i, (I_i - a\alpha - I_\ell)^{-1}Q)$$

and

$$(I_i, (I_i + x)^{-1}Q) = (I_i, (I_i - a\alpha - I_\ell)^{-1}Q),$$

thus

$$(I_i, -a^{-1}Q_i) = (I_i, (I_i + x)^{-1}Q).$$

$\mathcal{B}$ now has knowledge of all q_T pairs $(I_i, (I_i + x)^{-1}Q)$ because he can compute I_i, Q, Q_i from $(P, \alpha P, \alpha^2 P, \alpha^3 P, ...\alpha^q P)$ as described above. Armed with this information, he will be able to provide correctly formed values each time $\mathcal{A}$ queries H_1 for a given time, or requests the trapdoor value for the same time. To proceed, $\mathcal{B}$ starts $\mathcal{A}$ on input $s_{pub} = (-aU - I_\ell Q)$ and initializes a counter, $v = 0$. During the game we assume that: i) all H_1 queries are distinct, and ii) $\mathcal{A}$ produces her challenge request at T^*, for which he asks the hash value $H_1(T^*)$. $\mathcal{B}$ answers queries to random oracles as follows:

- H_1: $\mathcal{B}$ answers I_v and increments $v = 1, 2, ...$
- For $i = 2, 3, 4$ (H_i): On input γ_λ, $\lambda = 1, 2, ..., q_{H_i}$, $\mathcal{B}$ selects a random $\eta_{i,\lambda}$ and stores $(\gamma_\lambda, \eta_{2,\lambda})$ into a list L_i. Each incoming query input is matched against those already on the corresponding list; if the same query has been asked again, $\mathcal{B}$ returns the same value as before.
- Queries to time-server: On input $T_v, v = 1, 2, ...$, if $v = \ell$, $\mathcal{B}$ stops and reports "failure"; otherwise returns the trapdoor value for T_v, $-a^{-1}Q_v = (I_n + x)Q$, to $\mathcal{A}$.

After the find stage, $\mathcal{A}$ outputs $< m_0, m_1, T^* >$ and a valid public key u_{pub} to be challenged on. If $T^* \neq T_\ell$ ($\mathcal{B}$ did not guess correctly which T_i the attack will occur on), then $\mathcal{B}$ stops and reports "failure". Otherwise, he selects $\sigma \xleftarrow{R} \in \mathbb{Z}_q^*$ and a random string $c_3{}^*$ to return the challenge $C^* :< c_1{}^*, c_2{}^*, c_3{}^* >$, with $c_1{}^* = -a\sigma Q$, $c_2{}^* = r_2 Q$. To elucidate $\mathcal{B}$'s choice of c_1^*, recall that $\mathcal{B}$ should send something of the form $c_1{}^* = r_1(t + x)Q$, $c_2{}^* = r_2 Q$, with $t = I_\ell$ (corresponding to $H_1(T^*)$) and $x = -a\alpha - I_\ell$ (unknown server's private key). In the simulation set up by $\mathcal{B}$, it would be $c_1{}^* = r_1(I_\ell - a\alpha - I_\ell)Q = r_1(-a\alpha)Q$, for r_1 random. Now, assume that $r_1 = \frac{\sigma}{\alpha}$, so that $\sigma = r_1\alpha$. Then, sending $c_1{}^* = -a\sigma Q$ to $\mathcal{A}$, for random σ, would be precisely the same as if we had used $r_1 = \frac{\sigma}{\alpha}$ to encrypt. Realizing that $c_3{}^*$ is not properly formatted, would require $\mathcal{A}$ to query $H_4(d\beta c_2)$ with non-negligible probability (if not, one could construct an algorithm for inverting the *XOR* function with non-negligible probability). Whether or not the private key β is known to $\mathcal{A}$, it can be easily shown that computing the "correct" input to H_4 (which is $d\beta r_2 Q$) with non-negligible probability, implies that $\mathcal{A}$ must query H_3 on input $\hat{e}(Q,Q)^{r_1}$ with non-negligible probability to compute d, the latter being the output of a random oracle. $\mathcal{B}$ has the opportunity to detect that event and use $\hat{e}(Q,Q)^{r_1}$ to compute $\hat{e}(Q,Q)^{a^{-1}}$.

If $\mathcal{A}$ is successful in guessing the hidden bit with non-negligible probability, then using standard arguments it can be shown that $\mathcal{A}$ is very likely to query H_3 on $\hat{e}(Q,Q)^{r_1}$ at some time of the game, if the latter mimics perfectly the real attack environment. To produce its output, $\mathcal{B}$ selects a random γ from its L_3 list, so that with probability $\varepsilon(k)\frac{1}{q_{H_3}}$, γ is $\hat{e}(Q,Q)^{r_1}$. Then, $\gamma = \hat{e}(Q,Q)^{r_1} = \hat{e}(P,P)^{\frac{\sigma f^2(\alpha)}{\alpha}}$. If we let $f^2(z) = \sum_{j=0}^{2q-2} \phi_j z^j$ so that $f^2(\alpha)/\alpha = \phi_0/\alpha + \sum_{j=1}^{2q-2} \phi_j \alpha^{j-1}$, then we can solve for $\hat{e}(P,P)^{a^{-1}}$ from γ, as:

$$\hat{e}(P,P)^{a^{-1}} = \left(\gamma^{\sigma^{-1}} \prod_{j=1}^{2q-2} \hat{e}(P,P)^{(\alpha^{j-1})(-\phi_j)}\right)^{\phi_0^{-1}},$$

where the ϕ_j can be computed from the known coefficients of $f(z)$, and the $\hat{e}(P,P), \ldots, \hat{e}(P,P)^{2q-3}$ are computed from $\mathcal{B}$'s inputs $< P, \alpha P, \alpha^2 P, \alpha^3 P, \ldots, \alpha^q P >$. This contradicts the assumption that the q-BDHI problem is hard. We conclude that BasicAnTRE is IND-CTPA secure. Now, the IND-CTPA security of BasicAnTRE implies IND-CTCA security of AnTRE using a Lemma very similar to Lemma 2 of [11] (itself derived from Theorem 3 of [17]) with minor modifications. The proof (omitted here because of space limitations) can be found in a fuller version of this paper [12].

Lemma 1. *In the random oracle model, an IND-CTCA attacker $\mathcal{A}$ having non-negligible advantage ε against AnTRE when making q_D decryption queries and q_{H_i} queries to oracles $H_i, (i = 1, \ldots, 4)$, implies an IND-CTPA attacker $\mathcal{B}$ with non-negligible advantage against BasicAnTRE.*

We note that the proof [12] of the last lemma addresses the single-receiver case. If a message is sent to $N > 1$ receivers, then the corresponding ciphertexts differ only in their c_3 parts, and a malicious receiver may attempt to read his message early by obtaining multiple $c_{3,i}$ values, $i = 1, \ldots, N$, for the same message m and time T. Doing so can be shown to be computationally difficult [12]. The complete argument is omitted because of space limitations.

B.2 Proof of Theorem 2 - security against "curious" servers

We first show that BasicAnTRE is secure against chosen plaintext attacks (IND-CPA)[9] [10]. Assume that there exists a polynomial-time IND-CPA attacker $\mathcal{A}$ which has a non-negligible advantage, $\varepsilon(k)$, against BasicAnTRE, asking n_i queries to random oracles h_i, $i = 1, \ldots, 4$. We will show that there exists an algorithm, $\mathcal{B}$, which solves the CDH problem with non-negligible probability, in polynomial time, using $\mathcal{A}$ as a subroutine. The algorithm $\mathcal{B}$ will accept as inputs P, aP and bP, and will compute abP, for $a, b \in \mathbb{Z}_q^*$.

[9] This type of adversary is defined similarly to IND-CCA, but without access to a decryption oracle.

Our scheme follows the proof of Theorem 3 in Cathalo et. al [10], where $\mathcal{B}$ uses part of the challenge ciphertext, c_1, c_2 to elicit $\mathcal{A}$ to make a random oracle query with the desired input, in this case a known multiple of abP.

$\mathcal{B}$ operates by assigning any valid key pair $(s, S = sP)$, to the time server, and starts $\mathcal{A}$ with inputs P, bP (public key against which $\mathcal{A}$ is to be tested), and the server's secret key, s. $\mathcal{B}$ answers $\mathcal{A}$'s queries to H_1,...,H_4, as follows. For each H_i, $i = 1,\ldots,4$, $\mathcal{B}$ answers each query at random; each time, $\mathcal{B}$ records the input and the reply it gave on a list, L_i, so that if an oracle is queried again on the same input, $\mathcal{B}$ will return the same number.

After finishing its queries, $\mathcal{A}$ outputs m_0, m_1, T^*. At that point, $\mathcal{B}$ creates a challenge ciphertext to return to $\mathcal{A}$ as follows. He chooses r_1 at random and a message m also at random. He checks whether H_1 has previously been queried on T^*, in which case he sets t to its prior response, found in the list L_1. If the opposite is true, then $\mathcal{B}$ assigns t to be a random number, which he records in L_1. $\mathcal{B}$ then sets $c_1 = r_1S + r_1tP$, and $c_2 = aP$. To produce c_3, $\mathcal{B}$ first checks whether the value $\hat{e}(c_1, (s+t)^{-1}P) = \hat{e}(P,P)^{r_1}$ has been presented by $\mathcal{A}$ as a query to H_3. If so, $\mathcal{B}$ finds its previous reply from the list L_3 and sets d to that number, otherwise he sets d to a random number which it records on L_3, as specified previously. Finally, $\mathcal{B}$ chooses K at random, sets $c_3 = (m||H_2(m)) \oplus K$ and then sends the challenge (c_1, c_2, c_3, T^*) to $\mathcal{A}$.

From $\mathcal{A}$'s point of view, the challenge appears to be properly formated. To recognize that c_3 is not an encryption of either m_0 or m_1, $\mathcal{A}$ would have to request $H_4(dbc_2)$, for $d = H_3(\hat{e}(c_1, s_T))$, known to $\mathcal{B}$; doing so would enable $\mathcal{B}$ to obtain the solution to the CDH problem by computing $d^{-1}dbc_2 = abP$. On the other hand, because the simulation set up by $\mathcal{B}$ mimics a genuine attack environment perfectly, one can show using standard arguments that if $\mathcal{A}$ succeeds in guessing the hidden bit b, it is very likely to query oracle H_4 on input dr_2bP (e.g., if that were not the case, $\mathcal{A}$ could be used to produce an algorithm that inverts the *XOR* operation with non-negligible probability), and in particular, the probability of $\mathcal{A}$ doing so is some $\varepsilon'(k)$, non-negligible. In that case, $\mathcal{B}$ has the chance to detect $\mathcal{A}$'s query and compute the CDH solution. Thus, when $\mathcal{A}$ halts, $\mathcal{B}$ ignores her result, and selects at random an entry from his list L_4. For the number, g, that was his reply to the corresponding query, he computes and outputs $d^{-1}g$, having correctly guessed at the value of abP with probability $\varepsilon'(k)/n_4$, where n_4 is a polynomial bound on the number of H_4 queries made by $\mathcal{A}$ during her attack. This contradicts the assumed hardness of the CDH problem. We conclude that BasicAnTRE is IND-CPA secure.

The IND-CCA security of AnTRE follows from the IND-CPA security of BasicAnTRE, via a result whose proof is very similar to that of Lemma 1 and is omitted.

Lemma 2. *In the random oracle model, an IND-CCA attacker $\mathcal{A}$ having non-negligible advantage ε against AnTRE when making q_D decryption queries and q_{H_i} queries to oracles $H_i, (i = 1..4)$, implies an IND-CPA attacker $\mathcal{B}$ with non-negligible advantage against BasicAnTRE.*

As in the case of Lemma 1, the proof of Lemma 2 must account for the fact that if a message is sent to $N > 1$ receivers, then a malicious server might then have access to multiple $c_{3,i}$ values, $i = 1,\ldots,N$, for the same message m and time T, from which he could attempt to read the message. An argument for why this is computationally difficult is given in [12].