Low-cost Anonymous Timed-Release Encryption

D. Hristu-Varsakelis, K. Chalkias, and G. Stephanides
University of Macedonia
Department of Applied Informatics
Computational Systems and Software Engineering Laboratory
Thessaloniki, Greece
dcv@uom.gr, chalkias@java.uom.gr, steph@uom.gr

Abstract

We propose a new server-based efficient protocol for time-release encryption (TRE), or – as sometimes referred to — sending information "into the future". As with other recently-proposed schemes, ours is based on the use of bilinear pairings on any Gap Diffie-Hellman group, allowing absolute release time of the encrypted data. Our protocol possesses the required properties regarding user anonymity and server passivity. It also provides almost-costless scalability in settings with multiple time-servers, and improves significantly upon existing TRE schemes, in terms of computational and communication cost. This makes our approach well-suited to a number of emerging e-applications that require future decryption of confidential data.

Keywords: cryptographic protocols, timed-release encryption, bilinear pairings, multiple time-servers.

1. Introduction

The aim of *timed-release encryption* (TRE) is to support applications where encrypted confidential data must not be decrypted by anyone, including the designated receipient(s), until a predetermined future time. TRE was originally introduced in [19], and further studied in [25].The TRE problem arises in many emerging applications with significant impact to daily personal and business activity, such as electronic voting, which requires delayed opening of votes [24]; sealed-bid auctions, where bids must stay sealed until the bidding period is over [25]; and Internet-based contests where participants must not access the challenge problem before the contest starts [3]. Other examples include contract signing, where two or more suspicious parties need to exchange signatures on a contract [9], future release of documents (e.g., memoirs, wills, press releases) [25], delayed release of escrowed keys [2], and online gambling and lotteries [7, 8].

Existing approaches to TRE are based upon the use of either a *time-lock puzzle* (TLP) or a so-called *trusted-agent* (TA). The former method was initially proposed by [20] for protecting communications against passive adversaries. Later, [25] extended this technique to provide TRE. Briefly, the TLP approach consists of transforming a secret message in such a way that any serial or parallel machine must spend a significant amount of time solving a computational problem (puzzle) in order to reconstitute the message. Such techniques [25, 18, 14], do not require a trusted third-party, but they put immense computational overhead on the receiver, who must perform non-stop, non-parallelizable computation to retrieve a time-encrypted message. Except from "tying up" the receiver's CPU, TLP techniques cannot guarantee a precise release time because they depend on the computational power of the receiver's machine, and on the time at which the decryption process is started; thus they are impractical for many real-life scenarios.

To overcome the problems of TLPs, TA approaches make use of a so-called *time-server* who provides a common and absolute time reference to users. In these server-based schemes, users need to retrieve a piece of information (trapdoor) from the time-server(s), akin to a secret key which is necessary (in conjunction with the user's secret key) for the decryption process. Using TAs, one can set the decryption date with high precision. The tradeoff is that some interaction between the server and the users is required. In the early attempts at TRE [19, 25], the server was actively involved in the encryption or the decryption process and user-anonymity was compromised. The situation improved with [11], which proposed a scheme based on a conditional oblivious transfer protocol [17], in which anonymity was achieved for the sender only. The non-pairing based scheme of [22] was the first to achieve server passivity, meaning that the TA's only role is to publish *universal*[1] time-specific trapdoors (e.g., on a web page). That

[1]Here, *universal* means that a trapdoor is the same for all users who

approach was based on the quadratic residue assumption (QRA) and had a very high communication cost compared to all of the elliptic curve (EC)-based protocols examined here.

Since the early work on TA-based TRE, researchers have focused on minimizing server-user interaction, to ensure scalability and user-anonymity. New and innovative TRE techniques appeared especially after the introduction of *Identity-Based Encryption* (IBE) [5]. The latter used elliptic curve cryptography (ECC) and the efficient implementation of bilinear pairings on ECs, leading to a series of important developments. Here, we review some of the best-known and most efficient protocols.

Some of the newer pairing-based schemes allow a user to recover past time-specific trapdoors from the currently published one. Specifically, the schemes of [4] and [23] use the tree-like structure of [6] backwards, to construct previous trapdoors, while [8] uses an inverted hash chain, similarly to the S/Key system [15]. These approaches have a high communication cost, while the root of the tree-like structure and the hash chain, respectively, correspond to the last available trapdoor to be published, implying an upper bound on the "lifetime" of the system.

Of particular interest in TA-based TRE is the problem of eliminating (or at least reducing) the possibility of collusion between the receiver and an unscrupulous time-server who might allow early access to a message. An often-used solution is to arrange matters so that the receiver is forced to obtain trapdoors from more than one servers to decrypt a message (more servers means that a larger number of entities must be corrupted in order for cheating to take place). In that setting, Blake's and Chan's [3] scheme formed the point of departure for a number of recent works, including ours, being the first to provide efficient scalable, server-passive, user-anonymous TRE with support for multiple time-servers. The work in [16] described a similar, but computationally less efficient scheme, that could also support message pre-opening[2]. Improvements to [16] were later proposed by [10]. Also, [7] designed a user-anonymous TRE protocol that could make use of pre-computations (i.e., some of the calculations required to run the protocol can be performed off-line, prior to specifying a message or a receiver), and thus be faster than previous approaches. The main disadvantage of that work was the high additional cost when using multiple time-servers. See also [24] for an attempt at authenticated TRE.

The contribution of this paper is to propose a server-based TRE protocol which is inspired by [3] and which makes significant improvements in computational and communication cost, as well as scalability, over other published TRE approaches. With respect to computational cost, our modified protocol compares favorably to those mentioned previously and has the lowest cost in the case of unknown receivers. Its communication cost effectively combines the best features offered in other TRE protocols, using both a simple public key format and a small ciphertext space. Finally, under some conditions, the additional computational cost required in settings with multiple time-servers is negligible.

The remainder of this paper is structured as follows. In Section 2 we describe a new TA-based TRE protocol. In Section 3 we compare our protocol with [3] and with other modern TRE approaches, in terms of computational efficiency, communication cost and scalability.

decrypt messages released at the time instant for which the trapdoor was intended.

[2]Pre-opening refers to the ability of a sender to "speed-up" the decryption process by sending a trapdoor key before the designated time at which a message was scheduled to be opened.

2. Proposed Protocol

2.1. Preliminaries

We begin by fixing notation and by reviewing some related definitions and computational assumptions used in 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 (EC). We will let P denote the generator of $\mathbb{G}_1$ and $t \in \{0,1\}^\tau, \tau \in \mathbb{N}$ denote time. For instance, t could indicate the τ-bit string representation of a specific time instant (e.g., 09:30:00 AM August 29, 2007 GMT). Also, H_1, H_2 will be two secure hash functions, with $H_1 : \{0,1\}^\tau \mapsto \mathbb{G}_1^*$, $H_2 : \mathbb{G}_2^* \mapsto \{0,1\}^n$. Finally, $e : \mathbb{G}_1 \times \mathbb{G}_1 \mapsto \mathbb{G}_2$ will be a bilinear pairing.

Definition 1 Bilinear Pairings *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: $\hat{e}(aP, bQ) = \hat{e}(bP, aQ) = \hat{e}(abP, Q) = \hat{e}(P, abQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}_1$, $a, b \in Z_q^*$.*
- *Non-degeneracy: there exist $P, Q \in \mathbb{G}_1$ such that $\hat{e}(P, Q) \neq 1$.*
- *Efficiency: there exists an efficient algorithm to compute the bilinear map.*

We note that all pairing algorithms currently employed in cryptography are based on ECs, and thus make use of Millers algorithm [21]. Currently, admissible types of pairings include the Weil and Tate pairings and their variants [1, 13].

The security of our protocol is based on the assumed difficulty of the following problems:

Definition 2 Discrete Log Problem (DLP) in $\mathbb{G}_1$
Given $P, Q \in \mathbb{G}_1$, it is difficult to find an integer $a \in Z_q^*$ such that $Q = aP$.

Definition 3 Discrete Log Problem (DLP) in $\mathbb{G}_2$
Given $g_1, g_2 \in \mathbb{G}_2$, it is difficult to find an integer $a \in Z_q^*$ such that $g_2 = g_1^a$.

Definition 4 Computational Diffie-Hellman Problem (CDHP) in $\mathbb{G}_1$
Given $P, aP, bP \in \mathbb{G}_1$, for some unknowns $a, b \in Z_q^*$, it is difficult to find $abP \in \mathbb{G}_1$.

Definition 5 Bilinear Diffie-Hellman Problem (BDHP)
Given $P, aP, bP, cP \in \mathbb{G}_1$, for some unknowns $a, b, c \in Z_q^*$, it is difficult to find $\hat{e}(P,P)^{abc}$.

2.2. Proposed anonymous TRE

The proposed agent-based TRE scheme, termed *New-TRE*, involves two types of entities: a time-server which issues time-specific trapdoors with some pre-determined frequency (e.g., every minute), and users which may act as either senders or receivers. To send a message, m, that will be decrypted at (or after) a pre-defined time t, the following protocol is to be executed:

New-TRE.Setup:(run by the time-server) given a security parameter 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: $H_1 : \{0,1\}^\tau \mapsto \mathbb{G}_1^*$, $H_2 : \mathbb{G}_2^* \mapsto \{0,1\}^n$. These functions will be treated as random oracles when it comes to security considerations.
3. Generates the time-server's private key $s \in \xleftarrow{R} \mathbb{Z}_q^*$ 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 to be $C = \mathbb{G}_1 \times \{0,1\}^{n+\tau}$.
The public parameters are $params := \{k, q, \mathbb{G}_1, \mathbb{G}_2, P, S, \hat{e}, H_1, H_2, n, \tau, m, C\}$.

New-TRE.ReleaseT: (run by the time-server) given a time instant $t \in \{0,1\}^\tau$ and the server's private key $s \in \mathbb{Z}_q^*$, it returns the time-specific trapdoor $sk_T = sT \in \mathbb{G}_1^*$, where $T = H_1(t) \in \mathbb{G}_1^*$. We note that the trapdoor is in fact a time-server's short signature (as this proposed in [4]) on that time, t, and is inherently self-authenticating. Thus, there is no need for an additional server signature: a user can simply check whether $\hat{e}(S,T) \overset{?}{=} \hat{e}(P, sk_T)$.

New-TRE.KeyGen: (run by the receivers) given $params$, it chooses a private key $b \in \mathbb{Z}_q^*$ and produces receiver's public key $B = bP \in \mathbb{G}_1^*$.

New-TRE.Enc: (run by the senders) to encrypt $m \in \{0,1\}^n$ using the time information $t \in \{0,1\}^\tau$, the receiver's public key B and the server's public key S, the sender executes the following:
1. Choose $r \in \xleftarrow{R} \mathbb{Z}_q^*$.
2. Compute $T = H_1(t) \in \mathbb{G}_1^*$.
3. Compute $Q = rT \in \mathbb{G}_1^*$.
4. Compute $K = \hat{e}(S, Q) = \hat{e}(sP, rT) = \hat{e}(P,T)^{rs} \in \mathbb{G}_2^*$.
5. Compute $c_1 = rB = rbP \in \mathbb{G}_1^*$ and $c_2 = m \oplus H_2(K) \in \{0,1\}^n$, where $\oplus$ denotes the XOR function.
The ciphertext is $C := \langle c_1, c_2, t \rangle$.

New-TRE.Dec: (run by the receivers) given $C := \langle c_1, c_2, t \rangle$, the trapdoor sk_T and his private key b, the recipient computes:
1. $R = b^{-1}c_1 = b^{-1}brP = rP$. R can also be pre-computed (before the release time), and
2. the session key $K = \hat{e}(R, sk_T) = \hat{e}(rP, sT) = \hat{e}(P,T)^{rs} \in \mathbb{G}_2^*$.
3. He retrieves the message as $m = H_2(K) \oplus c_2$.

2.3. Sketch of Security Proof

The following sketch of proof is obtained by modifying that in [3].

1. The server's and receiver's private keys are safe because it is difficult to find s or b given P and sP or bP, respectively (it is an instance of the DL problem).

2. The server's key is also safe from an attacker who tries to compute s from the sT that the server produces for various times, t_i. Rewriting sT_i as $w_i sP$ for some unknown w_i, the attacker is faced with the problem of computing s from $P, sP, w_1 sP, w_2 sP, ...$, which is at least as difficult as the DL problem.

3. It is difficult for the receiver to decrypt a ciphertext without the release key, sT. To do so, he must compute $\hat{e}(P,T)^{rs}$ from P, b, rbP, sP. By rewriting $T = wP$ for some unknown w, we see that computing $\hat{e}(P,T)^{rs} = \hat{e}(P,P)^{wrs}$ from b, wP, sP, rbP is at least as hard as the Bilinear Diffie-Hellman problem. If the receiver tries to find sT from sT_i, $T_i \neq T$, then by rewriting $T = w_i T_i$ the receiver has the problem of finding $sw_i T_i$ from $T_i, w_i T_i, sT_i$), which is equivalent to the Computational Diffie-Hellman problem over a Gap Diffie-Hellman group. Thus, the receiver cannot decrypt a message before its release time unless he colludes with the time server.

4. A malicious server who wants to decrypt a message must compute $K = \hat{e}(P,T)^{rs}$ from s, P, bP, rbP and T. As s is known to the server, the problem can be transformed into finding $K^{s^{-1}} = \hat{e}(P,T)^r$. The last problem could be solved if one of r, rP or rT were known. However, the

only available quantity that "embeds" r is rbP. Using rbP, the malicious server cannot extract the value of r. This is because if we assume that $Q = bP$ (b is unknown), then the problem of finding r from rQ is an instance of the DLP in $\mathbb{G}_1$. Also, the problem of finding rP from bP and rbP is equivalent to the CDHP in $\mathbb{G}_1$. Moreover, the problem of finding rT from rQ, where $Q = bP$ (r, b unknown) is at least as difficult as breaking the short signature scheme of [4]. Finally, the malicious server could try to compute the pairing $\hat{e}(rbP, T) = \hat{e}(P, T)^{rb}$ but he then needs to solve the DLP in $\mathbb{G}_2$ in order to produce K.

The security of the protocol can also be further strengthened, to cover chosen time and ciphertext attacks [7], by applying the transformation of [12] as modified in [7] to account for, most importantly, the use of time as a parameter in the protocol.

3. Comparisons

We go on to compare New-TRE with four of the best-known, best-performing approaches to non-authenticated, non-interactive server-based anonymous TRE. These are the BC-TRE [3], HYL-TRE [16], DT-TRE [10], and CLQ-TRE [3] [7] protocols.

3.1. Computational Efficiency

For the purposes of comparing the computational efficiency of New-TRE with that of the four protocols mentioned previously, we will ignore operations whose cost is negligible compared to that of a scalar multiplication in $\mathbb{G}_1$. These include generating random numbers, integer multiplication, plain hashes and point additions in $\mathbb{G}_1$. Also, because some of the protocols enable pre-computations under certain circumstances, we distinguish between three cases of anonymous TRE: i) typical message transmission to *unknown* receivers, ii) transmission to *known* receivers (in which case there is no need to verify their public keys), and iii) TRE with multiple time-servers.

Table 1 lists the cost (in *msec*) of the basic operations required to run the protocol(s), taken from [24]. The results were obtained using the *MIRACL* open-source library [26] on a PIII 977MHZ with 512Mb of RAM, assuming a subgroup of order q in a supersingular EC E over F_p, where p is a 512 bit prime and q is a 160 bit prime. The pairing value belongs to a finite field of 1024 bits.

For New-TRE, the encryption phase requires 1 Mtp to compute T, 1 Sm for Q, 1 Pa for K and finally another 1 Sm for c_1, totaling 41.01 *msec*. As for the decryption phase, the recipient must perform 1 Sm operation to calculate the point value R and 1 Pa to produce K, thus the total decryption cost is approximately 35.15 *msec*. We point out that the cost of running New-TRE is not dependent on any *a-priori* knowledge of the receiver's public key.

[3]Unlike this work, [7] uses multiplicative notation for both groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

Table 1. Cost of basic operations *(in msec)*

Operation	Notation	Cost
Bilinear Pairing	Pa	31.71
Parallel Scalar Multiplication in $\mathbb{G}_1$	PSm	4.3
Scalar Multiplication in $\mathbb{G}_1$	Sm	3.44
Exponentiation in $\mathbb{G}_2$	Ex	3.93
Map-To-Point	Mtp	2.42
Inversion in $\mathbb{Z}_q$	Inv	1.81

Tables 2 and 3 summarize our comparisons of computational cost for the cases of *unknown* and *known* receivers, respectively, taking advantage of pre-computations, when possible. For example, the constant value $\hat{e}(P, P)$ in CLQ-TRE and the value b^{-1} in New-TRE, respectively, are considered to be pre-computed and thus do not figure into the total cost for those protocols. Because not all protocols provide for message pre-opening, we assumed that no such activity take place, for the sake of being able to make fair comparisons. For all protocols, we used the bilinearity of pairings to replace the more expensive exponentiation in $\mathbb{G}_2$ with scalar multiplication in $\mathbb{G}_1$, whenever possible, e.g., $\hat{e}(P, Q)^a = \hat{e}(aP, Q)$.

From Table 2, we observe that our approach is the fastest method in the general case of sending a message to *unknown* receivers, while CLQ-TRE has the lowest computational cost in the special case of *known* receivers, because of the pre-computation of $\hat{e}(P, P)$ (the sender computes no pairings "on-line"). Moreover, BC-TRE, HYL-TRE and New-TRE have the lowest complexity in the decryption phase. BC-TRE and CLQ-TRE are the only protocols whose cost depends on knowledge of the receiver's public key. This is because they use a slightly different public key format, with users' public keys consisting of two points in $\mathbb{G}_1$ instead of just one, as in the conventional cryptographic schemes (Diffie-Hellman keys). The implication for BC-TRE and CLQ-TRE is 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 it is properly formed and that the recipient will be able to decrypt the message. Such verification is not needed in the other three schemes.[4]

[4]When comparing the cost of implementations of the above schemes, we did not include the cost of a group membership test for the public keys. If that were taken into account, [3, 7] would require some additional checking because of the use of "two-point" public keys.

Table 2. Computational cost comparison sending to *unknown* receivers *(in msec)*

Protocol	Encryption	Decryption	Total
BC-TRE	$3Pa + 2Sm + 1Mtp = 104.43$	$1Pa + 1Sm = 35.15$	139.58
HYL-TRE	$1Pa + 1PSm + 2Sm + 1Mtp = 45.31$	$2Pa + 1Sm = 66.86$	112.17
DT-TRE	$1Pa + 3Sm + 1Mtp = 44.45$	$1Pa + 1Sm = 35.15$	79.6
CLQ-TRE	$2Pa + 1PSm + 1Ex = 71.65$	$1Pa + 1PSm + 1Ex = 39.94$	111.59
New-TRE	$1Pa + 2Sm + 1Mtp = 41.01$	$1Pa + 1Sm = 35.15$	76.16

Table 3. Computational cost comparison sending to *known* receivers *(in msec)*

Protocols	Encryption	Decryption	Total
BC-TRE	$1Pa + 2Sm + 1Mtp = 41.01$	$1Pa + 1Sm = 35.15$	76.16
HYL-TRE	$1Pa + 1PSm + 2Sm + 1Mtp = 45.31$	$2Pa + 1Sm = 66.86$	112.17
DT-TRE	$1Pa + 3Sm + 1Mtp = 44.45$	$1Pa + 1Sm = 35.15$	79.6
CLQ-TRE	$1PSm + 1Ex = 8.23$	$1Pa + 1PSm + 1Ex = 39.94$	48.17
New-TRE	$1Pa + 2Sm + 1Mtp = 41.01$	$1Pa + 1Sm = 35.15$	76.16

3.2. Communication Cost

The protocols' communication complexity depends on the bit-length of the transmitted public keys and the ciphertext space. As mentioned previously, in BC-TRE and CLQ-TRE the users' public keys consist of two EC points, while the rest use a single Diffie-Hellman public key. 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 HYL-TRE, DT-TRE and New-TRE schemes.

As for the ciphertext space, all of the TRE schemes under comparison require an EC point and a ℓ-bit string to be transmitted; the HYL-TRE scheme requires an additional EC point and pairing value, and the DT-TRE requires an additional EC point. The value of ℓ is the same for all schemes; it is either $\ell = n + \tau$ or $\ell = n + \tau + q$, depending on whether or not the Fujisaki-Okamoto transformation [12] is applied, where n is the length of the cleartext message, q is the security parameter (e.g., 160-bits) and τ is the bit-length of the string used to represents a time instant. We emphasize that in order to provide sufficient security, an EC point must be represented using at least 160 bits; a pairing value should be alloted approximately 1024 bits. To summarize, New-TRE has the lowest communication cost, while HYL-TRE has the highest one, in large part because of the pairing value being transmitted.

3.3. Encrypting using multiple time-servers

New-TRE can be easily modified to support multiple time servers, similarly to the BC-TRE scheme. Suppose that there are N time-servers, each using a secret key s_i and a generator $P_i \in \mathbb{G}_1$, where $i = 1..N$. Then, their corresponding server public keys are $P_i, S_i = s_iP_i$, and the trapdoors are of the form s_iT (where $T = H_1(t)$).

In the special case where all servers use the same generator $P_i = P$ (this could well be the case if, for example, the time-servers follow NIST recommendations for chosen ECs and generators), the additional computational burden for New-TRE is negligible. To encrypt a message m using i multiple servers, a sender follows the steps of the basic single-server protocol, but computes $K = \hat{e}(\sum S_i, Q) = \hat{e}(P,T)^{r\sum s_i}$ using the common generator, P. On the receiver side, once the trapdoors (s_iT) are published, the receiver can also compute $K = (R, \sum s_iT) = \hat{e}(P,T)^{r\sum s_i}$.

Thus, when all time-servers use the same generator, New-TRE is much faster than BC-TRE (the only protocol of those surveyed that explicitly handles multiple time-servers): with N servers, the additional computation required under New-TRE is only $N-1$ scalar additions in $\mathbb{G}_1$. There is no additional communication cost, besides the unavoidable retrieval of the servers' trapdoors. With BC-TRE on the other hand, a receiver's public key includes $N-1$ additional EC points due to that protocol's more complex public key format. An examination of that protocol reveals that a sender must perform an additional $2N$ pairings to verify the authenticity of each of an unknown receiver's public keys, and an additional $N-1$ scalar multiplications (these correspond to EC points which are appended to the ciphertext). The receiver must perform $N-1$ additional pairings to compute the decryption key.

If the time-servers have chosen different generators, a New-TRE receiver would need to send public keys of the form aP_i, for each different P_i, $i = 1..N$. In that case the additional computational and communication cost of New-

TRE are the same as those of BC-TRE.

It is important to note that in BC-TRE, it is the receiver who in effect chooses the time-servers, because the time-server's public key is used to create the receiver's public key. In our approach, a sender can use any time-server of his choice, and any number of them, without restrictions, which may increase the level of trust in transactions undertaken via New-TRE.

4. Conclusions

We have presented a new anonymous TRE protocol, inspired by that in [3], and compared it to other well-known TRE schemes from the recent literature. Ours has the lowest computational cost when sending data to unknown receivers, and is as good as [16] and [3] in decryption cost for both known and unknown receivers. In terms of communication cost, our protocol has an advantage overall, because it combines the best features of [16, 10] (simple public-key format) and [3, 7] (small ciphertext space) together in one approach. Our approach offers excellent scalability in multiple time-server settings. In particular, if all time-servers employ the same group generator, the additional cost is negligible (one scalar addition per additional server). Extensions of our protocol to include message pre-opening capabilities and release time confidentiality are the subject of ongoing work.

References

[1] P. Barreto, H. Kim, B. Lynn, and M. Scott. Efficient algorithms for pairing-based cryptosystems. In *Advances in Cryptology CRYPTO 2002, LNCS 2442, pp. 354 - 368*. Springer-Verlag, 2004.

[2] M. Bellare and S. Goldwasser. Encapsulated key-escrow. In *Technical Report MIT/LCS/TR-688*, 1996.

[3] I. F. Blake and A. C.-F. Chan. Scalable, server-passive, user-anonymous timed release cryptography. In *25th IEEE Int'l. Conf. on Distributed Computing Systems, pp. 504-513*. IEEE Computer Society, 2005.

[4] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology EUROCRYPT 2005, LNCS 3494, pp. 440-456*. Springer-Verlag, 2005.

[5] 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, 2000.

[6] 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.

[7] J. Cathalo, B. Libert, and J.-J. Quisquater. Efficient and non-interactive timed-release encryption. In *Intl. Conf. on Information and Communications Security, LNCS 3783, pp. 291-303*. Springer-Verlag, 2005.

[8] K. Chalkias and G. Stephanides. Timed release cryptography from bilinear pairings using hash chains. In *10th IFIP CMS, pp. 130-140*. Springer-Verlag, 2006.

[9] I. Damgard. Practical and provably secure release of a secret and exchange of signatures. In *Advances in Cryptology - EUROCRYPT '93, LNCS 765, pp. 200-217*. Springer-Verlag, 1994.

[10] A. W. Dent and Q. Tang. Revisiting the security model for timed-release public-key encryption with pre-open capability. In *Cryptology ePrint Archive: Report 2006/306*, 2006.

[11] G. Di 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.

[12] 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.

[13] S. Galbraith, H. K., and S. D. Implementing the Tate pairing. In *Algorithmic Number Theory Symposium ANTS V, LNCS 2369, pp. 324 - 337*. Springer-Verlag, 2002.

[14] J. Garay and M. Jakobsson. Timed release of standard digital signatures. In *Financial Cryptography '02, LNCS 2357, pp. 168-182*. Springer-Verlag, 2002.

[15] N. Haller. The s/key one-time password system. In *http://www.rfc-archive.org/getrfc.php?rfc=1760*, 2005.

[16] 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 *Information Security Conf., LNCS 3650, pp. 344-358*. Springer-Verlag, 2005.

[17] J. Killian. Basing cryptography on oblivious transfer. In *Proc. of STOC, pp. 20-31*, 1988.

[18] W. Mao. Timed release cryptography. In *Selected Areas in Cryptography 2001, LNCS 2259, pp. 342-357*. Springer-Verlag, 2001.

[19] T. May. Timed-release crypto. In *Manuscript, http://www.hks.net/cpunks/cpunks-0/1460.html*, 1993.

[20] R. C. Merkle. Secure communications over insecure channels. In *Communications of ACM, 21(4), pp. 294-299*, 1978.

[21] V. S. Miller. The weil pairing, and its efficient calculation. In *Journal of Cryptology, volume 17, pp. 235 - 261*, 2004.

[22] 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 *Intl. World Wide Web Conf., pp. 160-169*. ACM Press, 2003.

[23] D. Nali, C. Adams, and A. Miri. Time-based release of confidential information in hierarchical settings. In *Information Security, LNCS 3650, pp. 29-43*. Springer-Verlag, 2005.

[24] I. Osipkov, Y. Kim, and J.-H. Cheon. Timed-release public key based authenticated encryption. In *http://eprint.iacr.org/2004/231*, 2004.

[25] R. Rivest, A. L. Shamir, and D. A. Wagner. Time-lock puzzles and timed-release crypto. In *MIT Laboratory for Computer Science Technical Report 684*. Massachusetts Institute of Technology, 1996.

[26] L. Shamus Software. Miracl - multiprecision integer and rational arithmetic c/c++ library. In *http://indigo.ie/ mscott*, 2006.