An Implementation Infrastructure for Server-Passive Timed-Release Cryptography

Konstantinos Chalkias, Foteini Baldimtsi, Dimitrios Hristu-Varsakelis, and George Stephanides
Computational Systems and Software Engineering Laboratory
Department of Applied Informatics, University of Macedonia, Thessaloniki, Greece
{chalkias, foteini}@java.uom.gr, {dcv, steph}@uom.gr

Abstract

As online transactions become increasingly practical, a broad range of industrial and e-government applications have emerged which depend on time-based protection of confidential data. Despite theoretical progress in timed-release cryptography (TRC), there is still no implementation infrastructure that takes advantage of the latest TRC algorithms. The purpose of this paper is to propose such an infrastructure for pairing-based timed-release cryptography (PB-TRC) systems. Our model uses key generation centers (KGCs) which publish decryption keys periodically, and satisfies the security requirements of modern third-party based TRC schemes. Our approach combines the best features of existing models into a generic and complete infrastructure which is to support TRC. It is also "lighter" in terms of complexity and communication, and is as effective (in terms of security and related properties) as the TRC protocol it is used with.

Key words: *timed-release cryptography, infrastructure, passive server, key distribution center, bilinear pairings*

1 Introduction

The aim of timed-release encryption (TRE) is to encrypt a confidential message so that the resulting ciphertext cannot be decrypted by anyone, until a specific time in the future. Early work on this problem includes [15] and [18]. TRE is useful, and oftentimes necessary, in many real world applications such as e-voting [18], sealed-bid auctions [17], e-lotteries [8] and e-contests [2]. Recently, the rapid development of identity based encryption [5], motivated newer and more innovative TRE techniques. The main advantage of modern TRE schemes is that they depend on a fully passive time-server, whose sole role is to broadcast, at the right time, a piece of information (called a "trapdoor") that is required for decrypting time-encrypted messages.

Although by now there exist a significant number of TRE protocols, each with its own desirable features, there has been little work on the infrastructure(s) which will be required in order to implement the theoretical work. One exception we are aware of is [10] whose approach is computationally expensive and requires complex certificate management and time-trapdoor distribution. In [10], the time-server is actively involved in the sender-receiver communication process, so that sender anonymity is compromised. Furthermore, there is no support for multiple time-servers; the latter feature can be used to increase the security level by making it difficult for an "impatient" receiver to collude with the (single) time server in order to gain early access to a message. The main reason for these disadvantages is that [10] is based on the public key approach of [18]; a sender encrypts a timed-release message using a public key whose corresponding private part will be broadcast by a TA at the designated time. Another approach, named the "HP Time Vault Service" [16] does not include important services such as message pre-opening, multiple time-servers, support for PB-TRE algorithms, and the ability to choose between anonymous versus authenticated encryption.

This paper's contribution is to describe a "complete" and generic infrastructure over which TRE schemes can be implemented, and which can support a variety of PB-TRE protocols, including those which are most efficient and provide important security and privacy-related properties, such as those mentioned in the previous discussion. We view this work is an essential step towards putting PB-TRE schemes to practical use. The remainder of this paper is structured as follows. In Section 2 we give a list of security-related properties that are provided by newer TRE protocols and thus should be supported by any prospective TRE infrastructure. Section 3 gives a brief description of how modern TRE protocols work and reviews the current state of the art. Section 4 discusses the proposed infrastructure and compares it with the other two existing models [10, 16]. For a fuller version of this work, see [7].

978-0-7695-3324-7/08 $25.00 © 2008 IEEE
DOI 10.1109/IAS.2008.49

2 Security Requirements of Modern TRE

Before discussing an effective infrastructure for PB-TRE, we first review the security requirements that must be supported (assuming that they are also fulfilled by the encryption protocol to be implemented). By studying real-world applications which demand TRC, the following have come to be considered more or less essential.

- Access to the content of the confidential message must be infeasible until the specified release time. The decryption key should then be available only to the authorized entities.
- Anonymity: the model should provide both sender and receiver anonymity, if desired.
- Authentication and integrity: it must be possible to verify the authenticity and the integrity of the released message.
- Non-repudiation: after the release of the message, the receiver should not be able to deny knowledge of the message content.
- Multiple time-server support: it should be possible to support the use of multiple time-servers when encrypting/decrypting, in order to eliminate, or at least reduce, the possibility of collusion between the receiver and an unscrupulous time-server.
- Pre-open capability: a sender should have the option to allow early decryption of a message by sending to the receiver a trapdoor key (different from the one to be issued by the time-server) before the designated time.
- Confidentiality of release time: there should be an option to "hide" the disclosure time.
- Public part: an application may require that part of the message be public, i.e., viewable by anyone at any time.

Each of the PB-TRE protocols proposed in the literature has its own strengths and weaknesses achieving some combination of these properties. The infrastructure proposed in this paper may be used with any existing PB-TRE protocol.

3 How PB-TRE works

When Boneh and Franklin [5] announced their work on pairing-based IBE, they mentioned TRE as one of its possible applications. In IBE there is no need for public keys to be transmitted; thus, any alphanumeric sequence that uniquely identifies a user (e.g., their e-mail address) can be used as an encryption key. Modern PB-TRE protocols take advantage of this method, to ensure server-passiveness and sender anonymity. To illustrate the main idea, assume that Alice wants to broadcast a timed-release encrypted message, with a desired disclosure time of December, 1^{st} 2008, at 12:00 noon (GMT). Alice computes the IBE public key that corresponds to this date/time (represented as a string) and encrypts the message with this key. Unlike "traditional" IBE, this public key does not correspond to a person, but rather to a time instant. A trusted time-server (henceforth called a Key Generation Center - KGC) is then responsible for constructing the matching private key needed to decrypt the message. In order for this to work, the KGC must be equipped with an accurate clock, and must publish the time-related private keys (trapdoors) reliably at the intended times (e.g., at regular intervals), to be downloaded by anyone. Thus, in PB-TRE the sender does not interact at all with the KGC, while the KGC's sole responsibility is to periodically publish trapdoors. If message pre-opening is needed then the protocol is structured so that the sender can provide a "release key" to the receiver at any time; the latter acts as a secondary trapdoor and permits the receiver to decrypt immediately.

PB-TRE schemes can be classified in two categories with respect to the construction of the time-specific trapdoors: i) those using the pairing-based short signature scheme of Boneh and Boyen [3], termed BB, and ii) those using the approach of Boneh, Lynn and Shacham [4], termed BLS. The main reason for selecting BB/BLS signatures to produce TRE trapdoors is the fact that their length is half the size of ordinary signatures, such as DSA. Because of this, BB and BLS signatures are suitable for constrained channels; the KGC only publishes a small amount of data, reducing Denial of Service (DoS) attacks. We note that trapdoors are in fact self-signed (as instances of BB or BLS) so that a KGC does not need to attach any other kind of signature.

4 Proposed Infrastructure

The proposed PB-TRE infrastructure is illustrated in Figure 1 and consists of four main components:

Network Time Protocol (NTP) Servers: their role is to enable precise control over the release time of messages by providing an absolute time reference. NTP servers use the well-known NTP protocol to achieve clock synchronization between the KGCs, the key distribution centers (KDCs, detailed below) and the users. All publicly available NTP servers [11] use Coordinated Universal Time (UTC) as their time reference. The GPS clock can be used to keep the NTP servers aligned with UTC.

In our model we allow for multiple NTP servers. We do this in order to ensure robust synchronization by providing more than one time references. Thus, even if an NTP server suffers a temporary loss of accuracy, others will be available. Moreover, each user will have the possibility of choosing the closest NTP server to interact with, reducing transmission delays.

In Figure 1, NTP servers are connected with KGCs by a dashed line. This is to indicate that the communication

between them is neither continuous nor obligatory, as the KGCs could be equipped with their own clock and only periodically contact NTPs for synchronization. On the other hand, receivers *must* contact NTPs to ensure synchronization before trying to obtain a trapdoor. If an adversary tries to attack this part of our model by delaying, for example, the packets sent to/from the NTPs, he could cause a delay of some seconds in the worst case (a very long delay would raise suspicion on the part of a user, or may mean that an NTP is unreachable). In that case, the decryption of a message would also be subject to a small delay. To protect against attackers who intercept messages from the NTP to the user and change their contents (to throw off a user's clock) some type of secure NTP must be developed, although the problem is somewhat reduced if the user polls multiple NTPs (an attacker would have to tamper with the majority of NTPs in that case).

KGCs: a Key Generation Center, also known as a time-server, periodically releases the time trapdoors necessary for decryption. During the setup phase of a TRE protocol, the KGC generates a public/private key pair (s_{pub}, s_{pr}). The private key must be stored and secured locally, as it is used for the generation of the trapdoors. KGCs are not involved in the encryption or decryption process; their sole role is to issue trapdoors and send them to the KDCs for distribution. The frequency at which that is done can be specified during the initialization of the KGC, and may be different for each KGC.

Providing for multiple KGCs in our model increases security. By forcing a user to obtain trapdoors from more than one servers to decrypt a message we eliminate the possibility of collusion between the receiver and the time-server because more than one (in some cases all) KGCs must be corrupted in order for cheating to take place. Differences in frequency of trapdoor release is another reason for needing multiple time-servers: a sender can choose the server that provides the desired accuracy with respect to decryption time. It is also important to note that each trapdoor is in fact self-authenticated and thus the system is protected against KGC duplication.

KDCs: Key Distribution Centers are mainly involved with the publication of the trapdoors issued by the KGCs. Users apply to the KDCs in order to obtain the appropriate trapdoor for decryption of their message, either via traditional browsers or with programming methods, by querying a KDC with a date, time, and the KGC name. The KDC returns the corresponding trapdoor only if it has been generated by the time-server, otherwise returns a message indicating that the trapdoor has not been published yet. The KDCs also store information on the KGCs release frequency so that senders are to choose which time-server(s) to use. The implementation of KDCs requires a database system to store past trapdoors. This information should be stored in multiple KDCs in order to avoid denial of service attacks or collapse from high user demand, and to ensure data integrity if the database of a KDC is destroyed.

We propose the existence of two categories of KDCs, one to storing current or relatively recent trapdoors, and another for storing past ones. This way, users who want to obtain a trapdoor issued in the distant past do not increase the communication load of KDCs which publish current time trapdoors.

$\mathbf{R}_i$ and $\mathbf{S}_i$: by $R_{1,\ldots,N}$ and $S_{1,\ldots,N}$ we indicate the users, receivers and senders, respectively, involved in our model. Receivers are able to communicate with NTP servers (to keep synchronized) and with KDCs (to obtain trapdoors). On the other hand, senders do not need to contact an NTP server; they only communicate with the KDCs in order to obtain information on the KGCs frequency of trapdoor release, so that they may choose the KGCs that best match their desired decryption time.

We observe that there is a two-type connection between a sender and a receiver. The one shown with a single solid line indicates a sender transmitting a TRE encrypted message to one or more receivers. Besides that, there is also a one-way connection between the two types of entities, indicating support for *pre-opening*, i.e., transmission of the release key when a sender decides to reveal his message to the receiver before the pre-determined time.

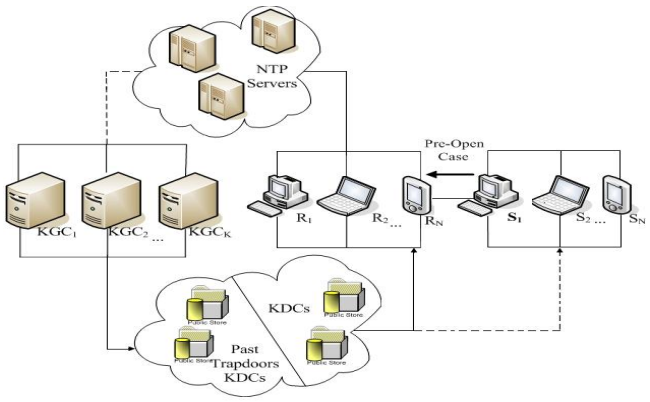

Figure 1. Proposed TRE infrastructure

4.1 Security and Other Properties

As we have previously stated, our model could be implemented using any of the PB-TRE protocols in the literature. Therefore, the choice of protocol used will largely determine which security properties hold in any given implementation. We distinguish between two main categories of properties: those attributable to the TRE protocol of choice and those which are "bound" to the infrastructure, keeping in mind that some properties may depend on both objects. Based on the choice of PB-TRE protocol, one could

guarantee various combinations of anonymity, authentication and message integrity, non-repudiation, pre-open capability, multiple KGCs support, and confidentiality of release time. These properties have been proved to hold by the authors of the protocol(s) in each case.

Besides these, the construction of our model ensures absolute and accurate disclosure time because the KGCs are designed to make infeasible the generation of decryption keys for future time instants. The role of NTP servers is essential in this context, because they ensure synchronization of the KGCs, reducing the possibility of a corrupt KGC with a counterfeit clock. Message pre-opening and multiple KGC support are also made available, if supported by the TRE protocol used in the implementation. Finally, the option to keep part of the message public is supported in our model by slightly altering the encryption algorithm of the TRE protocol to be used, so as to leave a portion of the message unencrypted. We stress the importance of incorporating this feature directly into the protocol used, and not through an external mechanism in the infrastructure. If the latter choice is made (e.g., [10]), the system becomes rather complex due to the requirement for a mechanism that provides a proof of "relation" between the ciphertext and the public part. To incorporating a "public part" feature into an existing TRE protocol, the following must take place. In addition to the encrypted message, the sender also transmits the hash of the public part of the message to the receiver. To eliminate the possibility of the message being altered en-route, that hash value should be encrypted with a secret value which the receiver can compute immediately, without waiting for any trapdoor. Such a secret key can often be the result of precomputations that the receiver performs while waiting for the message's release time. Moreover, in cases where the Fujisaki-Okamoto transformation is used, the value of the public part can be used as an input on the required hash functions.

4.2 Comparisons with Existing TRE Infrastructure Models

There are two proposed TRE infrastructures found in the literature. The first [10] is based on the TRE scheme of [18][1]. Although their approach is well-structured and practical, it leaves some work to be done in order to be used in emerging e-business and e-government applications.One of the drawbacks is the use of an external encryption module (EM) that encrypts the message on behalf of the sender. Its main purpose is to satisfy a special requirement according to which not even the sender should be able to decrypt the time-encrypted message. However, as the authors of [10] also recognize, a sender already knows the message content and does not need to decrypt it, making an EM unnecessary. Moreover, the possibility of malicious EMs in [10] must be examined further, because even though multiple EMs are to be used, parts of the message may still be revealed. This is because each EM deals with a part of the clear message without the use of any padding, so that if one EM is compromised, an attacker is able to obtain the corresponding part of the cleartext.

In [10], senders must request the appropriate time-related public key from the KGC before encrypting a message, so that they are no longer anonymous. It should be emphasized that this happens as a result of the infrastructure, in spite of the fact that the protocol to be implemented may provide anonymity. Senders' identities are known to the EMs as well. Furthermore, the infrastructure in [10] does not support some of the desired functionality, such as message pre-opening, confidentiality of release time and multiple time-server support. Finally, we note that [10] offers a sender the ability to "destroy" a time-encrypted document by asking the KGC to cancel the publication of a specific trapdoor. This feature may sometimes be useful (e.g., when a sender regrets their decision to send a TRE message) but may prove troublesome in a setting where trapdoors are universal (as in modern PB-TRE): if a trapdoor corresponding to a specific release time is somehow "cancelled", this would prevent the decryption of all messages with that same release time.

A second TRE infrastructure proposed in [16] sets up a server-passive TRE model by implementing their algorithm for QRA-based TRE. The authors have created a service in which identity based TRE (IB-TRE) is used. Their infrastructure model can be viewed as a simplified version of ours. A disadvantage of [16] is that the implementation infrastructure proposed therein is "tied" to the particular QRA-based TRE protocol described in the same work. QRA-based TRE has a high communication cost compared to pairing-based protocols. Also, there are no other QRA-based TRE protocols besides [16], so that corresponding infrastructure is not as generic or as scalable as the one described here. Finally, there is no provision for message pre-opening, confidentiality of release time, message public parts and multiple KGCs, and synchronization issues are not addressed.

In comparison, the infrastructure proposed here can be used with any of the existing PB-TRE algorithms. In each case, the system "inherits" the security features of the protocol that is being implemented, and can thus support a variety of newer security features and services. This makes our approach generic, flexible, and able to fulfill the requirements of the majority of e-applications that require delayed release of messages.

[1] We remind the reader that unlike PB-TREs, in [18] the sender has to acquire a public key (from a key distribution service or from the time-server itself) that corresponds to the desired release time.

4.3 Implementation

In the following we give an example of a typical TRE transaction between two users, Alice and Bob. Let us assume that Alice wants to time-encrypt a confidential document to Bob. At first she decides on the exact release time and date, for instance, January, 1^{st} 2008, 20:00 (GMT). The string representation of this date might be "GMT200801012000". Now, Alice must choose from a variety of PB-TRE protocols, depending on her requirements. Our model is independent of the PKI used, and supports any TRE scheme that makes use of BB or BLS trapdoor-types; it can easily operate with conventional Diffie-Hellman-like PKIs, but also with identity based [5] and certificateless [1] infrastructures. For convenience, we will assume that both users use "traditional" public keys, so that Alice must know Bob's digital certificate[2]. Alice could then choose from among the following non-IBE TRE protocols: [2, 14, 12, 13, 6, 9].

Assuming that Alice wants to send a user-anonymous message using multiple (say, three) KGCs for increased security, the multi-KGC version of [13] could be used. It is important that Alice be sure that at the designated time the selected KGCs will indeed publish time-trapdoors. To enable her to do this, before a KGC becomes active it must sign a document containing information about the first time-trapdoor to be published, the time interval between publication of each trapdoor and the trapdoor types that are published (BB, BLS or both). For instance, let us assume that a KGC started to issue 'BB and BLS'-based trapdoors at January, 1^{st} 2007, 12:00 noon (GMT), and that the period between trapdoor publications is 30 minutes. This KGC must initially sign the above information and publish this signature to KDCs. We will call this signature a "Time Policy File" (TPF) (see Figure 2). The complete list of TPFs for all KGCs will need to be downloaded once by a sender; then an update check is needed (e.g., daily, weekly). KDCs will then know when to expect time-trapdoors, while senders will be able to determine which server will broadcast time-trapdoors at the time of their choice (or as close to it as possible).

Figure 2. Illustrating the contents of a TPF

Before encrypting the message, Alice selects (from the TPF) three KGCs that will publish trapdoors on January, 1^{st} 2008, 20:00 (GMT). A KGC Chooser (KC) application can automate this procedure [7]. This software must take as inputs the predetermined time instant, the TPF file, the desired number of KGCs to be used, and the selected TRE algorithm [3]. The last input is necessary to determine the type of trapdoor, "BB or BLS"-based, because not all servers are required to publish both types. The algorithm "behind" the KC repeatedly selects a KGC and checks whether it publishes the required trapdoor-type at the desired release time, in which case the KGC will be used, until it finds the required number of KGCs.

After specifying the KGCs, Alice encrypts the message using the selected TRE algorithm and then sends the message to Bob. When Bob receives the time-encrypted message, he must wait until the release time to obtain the corresponding time-trapdoor and perform the decryption (if there exists a public part he is able to read it immediately). Of course, Bob should also synchronize his clock using the NTP protocol.

At the release time, the synchronized KGCs broadcast the corresponding time-trapdoors to the network of KDCs. Because trapdoors are self-authenticated, a KDC first verifies their validity before accepting them.

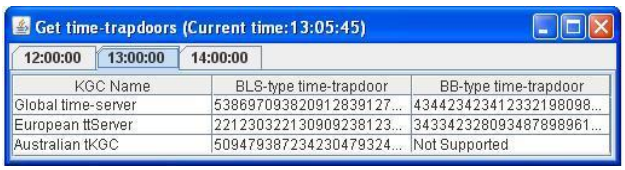

Figure 3. Interface for obtaining Trapdoors

After Bob has acquired the trapdoors from a KDC (see Figure 3), he is able to decrypt the message. We note that modern PB-TRE schemes are equipped with mechanisms that validate message integrity and are secure against chosen time and plaintext attacks (CTPA) as well as chosen time and ciphertext attacks (CTCA). The flexibility of the proposed approach makes it an ideal system for various applications. E-voting, sealed bid auctions, e-contests, e-lotteries, e-cash and e-payments, key escrow, contract signing are included in this category.

5 Conclusions and Results

This work improves upon the TRE implementation infrastructures of [10, 16] by proposing a generic, flexible infrastructure which can support new security features and desirable properties, inherited from the protocol of choice. Because of the scope of the applications which require or would benefit from TRE, any infrastructure must be sufficiently scalable to provide advanced security including user-anonymity and/or sender-authentication. Our model

[2] If IBE was to be used, Alice would produce Bob's public key only using, for example, his e-mail address.

[3] Additionally, the input may include a list of preferred KGCs.

fullfils this requirement and can work with a variety of modern TRE schemes. As in [16, 10] our approach depends on trusted authorities, called KGCs, who periodically issue self-authenticated time-specific trapdoors required for decryption. However, neither of the existing infrastructure models addresses the possibility of multiple KGCs, and are thus vulnerable to sender-KGC collusion. This reduces their applicability in applications such as e-voting and blind auctions, for example. To circumvent the problem of collusion, we propose a network of passive KGCs, each one with its own security parameters (e.g., private key and frequency of trapdoor publication), which can be scattered worldwide. Additionally, senders are able to select the number of KGCs whose trapdoors will be required to read a message, depending on the application and the required security level. To ensure fairness, we have also examined methods for decreasing possible delays in decryption, by synchronizing message receivers and KGCs through NTP servers, and by allowing for multiple KDCs. Current work is focused on implementation, including all related software for senders/receivers, together with a small network of KGCs and KDCs, and on embedding TRE capabilities in a number of applications including e-voting, timed-encrypted e-mail, and mobile messaging.

References

[1] S. S. Al-Riyami and K. G. Paterson. Certificateless public key cryptography. In *LNCS Vol. 2894, pp. 452-473*. Springer-Verlag, 2003.

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

[3] D. Boneh and X. Boyen. Short signatures without random oracles without random oracles. In *LNCS 3027, pp. 56-73*. Springer-Verlag, 2004.

[4] D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *http://eprint.iacr.org/2005/015*, 2005.

[5] D. Boneh and M.Franklin. Identity based encryption from the weil pairing. In *LNCS 2139, pp. 213-229*. Springer-Verlag, 2000.

[6] J. Cathalo, B. Libert, and J.-J. Quisquater. Efficient and non-interactive timed-release encryption. In *LNCS 3783, pp. 291-303*. Springer-Verlag, 2005.

[7] K. Chalkias, F. Baldimtsi, D. H. Varsakelis, and G. Stephanides. A practical model for modern timed-release encryption. In *In Technical report CR-02, http://csse.uom.gr/eprints/84/01/TRE-MODEL-CR-02.pdf*, 2008.

[8] K. Chalkias and G. Stephanides. Timed release cryptography from bilinear pairings using hash chains. In *CMS '06, 10th IFIP International Conference on Communications and Multimedia Security,pp. 130-140*. Springer-Verlag, 2006.

[9] K. Chalkias, D. H. Varsakelis, and G. Stephanides. Improved anonymous timed-release encryption. In *Volume 4734, pp. 311-326*. Springer-Verlag, 2007.

[10] R. F. Custódio, J. da S. Dias, F. C. Pereira, and A. E. Notoya. Temporal key release infrastructure. In *6th Annual PKI R&D Workshop at NIST in Gaithersburg, MD*, 2007.

[11] D. Deeths and G. Brunette. Using NTP to control and synchronize system clocks part i: Introduction to NTP. In *http://www.sun.com/blueprints*, 2001.

[12] A. W. Dent and Q. Tang. Revisiting the security model for timed-release public-key encryption with pre-open capability. In *http://eprint.iacr.org/2006/306.pdf*, 2006.

[13] D. Hristu-Varsakelis, K. Chalkias, and G. Stephanides. Low-cost anonymous timed-release encryption. In *3rd International Symposium on Information Assurance and Security (IAS '07)*, 2007.

[14] Y. Hwang, D. Yum, and P. J. Lee. Timed-release encryption with pre-open capability and its application to certified e-mail system. In *LNCS 3650, pp. 344-358*. Springer-Verlag, 2005.

[15] T. May. Timed-release crypto. In *manuscript*, 1993.

[16] M. C. Monte, K. Harrison, and M. Sadler. The HP time vault service: Innovating the way confidential information is disclosed at the right time. In *International World Wide Web Conference, pp. 160-169*. ACM Press, 2003.

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

[18] R. Rivest, A. 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.