New Approaches to Timed-Release Cryptography

Ivan Osipkov[1], Yongdae Kim[1], Jung Hee Cheon[2]

[1] University of Minnesota - Twin Cities, {osipkov,kyd}@cs.umn.edu
[2] Seoul National University, Korea, jhcheon@math.snu.ac.kr

Disclaimer

This paper was submitted to NDSS 2005 dated August 23, 2004. All content within this paper has been written/researched only with the knowledge of research contained in the reference list, which reflects our knowledge of related work prior to the date of submission to NDSS. The authors did not consult any other research that may or may not have been made publicly available through such channels as Internet and did not discuss the paper with anyone working on the same topic. Any resemblance to other works which were not cited is purely coincidental and refelects the phenomenon of independent discovery. Since August 24, the authors were made aware of additional related work. In partcular, the following was discovered:
"Scalable, Server-Passive, User-Anonymous Timed Release Public Key Encryption from Bilinear Pairing" by Ian F. Blake and Aldar C-F. Chan, available at `http://eprint.iacr.org/2004/211/`. Originally submitted August 25, withdrawn August 26 and updated version resubmitted Sept 9. We did not have a chance to obtain the first version due to their withdrawal.
This work will be cited and discussed in the next updated version.

Abstract. This paper introduces new approaches to Timed-release Cryptography including authenticated timed-release encryption and timed-release digital signature scheme with their variations. All of our proposed solutions are built from single mathematical primitive, Bilinear map. This simple construction allows our solution to be flexible and efficient. The proposed schemes are easily modifiable and extendable in order to satisfy different application requirements.

1 Introduction

The goal of timed-release cryptography is to encrypt a message in such a way that the receiver cannot decrypt it until some day in the future, in a sense “send a message into the future”. A solution to this problem has immediate applications in the real world. For example, in sealed-bid auctions, a dishonest auction board member could communicate to a bidder the amounts of all bids submitted. The bidder could then raise the highest bid by $1 and win the auction at the minimum expense. If a scheme exists that will not allow the bids to be opened until bidding is closed, one could prevent such situation from occurring [20]. Another relevant application is voting: if a submitted vote cannot be read until the voting process is over, other voters will not have information on who is winning, based on which they can change their mind, and will have nothing left but follow their original inclinations. Fair signature exchange without trusted third party (TTP) can also benefit from timed-release techniques and used to solve a number of applications and will be discussed more in the related work [2].

Longer-time applications include lottery [23] and check cashing. For example, a payroll may distribute checks to their employees ahead of time in a way that each check can be cashed starting only on specific date. Or, a person may decide to submit checks with the same requirement to her mortgage company thus avoiding need to keep track of when to pay [20]. Submitting sell/buy instructions of shares to a broker which should be executed/opened only at specific times is another relevant application [8]. Other applications include release of important documents and press releases among many others. A more focused application that can benefit from timed-release crypto is delaying release of escrow keys in order to limit abuse of opening all keys at the same time, which can be used to limit government surveillance abuse of encrypted communications [1].

We should note that in some applications the message sender may not be available at the disclosure time to help release the information (he may be dead, for example), or he may be unwilling to do so (in longer time-lines), in others it may simply not be in his interests to do so (fair exchange). In such a case, one needs to devise mechanisms which do not require sender participation in message disclosure. Other applications do make the sender available/interested (short time-line) as in the cases of lottery and voting, and we can take that into account when simplifying and/or designing solutions. Next, we take a look at the taxonomy of approaches to timed-release problems and compare them.

1.1 Taxonomy of Approaches

The problem of timed-release was first mentioned by May [16] and then elaborated on by Rivest *et al* [20]. May suggested three possible directions to solve it: 1) one can simply give the message to a legal attorney with legally-binding commitment that the message will not be disclosed until certain date, 2) use cryptography with something called “beacons” and/or 3) use a time-lock puzzle approach. The first approach suffers from efficiency (storage/communication overhead) problems, even though one can distribute trust among several attorneys, and cost problems [20]. And May mentions that the third approach would need more predictability to be useful. The “beacons” direction stayed unspecified by May. The work of Rivest *et al* [20] rejuvenated interest in the problem. Let us assume that the message sender is called Alice and

the receiver is Bob, and Alice wants to send a message to Bob in such a way that Bob will not be able to open it until certain time. The possible solutions that do not require Alice's participation after initial communication were divided into two parts:
Time-lock puzzle approach. Alice would encrypt her message in a way that Bob would have to perform non-parallelizable computation non-stop for the required wait time in order to decrypt it.
Agent-based approach. Alice could use trusted agents to encrypt the message in such a way that Bob will need some secret value, published by the agents on the required date, in order to decrypt the message.

Direct application of time-lock puzzle approach to "send message into the future" suffers from immense computational cost incurred by the message receiver which make it impractical for real-life scenarios. In addition, knowing computational complexity by Alice does not allow her to estimate how long it will take for Bob to decrypt the message, since Bob's computational resources cannot be ascertained. As a result, it may turn out that either Bob will decrypt message much earlier or much later than the designated date. Also, if the "future" is, say, 50 years ahead, how can Alice estimate what the current technology state-of-art will be in 10, 20, 30 years? As it turns out, though, one can use the negative side of computational cost to an advantage when the requirement is exactly to impose this complexity on another party (idea suggested by Merkle [17]) as in the case of fair exchange without TTP and some key-escrow contexts [2, 1, 23]. Fair exchange can then be used as a building block to address other timed-release applications.

Using agent-based approach relieves Bob from performing non-stop computation, fixes the date of decryption precisely and does not require Alice to have information on Bob's capabilities. This comes at a price, though. The agents have to be trusted and they have to be available at the end of the waiting period. If the agent gives required information to Bob before the due date, or it is late and/or does not provide Bob with information on due date, Alice's and/or Bob's goals may suffer. A natural way to deal with this problem is to distribute trust among several agents, using a threshold scheme for example [20]. Threshold centralized schemes suggested by Shamir [21] and distributed versions by Pederson [19] can be adapted for this purpose. Provided that no more than a threshold number of agents are dishonest, Bob will not be able to decrypt ahead of time. And if at least threshold number are honest and available, then Bob will be able to decrypt the message on due date.

1.2 Our Contribution

The contributions of this paper are as follows: Under the agent based paradigm,

- The proposed schemes are simple, flexible and extensible
  - The schemes are based on "single" cryptographic primitive resulting in a suite of useful protocols, some of which combine several cryptographic functionalities in a single stroke.
  - The agent is of "offline" type and can be run by anybody with access to the network and a minimally capable computer. The same agent can be used for various timed-release applications and multiple independent agents can be used by user to strengthen security. Furthermore, threshold techniques can be applied to agents to increase robustness. They would require only initial brief setup interaction between the agents after which agents would again function independently.
  - The encrypter/signer need not interact with the agent or obtain any published information. And decryptor/verifier needs only one value published by the agent on the target date.
- Performance
  - Agent: Minimal communication, computation and storage overhead. The values published by agent are self-authenticating and can be spread to or replicated at other possibly untrusted servers. The agent keeps a single secret that is used to generate the published values and need not be refreshed unless it is compromised.

- User: Minimal communication and key storage overhead. Only one public/private key pair is needed for all proposed protocols. Computational overhead is comparable to or less than in the existing approaches.

Organization: The rest of the paper is organized as follows: In Section 2, we introduce related and previous work of this paper. Section 3 describes our mathematical primitive and basic architecture along with summary of notation. The major results of this paper, i.e. new solutions for timed-release cryptography are presented in Section 4 and their extensions are discussed in Section 5. Performance analysis of the proposed protocols are presented in Section 6 and an exemplary application is available in Section 7.

2 Related Work

2.1 Time-lock Puzzle Approach

We mentioned before that time-lock puzzle is not suitable to "send message into the future" if applied directly. Besides its clear inefficiency for that purpose, one has to be very careful to guard against denial-of-service attacks through use special techniques such as zero-knowledge proofs to determine computational complexity. Still, it turns out that time-lock puzzle approach, can be used indirectly to address the applications such as sealed-bid auctions and lottery through use of timed fair commitment exchange [23]. In the work of Boneh and Naor [2] on timed commitments a zero-knowledge proof protocol was used to evaluate computational complexity of decryption, and (based also on additional properties) a strongly fair signature exchange protocol was proposed using timed commitments as a building block. The work of Mao [14] has further improved on the complexity of zero-knowledge protocol to make it more suitable for real-world applications. Fair exchange of digital signatures using time-lock puzzle was extended to arbitrary signatures by Garay and Pomerance [11, 10]. In the sealed-bid auction example mentioned above: the bidder can submit encrypted bid and exchange commitment signatures. After the bidding period is over, the bidder can release his encryption key. Similarly, it can easily be applied to lottery applications. More generally, one can apply it to situations in which the sender is available and willing to participate in document disclosure. Although, this approach has definite benefit of not requiring TTP, it assumes knowledge's of another party's computational resources and for practical applications it has significant overhead even with improvements [14, 10, 2], which are inherent limitations of time-lock puzzle.

2.2 Agent-based Approach

The agent-based approach, first suggested by May [16], was elaborated on in [20]. Rivest *et al* had suggested two agent constructions. In the first scheme, the agent would perform two duties: 1) it would publish every day a digitally signed and previously undisclosed secret key and 2) it would, on request, encrypt a user's message with a future key of her choosing, time-stamp it and and sign the result. Use of one-way function similar to one-time password Lamport scheme to generate the secret keys would make the published values self-authenticating (provided the very first key is signed) and would require publishing of only the current key from which all the previous ones can be easily determined. To deal with agent trust issues, one can use a threshold scheme in such a way that the decryption key on the designated day can be constructed by combining a threshold number of any published secret values published by agents. In addition, the authors suggest use of agents as a time-stamping signing service resulting in *timed-release signatures* that prove document existence at a certain time in the past. In the second offline scheme, the agent would precompute a pair of public/private keys, would publish all public ones and would release the private keys on the required days. In this way, no interaction with the agent is needed to encrypt timed-release message, but key storage overhead and the fact that all keys past keys need to be published is a deterrent, besides the fact that time-stamping service is no longer available.

A different scheme was proposed by Di Crescenzo *et al* [8]. In their design, the sender would use public key of the receiver and the release time together to encrypt the message. The result would be encrypted using agent's public key and non-malleable encryption primitive. For receiver to decrypt the message, it will have to interact with the agent using conditional oblivious transfer protocol. If the encrypted release time is not less than agent's current time, then at the end of oblivious transfer agent decrypts the message. Otherwise, protocol fails. In both cases, agent does not learn the identity of the sender. The computational and communication overheads are logarithmic with respect to the waiting time of the message before it can be decrypted.

The latest scheme was proposed in [15] and uses the identity-based encryption based on quadratic residues [7] as a "time-capsule". This capsule can be decrypted only on a certain day when agent publishes appropriate value and can carry inside encryptions and digital signatures depending on the goals that need to be achieved. More precisely, one can replace identity in the encryption primitive with future time and the job of the agent would be each day to compute the corresponding decryption key for current day and publish it to a web-server.

3 Background

3.1 Bilinear Maps

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two abelian groups of prime order q. We will use the additive notation for $\mathbb{G}_1$ (aP denotes the P added a times for element $P \in G_1$) and the multiplicative notation for $\mathbb{G}_2$ (g^a denotes the g multiplied a times for element g of $\mathbb{G}_2$).

A map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is called an admissible bilinear map if it satisfies the following conditions:

1. Bilinearity For any $P, Q \in G_1$ and $a, b \in Z_q$, $e(aP, bQ) = e(P, Q)^{ab}$.
2. Non-degeneracy $e(P, Q) \neq 1$ for at least one pair of $P, Q \in G_1$.
3. Efficiency There exist an efficient algorithm to compute the bilinear map.

The Weil pairing can be used to construct an admissible bilinear pairing. In that case, one can take $\mathbb{G}_1$ to be a subgroup of an elliptic curve and $\mathbb{G}_2$ a subgroup of the multiplicative group of a finite field. See the details of pairings and the conditions of curves in [6].

We make several comments about $\mathbb{G}_1$, $\mathbb{G}_2$ and $e(\cdot, \cdot)$. First, the Discrete Logarithm Problem (DLP) is assumed to be hard in $\mathbb{G}_1$ and $\mathbb{G}_2$. Second, the Decisional Diffie-Hellman Problem (DHP) is easy in $\mathbb{G}_1$ since $\forall a, b, c \in \mathbb{Z}_q$, if we are given aP, bP and cP, then $c = ab$ if and only if $e(aP, bP) = e(cP, P)$. Third, the Bilinear Diffie-Hellman Problem (BDHP) to find $e(P, P)^{abc}$ from $P, aP, bP, cP \in \mathbb{G}_1$ is assumed to be hard. It is called the Bilinear Diffie-Hellman (BDH) Assumption. In other words, the BDH assumption says that $e(P, P)^{abc}$ is hard to compute unless at least one of a, b, and c is computable. If one of them is known, $e(P, P^{abc}$ is easily computed.

For the rest of the paper, we denote by P a fixed generator of $\mathbb{G}_1$. This information along with details on $\mathbb{G}_1, \mathbb{G}_2$ and the bilinear map is assumed to be publicly known and fixed for the purposes of following protocols.

3.2 Basic Architecture

The central concept of our proposal to implement timed-release cryptography is a public agent called TiPuS (Timed-release Public Server). Every unit of time, say every day, it publishes new self-authenticating information I which combines public precomputable (i.e. it can be precomputed for any future or past day by anyone) information PI and its unique private information SI. (In other words, $I = f(PI, SI)$ for some function f.) It can be published on its web page or broadcasted when broadcast channel is available. Alice can encrypt (or sign) a message using PI with an option of combining it with public key of the receiver (and/or her private key). *Only when I is published on day T*, Bob (the receiver) will be able to decrypt (or verify) the message using I and optionally his private key (and/or Alice's public key).

We implement the above setting using admissible bilinear map defined in Section 3.1. Each TiPuS chooses a secret $s \in \mathbb{Z}_q$ and publishes authenticated sP. (For example, a CA certifies sP as the public key of the TiPuS.) On day i, it publishes $sH(i)$, that corresponds to I above where $s = SI, H(i) = PI$ and $f(SI, PI) = s \cdot H(i)$. (For simplicity, we denote $H(i)$ by P_i in the rest of the paper.) This value is *self-authenticated*: namely, each user can compute $e(sP, H(i))$ and verify if it is equal to $e(P, sH(i))$, since by bilinearity $(sP, H(i)) = e(P, sH(i)) = e(P, H(i))^s$.

Every user A has private/public key pair $(SK_A, PK_A) = (a, aP)$, and aP is certified by a CA. One can authenticate if A indeed possesses A. (for example, using short signature [4]) In Section 4 and 5, we will build several timed-release cryptographic protocols from this setting. Before going into detail, we emphasize few properties of TiPuS:

- TiPuS cannot be completely untrusted. Even so, some of our protocols ensure that even TiPuS cannot decrypt messages or forge signatures of other users. (For example, see Section 4.1)
- TiPuS can be shared for multiple applications, even though it publishes only single P_i (shared among all applications and all users) for day i. A malicious TiPuS may publish P_i to a specific user B before day i and, therefore, B can decrypt the ciphertext ahead of time. To prevent that, one can combine multiple TiPuS in a single protocol. (See Section 5.1).
- Due to simple protocol structure, several TiPuSes can be used for threshold timed-release cryptography easily. (See Section 5.2)

In the rest of the paper, we introduce variety of new approaches for timed-release cryptography from the above settings.

3.3 Summary of Notation

Throughout the rest of the paper, we use the following notations:

- P: a generator in $\mathbb{G}_1$.
- T: Target date. For example, Alice will encrypt a message to be decrypted at time T.
- P_T: $H(T)$. sP_T will be published by TiPuS at time T.
- $E_K(m)$: A symmetric key encryption providing authentication also. We use Encryption-then-MAC, encrypt the plaintext to get a ciphertext C and append a MAC of C.
- $D_K(c)$: A symmetric key decryption providing verification also. It should perform Decrypt-and-Verify.
- $h : \{0,1\}^* \to \{0,1\}^n$: A collision-free hash function with output size n
- $H : \{0,1\}^* \to G_1$: A collision-free full domain hash function [4].
- $H^* : \mathbb{G}_2 \to \mathbb{Z}_q$: A collision-free full domain hash function, which can be obtained by applying a standard hash function with output $n = \lfloor \log q \rfloor$ bits to the bit encoding of elements of $\mathbb{G}_2$.
- $\mathsf{TE}^{\mathsf{T}}_{\mathsf{A} \to \mathsf{B}}(\mathsf{m})$: Authenticated Timed-release encryption of message m from user A to B, which will be decrypted at time T.
- $\mathsf{TS}^{\mathsf{T}}_{\mathsf{A}}(\mathsf{m})$: Timed-release signature of message m by user A, which will be verified at time T.

4 New Approaches to Timed-Release Cryptography

4.1 (Authenticated) Timed-Release Encryption

The proposed authenticated timed-release encryption scheme has the following properties: Alice sends a ciphertext to Bob so that 1) Timed-release confidentiality: only Bob can decrypt it only at time T, in addition 2) Message authentication: Bob can verify that the message was sent by Alice, under the assumption that 1) Public timestamp of the ciphertext: Prior to time T, the ciphertext should be timestamped and 2)Trust on TiPuS : TiPuS has to be trusted not to collaborate with others. The last assumption will be weakened in Section 5.2. Rivest, Shamir and Wagner [20] provided few motivating applications for public timed-release encryption. For example in sealed bid auction, "the bidder wants to seal her bid so that it can only be opened after the bidding period is closed" [20].

Authenticated Timed-release Encryption: $\mathsf{TE}^{\mathsf{T}}_{\mathsf{A}\to\mathsf{B}}(\mathsf{m})$

Setting: Bob's authentic public key: bP, Private key of Alice: a, Certificate of Alice containing her public key aP: $Cert_A(aP)$.

Encryption: Alice computes $K = H^*[e(sP + bP, P_T)^a]$ and sends $(E_K(m), Cert_A(aP))$ to Bob.

Decryption: at time T after verifying the Alice's certificate, Bob can compute $K = H^*[e(aP, sP_T + bP_T)]$ as in the previous scheme and thus decrypt the message.

Discussion

Security To analyze the protocol, we need some analysis on the derivation of the key K:

$$\begin{aligned} e(sP + bP, P_T)^a &= e(asP + abP, P_T) \\ &= \underbrace{e(asP, P_T)}_{(a)} \cdot \underbrace{e(abP, P_T)}_{(b)} \qquad (1) \\ &= \underbrace{e(aP, sP_T)}_{(a)} \cdot \underbrace{e(abP, P_T)}_{(b)} \qquad (2) \end{aligned}$$

Alice can encrypt the message, since it knows public keys (sP, bP) of TiPuS and Bob as well as P_T. The fact that Bob can decrypt the message can be derived from Eq. (2). Eq. (2)-a can be computed by anyone at time T, since aP is public of A and sP_T is published by TiPuS at time T. Eq. (2)-b can be computed by only Bob at any time, since abP is long-term Diffie-Hellman key shared between Alice and Bob on $\mathbb{G}_1$ and P_T is a public value. Therefore, Bob can derive the same key K and decrypt the ciphertext.

When we look at Eq. (1)-a, we can find more interesting fact. In fact, asP can be thought as the long-term Diffie-Hellman key shared between Alice and $TiPuS$. Therefore, Alice can always precompute asP and abP, while 1) **Time-lock**: Bob cannot compute Eq. (1)-a prior to time T and 2) **Confidentiality from TiPuS** : Eq. (1)-b cannot be computed by TiPuS since it is derived from long-term Diffie-Hellman key between Alice and Bob.

- Confidentiality: Assuming the BDH problem, it is hard to compute $e((s+b)P, aP_T)$ (or $e((s+b)P, P_T)^a$) from P, P_T, aP, sP, bP unless a or $s+b$ is known. Thus only collaboration of Bob and TiPuS can compute K before time T. Even after time T, TiPuS will not be able to decrypt the message without Bob's help since the second underbrace in Eq. (1)-b is an ElGamal Encryption to Bob.
- Message Authentication: When Bob receives this message, he can be assure that the message was indeed generated by Alice, since nobody knows abP.

– Non-repudiation of Alice without revealing abP: Furthermore, Alice cannot repudiate that the message was indeed generated by Alice, when TiPuS is trusted and the ciphertext is publicly time-stamped by Bob prior to time t (See below sealed-bid auction example for public time-stamping). This is quite counter-intuitive, since the security of the whole system seems to depend on asP and abP.
  Since abP can be computed by only Alice and Bob, we need to consider if Bob can forge the ciphertext. In other words, if Bob provides a "proof" that the message was indeed encrypted by Alice, it should really be the case that Alice was the encrypter.
  1. Suppose Alice decides to repudiate the submission of the ciphertext after it was sent to Bob.
  2. Bob can time-stamp the message prior to day T, and, once Alice repudiates, Bob can submit K and bP_T to the authorities.
  3. If needed, authorities would wait until day T. Then they compute $K^* = K/e(aP, sP_T)$, and verify if $K^* = e(aP, bP_T)$ and $e(bP_T, P) = e(P_T, bP)$, using Alice's public key aP and Bob's public key bP.

  We note that TiPuS is assumed to be trusted. If the above protocol verifies, it proves to the authorities that Alice indeed generated the ciphertext. Note that this non-repudiation does not affect decryption of any previous ciphertext provided by either Alice or Bob.

Efficiency Alice needs one exponentiation in $\mathbb{G}_2$ and one bilinear map computation while Bob needs one exponentiation in $\mathbb{G}_1$ and one bilinear map computation. It has almost the same performance with a public timed-release encryption.

Variations Depending on the parameter setting, this protocol can be transformed to different timed-release protocols as follows:

Public Timed-release Encryption

When: $a = r$, rP is used instead of $Cert_a(aP)$ and $b = 0$

Meaning: A public timed-release encryption has the following properties: A plaintext is encrypted by the sender, while the ciphertext can be open to public only after certain period. In this case, identity of sender does not have to be part of message and, since it should be read by any user, no recipient need to be specified.

Security: In this case, Eq. (1) $= e(rsP, P_T)$ and Eq. (2) $= e(rP, sP_T)$. Therefore, confidentiality is completely dependent on TiPuS, since it can fast-forward the time. (i.e. sP_T can be always computable by TiPuS.)

Applications : One can use this protocol as a "time-capsule" that carries additional functionality. Other applications include anonymous public "whistle-blowing": a person that collected compromising material may want to stay anonymous but thinks that the information is something the public should know about. To make sure that time-wise he is not linked to this information, he decides that it should be revealed only after a certain time.

Authenticated Public Timed-release Encryption

When: $b = 0$

Meaning: First we consider encryption of a message that would convince the decryptor that it was generated by Alice and no one else. We note that in any agent-based authenticated public timed-release encryption, [1] agent that publishes required information on a certain time has to be assumed trusted, since the message has to be decryptable by anyone and thus requires only sender's public

[1] We again note that Time-lock puzzle can still achieve this by sacrificing a lot of computational overhead.

key and the information published by agent. In particular, in any such scheme agent itself will be able to decrypt the message once it is posted. Likewise, TiPuS itself can generate the ciphertext pretending to be Alice. Therefore, in this protocol we assume that *TiPuS is indeed trusted.* This is intended for a message that will be decrypted by anyone at time T, but the sender wants to be authenticated.

Security: In this case, Eq. (1) $= e(asP, P_T)$ and Eq. (2) $= e(aP, sP_T)$. Therefore, confidentiality is completely dependent on TiPuS as before. (i.e. asP can be always computable by TiPuS.) Also it can forge messages for Alice (Since it knows asP). However, as long as TiPuS is trusted and the message is publicly timestamped before time T, this provides message authentication. However, it does not provide non-repudiation.

Designated Timed-release Encryption

When: $a = r$, rP is used instead of $Cert_a(aP)$

Meaning: This is intended for a message that will be decrypted by a specific user (say Bob) at time T and the sender does not need to be specified. `APPLICATION!!!`

Security: In this case, Eq. (1) $= e(asP, P_T) \cdot e(abP, P_T)$ and Eq. (2) $= e(aP, sP_T) \cdot e(abP, P_T)$. Therefore, confidentiality is guaranteed: No one except Alice and Bob cannot decrypt the message.

Applications: If a "whistle-blower" in a company wants to anonymously inform the authorities but may prefer if any action would be taken only after he left the company. In that case, he can send an encrypted information so that only the appropriate authorities can read it but not until a certain date. Another example includes anonymous voting. An authorized voter submits non-authenticated designed encryption of his vote so that it can be decrypted only when voting booths are closed.

Authenticated Designated Timed-release Encryption : This is the basic protocol introduced above. This is intended for a message that will be decrypted by a specific user (say Bob) at time T and the sender has to be authenticated.

Key Compromise

In the above scheme, even though TiPuS secret s is compromised, K stays secure since abP is computed only by Alice or Bob. But if Alice's secret a is compromised, all the previous messages encrypted by Alice are disclosed even prior to time T since $K = H^*[e(sP + bP, P_T)^a]$. To prevent this, we can consider the following encryption scheme:

Authenticated Timed-release Encryption against Key Compromise

Setting: Bob's authentic public key: bP, Private key of Alice: a, Certificate of Alice containing her public key aP: $Cert_A(aP)$.

Encryption: Alice computes $K = H^*[e(sP+bP, P_T)^{ar}]$ and sends $(E_K(m), rP, arP, Cert_A(aP))$ to Bob.

Decryption: At time T after verifying the Alice's certificate, Bob can compute $K = H^*[e(arP, sP_T + bP_T)]$ as in the previous scheme and thus decrypt the message.

This scheme is obtained by taking a random $r \in \mathbb{Z}_q$ and using ar and arP instead of a and aP in the authenticated timed-release encryption. Thus this scheme is secure and authenticated if arP is of right form, which is verified by $e(aP, rP) = e(P, arP)$. In that case, even if the long term secret a of Alice is compromised, the session key K is still secure since r stays secret. However if b is compromised, K is computed after time T since $K = H^*[e(arP, sP_T) * e(arP, P_T)^b]$. In other words, the receiver can always decrypt the received message after T time, but the sender can not once he encrypted and erased the random r. Also, as the previous scheme, it is secure even when TiPuS secret s is compromise.

4.2 Timed-Release Digital Signature

We design a signature generation/verification scheme with non-repudiation that can be checked starting only at time T.

Timed-release Digital Signature: $\mathsf{TS}_{\mathsf{A}}^{\mathsf{T}}(\mathsf{m})$

Settings: Private key of Alice: a, Certificate of Alice containing her public key aP: $Cert_A(aP)$.

Signature: Alice computes $r = H^*(e(asP, P_T)) \mod q$ and sends $\{m, \frac{1}{r+a+h(m)}P\}$ to Bob.

Verification: at time T, Bob computes $r = H^*(e(aP, sP_T)) \mod q$, and verifies that $e(\frac{1}{r+a+h(m)}P, aP + (r + h(m))P)$ is equal to $e(P, P)$.

Security : If we publish r_1 and r_2P with the signature where r_1 and r_2 are random splits of r (i.e. $r_1+r_2 \equiv r \mod q$), the signature scheme is merely the Boneh-Boyen signature scheme [3] since r itself is random. Boneh-Boyen scheme was proved to be secure without random oracles. Thus this scheme is secure until r stays secret (Keeping r secret is more secure than publishing r_2P and $r_1 = r - r_2$). It was proved in Section 4.1 that r is not computable by anyone except Alice before time T if TiPuS is trusted. In addition, after r is revealed (that is, on or after T), r is no longer used for a new signature generation. Hence this signature scheme is secure if $TiPuS$ is trusted. Further, even though $TiPuS$ is corrupted, this scheme is secure (against existential forgery) under a weak chosen message attack. For the detail, refer to [3].

Efficiency : Alice takes one pairing computation and one exponentiation in $\mathbb{G}_2$. Bob takes two pairing computations and one exponentiation in $\mathbb{G}_1$. Note that asP is stored by Alice as a secret key and $e(P, P)$ is precomputed and stored.

Application : In some applications the message that is sent does not (should not) be encrypted, but the action that one can take with this message should be delayed. For example, a person gives a set of future payment checks to her mortgage company [20]. In this case, checks do not need to be encrypted and, in fact, mortgage company may want to double-check that the person did not make a mistake. Using timed-release signatures, the person can sign each so that the signature becomes valid only starting with certain date preventing mortgage company from early cashing. The same applies to the payroll check distribution example mentioned in the introduction.

5 Multiple TiPuSes

In this section, we consider how multiple independent TiPuSes can be used to strengthen the security and the flexibility of the system and then extend this construction to threshold TiPuSes to provide availability.

5.1 Combining TiPuSes

Suppose Alice wants to increase security and use several independent TiPuSes instead of just one, $\{TiPuS_i : i = 1, ..., m\}$. We can achieve this by replacing s in the above schemes by $\sum_{i=1}^{m} s_i$. To use the adapted protocols, Alice and Bob will have in addition to only perform m additions in $\mathbb{G}_1$ each time a cryptographic primitive from $\{TiPuS_i\}$ is used. More precisely, we have the following identities: $sP = \sum s_iP$, $sP_T = \sum s_iP_T$

We note that the chosen agents can be completely independent, do not have to know about each other and do not have to communicate between themselves. In this way, by using multiple TiPuSes, Alice can increase probability that sP_T will not be disclosed prior to time T. If at least one chosen TiPuS is honest, Alice does achieve this objective. If required, Alice and Bob can mutually agree on which agents to use.

Still, we should notice that, in this case, if at least one TiPuS fails to publish on the due date, then Bob will not be able to perform decryption/verification. The fact that this agent failed to publish, of course, becomes public knowledge and it may be blacklisted from further participation. Still, this does not help the current message since the message cannot be decrypted/verified by Bob. To increase robustness, we discuss a threshold adaptation next.

5.2 Threshold Adaptation

To adapt the previous scheme, the agents will have to perform initial setup between themselves after which they can return to fully independent functioning. We adapt the distributed threshold protocol of Pederson [19]. The initial setup will follow this protocol.

1. `Initial Setup` Very briefly, let $f_i(z) = f_{i,0} + f_{i,1}z + ... + f_{i,k-1}z^{k-1}$ be the random polynomial chosen by $TiPuS_i$ where $s_i = f_{i,0}$. Using secret channels, every $TiPuS_i$ obtains $d_i = \sum_{j=1}^{m} f_j(i)$. See [19] for full details.
2. `Post-Setup Functioning` Each TiPuS publishes authenticated d_iP and at time T publishes d_iP_T.
3. `User Actions` To calculate sP, a user chooses a subset of k TiPuSes, call it $\{TiPuS_{i_j} : j = 1, ..., k\}$, and computes $sP = \sum_{j=1}^{k} a_j d_{i_j} P$ where the Lagrange coefficient $a_j = \prod_{h \neq j} \frac{i_h}{i_h - i_j}$. Similarly, user computes sP_T (one should replace P by P_T in the previous calculation)

To admit additional $TiPuS_{m+1}$ into the group, each $TiPuS_i, i = 1, ..., m$ sends $f_i(m+1)$ to $TiPuS_{m+1}$ using a secret channel, the new TiPuS takes a random polynomial $f_{m+1}(z) = f_{m+1,0} + f_{m+1,1}z + ... + f_{m+1,k-1}z^{k-1}$ and then sends secretly $f_{m+1}(i)$ to $TiPuS_i$. Every $TiPuS_i$ thus obtains the new $d_i = \sum_{j=1}^{m+1} f_j(i)$ and the protocol then proceeds as before.

Provided that no more than $k-1$ agents are dishonest, one cannot construct sP_T prior to time T. Using $\{TiPuS_i\}$, a user just needs any k published values to construct sP and sP_T. In this way, we can improve not only security but robustness as well. The agents can continue to publish s_iP and s_iP_T if they want also to act as before.

6 Performance

To estimate the performance of our scheme, we first present experimental results for the cost of several cryptographic primitives in Table 1. We used Miracl library v.4.8.2 [22] in P3-977 MHz with 512 Mbytes memory. In MapToPoint and Pairing, we considered a subgroup of order q in a supersingular elliptic curve E over $\mathbb{F}_p$, where p is a 512 bit prime and q is a 160 bit prime. Note that the pairing value belongs to a finite field of 1024 bits.

We summarize the number of significant operations for each scheme in Table 2. Surprisingly, the schemes with additional functionalities have similar performance to the basic one. This shows that our advanced schemes are designed at minimum additional costs and more efficient than just combining basic timed-release encryption and other primitives.

Table 1. Cost of basic operations

Function	modulus (bits)	exponent (bits)	performance (msec)
RSA(Sig/Dec)	1024	1024	4.65
RSA(Ver/Enc)	1024	16 ($e = 2^{16} + 1$)	0.36
Expo in $\mathbb{F}_p$	1024	160	3.93
Scalar Mul in EC over $\mathbb{F}_p$	160	160	3.44
BLS sign	512	160	7.33
MapToPoint	-	-	2.42
Pairing	512	-	31.71

Table 2. Significant operations needed for each scheme

Scheme	Pairing	Exp. in $\mathbb{G}_1$	Exp. in $\mathbb{G}_2$
$\mathsf{TE}^{\mathsf{T}}_{*\to *}(\cdot)$/Enc	1	1	1
$\mathsf{TE}^{\mathsf{T}}_{*\to *}(\cdot)$/Dec	1	0	0
$\mathsf{ATE}^{\mathsf{T}}_{\mathsf{A}\to *}(\cdot)$/Dec	1	0	0
$\mathsf{ATE}^{\mathsf{T}}_{\mathsf{A}\to *}(\cdot)$/Dec	1	0	0
$\mathsf{TE}^{\mathsf{T}}_{*\to \mathsf{B}}(\cdot)$/Enc	1	2	0
$\mathsf{TE}^{\mathsf{T}}_{*\to \mathsf{B}}(\cdot)$/Dec	1	1	0
$\mathsf{ATE}^{\mathsf{T}}_{\mathsf{A}\to \mathsf{B}}(\cdot)$/Dec	1	0	1
$\mathsf{ATE}^{\mathsf{T}}_{\mathsf{A}\to \mathsf{B}}(\cdot)$/Dec	1	1	0
$\mathsf{TS}^{\mathsf{T}}_{\mathsf{A}}(\cdot)$/Sig	1	1	0
$\mathsf{TS}^{\mathsf{T}}_{\mathsf{A}}(\cdot)$/Ver	2	1	0

7 Application to Sealed-Bid Auction

In a sealed-bid auction, bidders submit their bids in closed form to the auction board. Once the bidding is closed, the bids are opened and the winning bid is chosen according to some publicly known metric [9]. Examples of such auctions include government construction bidding auctions, artwork and real estate sales among others. Because of special nature of such auctions, many security considerations need to be carefully analyzed and appropriate measure should be specified [18]. One of the problems that may occur in such auctions is cheating by the auction board [12]. A bid may be opened before the closing time and communicated to another bidder who can adjust his own bid appropriately (see more details in the introduction). Thus timing considerations play a central role in such applications. We identify the following as central requirements in sealed-bid auctions:

- The bids should not be opened until the bidding is closed
- The auction board should not be able to disavow bid submission
- Bidder should not be able to repudiate his bid
- Other requirements include correct calculation of winning bid, verification by bidders among many more, but they are beyond the scope of this paper and we presently dispense with these considerations. See [9, 13, 12, 5, 18] for more details.

Use of the authenticated designated timed-release encryption can resolve the first problem and third problems. A bid encrypted with such primitive cannot be opened until the time specified by the encrypter. Once the bid is submitted, the auction board can time-stamp it. In that case, as we saw before, the bidder will not be able to repudiate his bid later on. In order to make sure that the auction board cannot disavow a bid, one can use standard cryptographic mechanisms. In particular, each submitted bid can be signed by the auction board right after the closing period but before the bids can be opened (assume for example that there is a two-day gap before the closing of auction and bid opening). The signatures can be made public and posted somewhere. We note that it may not be in the interests of the auction board to throw out bids since their contents are known yet. Once the bids can be opened, the auction will not be able to disavow a bid.

To summarize, let Alice be the bidder and Bob be the auction board. Suppose the bidding closes on day 5 and bids are opened on day 7

1. Alice encrypts the bid using authenticated designated timed-release encryption protocol using Bob's authentic public key and her private key and sends the result to Bob prior to or on day 5. The message can be decrypted only by Bob and starting with day 7 only.
2. Bob time-stamps the submitted bid using trusted time-stamping service
3. On day 6, Bob computes digital signature of the submission and posts the result to a public board. Only these bids will participate in the auction.
4. On day 7, Bob can decrypt the bids.

Due to the properties of the encryption protocol, Bob not only knows that the bid did come from Alice but he can prove it to others without revealing any private information or shared keys. That means that Alice is bound to follow-up on her bid should she win. Provided that signature of Alice's submission had been posted, Bob will not be able to drop her bid once he opens it.

We note that this example certainly not a full solution, but it shows that the primitive presented in this paper can be very beneficial in solving some of its central problems.

8 Conclusions

We have presented novel solutions for timed-release cryptography. All of the proposed schemes are constructed from single mathematical primitive, which results in simple, flexible and extensible design. The centralized agent, TiPuS has minimal communication, computation and storage overhead. Users also have mininal communication and storage overhead with comparable computation overhead. We believe our solutions can be applied to many practical applications, which require "delay of time". The proposed protocols are efficient. In addition we show how our protocols allow easily to combine several independent agents to increase security of the proposed primitives. The scheme is also very easily adaptable to threshold design in which the agents after a brief setup session return to completely independent functioning. In this way, our design allows to achieve not only security but robustness as well.

References

1. M. Bellare and S. Goldwasser. Encapsulated key escrow. In *In MIT/LCS/TR-688*, 1996.
2. M. Bellare and C. Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Proc. of Asiacrypt '00, Lecture Notes in Computer Science, Vol. 1976*, 2000.
3. D. Boneh and X. Boyen. Short signatures without random oracle model. In *Proc. of Eurocrypt '04, Lecture Notes in Computer Science, Vol. 3027*, 2004.
4. D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *Proc. of Asiacrypt '01*, 2001.
5. F. Brandt. Fully private auctions in a constant number of rounds. In *In Proceedings of the 7th Annual Conference on Financial Cryptography (FC)*, 2003.
6. J. Cha and J. Cheon. An id-based signature from gap-diffie-hellman groups. In *In Public Key Cryptography - PKC 2003*, 2003.

7. C. Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, 2001.
8. G. D. Crescenzo, R. Ostrovsky, and S. Rajagopalan. Conditional oblivious transfer and timed-release encryption. In *Proc. of Eurocrypt '99*, 1999.
9. M. K. Franklin and M. K. Reiter. The design and implementation of a secure auction service. In *Proceedings of 1995 IEEE Symposium on Security and Privacy, pp. 2-14, Oakland, California*, 1995.
10. J. Garay and C. Pomerance. Timed fair exchange of arbitrary signatures. In *In Financial Crypto*, 2003.
11. J. A. Garay and C. Pomerance. Timed fair exchange of standard signatures. In *In Financial Cryptography '02*, 2002.
12. J. T. Harkavy and H. Kikuchi. On cheating in sealed-bid auctions. In *EC'03*, 2003.
13. J. T. M. Harkavy and H. Kikuchi. Electronic auctions with private bids. In *3 rd USENIX Workshop on Electronic Commerce, Boston, Mass., pp. 61–73*, 1998.
14. W. Mao. Timed-release cryptography. In *In Selected Areas in Cryptography VIII (SAC'01)*, 2001.
15. K. H. Marco Casassa Mont and M. Sadler. The hp time vault service: Exploiting ibe for timed release of confidential information. In *WWW2003*, 2003.
16. T. May. Timed-release crypto. `http://www.cyphernet.org/cyphernomicon/chapter14/14.5.html-`.
17. R. Merkle. Secure communications over insecure channels. In *Communications of the ACM*, 1978.
18. M. Naor, B. Pinkas, and R. Sumner. Privacy preserving auctions and mechanism design. In *Proceedings of ACM Conference on Electronic Commerce, pp. 129–139*, 1999.
19. T. P. Pederson. A threshold cryptosystem without a trusted party. In *In Advances in Cryptology-Eurocrypt 91*, 1991.
20. A. S. Ronald L. Rivest and D. A. Wagner. Time-lock puzzles and time-released crypto. In *MIT laboratory for Computer Science,MIT/LCS/TR-684*, 1996.
21. A. Shamir. How to share a secret. In *In Communications of ACM, Vol. 22*, 1979.
22. Shamus Software Ltd. Miracl: Multiprecision integer and rational arithmetic c/c++ library. `http://indigo.ie/~mscott/`.
23. P. F. Syverson. Weakly secret bit commitment: Applications to lotteries and fair exchange. In *1998 IEEE Computer Security Foundations Workshop (CSFW11)*, 1998.