Conditional Oblivious Transfer and Timed-Release Encryption

Giovanni Di Crescenzo[1], Rafail Ostrovsky[2], and Sivaramakrishnan Rajagopalan[2]

[1] Computer Science Department, University of California San Diego, La Jolla, CA, 92093-0114
giovanni@cs.ucsd.edu
Work done while at Bellcore.
[2] Bell Communications Research, 445 South Street, Morristown, NJ, 07960-6438, USA
{rafail,sraj}@bellcore.com

Abstract. We consider the problem of sending messages "into the future." Previous constructions for this task were either based on heuristic assumptions or did not provide anonymity to the sender of the message. In the public-key setting, we present an efficient and secure *timed-release encryption scheme* using a "time server" which inputs the current time into the system. The server has to only interact with the receiver and never learns the sender's identity. The scheme's computational and communicational cost per request are only logarithmic in the time parameter. The construction of our scheme is based on a novel cryptographic primitive: a variant of oblivious transfer which we call *conditional oblivious transfer*. We define this primitive (which may be of independent interest) and show an efficient construction for an instance of this new primitive based on the quadratic residuosity assumption.

1 Introduction

Time is a critical aspect of many applications in distributed computer systems and networks. Among other things, time is used to co-ordinate remote actions, to guarantee and monitor services, and to create linear order in some distributed transactions systems. Roughly speaking, applications of time in distributed systems fall in two categories: those that use relative time between events (e.g. one hour from the last reboot) and those that use absolute time (e.g. 0900 hours, May 2, 1999 GMT). We can concentrate on the second category as relative timing can be implemented using absolute time but not vice-versa. While the existence of a common view of current time is essential in systems that use absolute time, it is generally hard to implement in a distributed system – either the local clock is assumed to be an acceptable approximation of the universal time or there is a central time "server" that is available whenever needed and local clocks are periodically synchronized [24]. In some cases, the trustworthiness of the time server may be an issue as adversarial programs may try to change the value of

J. Stern (Ed.): EUROCRYPT'99, LNCS 1592, pp. 74–89, 1999.

the local clock or spoof a network clock to their advantage and this may have an unacceptable negative effect on the system. Such problems can be solved using authentication mechanisms as long as the applications only need the current time (or to be more accurate, the "latest time"). There are many applications that depend on a common assumption of an absolute time that is in the future, where, say, the opening of a document before a specified time is unacceptable. An example is the Internet programming contest where teams located all over the world need to be given access to the challenge problems at a certain time. Another example may be in trading stocks: suppose one wants to send an e-mail message from their laptop computer to a broker to sell a stock at a particular time in the future. The broker should not be able to see the message before that time and gain an unfair advantage, and yet one cannot rely on a service such as e-mail to work correctly and expeditiously if the message was sent exactly on release-time. The essence of the problem is this: the message has to be sent early but the broker should not be able to read the message before the release-time. Also, it would be preferable from a security perspective if the time server never learns the identity of the sender of the message.

The act of encrypting a document so that it cannot be read before a release-time has been called "sending information in to the future" or timed-release cryptography by May [22]. Rivest, Shamir and Wagner [27] give a number of applications for timed-release cryptography: electronic actions, sealed bids and chess moves, release of documents (like memoirs) over time, payment schedules, press releases, etc. Bellare and Goldwasser [2] propose that timed release may also be used in key escrow: they suggest that the delayed release of escrowed keys may be a suitable deterrent in some contexts to the possible abuse of escrow.

Prior techniques: There are two main approaches suggested in the literature: the first one is based on so-called *"time-lock puzzles,"* and the second one is based on trusted *"time-servers."* Time-lock puzzles were first suggested by Merkle [23] and extended by Rivest et al. [27]. The idea is that the time to recover a secret is given by the minimum computational effort needed by any machine, serial or parallel, to perform some computation which enables one to recover the secret. This approach has the interesting feature of not requiring any third party or trust assumption. On the other hand, it escapes a formal proof that a certain lock is valid for a certain time. The time-lock puzzle presented by Bellare and Goldwasser in [2] is based on the heuristic assumption that exhaustive search on the key space is the fastest method to recover the key of, say, 40-bit DES. In [27], Rivest et al. point out that this only works on average: for instance, for a particular key, exhaustive search may find the key well before the assigned time; they then propose a time-lock puzzle based on the hardness of factoring which does not have this problem, although it still uses a different heuristic assumption about the minimum time needed to perform some number-theoretic calculations. A major disadvantage of time-lock puzzles from our point of view is that they can only solve the relative time problem.

A "time-server" is a trusted third party that is expected to allow release of the message at the appointed time only. May [22] suggests that the third party be

a trusted escrow agent that stores the message and releases it at release-time. This does not scale well since the agent has to store all escrowed messages until their release-times. Moreover, no anonymity is guaranteed: the server knows the message, the release-time, and the identity of the two parties. In Rivest et al. [27] it was suggested that the server simply use the iterates of a one-way function (or a public-key sequence) and publish the next iterate value after one unit of time. A sender wishing to release a document at time t gets the server to encrypt the document (or a key to an encrypted version) with a secret key that the server will only publish at time t. This scheme has the advantage that the server does not have to remember anything except the seed to the sequence of one-way function iterates. This scheme requires the server to generate and publish a large number of keys, so that the overall computation and storage costs are linear in the time parameter. Furthermore, the receivers are anonymous but the sender is not anonymous in this scheme and the time server knows the release-time requested.

The Model. The general discussion above shows that it is necessary and advantageous to have a system in which the current absolute time is available with the facility of posting messages that can be read at a future time. For completeness, we first lay out some basic considerations. To begin with, time needs to be defined. In computers, as in real life, time is simply defined to be the output of a certain clock (such as the Denver Atomic Clock). We can assume the existence of such an entity which we will call the Time Server (or simply, server) that defines the time. Naturally, the server outputs (periodically or upon request) messages that indicate the current time (down to whatever granularity is needed or possible). The server is endowed with a universally known public key. However, the server is not required to remember any other keys, such as keys of senders and receivers (in this respect, our model is different from that of Rivest et al. [27]).

What we are concerned with here is timed release of electronic documents: it is straightforward to see that using encryption, we can reduce this problem to timed release of the decryption key (rather than the document itself which is assumed to be delivered ahead in encrypted form). Now, if the sender of a document is available at the time of its release, this problem is trivial since the sender can herself provide the decryption key at release-time. Thus, we can assume that the sender of the document is only present prior to, but not at, the release time. Furthermore, as May [22] suggested, if we have a trusted escrow agent that supplies the decryption key at the appointed time, again the problem is solved (but the solution does not scale well for the server). Next, the problem is also trivial if the receiver can be trusted by the sender to not decrypt the document before the release-time. Therefore, we assume that the the sender does not trust the receiver to not try to decrypt the document before the release-time. Finally, it may be that the sender is remote and hence may not be able to communicate directly with the server (this is not possible in [27]). Hence, we will also assume that the sender and server cannot interact at any time. However, the receiver can interact with the server and we will be interested in keeping this interaction to the minimum as well.

Putting all this together, the problem of timed-release encryption that we address in this paper can be restated as follows: how can a sender, without talking to the server, create a document with a release-time (defined using the notion of time as marked by the server) such that a receiver can decrypt this document only after the release-time has passed by interacting with the server and such that the server does not learn the identity of the sender?

Our results: We present a formal definition for the cryptographic task of a *timed-release encryption scheme*, and a solution to this task. Also, we introduce a new variant of the oblivious transfer protocol, which we call *conditional oblivious transfer*. We present a formal definition for this variant, and a construction for an instance of it. The properties of this construction will be crucial for the design of our timed-release encryption scheme.

Conditional Oblivious Transfer. The Oblivious Transfer protocol was introduced by Rabin [26]. Informally, it can be described as follows: it is a game between two polynomial time parties Alice and Bob; Alice wants to send a message to Bob in such a way that with probability 1/2 Bob will receive the same message Alice wanted to send, and with probability 1/2 Bob will receive nothing. Moreover, Alice does not know which of the two events really happened. There are other equivalent formulations of Oblivious Transfer (see, e.g., [8,9,15,3,21]). This primitive has found numerous applications (see, e.g., [19,16,29,17]).

In this paper, we consider a variant of Oblivious Transfer, which we call Conditional Oblivious Transfer. In this variant, Bob and Alice have private inputs and share a public *predicate* that is evaluated over the private inputs and is computable in polynomial time. The conditional oblivious transfer of (for simplicity), say, a bit b from Alice to Bob has the following requirements. If the predicate holds, then Bob successfully receives the bit Alice wanted to send him and if the predicate does not hold, then no matter how Bob plays, he will have no information about the bit Alice wanted to send him. Furthermore, no efficient strategy can help Alice during the protocol in computing the actual value of the predicate. Of course, such a game can be easily implemented as an instance of secure function evaluation [29,17,4,10,16], however, we are interested here in more efficient implementations of this particular game. To the best of our knowledge, such a variant has not been considered previously in the literature.

Timed-Release Encryption. The setting is as follows. There are three participants: the sender, the receiver and the server. First, the sender transmits to the receiver an encrypted messages and a release-time. Then, the receiver can engage in a conversation with a server. Our timed-release encryption scheme uses, in a crucial way, a protocol for conditional oblivious transfer. In particular, the server and receiver engage in a conditional oblivious transfer such that if the release-time is not less than the current time as defined by the server, then the receiver gets the message. Otherwise, the receiver gets nothing. Furthermore, the server does not learn any information about the release-time or the identity of the sender. In particular, the server does not learn whether the release-time is less than, equal to, or greater than the current time. Our protocol has minimal round-complexity: an execution of the scheme consists of a single message

from the sender to the receiver and one request-answer interaction between the receiver and the time server. Moreover, we present an implementation of our scheme, using efficient primitives and cryptosystems as building blocks, that only require communication and computation logarithmic in the size of the time parameter. Finally, we note that the trust placed on the server can be further decreased if more servers are available.

2 Notations and Definitions

In this section we present notations and definitions needed for this paper. We start with basic notations, then we define conditional oblivious transfer and timed-release encryption schemes. For the necessary number-theoretic background on quadratic residuosity and Blum integers, we refer the reader to [1,11].

2.1 Basic Notations and Model

An algorithm is a Turing machine. An *efficient* algorithm is an algorithm running in probabilistic polynomial time. An interactive Turing machine is a probabilistic algorithm with an additional communication tape. A pair of interactive Turing machines is an *interactive protocol*. The notation $x \stackrel{D}{\leftarrow} S$ denotes the *probabilistic experiment* of choosing element x from set S according to distribution D; we only write $x \leftarrow S$ in the case D is the uniform distribution over S. The notation $y \leftarrow A(x)$, where A is an algorithm, denotes the probabilistic experiment of obtaining y when running algorithm A on input x, where the probability space is given by the random coins (if any) of algorithm A. Similarly, the notation $t \leftarrow (A(x), B(y))(z)$ denotes the probabilistic experiment of running interactive protocol (A,B), where x is A's input, y is B's input, z is an input common to A and B, and t is the transcript of the communication between A and B during such execution. By $\mathrm{Prob}[R_1; \ldots ; R_n : E]$ we denote the probability of event E, after the execution of probabilistic experiments $R_1, \ldots , R_n$. Let $a \oplus b$ denote the string obtained as the bitwise logical xor of strings a and b. Let $a \circ b$ denote the string obtained by concatenating strings a and b. A language is a subset of $\{0,1\}^*$.

The predicate GE. Given two sequences of k bits $t_1, \ldots , t_k$, and $d_1, \ldots , d_k$, define predicate GE as follows: $\mathrm{GE}(t_1, \ldots , t_k, d_1, \ldots , d_k) = 1$ if and only if $(t_1 \circ \cdots \circ t_k) \geq (d_1 \circ \cdots \circ d_k)$, when strings $t_1 \circ \cdots \circ t_k$ and $d_1 \circ \cdots \circ d_k$ are interpreted as integers.

The public random string model. In the sequel, we define two cryptographic protocols: conditional oblivious transfer, and timed-release encryption, in a setting that is well known as the "public random string" model. In this model, the parties share a public and uniformly distributed string. It was introduced by Blum, De Santis, Feldman, Micali and Persiano in [6,5], and was motivated by the goal of reducing the ingredients needed for the implementation of zero-knowledge proofs. This model has been well studied in Cryptography since then as a minimal setting for obtaining non-interactive zero-knowledge proofs and several other cryptographic protocols.

2.2 Conditional Oblivious Transfer: Definition

We now give the formal definition of Conditional Oblivious Transfer.

Definition 1. Let Alice and Bob be two probabilistic Turing machines running in time polynomial in some security parameter n. Also, let x_A (x_B) be Alice's (respectively, Bob's) private input, let b be the private bit Alice wants to send to Bob, and let $q(\cdot,\cdot)$ be a predicate computable in polynomial time. We say that (Alice,Bob) is a CONDITIONAL OBLIVIOUS TRANSFER protocol for predicate q if there exists a constant a such that:

1. *Transfer Validity.* If $q(x_A, x_B) = 1$ then for each $b \in \{0,1\}$, it holds that
$$\text{Prob}\left[\sigma \leftarrow \{0,1\}^{n^a}; tr \leftarrow (\text{Alice}(x_A, b), \text{Bob}(x_B))(\sigma) : \text{Bob}(\sigma, x_B, tr) = b\right] = 1.$$
2. *Security against Bob.* If $q(x_A, x_B) = 0$ then for any Bob$'$, the random variables X_0 and X_1 are equally distributed, where, for $b \in \{0,1\}$,
$$X_b = [\,\sigma \leftarrow \{0,1\}^{n^a}; tr \leftarrow (\text{Alice}(x_A, b), \text{Bob}'(x_B))(\sigma) : (\sigma, tr)\,]$$
3. *Security against Alice.* For any efficient Alice$'$, there exists an efficient simulator M such that for any constant c, and any sufficiently large n, it holds that $|p_0 - p_1| \leq n^{-c}$, where, p_0 and p_1 are equal to, respectively,
$$\text{Prob}\left[\sigma \leftarrow \{0,1\}^{n^a}; tr \leftarrow (\text{Alice}'(x_A), \text{Bob}(x_B))(\sigma) : \text{Alice}'(\sigma, x_A, tr) = q(x_A, x_B)\right]$$
$$\text{Prob}\left[\sigma \leftarrow \{0,1\}^{n^a} : \text{M}(\sigma, x_A) = q(x_A, x_B)\right].$$

Notice that here we are defining the security against a possibly cheating Bob even if he is infinitely powerful (requirement 2), similar to [25]. In the sequel, we will also consider security with respect to a honest-but-curious Bob, meaning that Bob follows his program, but at the end may arbitrarily try to distinguish random variables X_0 and X_1. We also note that a definition suitable for the public-key setting can be easily obtained by the above one. In particular each party would use its private string too and would be given access also to the public keys of the other parties; namely, Alice and M would be given Bob's public key, and Bob would be given Alice's public key. Finally, we note that the above definition can be extended in a natural way to the case of a message containing more than one bit.

2.3 Timed-Release Encryption: Definition

First, we give an informal description. A timed release encryption scheme is a protocol between three polynomial time parties: a sender S, a receiver R and a server V. Time (a positive integer) is represented as a k-bit string and is entirely managed by V. Each message sent in an "encrypted" form from S to R will be associated with a release-time $d = (d_1, \ldots, d_k)$, where $d_i \in \{0,1\}$, for $i = 1, \ldots, 2^k$. R can check if the message it got from S is "well-formed". If

the message is "well-formed" then R is guaranteed to be able to decrypt some message after the release-time d. R is allowed to interact with V, while S never needs to interact with V. Also, for any efficient strategy of R, R can not decrypt before the release-time. Finally, V just acts as a time server; i.e., the conversation V sees reveals no information about the actual message, its release-time or which sender/receiver pair is involved in the current conversation. Time is managed by V by answering timing requests from R; namely, first R sends a message to V, then V answers. V's answer contains some information allowing R to decrypt if and only if the current time is greater than the release-time.

By (p_r, s_r) (resp., (p_v, s_v)) we will denote a pair of R's (resp., V's) public/secret keys; by σ a sufficiently long public random string; by $m \in \{0,1\}^*$ a message, by t and d the current time according to V's clock and the release-time of message m, both being represented as a k-bit string. We now present our formal definition of timed release encryption scheme.

Definition 2. Let S,R,V be three probabilistic Turing machines running in time polynomial in some security parameter n. Let $\perp$ denote a special symbol that may be output by any of S,R,V, on any input, meaning that such input is not of the specified form. We say that (S,R,V) is a TIMED-RELEASE ENCRYPTION SCHEME if there exists a constant a such that:

0. *Correctness.* For any $m \in \{0,1\}^*$ and any $d \in \{0,1\}^k$,

$$\begin{aligned}\text{Prob}\ [\ & \sigma \leftarrow \{0,1\}^{n^a}; (p_v, s_v) \leftarrow \mathrm{V}(\sigma); (p_r, s_r) \leftarrow \mathrm{R}(\sigma); \\ & (enc, d) \leftarrow \mathrm{S}(p_r, p_v, m, d); req \leftarrow \mathrm{R}(p_r, s_r, p_v, enc, d); \\ & ans \leftarrow \mathrm{V}(p_v, s_v, t, req, \sigma) : \\ & (t < d) \vee (\mathrm{R}(p_r, s_r, ans) = m)\ ] = 1.\end{aligned}$$

1. *Security against S.* For any probabilistic polynomial time S′, any constant c, and any sufficiently large n,

$$\begin{aligned}\text{Prob}\ [\ & \sigma \leftarrow \{0,1\}^{n^a}; (p_v, s_v) \leftarrow \mathrm{V}(\sigma); (p_r, s_r) \leftarrow \mathrm{R}(\sigma); \\ & (enc, d) \leftarrow \mathrm{S}'(p_r, p_v); req \leftarrow \mathrm{R}(p_r, s_r, p_v, enc, d); \\ & ans \leftarrow \mathrm{V}(p_v, s_v, t, req, \sigma) : \\ & (t < d) \vee (req = \perp) \vee (\mathrm{R}(p_r, s_r, ans) \neq \perp)\ ] \geq 1 - n^{-c}.\end{aligned}$$

2. *Security against R.* For any probabilistic polynomial time R′=(R'_1,R'_2,R'_3,R'_4), any constant c, and any sufficiently large n,

$$\begin{aligned}\text{Prob}\ [\ & \sigma \leftarrow \{0,1\}^{n^a}; (p_v, s_v) \leftarrow \mathrm{V}(\sigma); (p_r, s_r) \leftarrow \mathrm{R}'_1(\sigma, p_v); \\ & (m_0, m_1, aux) \leftarrow \mathrm{R}'_2(\sigma, p_v, p_r, s_r); i \leftarrow \{0,1\}; (enc, d) \leftarrow \mathrm{S}(p_r, p_v, m_i, d); \\ & req \leftarrow \mathrm{R}'_3(p_r, s_r, p_v, enc, d); ans \leftarrow \mathrm{V}(p_v, s_v, t, req, \sigma) : \\ & (t < d) \wedge \mathrm{R}'_4(p_r, s_r, aux, ans) = i\ ] \leq 1/2 + n^{-c}.\end{aligned}$$

3. *Security against V.* For any probabilistic polynomial time $V'=(V'_1,V'_2,V'_3)$, any constant c, and any sufficiently large n,

$$\begin{aligned}\text{Prob}\ [\ &\sigma \leftarrow \{0,1\}^{n^a}; (p_r, s_r) \leftarrow \mathrm{R}(\sigma); (p_v, s_v) \leftarrow \mathrm{V}'_1(\sigma, p_r);\\ &((m_0, d_0), (m_1, d_1), aux) \leftarrow \mathrm{V}'_2(\sigma, p_r); i \leftarrow \{0,1\};\\ &(enc, d_i) \leftarrow \mathrm{S}(p_r, p_v, m_i, d_i); req \leftarrow \mathrm{R}(p_r, s_r, p_v, enc, d_i) :\\ &\mathrm{V}'_3(p_v, s_v, t, req, aux, \sigma) = i\ ] \leq 1/2 + n^{-c}.\end{aligned}$$

Notes on Definition 2 :

- The validity of the encryption scheme is defined even with respect to malicious senders (requirement 1): even if S is malicious, and tries to send a message for which he claims the release-time to be d, then R can always decrypt after time d.
- The security against malicious R (requirement 2), and V (requirement 3) have been defined in the sense of semantic security [18] against chosen message attack. Extensions to chosen ciphertext attack can be similarly formalized.
- A scheme satisfying the above definition also protects the sender's anonymity: namely, the sender does not need to use his public or private keys when talking to the receiver, and never talks to the server.

3 A Conditional Oblivious Transfer Protocol for GE

In this section we show a conditional oblivious transfer protocol for predicate GE. Our result is the following

Theorem 3. Let GE be the predicate defined in Section 2. The protocol (Alice,Bob) presented in Section 3 is a conditional oblivious transfer protocol for GE, for which requirement 1 of Definition 1 holds with respect to any honest-but-curious and infinitely powerful Bob and requirement 3 of Definition 1 holds with respect to any probabilistic polynomial time Alice under the hardness of deciding quadratic residuosity modulo Blum integers.

The rest of this section is devoted to the proof of Theorem 3.

3.1 Our Conditional Oblivious Transfer Protocol

We first give an informal description of the ideas in our protocol and then give the formal description.

An informal description. We will use as a subprotocol (A,B) a simple variation of the oblivious transfer protocol given in [13,12], based on quadratic residuosity modulo Blum integers. For lack of space, we omit the description of such protocol but give the properties necessary for our construction here. By NQR-COT-Send$(b,(x,y))$ we denote the algorithm A on input a bit b and (x,y), where x is a Blum integer, $y \in Z_x^{+1}$. By NQR-COT-Receive$(mes,(x,p,q,y))$

we denote the algorithm B using the factors p, q of x to decode a message mes sent by A using (x, y), where the result of such decoding will be either b or $\perp$ (indicating an invalid message). We recall that in protocol (A,B), algorithm B will receive bit b sent by A if y is a quadratic non residue and the actual value of b will remain information-theoretically hidden with respect to a honest-but-curious B otherwise. Moreover, no efficient strategy allows A to guess whether B actually received the right value for b or not.

Informally, our COT protocol for the predicate GE works as follows. At the beginning of the protocol, Alice has a k-bit string $t = (t_1, \ldots, t_k)$ as her secret key and Bob has a k-bit string $d = (d_1, \ldots, d_k)$ as his secret key. Moreover, let b be the bit that Alice wants to send to Bob. First of all, Bob computes a Blum integer x, and a k-tuple $(D_1, \ldots, D_k)$ as his public key, where $D_i = r_i^2 y^{d_i} \bmod x$, where the d_i's are part of Bob's secret key. Similarly, Alice computes her public key as integers $T_1, \ldots, T_k \in Z_x^{+1}$, where T_i is a quadratic non residue if and only if $t_i = 1$. Now, one would like to use properly computed products of the T_i's and D_i's to play the role of integer y in the above mentioned protocol (A,B), where the products are computed according to the boolean expression that represents predicate GE over bit strings. A protocol implementing this would require $\Theta(k^2)$ modular multiplications. We show below a protocol which only requires $8k$ modular multiplications.

First of all Alice splits bit b into bit a and bit $a \oplus b$, for random a, and sends a using (x, T_1) and $a \oplus b$ using $(x, D_1 T_1 \bmod x)$ as inputs for the subprotocol (A,B) (notice that this allows Bob to receive b if and only if $t_1 > d_1$). Then, Alice will send a random bit c using $(x, -T_1 D_1 \bmod x)$ as input (this allows Bob to receive c if and only if $t_1 = d_1$). The gain in having the latter step is that it allows Alice to run the same scheme recursively on the tuple $(T_2, \ldots, T_k, , D_2, \ldots, D_k)$, using as input $b \oplus c$. Notice that if $t < d$ Bob will be able to compute only bits with distribution uniform and independent from b. In this protocol Alice only performs 8 modular multiplications for each i (this is because A's algorithm only requires 2 modular multiplications). We now proceed with the formal description of our scheme (Alice,Bob).

The algorithm Alice: On input $b, t_1, \ldots, t_k \in \{0,1\}$, Alice does the following:

1. **Receive:** $x, D_1, \ldots, D_k$ from Bob and set $b_1 = b$.
2. For $i = 1, \ldots, k$,
   uniformly choose $a_i, c_i \in \{0,1\}, r_i \in Z_x^*$ and compute $T_i = r_i^2 (-1)^{t_i} \bmod x$;
   if $i = k$ then set $c_i = b_i$;
   compute $mes_{i1} = \text{NQR-COT-Send}(a_i, (x, T_i))$;
   compute $mes_{i2} = \text{NQR-COT-Send}(a_i \oplus b_i, (x, D_i T_i \bmod x))$;
   compute $mes_{i3} = \text{NQR-COT-Send}(c_i, (x, -D_i T_i \bmod x))$;
   set $b_{i+1} = b_i \oplus c_i$;
   set $p_A = (T_1, \ldots, T_k)$ and $mes = ((mes_{11}, mes_{12}, mes_{13}), \ldots, (mes_{k1}, mes_{k2}, mes_{k3}))$;
   send: (p_A, mes) to Bob.

THE ALGORITHM BOB: On input a sufficiently long string σ and $d_1, \ldots, d_k \in \{0,1\}$, Bob does the following

1. Uniformly choose two n-bit primes p, q such that $p \equiv q \equiv 3 \bmod 4$ and set $x = pq$; for $i = 1, \ldots, k$, uniformly choose $r_i \in Z_x^*$ and compute $D_i = r_i^2(-1)^{d_i} \bmod x$; let $p_B = (x, D_1, \ldots, D_k)$ and **send:** p_B to Alice.
2. **Receive:** $((T_1, \ldots, T_k), (mes_{11}, mes_{12}, mes_{13}, \ldots, mes_{k1}, mes_{k2}, mes_{k3}))$ by Alice.
3. For $i = 1, \ldots, k$,
   compute $a_i = \text{NQR-COT-Receive}(mes_{i1}, (x, p, q, T_i))$;
   compute $e_i = \text{NQR-COT-Receive}(mes_{i2}, (x, p, q, D_i T_i \bmod x))$;
   if $a_i \neq \perp$ and $e_i \neq \perp$ then
      output: $a_i \oplus e_i \oplus c_{i-1} \oplus \cdots \oplus c_1$ and halt;
   else compute $c_i = \text{NQR-COT-Receive}(mes_{i3}, (x, p, q, -D_i T_i \bmod x))$;
   if $i = k$ and $c_i \neq \perp$ then **output:** c_i and halt;
   output: $\perp$.

3.2 Conditional Oblivious Transfer: The Proof

We need to prove three properties: transfer validity, security against Alice and security against Bob.

Transfer validity. Assume predicate q is true; i.e., $t_1 \circ \cdots \circ t_k \geq d_1 \circ \cdots \circ d_k$. Notice that if $t_1 > d_1$ then T_1 and $D_1 T_1 \bmod x$ are quadratic non residue and by the validity property of NQR-COT-Receive, Bob can compute a_1, e_1. and therefore b as $a_1 \oplus e_1$. Now, assume that $t_j = d_j$, for $j = 1, \ldots, i-1$ and $t_i > d_i$, for some $i \in \{1, \ldots, t\}$. Then, since $t_j = d_j$, the integer $-T_j D_j \bmod x$ is a quadratic non residue modulo x and Bob can compute c_j, for each $j = 1, \ldots, i-1$, by the validity property of NQR-COT-Receive; moreover, since $t_i > d_i$, Bob can compute both a_i and e_i from Alice's message. Since $e_i = a_i \oplus b \oplus c_1 \oplus \cdots \oplus c_{i-1}$, Bob can compute b as $a_i \oplus e_i \oplus c_1 \oplus \cdots \oplus c_{i-1}$. Finally, the case of $t_j = d_j$, for $j = 1, \ldots, k$, is similarly shown, by just observing that c_k is set equal to $b \oplus c_1 \oplus \cdots \oplus c_{k-1}$.

Security against Bob. To prove security against any honest-but-curious algorithm Bob$'$, first, assume that x is a Blum integer, $D_1, \ldots, D_k \in Z_x^{+1}$, and predicate q is false; i.e., $t_1 \circ \cdots \circ t_k < d_1 \circ \cdots \circ d_k$. Consequently, for some $i \in \{1, \ldots, k\}$ it must be that $t_j = d_j$, for $j = 1, \ldots, i-1$, and $t_i < d_i$. Note that according to Alice's algorithm, b can be written as $a_1 \oplus e_1$ or as $c_1 \oplus \cdots \oplus c_{l-1} \oplus a_l \oplus e_l$, for some l. Then, since $T_j D_j$, for $j = 1, \ldots, i-1$, is a quadratic residue modulo x, for each j, it holds that at most c_j and a_j can be computed from mes_{j1}, mes_{j3}, but e_j is information-theoretically hidden given mes_{j2}; from the properties of NQR-COT-Send. Notice that both a_j and c_j are independently and uniformly distributed bits. Then, since $t_i < d_i$, Bob$'$ has no information about either a_i or c_i; this guarantees that even for any $i' > i$ such that $t_{i'} > d_{i'}$, Bob$'$ will obtain $a_{i'}$ and $a_{i'} \oplus b_{i'}$, but not b since $b_{i'} = b \oplus c_1 \oplus \cdots \oplus c_{i'-1}$. Moreover, even for such values i', the values received by Bob$'$ are again independently and uniformly distributed bits. Hence, for any b, Bob$'$ only sees uniform and independent bits. Therefore, the two variables X_0, X_1 are equally distributed.

Security against Alice. Notice that Alice's role in the protocol consists of a single message to Bob. Therefore, if after the protocol, Alice has a non-negligible advantage over any efficient simulator M in deciding the predicate q, then she has the same advantage when she is given only Bob's public message p_B before running the protocol. Therefore, there exists an efficient M that has the same advantage in deciding predicate q as Alice. Finally, using a standard hybrid argument, M has a non-negligible advantage in deciding the quadratic residuosity modulo the Blum integer x of one of the D_i's, and therefore any $y \in Z_x^{+1}$.

This concludes the proof of Theorem 3.

4 A Timed-Release Encryption Scheme

In this section, we present our construction of a timed-release encryption scheme which can be viewed as a transformation from any ordinary encryption scheme into a timed-release one. It uses as additional tools, a non-malleable encryption scheme and a conditional oblivious transfer protocol for the predicate GE. Our scheme can be based on several different intractability assumptions, according to what goal one is interested in (i.e., generality of the assumptions, efficiency in terms of communication, and efficiency in terms of computation). We discuss all these variants after our formal description and proof. Our result is the following

Theorem 4. The scheme (S,R,V) defined in Section 4.1 is a timed-release encryption scheme.

4.1 Description of Our Scheme

We start with an informal description of the ideas needed for our scheme. A first idea for the construction of our scheme would be to use the conditional oblivious transfer designed in Section 3 as follows. Assume the receiver has obtained the release-time $d = (d_1, \ldots, d_k)$ of the message from the sender. Since the server has the current time $t = (t_1, \ldots, t_k)$, the server and the receiver can execute the conditional oblivious transfer protocol for predicate GE, where the server plays as Alice on input t and the receiver plays as Bob on input d. Additionally, the receiver, by running this protocol should get the information required to decrypt and compute the message. The properties of conditional oblivious transfer guarantee that the receiver will be able to receive some private information if and only if the time of the receiver's request was not earlier than the release-time. First, we have to decide what secret information should be sent from the server to the receiver in the event that the release-time is past. This can be as follows: the sender will first encrypt the message using the receiver's public key, and then encrypt this encryption using the server's public key. Let z_0 be the resulting message. Then the private information sent by the server to the receiver could be the decryption of z_0 under the server's public key. Note this is the encryption of the message under the receiver's key and therefore this would

give the receiver a way to compute the message. Moreover, the sender does not get any information about the message since he only sees an encryption of it.

A second issue is about the release-time. So far, we have assumed that the receiver encrypts the same release-time that he obtains from the sender, and the server uses those encryptions for the conditional oblivious transfer. However, a malicious receiver could simply replace the release-time with an earlier one and obtain the message earlier. Now, a first approach to prevent this is the following: the server will compute the bit-by-bit encryption of the release-time needed for the conditional oblivious transfer and send it to the receiver, together with a further encryption of it under the server's public key. Let z_1 be the resulting message. The idea would be that the receiver will be required to send z_1 to the server so that the server can verify that he is using the right encryptions. Still, the receiver can compute a faked encryption z_1' and repeat the same attack as before. However, now we can have the sender encrypt under the server's key the concatenation of the encryption of the release-time (under the receiver's key) and the encryption of the message (under the receiver's key). In other words, z_0 and z_1 are actually merged into a single encryption z. Now, the only attack the receiver can try is to modify z into something which may be decrypted as encryptions of the same message and a different release-time. However, this can be prevented by requiring that the encryption scheme of the server is non-malleable. the preceding discussion gives us our timed-release encryption scheme described formally below.

A formal description of our scheme. Let $(\text{nm-G}, \text{nm-E}, \text{nm-D})$ be a non-malleable encryption scheme, and denote by $(nm\text{-}pk, nm\text{-}sk)$ the pair of public and secret keys output by nm-G. Also, let (Alice,Bob) denote the conditional oblivious transfer protocol for predicate GE given in Section 3. We now describe the three algorithms S, R, V; in each algorithm, when necessary, we differentiate between the key-generation phase and the encryption/decryption phase. Also, we assume wlog that the message m to be encrypted is a single bit.

The algorithm S:

Key-Generation Phase: no instruction required.

Encryption Phase:

1. Let (m, d) be the pair message/release-time input to S, where $m \in \{0, 1\}$;
2. let (x) be R's public key;
3. uniformly choose $r \in Z_x^*$ and compute $c_m = r^2(-1)^m \bmod x$;
4. let $d = d_1, \ldots, d_k$, where $d_i \in \{0, 1\}$, for $i = 1, \ldots, k$;
5. for $i = 1, \ldots, k$, uniformly choose $r_{2i} \in Z_x^*$ and compute $D_i = r_{2i}^2(-1)^{d_i} \bmod x$;
6. let $c_d = (D_1, \ldots, D_k)$;
7. compute $cc = \text{nm-E}(nm\text{-}pk, c_d \circ c_m)$ and **output:** (cc, c_d, d).

The algorithm R:

Key-Generation Phase:

1. Uniformly choose two $n/2$-bit primes p, q such that $p \equiv q \equiv 3 \bmod 4$ and set $x = pq$;
2. let L be the language $\{x \mid x \text{ is a Blum integer }\}$;

3. using σ, p, q, compute a non-interactive zero-knowledge proof Π for L;
4. **output:** (x, Π).

Decryption Phase:

1. Let (cc, c_d, d) be the triple received by S;
2. let $d = d_1, \ldots, d_k$, where $d_i \in \{0, 1\}$, for $i = 1, \ldots, k$;
3. for $i = 1, \ldots, k$,
   using p, q, set $d'_i = 1$ if $(x, D_i) \in NQR$ or $d'_i = 0$ otherwise;
   if $d'_i \neq d_i$ then **output:** $\perp$ and halt.
4. run step 1 of algorithm Bob, by sending $(x, D_1, \ldots, D_k)$ to V;
5. send cc, Π to V;
6. run step 2 of algorithm Bob, by receiving $(T_1, \ldots, T_k), mes$ from V;
7. run step 3 of algorithm Bob, by decoding mes as c_m;
8. if $c_m \neq \perp$ then compute $m = \mathrm{D}(sk, pk, c_m)$ and **output:** m else **output:** $\perp$.

The algorithm V:

Key-Generation Phase:

1. Run algorithm nm-G to generate a pair $(nm\text{-}pk, nm\text{-}sk)$;
2. output: $nm\text{-}pk$.

Timing service phase:

1. run step 1 of algorithm Alice, by receiving $(x, D_1, \ldots, D_k)$ from R;
2. receive cc, Π from R;
3. verify that the proof Π is accepting;
4. compute $(c'_d, c'_m) = \text{nm-D}(nm\text{-}sk, nm\text{-}pk, cc)$;
5. if $c_d \neq c'_d$ or the above verification is not satisfied then **output** $\perp$ to R and **halt**;
6. let $t = (t_1, \ldots, t_k)$ be the current time, where $t_i \in \{0, 1\}$, for $i = 1, \ldots, k$;
7. for $i = 1, \ldots, k$, uniformly choose $r_i \in Z_x^*$ and compute $T_i = r_i^2(-1)^{t_i} \bmod x$;
8. run step 2 of algorithm Alice, by computing mes;
9. **output:** $(T_1, \ldots, T_m), mes$.

Round complexity: In the above description, we use the specific conditional oblivious transfer protocol of Section 3, based on quadratic residuosity, since this protocol shows that the entire timed-release can be implemented with minimum interaction. Notice that the sender does not interact at all with the server. Moreover, the sender only sends one message to the receiver in order to encrypt a message, after the receiver has published his public key. Finally the interaction between receiver and server is one round (after both parties have published their own public keys).

Efficiency: In the above description, we can use a generic non-malleable encryption scheme. A practical implementation would use, for instance, the scheme by Cramer and Shoup [7], that is based on the hardness of the decision Diffie-Hellman problem. Recall that the scheme in [7] requires about 5 exponentiations from its parties. The rest of the communication between sender and receiver is based on computing an encryption of the message m and release-time d, which requires at most k modular products (which is less expensive than one exponentiation, since k is the number of bits to encode time, and therefore a very small constant). Then, the interaction between server and receiver requires only $8nk$

modular products (which is about $8k$ n-bit exponentiations). We observe that the communication complexity is $12nk + n \log t$ and the storage complexity is $6n + n \log t$, where t is the soundness parameter required for the non-interactive zero-knowledge proof.

Complexity Assumptions: We remark that by using the non-malleable encryption scheme in [14], and implementing the conditional oblivious transfer protocol using well-known results on private two-party computation [28,17,16], our scheme can be implemented using any one-way trapdoor permutation.

5 Timed-Release Encryption: The Proof

We would like to prove Theorem 4. First of all observe that S,R,V run in probabilistic polynomial time; in particular the non-interactive zero-knowledge proof Π can be efficiently computed and verified using the protocol in [11]. Now we need to prove four properties: correctness, security against S, security against R and security against V. The correctness requirement directly follows from the properties of the conditional oblivious transfer and the encryption schemes used as subprotocols. We now concentrate on the remaining three properties.

Security against S. We need to show that for any probabilistic polynomial time S′, if R does not output $\perp$ and $t \geq d$ then R can compute the message m sent by S′. Notice that if R does not output $\perp$ then the release-time has a right format; then, since $t \geq d$, by the transfer validity property of the conditional transfer protocol used, R will always receive c_m and then compute m with probability 1.

Security against R. Assume that S and V are honest. Consider the following experiment for any probabilistic polynomial time algorithm $\mathrm{R}'=(\mathrm{R}'_1,\mathrm{R}'_2,\mathrm{R}'_3,\mathrm{R}'_4)$. Let $((x,\Pi),(p,q))$ be the pair computed by R'_1 on input σ, p_v and let (m_0,m_1) be the two messages returned by R'_2 on input σ, x, p, q. Let b a uniformly chosen bit, and let $((cc,c_d),d)$ be the output of S on input message m_b, the public key pk by R' and the public key $nm\text{-}pk$ of V. Now, let $req = (x,\Pi,cc',c'_d)$ be the request made by R'_3 to V, and let ans be V's reply at some time $t < d$. We now want to show that for any t such that $t < d$, the probability p that $\mathrm{R}'_4(x,\Pi,p,q,ans) = b$ is at most $1/2 + n^{-c}$, for any constant c and all sufficiently large n. We divide the proof in three cases.

Case 1: $cc' = cc$ and $c'_d = c_d$. Assume that there exists a t such that $t < d$ and the above probability p is at least $1/2 + n^{-c}$, for some constant c and infinitely many n. Now, we explain how to turn R′ into an algorithm B′ that can break the scheme (nm-G, nm-E, nm-D). The idea is of course to simulate an entire execution of the protocol, and then use R′ in order to break the mentioned scheme. Specifically, B′ uses R'_1 in order to generate the pair of public/private keys. Now, given the two messages m_0, m_1 output by R'_2 as candidates to be encrypted using the timed release encryption scheme, B′ will compute the two messages $nm\text{-}m_0, nm\text{-}m_1$ that are the candidates to be encrypted under the non-malleable scheme. These two messages are computed by encrypting the messages m_0, m_1, respectively, using the public key output by R'_1. Now, a bit b is uniformly

chosen in the attack experiment associated to the non-malleable scheme, and $nm\text{-}m_b$ is encrypted using such encryption scheme (this is the encryption cc). This automatically chooses message m_b in the experiment associated with the timed release scheme. Now, B′ uses R'_3 to send a request (x, Π, cc', c'_d) to V; recall that we are assuming that $cc = cc'$ and $c_d = c'_d$, therefore, B′ will now simulate the server using the assumed time t. Notice that he does not need to know the string c_m that is part of the decryption of cc' since when $t < d$, by Property 2 of the conditional oblivious transfer (Alice,Bob), the receiver is only obtaining transfers of uniformly distributed bits, which are therefore easy to simulate. Finally, B′ runs algorithm R'_4 on the (simulated) answer obtained by V. Now, notice that since the simulation of V is perfect, the probability that B′ breaks the non-malleable encryption scheme is the same as p, which contradicts the security of scheme (nm-G, nm-E, nm-D).

Case $cc' = cc$ and $c'_d \neq c_d$. This case cannot happen, since V decrypts cc' as (c_d, c_m) and can see that $c'_d \neq c_d$, and therefore outputs $\perp$ and halts.

Case $cc' \neq cc$. This case contradicts the non-malleability of the scheme used by V. This is because given history c_d about plaintext $pl = (c_d, c_m)$, and ciphertext cc, R′ is able to compute a ciphertext cc' of some related plaintext $pl' = (c_d, c'_m)$, i.e., such that c'_m is a valid encryption of m under the key of R′. The fact that c'_m is a valid encryption of m under such key is guaranteed by our original contradiction assumption that R′ successfully breaks the timed release encryption scheme.

Security against V. We see that the server V receives a tuple (x, Π, cc, c_d), and he can decrypt cc as (c_m, c'_d) and check that $c'_d = c_d$. Namely, he obtains encryptions of the message m and the release-time d under the receiver's key. The semantic security of the encryption scheme used guarantees that the server does not obtain any additional information about m, d. Moreover, notice that the tuple (x, Π, cc, c_d) is independent from the sender's identity and the receiver's identity. Therefore, V does not obtain any information about the sender's or the receiver's identity either.

This concludes the proof of Theorem 4.

References

1. E. Bach and J. Shallit, *Algorithmic Number Theory*, MIT Press, 1996.
2. M. Bellare and S. Goldwasser, *Encapsulated Key-Escrow*, MIT Tech. Report 688, April 1996.
3. G. Brassard, C. Crépeau, and J.-M. Robert, *Information Theoretic Reductions among Disclosure Problems*, in Proc. of FOCS 86.
4. M. Ben-or, S. Goldwasser, and A. Wigderson, *Completeness Theorems for Non-Cryptographic Fault-Tolerant Distributed Computation*, in Proc. of STOC 88.
5. M. Blum, A. De Santis, S. Micali, and G. Persiano, *Non-Interactive Zero-Knowledge*, SIAM Journal of Computing, vol. 20, no. 6, Dec 1991, pp. 1084–1118.
6. M. Blum, P. Feldman, and S. Micali, *Non-Interactive Zero-Knowledge and Applications*, in Proc. of STOC 88.

7. R. Cramer and V. Shoup, *A Practical Cryptosystem Provably Secure under Chosen Ciphertext Attack,* in Proc. of CRYPTO 98.
8. C. Crépeau, *Equivalence between Two Flavors of Oblivious Transfer*, in Proc. of CRYPTO 87.
9. C. Crépeau and J. Kilian, *Achieving Oblivious Transfer Using Weakened Security Assumptions*, in Proc. of FOCS 1988.
10. D. Chaum, C. Crepeau, and I. Damgard, *Multiparty Unconditionally Secure Protocols*, in Proc. of STOC 88.
11. A. De Santis, G. Di Crescenzo, and G. Persiano, *The Knowledge Complexity of Quadratic Residuosity Languages,* Theoretical Computer Science, vol. 132, (1994), pp. 291–317.
12. A. De Santis, G. Di Crescenzo, and G. Persiano, *Zero-Knowledge Arguments and Public-Key Cryptography,* Information and Computation, vol. 121, (1995), pp. 23–40.
13. A. De Santis and G. Persiano, *Public Randomness in Public-Key Cryptography,* in Proc. of EUROCRYPT 92.
14. D. Dolev, C. Dwork, and M. Naor, *Non-Malleable Cryptography,* in Proc. of STOC 91.
15. S. Even, O. Goldreich and A. Lempel, *A Randomized Protocol for Signing Contracts*, Communications of ACM, vol. 28, 1985, pp. 637-647.
16. O. Goldreich, *Secure Multi-Party Computation*, 1998. First draft available at `http://theory.lcs.mit.edu/~oded`
17. O. Goldreich, S. Micali, and A. Wigderson, *How to Play any Mental Game*, in Proc. of STOC 87.
18. S. Goldwasser and S. Micali, *Probabilistic Encryption*, in Journal of Computer and System Sciences. vol. 28 (1984), n. 2, pp. 270–299.
19. J. Kilian, *Basing Cryptography on Oblivious Transfer* , in Proc. of STOC 88.
20. J. Kilian, S. Micali and R. Ostrovsky *Minimum-Resource Zero-Knowledge Proofs*, in Proc. of FOCS 89.
21. E. Kushilevitz, S. Micali, and R. Ostrovsky, *Reducibility and Completeness in Multi-Party Private Computations*, Proc. of FOCS 94 (full version joint with J. Kilian to appear in SICOMP).
22. T. May, *Timed-Release Crypto*, Manuscript.
23. R.C. Merkle, *Secure Communications over insecure channels* Communications of the ACM, 21:291-299, April 1978.
24. R. Ostrovsky and B. Patt-Shamir, *Optimal and Efficient Clock Synchronization Under Drifting Clocks*, in Proc. of PODC 99, to appear.
25. R. Ostrovsky, R. Venkatesan, and M. Yung, *Fair Games Against an All-Powerful Adversary,* in Proc. of SEQUENCES 91, Positano, Italy. Final version in AMS DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 13, pp. 155–169, 1993.
26. M. Rabin, *How to Exchange Secrets by Oblivious Transfer*, TR-81 Aiken Computation Laboratory, Harvard, 1981.
27. R. Rivest, A. Shamir, and D. Wagner, *Time-Lock Puzzles and Timed-Release Crypto*, manuscript at `http://theory.lcs.mit.edu/ rivest`.
28. A.C. Yao, *Protocols for Secure Computations*, in Proc. of FOCS 82.
29. A.C. Yao, *How to Generate and Exchange Secrets*, in Proc. of FOCS 86.