Timed Fair Exchange of Standard Signatures*

[Extended Abstract]

Juan A. Garay† Carl Pomerance†

Abstract

In this paper we show how to achieve timed fair exchange of digital signatures of standard type. Timed fair exchange (in particular, contract signing) has been considered before, but only for Rabin and RSA signatures of a special kind.

Our construction follows the gradual release paradigm, and works on a new "time" structure that we call a *mirrored time-line.* Using this structure, we design a protocol for the timed fair exchange by two parties of arbitrary values (values lying on their respective mirrored time-lines). Finally, we apply the blinding techniques of Garay and Jakobsson to turn this protocol into a protocol for the timed fair exchange of standard signatures.

The length of these mirrored time-lines makes another problem apparent, which is making sure that the underlying sequence has a period large enough so that cycling is not observed. We also show how to construct these structures so that, under reasonable assumptions, this is indeed the case.

Key words: Timed-release cryptography, timed commitments, contract signing, blind signatures.

1 Introduction

The exchange of digital signatures, and, in particular, contract signing, where the signatures are on a common piece of text, constitutes an important part of any business transaction, especially in settings such as the World Wide Web, where participants do not trust each other to some extent already. Recently, considerable efforts have been devoted to develop protocols that mimic the features of "paper contract signing," especially *fairness.* A contract signing protocol, or, more generally, an exchange of digital signatures, is *fair* if at the end of the protocol, either both parties have valid signatures, or neither does. In some sense, this corresponds to the "simultaneity" property of traditional paper contract signing. That is, a paper contract is generally signed by both parties at the same place and at the same time, and thus is fair.

Early work on fair exchange of secrets/signatures focused on the gradual release of secrets to obtain simultaneity, and thus fairness [Blu83, EGL85, Gol83] (see [Dam95] for more recent results). The basic idea is that if each party alternately releases a small portion of the secret, then neither party has a considerable advantage over the other. Unfortunately, such a solution has several drawbacks in real situations. One problem is that of uncertain termination: if the protocol stops

*A shorter version of this paper is to appear in *Proc. FINANCIAL CRYPTO '03*, Gosier, Guadeloupe, January 2003, published by Springer-Verlag.

†Bell Labs – Lucent Technologies, 600 Mountain Ave, Murray Hill, NJ 07974. E-mail: {garay,carlp}@research.bell-labs.com.

prematurely and one of the participants does not receive a message, he will never be sure whether the other party is continuing with the protocol, or has stopped—and perhaps even has engaged in another contract signing protocol! The other problem is how to enforce that neither party has a considerable advantage over the other. If the method is not designed properly, the party with more powerful computational resources could abort and employ all of his resources to complete the computation (e.g., recover the signature by parallel search of the remaining bits), while it might take much longer or even be infeasible for the weaker party to do so.

These problems have recently been tackled by Boneh and Naor [BN00] using tools based on *moderately-hard problems* [DN92]. (A moderately-hard problem is one which is not computationally infeasible to solve, but also not easy.) They propose an elegant "timing" mechanism based on modular exponentiation, which is a problem believed not to be well suited for parallelization. Indeed, considerable efforts have been invested in finding efficient exponentiation algorithms and still the best methods are sequential. Another important property of this mechanism is verifiability of the amount of work (number of steps) that would guarantee that a certain value is obtained. Using this mechanism, they introduce a variety of timed primitives, including timed commitments, timed signatures, and in particular, timed contract signing, where they show how to fairly exchange Rabin and RSA signatures of a special kind (namely, with modulus that is a Blum integer that fits the time structure).

In this paper we show how to achieve fair exchange—and contract signing—of standard signatures (e.g., RSA, DSA, Schnorr) without the modulus restriction above; more specifically, of signatures that allow for blinding [Cha82]. We build on the construction for timed release of standard signatures due to Garay and Jakobsson [GJ02]; however, the timed release is a "one way" operation, from a signer to a receiver, and cannot be directly applied to achieve a fair exchange. We answer this challenge by introducing a new time structure, which we call a *mirrored time-line*. In a nutshell, Garay and Jakobsson called a time-line the [BN00] structure (vector) of the form:

$$\{g^{2^{2^i}}\}_{i=0}^{k} \pmod{N}, \tag{1}$$

for N a Blum integer and generator g satisfying certain properties, and used an undisclosed value in the vicinity of the kth point (specifically, the kth point's square root) as the signature's blinding factor. A *mirrored time-line* is basically obtained by "concatenating" a time-line with its symmetric image (in other words, in the first half of a mirrored time-line the distance between the points increases exponentially, while in the second half it *decreases* like-wise). We then design a protocol—a "walk" on the mirrored time-line—which allows each party to successively approach the other party's undisclosed value in a synchronized way. Finally, by using the blinding techniques of [GJ02], we obtain a protocol for the timed fair exchange of standard signatures.[1]

Increasing the length of these time structures poses another problem, which is making sure that the underlying sequences do not cycle, or at least that their period is large enough, otherwise no guarantees could be given that a time-line would be traversed sequentially. (In fact, this is also a problem with "regular," shorter time-lines, but our construction makes it more apparent.) We also show in this paper how to construct time-lines so that, under reasonable assumptions, the period of the underlying sequence will be large enough so that no cycling will occur. To our knowledge, although work has been done estimating the period of more general sequences [FPS01a, FPS01b], the case of the period of sequences such as the ones above has not been considered before.

[1] We observe that, as pointed out in [GJ02], these techniques are more general and can also be applied to other cryptographic functions, allowing, for example, the fair exchange of (verifiable) ciphertexts (e.g., by using ElGamal encryption); in this extended abstract we concetrate on the signatures application.

Prior and related work. We already mentioned the work on fair exchange and contract signing based on the gradual exchange approach. An alternative approach to achieve fairness instead of relying on the gradual release of secrets has been to use a trusted third party, who is essentially a judge that can be called in to handle disputes between contract signers. There is also a large body of work following this approach; see, e.g., [ASW98] and references therein. Our solution follows the former paradigm.

Regarding work on "time," or relating time to computational effort, we already mentioned the work of Dwork and Naor [DN92] on moderately-hard functions, which they used to combat junk e-mail. Timed primitives have also been designed for cryptographic key escrow. In [Sha95], Shamir suggested to escrow only partially a DES key, so that to recover the whole key, the government would need to first obtain the escrowed bits of the key, and then search exhaustively for the remaining bits. A problem with this type of approach is that the search can be parallelized, thus making it difficult to give a precise bound on the number of steps needed for the recovery. Bellare and Goldwasser [BG96, BG97] later suggested the notion of "time capsules" for key escrowing in order to deter widespread wiretapping, and where a major issue is the verification at escrow time that the right key will be recovered.

In another line of work, Rivest, Shamir and Wagner [RSW96] suggested "time-lock puzzles" for encrypting data, where the goal is to design puzzles that are "intrinsically sequential," and thus, putting computers to work together in parallel does not speed up finding the solution. They base their construction on a problem that, as we mentioned earlier, seems to satisfy that: modular exponentiation. In their work, however, no measures are taken to verify that the puzzle can be unlocked in the desired time.

Using a function similar to that of [RSW96]—specifically, the vector (1) above—Boneh and Naor [BN00] defined the notion of (verifiable) *timed commitments*, an extension to the standard notion of commitments in which a potential forced opening phase permits the receiver to recover (with effort) the committed value vithout the help of the committer. They show how to use timed commitments to improve a variety of applications involving time, including timed signatures of a special kind, and, in particular, contract signing. They show how to exchange Rabin and RSA signatures when the respective moduli coincide with the one used in vector (1).

Finally, in [GJ02] Garay and Jakobsson show how to generate vector (1)-type structures very efficiently—they call these "derived time-lines," and use them together with blinding techniques for the timed release of standard signatures.

Our work. Our contributions are two-fold:

- We achieve timed fair exchange (and contract signing) of standard signatures—specifically, of those which admit blinding (e.g., RSA, DSA, Schnorr). We achieve this in three steps. First, we extend the time structure (1) to what we call *mirrored time-lines*: points whose "distance" increases in exponential fashion, followed by points whose distance decreases likewise. Second, using this new structure, we design a protocol for the timed fair exchange by two parties of arbitrary values (values lying on their respective mirrored time-lines). Finally, we apply the blinding techniques of [GJ02] to turn this protocol into a protocol for the timed fair exchange of standard signatures.

- Using a "layered safe prime" construction, we show how under reasonable assumptions, namely, the Hardy–Littlewood version of the prime k-tuples conjecture [HL23], the period of the underlying sequence can be made large enough so that cycling will not occur in a time-line whose modulus is the product of two such safe primes.

Organization of the paper. Section 2 contains the necessary background material for the rest of the paper. The notion of mirrored time-lines, together with considerations on their periods as well as a protocol to construct them, is presented in Section 3. Section 4 is devoted to timed fair exchange. First we present the protocol for the fair exchange of arbitrary values (Section 4.1), and then we show how to turn it into a protocol for the fair exchange of signatures (Section 4.2). We conclude with some remarks on the efficiency of our protocol.

2 Preliminaries

The generalized BBS assumption. Let N be a Blum integer, i.e., $N = p_1 p_2$, where p_1 and p_2 are distinct primes each congruent to 3 mod 4. Recall the notion of a Blum-Blum-Shub (BBS) sequence $x_0, x_1, \cdots, x_n$, with $x_0 = g^2 \pmod N$ for a random $g \in \mathbb{Z}_N$, and $x_i = {x_{i-1}}^2 \pmod N$, $1 \le i \le n$. It is shown in [BBS86] that the sequence defined by taking the least significant bit of the elements above is polynomial-time unpredictable (unpredictable to the left and to the right), provided the quadratic residuosity assumption (QRA) holds. Recall also that these sequences are periodic (although not always purely periodic).

In [BN00], Boneh and Naor postulate the following generalization of unpredictability of BBS-type sequences. Let N and g be as above, and k an integer such that $l < k < u$. Then given the vector

$$< g^2, g^4, g^{16}, \ldots, g^{2^{2^i}}, \ldots, g^{2^{2^{k-1}}}, g^{2^{2^k}} > (\mathrm{mod} N), \tag{2}$$

the (l, u, δ, ϵ) generalized BBS assumption states that no PRAM algorithm whose running time is less than $\delta \cdot 2^k$ can distinguish the element $g^{2^{2^{k+1}}}$ from a random quadratic residue R^2, with probability larger than ϵ. The bound l precludes the parallelization of the computation, while the bound u the feasibility of computing square roots through factoring. We refer to [BN00] for further details on this assumption.

Time-lines. In [GJ02], Garay and Jakobsson called the partial description of the BBS sequence $\{g^{2^{2^i}}\}_{i=0}^{k} \pmod N$ as given by input vector (2) a *value-hiding time-line*, calling the value $g^{2^{2^k-1}}$ the time-line's *hidden value*. They defined the notion of a (T, t, ϵ) *time-line commitment*, which allows a committer to give the receiver a *timed* commitment to a hidden value. At a later time, she can reveal this value and prove that it's the correct one. However, in case the committer fails to reveal it, the receiver can spend time T and retrieve it.

More specifically, a (T, t, ϵ) time-line commitment consists of three phases: the **commit** phase, where the committer commits to the time-line's hidden value by giving the receiver the description of the time-line and a proof of its well-formedness; the **open** phase, where the committer reveals information to the receiver that allows him to compute the hidden value—and be convinced that this is indeed the committed value; and the **forced open** phase, which takes place when the committer refuses to execute the open phase; in this case, the receiver executes an algorithm that allows him to retrieve the hidden value, and produces a proof that this is indeed the value. A time-line commitment scheme must satisfy the following security constraints:

Binding: The value committed to is uniquely defined by the commitment. In particular, it is not possible to open up the commitment to a value other than that which will be obtained by the forced opening of the commitment (corresponding to the iterated squaring of the time-line's starting value.)

Soundness: After receiving the time-line, the receiver is convinced that the forced open phase will produce the hidden value in time T.

Privacy: Every PRAM algorithm whose running time is at most $t < T$ on polynomially many processors, given the transcript of the time-line commit protocol as input, will succeed in computing the time-line's hidden value with probability at most ϵ.

Proving the well-formedness of a time-line involves a computational effort (see [BN00], and our variant in Section 3). Garay and Jakobsson [GJ02] show how once a time-line is established and its properties verified, time-line commitments can be generated at a low cost. At a high level, the idea is for the committer to create a new time-line from the original time-line by applying a secret transformation value—the *shifting* exponent—to the end points of the master time-line, in such a way that verification of the new time-line's properties is easy. We will be using time-line commitments in Section 4. Further details can be found in [GJ02].

3 Mirrored Time-Lines

Recall that, in a nutshell, a time-line is the partial description of a BBS sequence, as given by the vector (2) in Section 2. In this section we introduce a time structure that is an extension of a time-line. This new time structure will be used by our applications of Section 4.

Let N be a Blum integer as before (i.e., $N = p_1 p_2$, where p_1 and p_2 are distinct primes each congruent to 3 mod 4) and $g \in_{\mathrm{R}} \mathbb{Z}_N$. Let $u_i \stackrel{\mathrm{def}}{=} g^{2^{2^i}} \bmod N$, $0 \le i \le K$, and $v_j \stackrel{\mathrm{def}}{=} g^{2^{2^{K+1}-2^{K-j}}} \bmod N$, $1 \le j \le K$. We call a *mirrored time-line* (MTL) the result of "concatenating" two such vectors, and throwing in the initial term g, and the final term $u_{k+1} = g^{2^{2^{K+1}}} \bmod N$, i.e.:

$$\langle\, g, \underbrace{g^2, g^4, \cdots, g^{2^{2^{K-1}}}}_{\{u_i\}_{i=0}^{K-1}}, \underbrace{g^{2^{2^K}}}_{u_K = v_0}, \underbrace{g^{2^{2^K+2^{K-1}}}, g^{2^{2^K+2^{K-1}+2^{K-2}}}, \cdots, g^{2^{2^K+2^{K-1}+\cdots+1}}}_{\{v_j\}_{j=1}^{K}}, \underbrace{g^{2^{2^{K+1}}}}_{u_{K+1}} \,\rangle \tag{3}$$

(everything mod N). In other words, in the first half of an MTL the distance between the points grows geometrically, as with original time-lines,while in the second half it *decresases* like-wise. Sometimes we will call element $u_K (= v_0)$ the *pivot* of the MTL.[2] Obviously, the length of an MTL is double that of a basic time-line—for the same K. In fact, our applications will require using larger values of K (e.g., $K \approx 80$) than the ones used by the timed applications of [BN00, GJ02] ($K \in [30, 50]$); thus, measures must be taken to guarantee that the period of the underlying sequence is large enough. In the following, we present a protocol for MTL generation between two parties, the prover (the party generating the MTL) and the verifier, with the following properties:

1. It assures the verifier that with high probability all the points in the received MTL lie on the same time-line; and

2. it guarantees the prover that the period of the underlying sequence is larger than the length of the MTL.

[2] The main application of the MTL, the fair exchange of signatures, will only be using the second half of the time-line; however, the first half of the time-line will be needed to prove the well-formedness of the second half (cf. Section 3.2).

As in [BN00, GJ02], property 1 can be achieved rather efficiently, with roughly $2K$ zero-knowledge proofs that can be run in parallel. Property 2, however, requires some additional considerations, which we outline below.

3.1 Obtaining large periods

Given a nonzero integer g and a positive integer n let $\mathrm{Per}(g,n)$ be the period of the (ultimately) periodic sequence $(g^i \bmod n)_{i\geq 0}$, let $\mathrm{Per}_1(g,n)$ be the period of $(g^{2^i} \bmod n)_{i\geq 0}$, and let $\mathrm{Per}_2(g,n)$ be the period of $(g^{2^{2^i}} \bmod n)_{i\geq 0}$. We have

$$\mathrm{Per}_2(g,n) = \mathrm{Per}(2, \mathrm{Per}_1(g,n)) = \mathrm{Per}(2, \mathrm{Per}(2, \mathrm{Per}(g,n))). \tag{4}$$

Since $\mathrm{Per}(g,n)$ is smaller than n, and in fact can be much smaller, it is not clear that a multiple layering of the Per function is adding complexity. In fact it may be that the reverse is occurring, and a sequence $(g^{2^{2^i}} \bmod n)_{i\geq 0}$ has a very short period. Thus, in using a time-line sequence as a means of imposing a given degree of computational intractability, it seems interesting to know that the period of such a sequence is suitably large.

If g and n are coprime integers, with $n > 0$, let $\mathrm{ord}(g,n)$ be the multiplicative order of g in $\mathbb{Z}_n^*$. More generally, even when g and n are not coprime, let $\mathrm{ord}^*(g,n)$ be $\mathrm{ord}(g,n^*)$, where n^* is the largest divisor of n that is coprime to g. The following result is well-known.

Lemma 3.1 *For nonzero integers g,n with $n > 0$, $\mathrm{Per}(g,n) = \mathrm{ord}^*(g,n)$.*

Proof Write $n = n_0 n^*$. By the Chinese remainder theorem, our sequence $(g^i \bmod n)_{i\geq 0}$ is equivalent to the sequence of pairs $(g^i \bmod n_0, g^i \bmod n^*)_{i\geq 0}$. For i large, $g^i \equiv 0 \bmod n_0$, so it has an ultimate period of length 1. Further, $(g^i \bmod n^*)_{i\geq 0}$ is purely periodic with period $\mathrm{ord}(g,n^*)$. Thus $(g^i \bmod n)_{i\geq 0}$ becomes periodic as soon as $g^i \equiv 0 \bmod n_0$, and the length of the period is $\mathrm{ord}^*(g,n)$. ■

■

We have at least some lower bound before a repeat in the sequence $(2^i \bmod n)_{i\geq 0}$. Indeed, if $2^j \equiv 2^i \bmod n$ for $j > i \geq 0$, then $j > \log n$. It follows then from (4) that the first $\lfloor \log\log(\mathrm{Per}(g,n)) \rfloor$ terms of the sequence $(g^{2^{2^i}})_{i\geq 0}$ are distinct.

On the other hand, since $\mathrm{Per}_2(g,n) < \mathrm{Per}_1(g,n) < \mathrm{Per}(g,n)$ (from (4) and Lemma 3.1) if we have g,n with $\mathrm{Per}_2(g,n)$ large, it follows that $\mathrm{Per}_1(g,n)$ and $\mathrm{Per}(g,n)$ are large as well.

Here are some possible strategies for choosing a pair g,n with $\mathrm{Per}_2(g,n)$ large:

- Actually compute $\mathrm{Per}_2(g,n)$ for randomly chosen parameters g,n (with n a Blum integer) to see if it is long enough.
- Ignore the problem, hoping that a randomly chosen g,n will do.
- Use *layered* safe primes as described below.
- Use a combination of the above strategies; that is, use safe primes that are not layered, or layered to only the next level.

Computing the period for randomly chosen parameters may in fact be difficult, depending on one's ability to factor various numbers that arise. In some sense, using layered safe primes allows for the computation of the period in direct fashion.

Ignoring the problem, which has been the tacit strategy till now, would work in practice if there are not many pairs g, n where $\mathrm{Per}_2(g,n)$ is very small. Such a result was shown in [FPS01a, FPS01b] for the sequence $(g^{h^k} \bmod n)_{k\geq 0}$, where g, h, n are chosen at random, with n the product of 2 primes of the same magnitude. A somewhat weaker, but still adequate result is shown in the same papers for the special case $h = 2$. It seems likely that for most Blum integers n and for most residues $g \bmod n$, $\mathrm{Per}_2(g,n) > n^{1-\epsilon}$ for any fixed $\epsilon > 0$, but this remains unproved at present. A paper which takes a step in this direction is [MP03].

A prime p is "safe" if $(p-1)/2$ is also prime. (In number theory, a prime q with $2q+1 = p$ also prime is known as a Sophie Germain prime.) It has long been recommended to use safe primes in RSA moduli in order to foil the $p-1$ factoring algorithm. However, almost surely this algorithm would be useless with random primes in an RSA modulus (see [PS95]). Despite this, since it is so easy to use safe primes, the thought then is "Why not? Let's use safe primes." The same logic can be applied to time-line sequences.

Consider integers s such that

$$s, \quad r = 2s+1, \quad q = 2r+1, \quad \text{and} \quad p = 2q+1$$

are all prime. By the Hardy–Littlewood version of the prime k-tuples conjecture [HL23], the number of such integers $s \leq x$ is

$$\sim cx/(\ln x)^4,$$

as $x \to \infty$, where

$$c = \frac{2}{3} \prod_{p \text{ prime}} \left(1 - \frac{6p^2 - 4p + 1}{(p-1)^4}\right) \approx 5.5349.$$

While it is perhaps not so easy to find large examples of such numbers s, it is not intractable. For example, in the vicinity of 2^{500}, about 1 in 26 billion integers will (conjecturally) be valid examples. A sieve may be used to isolate numbers s which are more likely to work, with a primality test to finish the job. If successful values s_1, s_2 are found, one can construct the modulus $N = p_1 p_2$ with these layered safe primes. Doing so ensures that if $\gcd(g^3 - g, N) = 1$ then the period of the sequence

$$g^{2^{2^i}} \bmod N, \quad i = 0, 1, \ldots, \tag{5}$$

is either $s_1 s_2$ or $2 s_1 s_2$, where $s_1 s_2 \approx 2^{-6} N$. Indeed, the condition $\gcd(g^3 - g, N) = 1$ implies that g is coprime to N and that $\mathrm{ord}(g, p_i) > 2$ for $i = 1, 2$. Thus, $q_1 q_2$ divides $\mathrm{ord}(g, N)$. Similarly, $r_1 r_2$ divides $\mathrm{ord}(2, q_1 q_2)$ and $s_1 s_2$ divides $\mathrm{ord}(2, r_1 r_2)$. It follows from Lemma 3.1 and (4) that $s_1 s_2$ divides $\mathrm{Per}_2(g, N)$.

We thus have the following theorem.

Theorem 1 *Assuming the Hardy–Littlewood version of the prime k-tuples conjecture there are $\sim 1.4986 \cdot 2^m/(m-3)^4$ safe primes p with m bits such that $(p-1)/2 = q$ and $(q-1)/2 = r$ are also safe primes. Further, if N is the product of any 2 of these primes p and g is any integer satisfying $\gcd(g^3 - g, N) = 1$, then the period of the sequence (5) is at least 2^{2m-8}.*

As noted above, having $\mathrm{Per}_2(g, N)$ large ensures that $\mathrm{Per}_1(g, N)$ is also large, since $\mathrm{Per}_1(g, N) > \mathrm{Per}_2(g, N)$. This then ensures that the points u_i, v_j on an MTL under consideration are all distinct. Indeed, if r_1, r_2 have 500 or more bits each (to make N hard to factor), and if $K \approx 80$ as recommended, we are only looking at a subset of the first $2^{\approx 80}$ terms of a sequence with period more than 2^{998}.

A less "high tech" way is likely to work to find Blum integers N where the period of (5) is large. Suppose q_1, q_2 are random primes near 2^{500}. In [FPS01a, FPS01b] it is shown that for random choices of g, h it is highly likely that the period of the sequence

$$g^{h^i} \bmod q_1 q_2, \quad i = 0, 1, \ldots,$$

is greater than $(q_1 q_2)^{1-\epsilon}$. It is reasonable to conjecture that the same is true for the sequence

$$2^{2^i} \bmod q_1 q_2,$$

and that this remains true if in addition we insist that both $p_1 = 2q_1 + 1, p_2 = 2q_2 + 1$ are prime. Assuming this is so, it is an easy matter then to select random primes q_1, q_2 near 2^{500} such that

(1) $p_1 = 2q_1 + 1$, $p_2 = 2q_2 + 1$ are prime, and

(2) the period of the sequence $2^i \bmod q_1 q_2$ exceeds 2^{900}.

(One can either hope that property (2) is satisfied without checking it, or one can choose the primes q_1, q_2 such that $\phi(q_1 q_2)$ is easy to factor, so that the period of the sequence might actually be checked.) Then, if $N = p_1 p_2$ and g is chosen with $\gcd(g^3 - g, N) = 1$, then $\mathrm{Per}_1(g, N) > 2^{900}$ and the first 900 terms of the sequence (5) are distinct. Both conditions give a quite comfortable margin to avoid periodicities.

In what follows, we take this last approach of using safe primes p_1, p_2 for N, with $p_i = 2q_i + 1$ for $i = 1, 2$, and with $\mathrm{ord}(2, q_1 q_2) > 2^{900}$.

3.2 The mirrored time-line protocol

We now outline the steps of the MTL protocol.

1. **Setup.** The prover chooses a modulus $N = p_1 p_2$, for p_1, p_2 safe m-bit primes as in Theorem 1. The prover then proves to the verifier in zero-knowledge that N is the product of two safe primes, using the protocol of Camenisch and Michels [CM99a]. The prover also chooses a generator g with $g^3 - g \in \mathbb{Z}_N^*$. Then the order of g in $\mathbb{Z}_N^*$ is either $2q_1q_2$ or q_1q_2, and the order of g^2 is $q_1 q_2$. The condition that $g^3 - g \in \mathbb{Z}_N^*$ is easily checked by both the prover and the verifier.

2. **Compute the** MTL**.** The prover computes the elements in the MTL, $u_i = g^{2^{2^i}} \bmod N$, $0 \le i \le K$, and $v_j = g^{2^{2^{K+1} - 2^{K-j}}} \bmod N$, $1 \le j \le K$. Knowing $\phi(N)$ allows the prover to perform the computations efficiently, by first computing, e.g., $a_i = 2^{2^i} \bmod \phi(N)$ and then $u_i = g^{a_i} \bmod N$. The computation of the v_i's is performed similarly.

3. **Prove well-formedness.** The prover proves to the verifier that $u_i = g^{2^{2^i}} \bmod N$, for $0 \le i \le K+1$, and $v_j = g^{2^{2^{K+1} - 2^{K-j}}} \bmod N$, for $1 \le j \le K$. The proof for the u_i's is done as in [BN00], by showing that each triple $< g, u_i, u_{i+1} >$, $0 \le i < K$, is of the form $< g, g^x, g^{x^2} >$, for some x.[3] Similarly, the correctness of the "v line" is established by a zero-knowledge proof that the tuples $< g, u_{K-1}, u_K, v_1 >$, and $< g, u_{K-j}, v_{j-1}, v_j >$, for $2 \le j \le K$, are Diffie-Hellman tuples. These $2K$ proofs are performed in parallel.

We now argue how the properties mentioned at the beginning of the section are satisfied upon completion of the protocol.

[3] See also [Mao01] for a more efficient protocol.

1. The probability that a cheating prover successfully convinces the verifier that a wrong value lies on the MTL is $2K\varepsilon_s$, where ϵ_s is the soundness error of the zero-knowledge protocol of step 3. Given the safe-prime structure of N, and using the ZK protocol presented in [Mao01] yields

$$\varepsilon_s = \frac{2q_1 + 2q_2 - 1}{2q_1 q_2} \approx \frac{4}{\sqrt{N}}.$$

   These protocols can be repeated to lower the error.

2. A sufficiently large period of the MTL follows from Theorem 1.

Setting up an MTL requires a computational effort, particularly in the Setup phase for the proof of the safe-prime structure of N [CM99a]. In our applications of Section 4, however, this effort will be incurred only once, as the prover will use one MTL (call it the *master* MTL) to spawn many other MTL's much more efficiently. We also note the "zero-knowledgeness" of the MTL protocol above with respect to $\phi(N)$; this will guarantee the timed-release property of the applications that follow.

4 Timed Fair Exchange of Standard Signatures

We now turn to applications of our MTL construction. One important application is fair exchange. Recall that in this problem two parties, Alice and Bob, have an item that they wish to exchange in such a way that either both get the other's item, or nobody does. In particular, we will be considering the case of exchange of digital signatures, a specific instantiation of which is *contract signing*, where each party would like to receive the other's signature on a known piece of text. But before we do that, we first consider the simpler case of the parties fairly exchanging their respective time-lines' hidden values (see Section 2). We will then use this building block to allow for the exchange of standard signatures, by using the committed hidden values to blind the respective signatures—and prove that this is indeed the case. Note that the simpler building block already enables other timed applications, such as collective (two-party) coin tossing [Blu81].

4.1 Fair exchange of hidden values

Assume that the two parties, Alice and Bob, have agreed on an integer parameter K. Let $T = 2^K$. For this application, we envision K to be large (in particular, larger than for previous timed applications), e.g., $K \approx 80$. Further, assume that both parties have performed (T, t, ϵ) time-line commitments to the other party, using the protocol of [GJ02]. As a result, Alice and Bob are committed to their respective time-lines' hidden values which now they would like to exchange fairly. We adapt the definition of fairness of a timed contract signing protocol of [BN00] to the case of exchange of hidden values:

Fairness: A hidden-value fair exchange protocol is said to be (a, ϵ)-*fair* if for any of the parties (wlog, Alice) working in time t smaller than some security parameter, and running the fair exchange protocol with the other party (Bob), the following holds: If at some point Alice aborts the protocol and succeeds in recovering Bob's hidden value with probability p_A, then Bob running in time $a \cdot t$ can recover Alice's hidden value with probability p_B, such that $|p_A - p_B| \leq \epsilon$.

Intuitively, a is the *advantage* of the party that aborts. Below we present a hidden-value fair exchange protocol that is $(2, \epsilon)$-fair, for negligible ϵ. The general idea is as follows. First, the parties each generate an MTL as described in Section 3.[4] To these MTL's the parties apply the time-line commitment protocol of [GJ02]; as a result, new, *derived* time-lines are generated (much more efficiently), and the parties become committed to the new time-lines' hidden values. It turns out that a much shorter description (only three points) of these new time-lines is sufficient for the commitment protocol: the end points and the new time-line's pivot; the hidden value is the (principal) square root of the right end point, and remains secret. Having established this, the parties proceed to perform the exchange: starting from the time-line's pivot, the parties now take turns, revealing and verifying the v points in ascending order (i.e., from left to right) on the new time-line., Thus, after K rounds, they both reach and verify each other's hidden values. Should any of the parties abort in the middle of the exchange, the other party is assured that the hidden value can be reached by repeated squaring of the last revealed value. Note that it is possible for a party to abort after receiving the first party's commitment and without having sent his own; given our choice of parameters K and T, however, the amount of work—and time—required to get to the first party's hidden value will be gargantuan. We now describe the protocol in detail.

1. **Setup.** Both Alice and Bob generate their own *mirrored* time-line, MTL_A and MTL_B respectively, using the protocol from Section 3, which gets verified by the other party.

2. **Commit phase.** Both parties, acting as committers, execute the time-line commitment protocol of [GJ02] using their respective MTL's as the *master* time-line. Specifically, each party $i \in \{\text{Alice, Bob}\}$ acting as the committer, performs the following steps on the mirrored time-line he generated (when necessary, we will be using superscripts to identify the elements of each party's time-line; in the following we omit them for simplicity):

   2.1 *Choose shifting exponent.* The committer picks a random integer $\alpha \in \mathbb{Z}_{\phi(N)}$. (If the committer does not know the value of $\phi(N)$ because of use of an MTL from a trusted third party, then the committer may choose α at random in $[1, N/2]$.)

   2.2 *Compute new time-line.* The committer computes $g' = g^\alpha$, $u'_K = u_K^\alpha$, $v'_K = v_K^\alpha$, and $u'_{K+1} = u_{K+1}^\alpha \pmod N$. He outputs $\langle g', u'_K, u'_{K+1}\rangle$, and **keeps** v'_K **secret**. v'_K is party i's hidden value.

   2.3 *Prove well-formedness of new time-line.* The committer does not reveal α, but instead proves to the receiver that

   $$\log_g g' = \log_{u_K} u'_K = \log_{u_{K+1}} u'_{K+1} \, (= \alpha),$$

   i.e., that the new time-line is correctly derived from the mirrored time-line, using a (statistical) zero-knowledge proof of equality of two discrete logs modulo the same number (N) [CEvdG87, CP92, Bao98]. Call this type of proof EQLOG-1(α, N). (For conciseness, we will sometimes use "$[\cdot]$" to refer to these zero-knowledge proofs.)

3. **Exchange phase.** Now the gradual exchange starts, with Alice and Bob taking turns in revealing their respective v''s in ascending order (i.e., halving the distance to the hidden value in each round), together with a proof that they lie on the shifted time-line. Specifically, the following is being executed from $j = 1$ to K: Alice sends to Bob

   $$v'^A_j = (v^A_j)^\alpha \bmod N_A, \quad [\log_{g^A} g'^A = \log_{v^A_j} v'^A_j],$$

[4] This phase of the protocol has to be performed only once, and then the resulting time-lines can be used repeatedly for several exchanges. It is also possible for both parties to use the same mirrored time-line, say, a "public" mirrored time-line generated by a third party which each party verifies. For simplicity, we assume that each party generates its own.

using an EQLOG-1 proof as before. Bob, after verifying the proof, responds with

$$v'^B_j = (v^B_j)^\beta \bmod N_B, \quad [\log_{g^B} g'^B = \log_{v^B_j} v'^B_j].$$

4. **Forced retrieval.** Should any of the parties, say, Alice, stop the exchange at round ℓ, $1 \leq \ell < K$, then Bob proceeds to successively square $v'^A_{\ell-1}$ $(\mathrm{mod} N_A)$ to get to v^A_K, Alice's hidden value, in approximately $2^{K-\ell}$ modular multiplications.

We now argue for the security of the protocol. The well-formedness of the derived time-line in Step 2 follows from the well-formedness of the MTL construction (Section 3) and the proofs of correct shifting, both in Step 2 and in the gradual exchange of Step 3. Thus, the parties are assured that the received points lie on the new time-line. Additionally, the new time-line inherits the long-period guarantee of the original MTL; thus, assuming the generalized BBS assumption holds, no "shortcuts" are possible for any adversary working in time $t < T$, and the probability of obtaining the other party's hidden value is at most ϵ. For the fairness, suppose, wlog, that Alice aborts the protocol after Bob reveals only $\ell < K$ points v' on his time-line. Then she can compute the hidden value in approximately $2^{K-\ell}$ multiplications (but not faster, based on the generalized BBS assumption). Bob, on the other hand, has one v' less than Alice has, and can compute Alice's hidden value using approximately $2 \cdot 2^{K-\ell} = 2^{K-\ell+1}$ modular multiplications; thus, his workload is roughly twice that of Alice's. Thus we have the following.

Lemma 4.1 *Assume the hardness of the discrete logarithm problem, the Hardy–Littlewood version of the prime k-tuples conjecture, and that the generalized BBS assumption holds for some parameters (l, u, δ, ϵ). Then the protocol above is a $(2, \epsilon)$-fair hidden-value exchange protocol.*

As in [BN00], it is possible to argue the "zero-knowledgness" of a party's hidden value against an adversary willing to invest a time less than T, by constructing a simulator operating in time proportional to the running time of the adversary that outputs a transcript of the execution that is indistinguishable from the real execution. This property will additionally guarantee the feature termed "abuse freeness" in [GJM99] (aka "strong fairness" [BN00]) in contract signing protocols, which we now consider.

4.2 Fair exchange of signatures

In [GJ02], Garay and Jakobsson show how to use time-line commitments to perform the timed release of signatures that allow for "blinding" [Cha82] (e.g., RSA, DSA, Schnorr). Timed release of signatures is closely related to our timed fair exchange problem, except that it is only "one way," from a signer to a verifier: the signer time-commits to a signature, and the verifier knows that if the signer refuses to reveal the signature, he will be able to retrieve it using a "forced-open" phase. In [GJ02] the time-line's hidden value is used as the blinding factor for the signature—together with proofs that, once recovered, the hidden value will produce correct unblinding of the signature. By applying this transformation to our hidden-value fair exchange protocol, we obtain a fair exchange protocol for standard signatures.

Note that the **Fairness** condition of Section 4.1 can be readily modified to account for the fair exchange of signatures. Two other standard conditions for the case of exchange of signatures are **Completeness** (the signature verification algorithms will output "Accept" on signatures that are the result of correct execution of the protocol), and **Unforgeability** (the probability that any polynomial-time adversary will produce a fake signature/contract is negligible); see, e.g., [ASW98, BN00] for further details on these and other definitions.

We now show how to augment the Commit phase of Section 4.1 to perform the signature blinding. We exemplify for the case of RSA signatures,[5] but keep in mind that the technique also applies to other (e.g., discrete log-based) signatures; also note that the signatures being exchanged could be of different types. The transformation is taken from [GJ02] almost *verbatim* (with some details omitted for readability). We assume that steps 2.1–2.3 regarding the time-line commitment have already taken place, and, again, we omit indices identifying the parties for readibility, but bear in mind that both parties are performing this phase, as signers.

2. **Commit$^+$ phase.** Let n be the signer's RSA modulus, (e, n) be the signer's public key, and d his secret key, chosen in such that way that $x^{ed} = x \bmod n$ for all values $x \in \mathbb{Z}_n$. Additionally, the signer will be performing auxiliary (standard) commitments using a subgroup of order N in $\mathbb{Z}^*_{N'}$, for $N' = \kappa N + 1$ a prime; let h be a generator of this subgroup.

   2.4 *Normal signature generation.* Let M be the message to be signed.[6] The signer computes $s = M^d \bmod n$.

   2.5 *Application of blinding factor.* The signer blinds the signature by computing

   $$\tilde{s} = s^{1/v_K} \bmod n,$$

   where v_K is his time-line's hidden value. He sends the pair $(M, \tilde{s})$ to the verifier.

   2.6 *Auxiliary commitments and proof of uniqueness.* The signer computes

   $$b = h^{v_K} \text{ and } B = h^{u_{K+1}} \pmod{N'}.$$

   The signer proves to the verifier that $\log_h b = \log_b B \ (= v_K)$, using EQLOG-1$(v_K, N')$.

   Let INTVL$(x \in [a, b])$ denote a zero-knowledge proof of knowledge that x lies exactly (i.e., with expansion rate 1) in interval $[a, b]$ [Bou00]. The signer proves to the verifier that the blinding factor lies in the right interval with INTVL$(v_K \in [0, N-1])$.

   2.7 *Proof of correct blinding.* The verifier computes $X = \tilde{s}^e \bmod n$.

   Let EQLOG-2(x, n_1, n_2) denote a zero-knowledge proof of knowledge of equality of two discrete logs (x) in *different* moduli (n_1 and n_2) [BT99, CM99b]; these proofs additionally require the strong RSA assumption [BP97]. The signer proves that

   $$\log_X (M \bmod n) = \log_h (b \bmod N') \ (= v_K)$$

   using EQLOG-2(v_K, n, N').

It is shown in [GJ02] that the above sub-protocol produces a (T, t, ϵ) timed RSA signature scheme. This, together with Lemma 4.1, allow us to conclude

Theorem 2 *Assume the hardness of the discrete logarithm problem, the Hardy–Littlewood version of the prime k-tuples conjecture, and that the strong RSA assumption and the generalized BBS assumption hold, the latter for some parameters (l, u, δ, ϵ). Let $\mathcal{S} = (G, S, V)$ be a signature scheme that allows for blinding (e.g., RSA, DSA, Schnorr). Then it is possible to construct protocols for the timed fair exchange of signatures based on $\mathcal{S}$.*

[5] Timed contract signing using RSA (and Rabin) signatures was also discussed in [BN00], but for the case when the same modulus is used for both the signature and the time-line construction. Here we consider the general case of arbitrary moduli.

[6] For simplicity, we omit here the issue of secure padding schemes for digital signatures.

4.3 Efficiency

Regarding efficiency, the number of rounds of our protocol is roughly the same as the protocol of [BN00] for special signatures (K). The computational cost of our protocol in each round is higher, though: generation and verification of a zero-knowledge proof of knowledge, vs. a simpler verification operation (e.g., a squaring in the case of Rabin signatures) in the case of [BN00]. On the other hand, the setup cost is lower, as the cost involved in the generation of the mirrored time-line is incurred only once, after which derived time-lines are generated for each execution of the protocol, which require a constant number of proofs of knowledge.

Acknowledgements: The authors thank Markus Jakobsson for valuable discussions on the subject, and the anonymous reviewers for FC'03 for their useful comments.

References

[ASW98] N. Asokan, V. Shoup, and M. Waidner. Fair exchange of digital signatures. *Advances in Cryptology—EUROCRYPT 98*, volume 1403 of *Lecture Notes in Computer Science*, pages 591–606, Springer-Verlag, May/June 1998.

[Bao98] F. Bao. An efficient verifiable encryption scheme for encryption of discrete logarithms. In *Proc. CARDIS'98*, 1998.

[Ble00] D. Bleichenbacher. On the distribution of DSA session keys. Manuscript, 2000.

[Blu81] M. Blum. Coin flipping by telephone: A protocol for solving impossible problems. In *Advances in Cryptology—CRYPTO '81*, pages 11–15. ECE Report 82-04, 1982.

[Blu83] M. Blum. How to exchange (secret) keys. *ACM Transactions on Computer Systems*, 1(2):175–193, May 1983.

[BBS86] L. Blum, M. Blum, and M. Shub. A simple unpredictable pseudo-random number generator. *SIAM Journal on Computing*, 15(2):364–383, May 1986.

[BCDvdG87] E. Brickell, D. Chaum, I. Damgård, and J. van de Graaf. Gradual and verifiable release of a secret (extended abstract). In *Advances in Cryptology—CRYPTO '87*, volume 293 of *Lecture Notes in Computer Science*, pages 156–166. Springer-Verlag, 1988, 16–20 August 1987.

[BG96] M. Bellare and S. Goldwasser. Encapsulated key escrow. In MIT/LCS/TR-688, 1996.

[BG97] M. Bellare and S. Goldwasser. Verifiable partial key escrow. In *Proc. ACM CCS*, pages 78–91, 1997.

[BN00] D. Boneh and M. Naor. Timed commitments (extended abstract). In *Advances in Cryptology—CRYPTO '00*, volume 1880 of *Lecture Notes in Computer Science*, pages 236–254. Springer-Verlag, 2000.

[Bou00] F. Boudot. Efficient proofs that a committed number lies in an interval. In *Advances in Cryptology—EUROCRYPT '00*, volume 1807 of *Lecture Notes in Computer Science*, pages 431–444. Springer-Verlag, 2000.

[BP97] N. Barić and B. Pfitzmann. Collision-free accumulators and fail-stop signature schemes without trees. In *Advances in Cryptology–Eurocrypt '97*, pp.480–494, 1997.

[BT99] F. Boudot and J. Traoré. Efficient publicly verifiable secret sharing schemes with fast or delayed recovery. In *Proc. 2nd International Conference on Information and Communication Security*, volume 1726 of *Lecture Notes in Computer Science*, pages 87–102. Springer-Verlag, 1999.

[Cha82] D. Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology: Proceedings of Crypto 82*, pages 199–203. Plenum Press, New York and London, 1983, 23–25 August 1982.

[CDS94] R. Cramer, I. Damgård, and B. Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *Advances in Cryptology—CRYPTO '94*, volume 839 of *Lecture Notes in Computer Science*, pages 174–187. Springer-Verlag, 21–25 August 1994.

[CEvdG87] D. Chaum, J. Evertse, and J. van de Graaf. An improved protocol for demonstrating possession of discrete logarithms and some generalizations. In *Advances in Cryptology—EUROCRYPT 87*, volume 304 of *Lecture Notes in Computer Science*, pages 127–141. Springer-Verlag, 1988, 13–15 April 1987.

[CFT98] A. Chan, Y. Frankel, and Y. Thiounis. Easy come – easy go divisible cash. In *Advances in Cryptology—EUROCRYPT 98*, volume 1403 of *Lecture Notes in Computer Science*, pages 561–575. Springer-Verlag, 1998.

[CM99a] J. Camenisch and M. Michels. Proving in Zero-Knowledge that a Number is the Product of Two Safe Primes. In *Advances in Cryptology - EUROCRYPT '99*, . volume 1592 of *Lecture Notes in Computer Science*, pages 106-121, Springer Verlag, 1999.

[CM99b] J. Camenisch and M. Michels. Separability and efficiency for generic group signature schemes (extended abstract). In *Advances in Cryptology—CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science*, pages 414–430. Springer-Verlag, 1999.

[CP92] D. Chaum and T. Pedersen. Wallet databases with observers (extended abstract). In CRYPTO'92 [CRY92], pages 89–105.

[CRY92] *Advances in Cryptology—CRYPTO '92*, volume 740 of *Lecture Notes in Computer Science*. Springer-Verlag, 1993, 16–20 August 1992.

[Dam95] I. B. Damgård. Practical and provably secure release of a secret and exchange of signatures. *J. of Crypt.*, 8(4):201–222, Autumn 1995.

[DN92] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In CRYPTO'92 [CRY92], pages 139–147.

[EGL85] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28(6):637–647, June 1985.

[FS86] A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Advances in Cryptology—CRYPTO '86*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer-Verlag, 1987, 11–15 August 1986.

[FPS01a] J. B. Friedlander, C. Pomerance, and I. E. Shparlinski. Period of the power generator and small values of Carmichael's function. *Math. Comp.* **70** (2001), 1591–1605.

[FPS01b] J. B. Friedlander, C. Pomerance, and I. E. Shparlinski. Small values of the Carmichael function and cryptographic applications. In *Progress in Computer Science and Applied Logic*, Vol. 20, pages 25–32, Birkhäuser Verlag, Basel, Switzerland, 2001.

[Gol83] O. Goldreich. A simple protocol for signing contracts. In *Advances in Cryptology—CRYPTO '83*, pages 133–136.

[GJ02] J. Garay and M. Jakobsson. Timed Release of Standard Digital Signatures. In *Financial Cryptography '02*, *Lecture Notes in Computer Science*, Springer-Verlag, to appear.

[GJM99] J. Garay, M. Jakobsson and P. MacKenzie. Abuse-free Optimistic Contract Signing. In *Advances in Cryptology - CRYPTO '99*, volume 1666 of *Lecture Notes in Computer Science* Springer-Verlag, pages 449-466, August 1999.

[GMP01] S. Galbraith, W. Mao, and K. Paterson. A cautionary note regarding cryptographic protocols based on composite integers. In HPL-2001-284, 2001.

[HL23] G. H. Hardy and J. E. Littlewood, Some problems in "Partitio Numerorum," III: On the expression of a number as a sum of primes. *Acta Math.* **44** (1923), 1–70.

[Mao98] W. Mao. Guaranteed correct sharing of integer factorization with off-line shareholders. In *Proc. Public Key Cryptography '98*, pages 27–42, 1998.

[Mao01] W. Mao. Timed-Release Cryptography. In *Selected Areas in Cryptography VIII (SAC'01)*, volume 2259 of *Lecture Notes in Computer Science*, pages 342-357, Springer-Verlag, Toronto, Ontario, Canada, August 2001.

[MP03] G. Martin and C. Pomerance. The normal order of iterates of the Carmichael λ-function. in progress.

[May93] T. May. Timed-release crypto. In `http://www.hks.net.cpunks/cpunks-0/1460.html`, 1993.

[PS95] C. Pomerance and J. Sorenson. Counting the integers factorable via cyclotomic methods. *J. Algorithms* 19: 250–265, 1995.

[RSW96] R. Rivest, A. Shamir, and D. Wagner. Time-lock puzzles and timed-release crypto. In MIT/LCS/TR-684, 1996.

[Sha84] A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology: Proceedings of CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53. Springer-Verlag, 1985, 19–22 August 1984.

[Sha95] A. Shamir. Partial key escrow: A new approach to software key escrow. In *Key Escrow Conference*, 1995.