One-Time Trapdoor One-Way Functions

Julien Cathalo*[1] and Christophe Petit**[2]

[1] Smals
Avenue Fonsny 20, 1060 Bruxelles, Belgium
[2] UCL Crypto Group
Université catholique de Louvain, Place du Levant 3, 1348 Louvain-la-Neuve, Belgium

Abstract. Trapdoors are widely used in cryptography, in particular for digital signatures and public key encryption. In these classical applications, it is highly desirable that trapdoors remain secret even after their use. In this paper, we consider *positive* applications of trapdoors that do *not* remain secret when they are used. We introduce and formally define *one-time trapdoor one-way functions* (OTTOWF), a primitive similar in spirit to classical trapdoor one-way functions, with the additional property that its trapdoor always becomes public after use. We provide three constructions of OTTOWF. Two of them are based on factoring assumptions and the third one on generic one-way functions. We then consider potential applications of our primitive, and in particular the fair exchange problem. We provide two fair exchange protocols using OTTOWF, where the trapdoor is used to provide some advantage to one of the parties, whereas any (abusive) use of this trapdoor will make the advantage available to the other party as well. We compare our protocols with well-established solutions for fair exchange and describe some scenarios where they have advantageous characteristics. These results demonstrate the interest of one-time trapdoor one-way functions, and suggest looking for further applications of them.

Keywords Cryptographic primitive, trapdoor one-way function, fair exchange

1 Introduction

In cryptography, a *trapdoor* is a secret piece of information that provides its holder with some special power or advantage. This concept is usually formalized through the definition of a *trapdoor one-way function* (TOWF), a function that is computationally easy to compute and hard to invert but that becomes easy to invert with the help of the trapdoor. TOWF are a fundamental tool for public key cryptography, for example to build digital signature or public key encryption schemes.

* The work of this author was done while he was a post-doctoral researcher at the UCL Crypto Group and was supported by the Belgian Walloon Region under its RW-WIST Programme, ALAWN Project.

** Research Fellow of the Belgian Fund for Scientific Research (F.R.S.-FNRS) at Université catholique de Louvain (UCL).

The two most famous trapdoor one-way functions are arguably RSA [31] and Rabin [30] TOWF. Both of them rely on the factoring assumption, the assumption that factoring "big" composite numbers is "hard". Even if the security link between Rabin and factoring is better (in fact, breaking Rabin TOWF is equivalent to factoring), the RSA TOWF has been much more used in practice. The reason is that any use of Rabin TOWF will leak its trapdoor with a probability of one half, whereas RSA trapdoor remains completely secret even after many uses.

If trapdoors provide their owners with some specific advantage, their leakage will, on the other hand, strongly limit this power. Fresh parameters will have to be generated after each use, which is an important limitation in many applications. For this reason, leaking trapdoors have usually been considered as undesirable and RSA has been preferred to Rabin in applications. However, we remark that limiting the use of a trapdoor also seems appealing and potentially very useful in other applications:

- To ensure that the trapdoor is only used once, assuming that it becomes useless as soon as it becomes public. This seems appealing for an electronic cash system, to ensure that coins are only spent once.
- To prove that the trapdoor has been used, assuming that it becomes public if and only if it is used. This could be useful in a digital right management system supporting delegation, to trace the fact that the proxy has used its trapdoor.
- To ensure some fairness between various parties in a protocol, since the special power given to the holder of the trapdoor will also become available to the other parties after its use.

In this paper, we therefore introduce *one-time trapdoor one-way functions* (OTTOWF), a cryptographic primitive close to trapdoor one-way functions but with the additional property that trapdoors *always* become public after their use. We provide a formal definition and three constructions of OTTOWF satisfying various flavors of this definition. Our first construction is a natural extension of Rabin TOWF. It relies on the factoring assumption for RSA numbers. The second construction is a modification of Paillier's trapdoor one-way permutation [28] and relies on a different factoring assumption. Finally, our third construction is based only on generic one-way functions (but is not a trapdoor one-way function in the usual sense).

As an example of application, we consider the fair exchange problem, a very important problem for e-commerce. A *fair* exchange of signatures between two parties requires that both parties are guaranteed to give their own signature only if they receive the other signature in exchange. We design two fair exchange protocols using OTTOWF. In these protocols, we use the trapdoor of an OTTOWF to provide one of the parties with some protection mechanism, and we use its leakage property to ensure fairness to the other party even if the first party abusively uses its trapdoor. A comparison with state-of-the-art protocols reveals some advantages of our approach in particular situations. We believe that these

results illustrate the interest of our new primitive, and encourage research for further applications.

This paper is organized as follows. We define our notations and we recall standard security notions in Section 2. We introduce one-time trapdoor one-way functions in Section 3.1 and give our three constructions in Sections 3.2, 3.3 and 3.4. We provide and analyze two new protocols for fair exchange in Section 4, and we conclude the paper in Section 5.

2 Preliminaries

2.1 Notations

In this paper, we will use the notation ℓ for the *security parameter* involved in our definitions. For any $\ell \geq 2$, we write $\mathcal{P}_\ell$ for the set of prime numbers p such that $2^{\ell-1} \leq p < 2^\ell$. By $\{\mathcal{S}(\ell), \ell = 1, 2, \dots\}$ we mean a sequence of sets $\mathcal{S}(1), \mathcal{S}(2), ...$, one set for each value of the security parameter ℓ. For any function f, we write $\mathrm{Dom}(f)$ and $\mathrm{Im}(f)$ for its domain and codomain. We say that a function $f : \mathbb{N} \to \mathbb{R}^+$ is *negligible* if for any polynomial p, there exists $\ell_0 \in \mathbb{N}$ such that $f(\ell) < 1/p(\ell)$ for any $\ell \geq \ell_0$. We say that a function $g : \mathbb{N} \to [0, 1]$ is *overwhelming* if there exists a negligible function f such that $g = 1 - f$.

When x and y are two bitstrings, we write $\langle x, y \rangle$ for their concatenation. We denote by 1^ℓ a bitstring made of ℓ consecutive 1. We write $\{0,1\}^*$ for the set of all bitstrings with a finite length. We say that an algorithm $\mathbf{A}$ is *probabilistic polynomial time* or just PPT if there exists a polynomial p such that the average running time of $\mathbf{A}$ on inputs of size ℓ is smaller than $p(\ell)$. We say that a computational task is *computationally infeasible* if no PPT algorithm succeeds in performing the task with a non-negligible probability.

We say that a set $\mathcal{R}$ is *samplable* if there exists a PPT algorithm that randomly picks an element from the uniform distribution on $\mathcal{R}$. When $\mathcal{R}$ is samplable, we write $r \xleftarrow{\$} \mathcal{R}$ for the experiment of uniformly selecting a random value from $\mathcal{R}$ and assigning it to r. We say that a set $\mathcal{R}$ is *decidable* if there exists a PPT algorithm that on input (bitstring representations of) x and $\mathcal{R}$, returns 1 if and only if $x \in \mathcal{R}$. If $\mathbf{Alg}$ is a deterministic algorithm we note $\mathbf{Alg}(x)$ for the value outputted by $\mathbf{Alg}$ on input x. If $\mathbf{Alg}$ is probabilistic, $y \in \mathbf{Alg}(x)$ means that $\mathbf{Alg}$ returns y on input x for some values of its internal random coins, whereas $y = \mathbf{Alg}(x)$ denotes the experiment of running $\mathbf{Alg}$ on input x and assigning its output to y. By $\Pr[A : B; C]$ we mean the probability that a logical expression A holds given an experiment that consists in successively executing B and C.

2.2 One-way function

Intuitively, a one-way function (OWF) is a function that is easy to compute but computationally hard to invert.

Definition 1 *A function $f : \{0,1\}^* \to \{0,1\}^*$ is a* one-way function *if it is:*

- ***Easy to compute:*** *there exists a PPT algorithm that on input x returns $f(x)$.*

– ***Hard to invert:*** *there exists no PPT algorithm* $\mathbf{A}$ *such that* $\Pr\left[f(x') = y : x \xleftarrow{\$} \{0,1\}^\ell; y = f(x); x' = A(y)\right]$ *is a non negligible function of* ℓ.

2.3 Trapdoor one-way function

Intuitively, a trapdoor one-way function (TOWF) is a one-way function that becomes easy to invert given an additional piece of information called a trapdoor.

Definition 2 *A* trapdoor one-way function *is a set of three probabilistic polynomial time algorithms:*

– **Setup***: probabilistic algorithm that on input* 1^ℓ*, returns* $\langle f, t\rangle$ *where* $f \in \mathcal{F}(\ell)$ *is the description of a function, together with a samplable set* $Dom(f)$ *and two sets* $Im(f)$ *and* $\mathcal{T}(f)$*, and* $t \in \mathcal{T}(f)$ *is a corresponding trapdoor. (The sets* $\mathcal{F}(\ell)$*,* $\ell \in \mathbb{N}_0$ *are implicitly defined by* **Setup**.*)*
– **Eval***: deterministic algorithm that on input* $\langle f, x\rangle$ *where* $f \in \mathcal{F}(\ell)$ *and* $x \in Dom(f)$*, returns* $y \in Im(f)$.
– **Preimage***: probabilistic algorithm that on input* $\langle f, t, y\rangle$ *where* $f \in \mathcal{F}(\ell)$ *is the description of a function,* $t \in \mathcal{T}(f)$ *is a trapdoor and* $y \in Im(f)$*, returns* $x \in Dom(f)$.

Moreover, the algorithms satisfy the following properties:

– ***One-wayness****: there exists no PPT algorithm* $\boldsymbol{A}$ *such that* $\Pr\left[f(x') = y : \langle f, t\rangle = \mathbf{Setup}(1^\ell); x \xleftarrow{\$} Dom(f); y = \mathbf{Eval}(f, x); x' = \boldsymbol{A}(f, y)\right]$ *is non negligible.*
– ***Trapdoor****: for any* $\langle f, t\rangle = \mathbf{Setup}(1^\ell)$*, any* $x \in Dom(f)$*, and any* $x' \in \mathbf{Preimage}(f, t, \mathbf{Eval}(f, x))$*, we have* $\mathbf{Eval}(f, x') = y$.

2.4 Signature scheme

Informally, a digital signature scheme is made of three algorithms, **Setup**, **Sign** and **Ver**. The **Setup** algorithm creates a pair of private and public keys; the private key is used for signing and the public key for verifying a signature.

Definition 3 *A* digital signature scheme *is a set of three PPT algorithms:*

– **Setup***: a probabilistic algorithm that on input* 1^ℓ*, returns* $\langle SK, PK\rangle$ *where* $PK \in \mathcal{P}(\ell)$ *is a public key (implicitly defining a set* $\mathcal{S}(PK)$*, a samplable set* $\mathcal{M}(PK)$ *and a decidable set* $\Sigma(PK)$*) and* $SK \in \mathcal{S}(PK)$ *is a corresponding private key. (The sets* $\mathcal{P}(\ell)$*,* $\ell \in \mathbb{N}_0$ *are implicitly defined by* **Setup**.*)*
– **Sign***: a probabilistic algorithm that on input* $\langle m, SK\rangle$ *where* $m \in \mathcal{M}(PK)$ *is a message and* $SK \in \mathcal{S}(PK)$ *is a private key corresponding to some public key* PK*, returns a signature* $\sigma \in \Sigma(PK)$.
– **Ver***: a deterministic algorithm that on input* $\langle m, PK, \sigma\rangle$ *where* $m \in \mathcal{M}(PK)$ *is a message,* $PK \in \mathcal{S}(\ell)$ *is a public key and* $\sigma \in \Sigma(PK)$ *is a signature, returns either 0 or 1.*

Moreover, the algorithms are required to satisfy the following properties:

- ***Correctness**: for any $\ell \in \mathbb{N}$, for any $\langle SK, PK\rangle \in \mathbf{Setup}(1^\ell)$, any $m \in \mathcal{M}(PK)$ and any $\sigma \in \mathbf{Sign}(m, SK)$, we have $\mathbf{Ver}(m, PK, \sigma) = 1$.*
- ***Existential unforgeability against adaptive chosen message attacks** [21]: no PPT algorithm $\boldsymbol{A}$ can succeed in the following game with a non-negligible probability.*
  - *A* challenger *algorithm runs $\mathbf{Setup}(1^\ell)$, keeps the private key SK and forwards the public key PK to $\boldsymbol{A}$.*
  - *$\boldsymbol{A}$ chooses a polynomial number of messages $m_i \in \mathcal{M}(PK)$ and queries the challenger for signatures $\mathbf{Sign}(m_i, SK)$ on these messages.*
  - *$\boldsymbol{A}$ outputs $\langle m, \sigma\rangle$ where $m \in \mathcal{M}(PK)$, $\sigma \in \Sigma(PK)$, $\mathbf{Ver}(m, PK, \sigma) = 1$ and m was not queried before to the challenger.*

2.5 The factoring assumption(s)

The *factoring assumption* states that (some classes of) integers are computationally hard to factor. More formally, it tells that there is no PPT algorithm $\mathbf{A}$ for which $\Pr\left[p = \mathbf{A}(n(p,q)); p \xleftarrow{\$} \mathcal{P}_\ell; q \xleftarrow{\$} \mathcal{P}_\ell\right]$ is a non negligible function of ℓ. In this paper, we consider RSA and Paillier composite numbers, respectively $n(p,q) := pq$ and $n(p,q) := p^2q$. The factoring assumption in those cases is widely believed to be true, and it has been intensively used in cryptography.

3 One-time trapdoor one-way functions

We now define *one-time trapdoor one-way functions* (OTTOWF). Intuitively, a one-time trapdoor one-way function is a one-way function with a trapdoor and the additional property that any use of the trapdoor will reveal it. More precisely, there exists an extraction algorithm that can extract a trapdoor from any couple of messages with the same image, provided that the first message was generated "normally" and the second one was computed with the trapdoor. We give three practical constructions of OTTOWF. The first one is a natural extension of Rabin's trapdoor function; the second one is inspired by Paillier's trapdoor permutation [28]; the third one is based on generic one-way functions.

3.1 Definition

Definition 4 *A* one-time trapdoor one-way function *is given by a set of five probabilistic polynomial time algorithms:*

- **Setup**: *probabilistic algorithm that upon input of a security parameter ℓ, returns $\langle k, t\rangle$ where $k \in \mathcal{K}(\ell)$ is a key (implicitly defining a set $\mathcal{T}(k)$, a samplable and decidable set $\mathcal{R}(k)$ and a decidable set $\mathcal{H}(k)$) and $t \in \mathcal{T}(k)$ is a corresponding trapdoor. (The sets $\mathcal{K}(\ell)$ for $\ell \in \mathbb{N}_0$ are implicitly defined by* **Setup**.*)*
- **Eval**: *algorithm that upon input of $\langle k, r\rangle$ where $k \in \mathcal{K}(\ell)$ is a key and $r \in \mathcal{R}(k)$ is a message, outputs a hash value $h \in \mathcal{H}(k)$.*

- **Verify**: *deterministic algorithm that upon input of* $\langle k, r, h\rangle$ *where* $k \in \mathcal{K}(\ell)$ *is a key,* $r \in \mathcal{R}(k)$ *is a message and* $h \in \mathcal{H}(k)$ *is a hash value, outputs either 0 or 1.*
- **Preimage**: *deterministic algorithm that upon input of* $\langle k, h, t\rangle$ *where* $k \in \mathcal{K}(\ell)$ *is a key,* $h \in \mathcal{H}(k)$ *is a hash value and* $t \in \mathcal{T}(k)$ *is a trapdoor, outputs a message* $r \in \mathcal{R}(k)$*.*
- **TrapExtr**: *deterministic algorithm that upon input of* $\langle k, r, r'\rangle$ *where* $k \in \mathcal{K}(\ell)$ *is a key and* $r, r' \in \mathcal{R}(k)$ *are two messages, outputs either* $\perp$ *or a trapdoor* $t \in \mathcal{T}(k)$*.*

Moreover, these algorithms are required to satisfy the following properties :

- ***Correctness:*** *For any* $\langle k, t\rangle \in \mathbf{Setup}(1^\ell)$ *and any* $r \in \mathcal{R}(k)$*, we have* $\mathbf{Verify}(k, r, \mathbf{Eval}(k, r)) = 1$*.*
- ***Onewayness:*** *There exists no PPT algorithm* $\boldsymbol{A}$ *such that* $\Pr\left[\mathbf{Verify}(k, r', h) = 1 : \langle k, t\rangle \in \mathbf{Setup}(1^\ell); r \xleftarrow{\$} \mathcal{R}(k); h = \mathbf{Eval}(k, r); r' = \boldsymbol{A}(k, h)\right]$ *is non negligible.*
- ***Trapdoor:*** *For any* $\langle k, t\rangle \in \mathbf{Setup}(1^\ell)$*, any* $r \in \mathcal{R}(k)$ *and any* $h \in \mathbf{Eval}(k, r)$*, we have* $\mathbf{Verify}(k, \mathbf{Preimage}(k, h, t), h) = 1$*.*
- ***Fairness:*** *For any* $\langle k, t\rangle \in \mathbf{Setup}(1^\ell)$*, the probability* $\Pr\left[\mathbf{TrapExtr}(k, r, \mathbf{Preimage}(k, \mathbf{Eval}(k, r), t)) = t : r \xleftarrow{\$} \mathcal{R}(k)\right]$ *is overwhelming.*

We say that an OTTOWF has verifiable setup *if the sets* $\mathcal{K}(\ell)$ *are decidable. We say it is* deterministic *if the algorithm* **Eval** *is deterministic. We say it is* surjective *if for any* $k \in \mathcal{K}(\ell)$ *and any* $h \in \mathcal{H}(k)$*, there exists* $r \in \mathcal{R}(k)$ *such that* $\mathbf{Eval}(k, r) = h$*. We say it has* trivial verification *if the algorithm* **Verify** *upon input of* $\langle k, r, h\rangle$ *simply computes* $\mathbf{Eval}(k, r)$ *and compares the result with* h*. We say it has* uniform inversion *if* $\mathbf{Preimage}(k, h, t)$ *can return any valid preimage of* h *with the same probability.*

The onewayness and trapdoor properties simply state that finding preimages is hard without the trapdoor and easy with the trapdoor. The fairness property states that, given a message and a preimage of the hash of this message computed with the trapdoor, anyone can recover the trapdoor. For *surjective* OTTOWF the trapdoor property is equivalent to the following property: for any $\langle k, t\rangle \in \mathbf{Setup}(1^\ell)$ and any $h \in \mathcal{H}(k)$, we have $\mathbf{Verify}(k, \mathbf{Preimage}(k, h, t), h) = 1$. The trapdoor and one-wayness properties together imply that it is hard to recover the trapdoor from the key. Finally, our definition implies that for random r values, the probability that $\mathbf{Preimage}(k, \mathbf{Eval}(k, r), t) = r$ is negligible. Indeed, if the probability that $\mathbf{Preimage}(k, \mathbf{Eval}(k, r), t) = r$ was non-negligible, an adversary against the one-wayness property would obtain the trapdoor with a non-negligible probability by computing $\mathbf{TrapExtr}(k, r, r)$ on random r values.

We point out that an OTTOWF is not necessarily a TOWF in the sense of Definition 2. Indeed, the correctness property of OTTOWF does not require that $\mathbf{Eval}(\mathbf{Preimage}(h)) = h$ for all h, but only that the **Preimage** algorithm finds

a value that passes the **Verify** algorithm. However, any deterministic OTTOWF with trivial verification *is* a TOWF.

Definition 4 is sufficiently flexible to allow at least three different constructions based on various assumptions. The construction we give based on Rabin is deterministic and it has trivial verification and uniform inversion but it is not surjective. The construction based on Paillier's permutation is deterministic, surjective and it has trivial verification and uniform inversion. Both of these constructions are TOWF in the sense of Definition 2. The construction from generic OWF is deterministic and surjective, but it has non uniform inversion and it is not a TOWF since it does not have trivial verification. (We remark that otherwise, we would have a reduction from the existence of TOWF to the existence of OWF, a long-standing open problem.) The construction based on OWF has a verifiable setup, whereas the constructions based on Rabin and Paillier can be easily modified to also have a verifiable setup, at the cost of increasing the key size.

3.2 OTTOWF based on Rabin's TOWF

In this section, we give a deterministic non surjective OTTOWF with trivial verification and uniform inversion under the factoring assumption for RSA moduli $n = pq$. It is well-known that for Rabin's TOWF, two random domain elements with the same image will reveal the trapdoor with a probablity $1/2$. In the following construction, Rabin's TOWF is modified to increase this probability up to essentially 1.

Let $N : \mathbb{Z} \to \mathbb{Z}$ be a function such that for any polynomial p, there exists $\ell_0 \in \mathbb{N}$ such that $N(\ell) > \log(p(\ell))$ for any $\ell \geq \ell_0$. We define the algorithms **Setup**, **Eval**, **Verify**, **Preimage** and **TrapExtr** as follows:

- **Setup**: on input 1^ℓ, randomly choose two primes $2^{\ell-1} < p, q < 2^\ell$ and compute their product $n = pq$. Set $N = N(\ell)$. Output $\langle k, t\rangle$ where $k = \langle n, N\rangle$ and $t = \langle p, q\rangle$. We have $\mathcal{H}(k) = \mathcal{R}(k) = \mathbb{Z}_n$.
- **Eval**: on input $\langle n, N\rangle$ and $r = (r_1, \ldots, r_N) \in \mathbb{Z}_n^N$, output $h = (r_1^2 \bmod n, \ldots, r_N^2 \bmod n)$.
- **Verify**: on input $k = \langle n, N\rangle$, $r \in \mathbb{Z}_n^N$ and $h \in \mathbb{Z}_n^N$, output 1 if $h = \mathbf{Eval}(k, r)$ else output 0.
- **Preimage**: on input $\langle n, N\rangle$, $h = (h_1, \ldots, h_N) \in \mathbb{Z}_n^N$ and $t = \langle p, q\rangle$, compute $E := \{\left(\pm q^{-1} \bmod p\right) q + \left(\pm p^{-1} \bmod q\right) p\}$ and pick random $\epsilon_i \in E$ for $i = 1, \ldots, N$. For any $i \in \{1, \ldots, N\}$, compute a square root of h_i modulo p and q, then compute a square root r_i of h_i modulo n with the Chinese remainder theorem. Output $r = (r_1\epsilon_1 \bmod n, \ldots, r_N\epsilon_N \bmod n)$.
- **TrapExtr**: on input $\langle n, N\rangle, r = (r_1, \ldots, r_N), r' = (r'_1, \ldots, r'_N)$, try to find $i \in \{1, \ldots, N\}$ such that $r_i \not\equiv \pm r'_i \pmod n$. If such i is found, find the factors of n using $gcd(r_i - r'_i, n)$ and output $t = \langle p, q\rangle$. Otherwise, output $\perp$.

Proposition 1 *Under the factoring assumption for RSA moduli, the above construction is a deterministic (non surjective) OTTOWF with trivial verification and uniform inversion.*

This construction is not *surjective* since only one fourth of the elements of $\mathcal{H}(k) := \mathbb{Z}_n$ belong to the set $\mathbb{QR}_n$ of quadratic residues modulo n. Making it surjective would require to define $\mathcal{H}(k) := \mathbb{QR}_n$, but deciding whether a random element of $\mathbb{Z}_n$ is a quadratic residue is believed to be a hard problem [20].

The construction as such does not have *verifiable setup* since it is presumably hard to decide whether a composite number is an RSA modulus or not. However, the setup can be made verifiable by adjoining to the key a non-interactive zero-knowledge (NIZK) proof that the modulus was correctly generated. Such a proof can be generated using the techniques of Gennaro et al. [19], at the cost of increasing the keysize. This solution requires a slight adaptation of the correctness property of Definition 4 to take into account the soundness and completeness errors of the NIZK proof. Details are left aside in this paper.

3.3 OTTOWF based on Paillier's trapdoor permutation

In the full version of this paper [13], we describe a second construction based on the factoring assumption, but this time for moduli $n = p^2q$. Compared to the previous one, this construction has the advantage of requiring much smaller parameters (key sizes, preimage sizes and image sizes). The idea is to extend the domain of Paillier's trapdoor permutation [28] to make it non-injective. The trapdoor is made of two prime integers p and q that have the same size ℓ. On input $r = \langle r_1, r_2 \rangle$ where r_1 has ℓ bits and r_2 has $3\ell - 3$ bits, the **Eval** algorithm returns $h = g^{r_1} r_2^n \bmod n$ with $n = p^2q$. The preimage algorithm is similar to the inversion algorithm of [28]. However, preimages are no longer unique because of the domain extension, and two different preimages will actually leak the trapdoor. The resulting construction is a deterministic, surjective OTTOWF with trivial verification and uniform inversion. It has *verifiable setup* under an additional reasonible hardness assumption, or if we append to the key a NIZK proof that the modulus n was well-formed [13].

3.4 OTTOWF based on one-way functions

One-time trapdoor one-way functions can also be built from generic one-way functions. Our construction is loosely inspired by the (non-injective) trapdoor one-way function of [7]. The trapdoor is a domain element of the OWF and its image is part of the public key. The algorithm **Verify** always accepts the trapdoor as a valid "preimage" of any hash value. Since the algorithm **Verify** is not *trivial*, this construction is not a TOWF in the sense of Definition 2.

Let f be a one-way function in the sense of Definition 1. We build the algorithms **Setup**, **Eval**, **Verify**, **Preimage**, **TrapExtr** as follows:

- **Setup**: on input 1^n, randomly choose $t \in \{0,1\}^\ell$. Compute $\beta = f(t)$. Output $\langle k, t \rangle$ where $k = \langle f, \beta \rangle$. The key implicitly defines $\mathcal{T}(k) := \{0,1\}^\ell$, $\mathcal{R}(k) := \{0,1\}^\ell$ and $\mathcal{H}(k) := \{0,1\}^\ell$.
- **Eval**: on input $\langle \langle f, \beta \rangle, r \rangle$, return $f(r)$.
- **Verify**: on input $\langle \langle f, \beta \rangle, r, h \rangle$, return 1 iff $f(r) = h$ or $f(r) = \beta$.
- **Preimage**: on input $\langle \langle f, \beta \rangle, h, t \rangle$, return t.
- **TrapExtr**: on input $\langle \langle f, \beta \rangle, r, r' \rangle$, return r'.

Proposition 2 *The above construction is a deterministic, surjective OTTOWF with verifiable setup (but non trivial verification and non uniform inversion).*

4 Application to fair exchange

In this section, we apply our new primitive to the fair exchange problem. We introduce two new protocols that use OTTOWF for the fair exchange of signatures, and we compare them with state-of-the-art solutions.

4.1 Fair exchange of signatures

In our increasingly electronic societies, more and more business transactions are conducted over the internet. A large amount of electronic transactions may be seen as two parties exchanging their digital signatures on a predetermined contract. *Fairness* is a fundamental requirement of contract signing protocols: no party wants to send his signature if he does not get the other party's signature in exchange. This problem of exchanging signatures in a fair manner has been extensively investigated in the last 30 years [8, 10, 15, 26, 1, 2, 18, 5, 4, 11, 27, 29, 17, 9, 12, 14, 6, 3, 25, 35, 16, 34, 24, 23, 22, 33]. Most fair exchange protocols have four communication rounds, during which the two parties, the *initiator* and the *responder*, first exchange *partial signatures* and then *full signatures*. Many protocols involve a *semi-trusted third party* (STTP) to ensure fairness. Such an STTP has been called *optimistic* when it is only invoked to resolve conflicts when "something goes wrong" in the protocol [1], either when a party attempts to cheat or when the communication network is defective. Other protocols do not require an STTP, but at the price of a reduced notion of fairness. *Concurrent signatures* (CS) [14] and *verifiably committed signatures* (VCS) [17] have emerged as the two most convincing approaches proposed so far for fair exchange, respectively without and with STTP.

4.2 Fair exchange without STTP

Concurrent signatures The main idea of concurrent signatures (CS) [14] is to use *ambiguous signatures* to construct the partial signatures. From the point of view of any third party, partial signatures are meaningless since they could have been generated either by the initiator or by the responder. The ambiguity of both partial signatures is removed when the initiator releases a piece of information called the keystone. Full signatures are made of a partial signature plus the keystone.

Concurrent signatures do not require any STTP but on the other hand, they only provide a reduced notion of fairness. In particular, if the initiator aborts after the second step of the protocol, he obtains a full signature of the responder while the responder only has a partial (meaningless) signature from the initiator. The fairness of CS is guaranteed only once the initiator uses the full signature he has built, and only if this signature is seen by the responder (since he therefore gets the keystone).

The ambiguity property ensures that nobody is even able to see that the initiator is willing to sign, unless they are able to see that the responder is willing to sign as well. On the other hand, ambiguity creates an asymmetry in the abortion facilities given to the parties. While the responder has committed to provide a full signature after the second step, the initiator can abort the protocol unilaterally, and the responder will not even be able to prove to a third party that he had received a first message from the initiator. In a practical scenario, this allows some vendor to pretend to sell the same good to various candidates, and only conclude the transaction with the most offering one.

A trivial non-ambiguous solution When we are willing to give up ambiguity for committing partial signatures, a very simple protocol comes in mind. In this protocol, partial signatures are regular signatures of the parties, and full signatures are the concatenation of both partial signatures. A drawback of this protocol is that neither the initiator nor the responder can directly produce a valid full signature on a message of their choice. This means that in an application where they also need to be able to directly sign messages, such "direct" signatures will have a different format than the full signatures.

Protocol 1: OTTOWF for fair exchange without STTP We now describe our first fair exchange protocol (see Figure 1). It uses an unforgeable signature scheme and a OTTOWF with verifiable setup. The message is concatenated with the image of a randomness by a OTTOWF, and the result is signed to form a partial signature. A full signature is made of a partial signature plus the corresponding randomness. The protocol follows the usual "partial, then full signatures" four communication rounds approach. With this approach, the responder has an *a priori* advantage since he can aborts the protocol after receiving a full signature. We compensate this advantage with the trapdoor, which gives the initiator the power to convert the responder's partial signature into a full signature. Finally, the consequences of any abusive use of the trapdoor are limited by the fairness property of the OTTOWF.

Our protocol requires an OTTOWF with verifiable setup, so it can be used with the construction of Section 3.4, or the constructions of Sections 3.2 and 3.3 extended with NIZK proof of correctness for the parameters. The construction based on Paillier's trapdoor permutation can also be used without NIZK proofs but under an additional complexity assumption. The signature scheme must of course be existentially unforgeable, but despite this minimal requirement any scheme can be chosen, either for optimizing the efficiency or for obtaining some specific extra properties. We now briefly analyze the security.

- The checks at the beginning of Step 2 ensure the responder that the partial signature he has received is valid, that the parameters for the OTTOWF are correct (in particular, that he will be able to recover a trapdoor if the initiator uses his one) and that if he ever gets a trapdoor he will be able to find preimages to h_I (and hence to compute a full signature of the initiator).

– The checks at the beginning of Step 3 ensure the initiator that he can extract a preimage of h_R and that the partial signature is valid.
– If the initiator simply aborts after Step 2, none of the parties has a valid full signature so the protocol is fair.
– The initiator can abort after Step 2 and use his trapdoor to build a full signature of the responder. However, the fairness property of OTTOWF ensures that the responder, after seeing this signature, will in turn be able to build a full signature of the initiator. (Fairness is only provided in the same reduced sense as in concurrent signatures.)
– If the responder aborts after Step 3, the initiator can extract a preimage of h_R to get a full signature.

This protocol resembles concurrent signatures protocol but it is not anonymous. Therefore after Step 2 of our protocol both parties have committed themselves to signing, and the responder can prove to a court or any other third party that the initiator has committed himself to signing. On the other hand, unlike the "trivial protocol" above, both parties can also generate full signatures without any communication with the other party.

4.3 Fair exchange with STTP

Stronger fairness guarantees can be achieved if the parties are willing to partially rely on a *semi-trusted third party* (STTP).

Verifiably committed signatures *Verifiably committed signatures* schemes (VCS) [17] rely on an STTP that is able to convert partial signatures into full signatures. VCS have the usual four communication rounds; the parties first exchange partial signatures and then full signatures. If one of the parties attempts to cheat or aborts the protocol, the STTP can be called by the other party to convert partial signatures into full signatures. In practice, it is clearly desirable to reduce as much as possible both the work of the STTP and the trust the parties have to place on it. The definition of VCS [17] also requires that signatures converted by the SSTP are computationally indistinguishable from "regular" full signatures.

Protocol 2: OTTOWF for fair exchange with STTP In our second protocol, we assume the existence of a STTP, and we let the STTP generate the parameters of an OTTOWF and distribute the key to the parties. The protocol is described in Figure 1. It is very similar to verifiably committed signatures [17], but the fairness property of the OTTOWF allows further reducing the trust placed into the STTP. Like in our first protocol, partial signatures are signatures of the message concatenated with the image of a randomness by an OTTOWF, and a full signature is made of a partial signature and the randomness. Since the STTP holds the trapdoor of the OTTOWF, he can be called whenever one of the parties attempts to cheat or when a communication problem occurs.

Like our first protocol, this protocol requires an unforgeable signature scheme and an OTTOWF with verifiable setup. Upon the appending of NIZK proofs

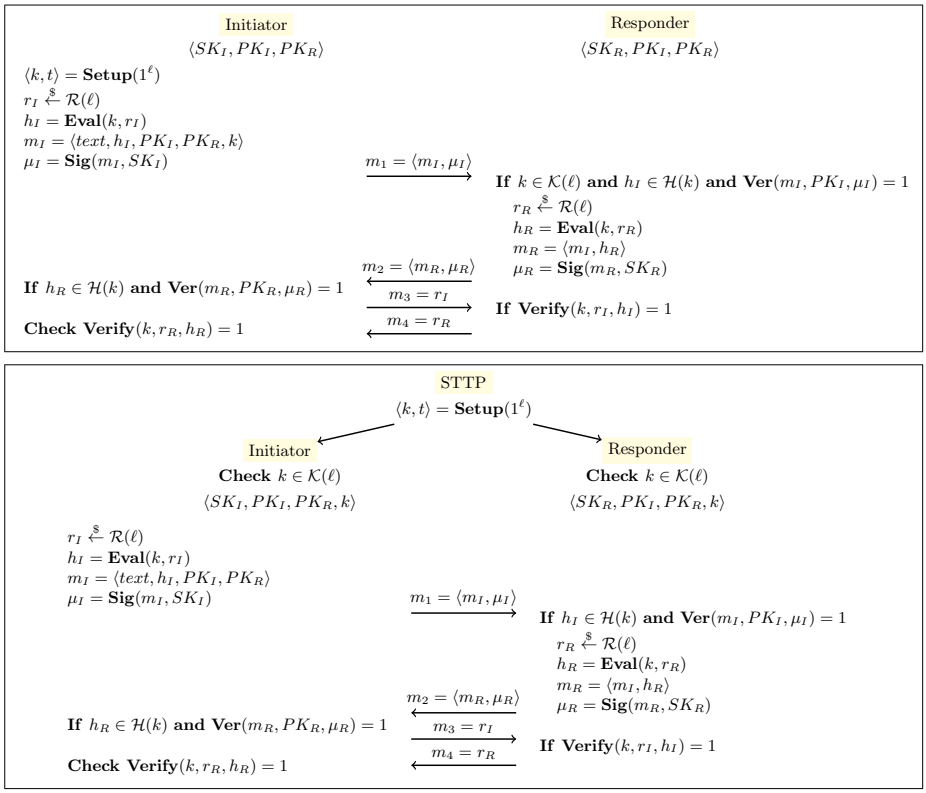

Fig. 1. Description of Protocol 1 (the trapdoor is generated by the initiator) and Protocol 2 (the trapdoor is generated by a semi-trusted third party). We assume that the private and public key pairs have been previously distributed by a PKI.

for the constructions based on Rabin and Paillier, the three constructions of Section 3 can be used. If the OTTOWF used has uniform inversion, the protocol becomes a particular case of VCS since converted full signatures and regular signatures become indistinguishable. In particular, this protocol is a VCS if it is used with the extended Rabin or the modified Paillier OTTOWF constructions. The construction based on Paillier can be used without NIZK proof as well under an additional complexity assumption.

We now analyze the security of the protocol. As a first step, we show that it is fair if the STTP behaves honestly. The checks at the beginning of Steps 2 and 3 ensure that the STTP will be able to convert a partial signature into a full one, and that the partial signatures sent were valid. Therefore, if either the initiator or the responder aborts the protocol at Step 3 or 4 respectively, the other party can go to the STTP and provide a proof that the first two rounds have taken place. In that case and after verifying that both partial signatures

were valid, the STTP will release his trapdoor t, and both parties will be able to convert partial signatures into full signatures.

The advantage of our approach appears when the STTP colludes with the initiator. For VCS, this breaks down all fairness properties, since the STTP can convert the responder's partial signature after Step 2. Fairness of VCS can be recovered in that situation, but only with the help of an external dispute resolution system (a kind of "super honest" STTP, like a court of justice for example). This restricted notion of fairness is called *weak fairness* by Asokan et al. [1]. On the other hand, when the STTP colludes with the initiator our second protocol actually amounts to our first one, and it therefore provides the same fairness guarantees to the responder without any external resolution system.

If the STTP colludes with the responder, then he can either convert the initiator's partial signature after Step 1, or not convert the responder's partial signature when this one aborts after Step 3. Both attacks are also possible against VCS, and both for VCS and for our protocol they require an external dispute resolution system to solve them.

Our protocol reduces the trust that the responder has to place on the STTP, which is useful in practical scenarios where one of the parties is much more likely to corrupt the STTP. To be "fair" in our comparison with VCS, we mention a drawback of our approach with respect to at least "the spirit" of VCS. In many cases for VCS, in particular when they are built based on *verifiably encrypted signatures* [2], the STTP can solve many conflicts without refreshing its parameters. In our protocol, the STTP must refresh his parameters every time he uses them, since the trapdoor becomes public after each use. Therefore, our protocol will become practical only for small infrastructures or for very *optimistic* contexts (in its usual sense in fair exchange), when a very large proportion of the protocol instantiations will conclude normally.

5 Conclusion

In this paper, we considered *leaking trapdoors*, trapdoors that are revealed as soon as they are used. We introduced *one-time trapdoor one-way functions* (OTTOWF), a new cryptographic primitive capturing this notion. We provided three OTTOWF constructions based on classical cryptographic assumptions. Finally, we used our primitive to build two new fair exchange protocols that improve state of the art solutions.

Acknowledgments

We would like to thank Benoit Libert for his careful and very useful review. We also thank Sylvie Baudine for her help in improving the paper.

References

1. N. Asokan, M. Schunter, and M. Waidner. Optimistic protocols for fair exchange. In *ACM Conference on Computer and Communications Security*, pages 7–17, 1997.

2. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures (extended abstract). In *EUROCRYPT*, pages 591–606, 1998.
3. G. Ateniese. Verifiable encryption of digital signatures and applications. *ACM Trans. Inf. Syst. Secur.*, 7(1):1–20, 2004.
4. F. Bao. An efficient verifiable encryption scheme for encryption of discrete logarithms. In J.-J. Quisquater and B. Schneier, editors, *CARDIS*, volume 1820 of *Lecture Notes in Computer Science*, pages 213–220. Springer, 1998.
5. F. Bao, R. H. Deng, and W. Mao. Efficient and practical fair exchange protocols with off-line ttp. In *IEEE Symposium on Security and Privacy*, pages 77–85. IEEE Computer Society, 1998.
6. F. Bao, G. Wang, J. Zhou, and H. Zhu. Analysis and improvement of micali's fair contract signing protocol. In H. Wang, J. Pieprzyk, and V. Varadharajan, editors, *ACISP*, volume 3108 of *Lecture Notes in Computer Science*, pages 176–187. Springer, 2004.
7. M. Bellare, S. Halevi, A. Sahai, and S. P. Vadhan. Many-to-one trapdoor functions and their ralation to public-key cryptosystems. In H. Krawczyk, editor, *CRYPTO*, volume 1462 of *Lecture Notes in Computer Science*, pages 283–298. Springer, 1998.
8. M. Ben-Or, O. Goldreich, S. Micali, and R. L. Rivest. A fair protocol for signing contracts (extended abstract). In W. Brauer, editor, *ICALP*, volume 194 of *Lecture Notes in Computer Science*, pages 43–52. Springer, 1985.
9. D. Boneh, C. Gentry, B. Lynn, and H. Shacham. Aggregate and verifiably encrypted signatures from bilinear maps. In E. Biham, editor, *EUROCRYPT*, volume 2656 of *Lecture Notes in Computer Science*, pages 416–432. Springer, 2003.
10. H. Bürk and A. Pfitzmann. Digital payment systems enabling security and unobservability. *Computers & Security*, 8(5):399–416, 1989.
11. J. Camenisch and I. Damgård. Verifiable encryption, group encryption, and their applications to separable group signatures and signature sharing schemes. In T. Okamoto, editor, *ASIACRYPT*, volume 1976 of *Lecture Notes in Computer Science*, pages 331–345. Springer, 2000.
12. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In D. Boneh, editor, *CRYPTO*, volume 2729 of *Lecture Notes in Computer Science*, pages 126–144. Springer, 2003.
13. J. Cathalo and C. Petit. One-time trapdoor one-way functions (full version). `http://www.dice.ucl.ac.be/~petit/files/ISC2010full.pdf`.
14. L. Chen, C. Kudla, and K. G. Paterson. Concurrent signatures. In C. Cachin and J. Camenisch, editors, *EUROCRYPT*, volume 3027 of *Lecture Notes in Computer Science*, pages 287–305. Springer, 2004.
15. I. Damgård. Practical and provably secure release of a secret and exchange of signatures. In *EUROCRYPT*, pages 200–217, 1993.
16. Y. Dodis, P. J. Lee, and D. H. Yum. Optimistic fair exchange in a multi-user setting. In T. Okamoto and X. Wang, editors, *Public Key Cryptography*, volume 4450 of *Lecture Notes in Computer Science*, pages 118–133. Springer, 2007.
17. Y. Dodis and L. Reyzin. Breaking and repairing optimistic fair exchange from PODC 2003. In M. Yung, editor, *Digital Rights Management Workshop*, pages 47–54. ACM, 2003.
18. J. A. Garay, M. Jakobsson, and P. D. MacKenzie. Abuse-free optimistic contract signing. In M. J. Wiener, editor, *CRYPTO*, volume 1666 of *Lecture Notes in Computer Science*, pages 449–466. Springer, 1999.
19. R. Gennaro, D. Micciancio, and T. Rabin. An efficient non-interactive statistical zero-knowledge proof system for quasi-safe prime products. In *ACM Conference on Computer and Communications Security*, pages 67–72, 1998.

20. S. Goldwasser and S. Micali. Probabilistic Encryption and How to Play Mental Poker Keeping Secret All Partial Information. In *STOC*, pages 365-377. ACM, 1982.
21. S. Goldwasser, S. Micali, and R. L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17:281–308, 1988.
22. Q. Huang, G. Yang, D. S. Wong, and W. Susilo. Ambiguous optimistic fair exchange. In J. Pieprzyk, editor, *ASIACRYPT*, volume 5350 of *Lecture Notes in Computer Science*, pages 74–89. Springer, 2008.
23. Q. Huang, G. Yang, D. S. Wong, and W. Susilo. Efficient optimistic fair exchange secure in the multi-user setting and chosen-key model without random oracles. In T. Malkin, editor, *CT-RSA*, volume 4964 of *Lecture Notes in Computer Science*, pages 106–120. Springer, 2008.
24. J. Liu, R. Sun, W. Ma, Y. Li, and X. Wang. Fair exchange signature schemes. *Advanced Information Networking and Applications Workshops, International Conference on*, 0:422–427, 2008.
25. S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters. Sequential aggregate signatures and multisignatures without random oracles. In S. Vaudenay, editor, *EUROCRYPT*, volume 4004 of *Lecture Notes in Computer Science*, pages 465–485. Springer, 2006.
26. W. Mao. Verifiable escrowed signature. In V. Varadharajan, J. Pieprzyk, and Y. Mu, editors, *ACISP*, volume 1270 of *Lecture Notes in Computer Science*, pages 240–248. Springer, 1997.
27. S. Micali. Simple and fast optimistic protocols for fair electronic exchange. In *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 12–19, New York, NY, USA, 2003. ACM.
28. P. Paillier. A trapdoor permutation equivalent to factoring. In H. Imai and Y. Zheng, editors, *Public Key Cryptography*, volume 1560 of *Lecture Notes in Computer Science*, pages 219–222. Springer, 1999.
29. J. M. Park, E. K. P. Chong, and H. J. Siegel. Constructing fair-exchange protocols for e-commerce via distributed computation of rsa signatures. In *PODC '03: Proceedings of the twenty-second annual symposium on Principles of distributed computing*, pages 172–181, New York, NY, USA, 2003. ACM.
30. M. O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. Technical report, Cambridge, MA, USA, 1979.
31. R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.
32. J. Rompel. One-way functions are necessary and sufficient for secure signatures. In *STOC*, pages 387–394. ACM, 1990.
33. M. Rückert and D. Schröder. Security of verifiably encrypted signatures and a construction without random oracles. In H. Shacham and B. Waters, editors, *Pairing*, volume 5671 of *Lecture Notes in Computer Science*, pages 17–34. Springer, 2009.
34. J. Zhang and J. Mao. A novel verifiably encrypted signature scheme without random oracle. In E. Dawson and D. S. Wong, editors, *ISPEC*, volume 4464 of *Lecture Notes in Computer Science*, pages 65–78. Springer, 2007.
35. H. Zhu and F. Bao. Stand-alone and setup-free verifiably committed signatures. In D. Pointcheval, editor, *CT-RSA*, volume 3860 of *Lecture Notes in Computer Science*, pages 159–173. Springer, 2006.