Probabilistic Escrow of Financial Transactions with Cumulative Threshold Disclosure

Stanisław Jarecki[1] and Vitaly Shmatikov[2]

[1] University of California, Irvine
[2] University of Texas at Austin

Abstract. We propose a scheme for privacy-preserving escrow of financial transactions. The objective of the scheme is to preserve privacy and anonymity of the individual user engaging in financial transactions until the cumulative amount of all transactions in a certain category, for example all transactions with a particular counterparty in any single month, reaches a pre-specified threshold. When the threshold is reached, the escrow agency automatically gains the ability to decrypt the escrows of all transactions in that category (and only that category).

Our scheme employs the *probabilistic polling* idea of Jarecki and Odlyzko [JO97], amended by a novel robustness mechanism which makes such scheme secure for malicious parties. When submitting the escrow of a transaction, with probability that is proportional to the amount of the transaction, the user reveals a share of the key under which all his transactions are encrypted. Therefore, the fraction of shares that are known to the escrow agency is an accurate approximation of the fraction of the threshold amount that has been transacted so far. When the threshold is reached, with high probability the escrow agency possesses all the shares that it needs to reconstruct the key and open the escrows. Our main technical contribution is a novel tool of *robust probabilistic information transfer*, which we implement using techniques of optimistic fair 2-party computation.

1 Introduction

Increasing demands by law enforcement and regulatory agencies to monitor financial transactions are gradually eroding individual and organizational privacy. A common legal requirement is that all transactions exceeding a certain threshold (*e.g.*, $10,000 for currency transactions in the U.S.) must be reported to the financial authorities. Moreover, if a financial institution suspects that a customer is engaged in "structuring" his transactions so as to avoid the reporting requirement (*e.g.*, making regular cash deposits just under the reporting threshold), the institution is required to report these transactions, too. This may lead to an unnecessary loss of privacy, since the transactions in question may be innocuous.

Building on the transaction escrow scheme of [JS04], we propose an efficient technical solution to the problem of reporting structured transactions. Our goal is to balance individual privacy with the legally mandated cumulative threshold

A.S. Patrick and M. Yung (Eds.): FC 2005, LNCS 3570, pp. 172–187, 2005.

disclosure requirement, *e.g.*, "all transactions of any individual totaling T or more must be disclosed". Our scheme guarantees the following properties:

Privacy: With high probability, an individual whose transactions total less than the pre-specified threshold T enjoys *provable* anonymity and privacy. In particular, a malicious escrow agency cannot feasibly open the escrowed transactions whose cumulative amount is less than this threshold.

Cumulative Threshold Disclosure: Once the total amount of some individual's escrowed transactions exceeds the pre-specified threshold T, then with high probability the escrow agency is able to (i) efficiently identify these transactions among all escrows it has collected, and (ii) automatically open these (and only these) escrows without help from their creator.

We achieve these properties assuming a trusted third party (TTP), which is only invoked *optimistically*. The role of the TTP can be naturally played by the Key Certification Authority, whose presence is required in any case in any realistic transaction escrow system. Our protocols are *optimistic* in the sense that the TTP is contacted only if one of the parties notices that the other one misbehaves. The effect of interaction with the TTP is equivalent to interaction with an honest counterparty in the protocol, hence there is no incentive for either player to diverge from the protocol specification. Therefore, in practice the TTP should only be invoked in the (rare) cases of certain communication failures.

Both privacy and cumulative threshold disclosure properties in our scheme are *probabilistic*: (1) there is a small *probability of erroneous disclosure*, *i.e.*, that some individual's transactions will be revealed to the escrow agency even though they total less than the pre-specified threshold, and (2) there is also a small *probability of erroneous non-disclosure*, *i.e.*, that some individual's transactions will not be disclosed even though they total more than the threshold. Both probabilities decrease sharply with the distance separating the cumulative transaction amount and the threshold T (*i.e.*, it is highly unlikely that privacy of some individual will be compromised if the cumulative amount of his transactions is significantly below T, or that he will avoid disclosure if it is significantly higher than T). Our scheme provides a tradeoff between the computation and communication complexity of interaction between the user and the escrow agency, and the sharpness of the slope of these functions.

Overview of Transaction Escrow. The concept of verifiable transaction escrow was introduced in [JS04], but the escrow scheme in [JS04] does not support *cumulative* disclosure conditions which are the focus of the present paper. Following [JS04], we refer to the individual performing the transaction, *e.g.*, a bank transfer or a stock purchase, as the *user* (U), and the escrow agency collecting the escrows as the *agency* (A). We assume that U and A communicate over an anonymizing channel. In particular, U may send information to and engage in zero-knowledge protocols with A through a proxy, without revealing U's true identity. We refer to the full description of any transaction as the transaction *plaintext*. We'll say that transactions are *related* if they belong to the same *category*, and to simplify the exposition, we'll equate the category with the user's

identity. In real applications, the category of a transaction might be more fine-grained, and determined not only by the user's identity, but also by any predicate on the transaction plaintext, such as the type of the transaction, the payee's identity, the jurisdiction of the payee's financial institution, *etc.*

A transaction escrow scheme, as introduced in [JS04], must ensure **category-preserving anonymity**: the only information the escrow agency can learn from any two escrowed transactions is whether or not they originate from the same user. Importantly, the agency does not learn which user this is.[1] The scheme of [JS04] can also support **simple threshold disclosure**: the agency can efficiently identify and de-escrow all transactions that belong to the same category once the *number* of such transactions reaches a pre-specified threshold.

Cumulative Disclosure Conditions for Financial Transactions. A simple threshold disclosure condition described above cannot efficiently support monitoring of financial flows, because financial oversight laws usually call for transactions of a certain type to be reported to the monitoring agency based on the total *value* of the transactions and not just their *number*. Indeed, this objective is difficult to achieve with any system in which disclosure is based just on the number of transactions. No matter how we set the limit which determines when a single transaction needs to be escrowed and the number of transactions that should lead to automatic disclosure, the person performing the transactions can divide his transactions into small pieces, each of which stays below the threshold level.

2 Overview of Escrow with Cumulative Disclosure

Let T be the pre-specified cumulative disclosure threshold for transactions originating from a single individual (*e.g.*, $10,000 for financial transactions in the U.S.). Conceptually, we split the threshold T into d parts, *e.g.*, $d = 20$ (in section 6, we discuss how to choose d and we describe the trade-off between efficiency and the probability of erroneous disclosure or non-disclosure). All transactions that belong to the same category are encrypted with the same key, using a verifiable anonymous encryption scheme. The key itself is split by the user into d shares using standard verifiable secret sharing techniques [Fel87].

Our scheme follows the "probabilistic polling" idea proposed by Jarecki and Odlyzko for a micropayment scheme [JO97]. Whenever the user performs a transaction for some amount $t \leq \frac{T}{d}$ (higher amounts need to be subdivided into pieces of at most $\frac{T}{d}$ value) and submits the corresponding escrow to the agency, the user must also reveal one share of his encryption key with probability exactly equal to $\frac{d}{T} * t$. If the probability of submitting a share is set in this way, then regardless of the size t of the individual transactions that make up a cumula-

[1] Note that this requirement precludes the traditional escrow solutions where plaintext data is encrypted under escrow agency's public key, as the escrow agency would then in principle be always able to decrypt all the escrowed data.

tive amount A, the expected number of shares generated by a user who escrows $n = \frac{A}{t}$ transactions will be $n(\frac{d}{T} * t) = \frac{A}{T}\ d$, which is independent of t.

When total amount reaches $A = T + \delta$, regardless of the pattern of transactions, with probability that grows steeply with δ the escrow agency will have obtained d shares, enabling it to reconstruct the key and open all escrows of that user. Because the agency cannot feasibly decrypt the escrows until it collects d shares, all transactions of a user whose cumulative transaction value A is $A = T - \delta$ will stay secret with probability that again increases sharply with δ.

Robust Probabilistic Information Transfer with a Fair Coin Toss. To guarantee that the share is transferred from the user to the agency with the required probability, we develop a joint coin tossing protocol between the user and the agency based on fair exchange of random contributions (encrypted under the trusted third party's public key) using the standard techniques of optimistic fair exchange of secrets (see, *e.g.*, [ASW00]). In addition to committing to his random contribution, the user verifiably commits to a share of the escrow key, using the verifiable encryption scheme of Camenisch and Shoup [CS03], and "signs" (see below) the transcript of the protocol up to that point. The parties then de-commit their contributions to the joint coin toss, and if the resulting coin toss indicates that the share must be revealed, the user is expected to open his commitment. If the user refuses to de-commit his random contribution correctly, or refuses to reveal the share itself, the agency can appeal to a trusted third party, who will open the escrow and reveal the user's share if the joint coin toss should indeed result in a transfer of the share. Thus neither the user, nor the agency can skew the probability with which the key share is transferred between them.

Note that the agency must be able verify the user's signatures without learning his identity. Since the TTP is allowed to know the user's identity, we combine the unlinkable credentials technique of [CL01] with the verifiable encryption of [CS03] and have the user issue signatures under a public key which is itself encrypted under the TTP's public key. The escrow agency does not learn the user's public key, but can verify that (1) some CA-certified valid public key was used and the TTP will be able to identify it, and (2) the transcript was signed under that key, and the TTP will be able to verify it.

There is a small privacy leak in our scheme since the escrow agency must know the probability with which the information is to be transferred. Since this probability is proportional to the transaction value, the agency essentially learns this value. This leak appears harmless in practice since the agency does *not* learn the identity of the user, or anything else about the transaction plaintext, except that the transaction must be related to some other previously escrowed transactions, and thus that they all originate from the same (unknown) user.

Related Work. Our scheme employs Shamir's polynomial secret sharing in such a way that user's revelation of enough shares enable the escrow agency to recover

the user's keys and decrypt his/her escrowed data. Similar idea was proposed for secure metering of web accesses by Naor and Pinkas [NP98], but in our scheme this idea is is extended so that (1) it can be used in conjunction with a PKI system, so that the secret sharing is determined by the user's private key, (2) the generated shares must be linkable to each other but unlinkable to their originator, and (3) the shares need to be generated only with some probability, and this probabilistic generation must be fair.

Our notion of a probabilistic information transfer owes much to works on 2-party coin tossing [Blu82] and two-party secure computation in general [Can00]. Our implementation of this functionality utilizes the techniques and the model of the 2-party computation with off-line trusted third party, used *e.g.*, by the secret exchange protocol of Asokan, Shoup, and Waidner [ASW00], and by the general fair 2-party computation protocol of Cachin and Camenish [CC00].

3 Model and Definitions

A transaction escrow system involves an *Escrow Agency* and any number of *Users*. Users engage in financial transactions such as stock purchases, wire transfers, *etc.* For the purposes of this paper, we will focus on one application, in which the transactions are wire transfers (or more properly, wire transfer requests) and the counterparties of these transactions (*i.e.*, the entities the perform them on users' behalf) are banks and other financial services providers. As mentioned in the introduction, each transaction is fully described by its *plaintext*, and we define the *category* of the transaction as simply the user's identity. To make this identity unambiguous, we assume a global PKI with a trusted Certification Authority who issues a unique public key credential to every user.

In our scheme the user, knowing the plaintext of his intended transaction, first performs a protocol with the escrow agency in which he sends to the agency a transaction *escrow* and in return receives the agency's receipt. The user then engages in the transaction with the counterparty, and the counterparty verifies that the user holds a valid receipt for this transaction. Note that we have no hope of escrowing transactions in which a counterparty aids the user in avoiding the escrow protocol by foregoing the verification of the escrow receipt. Simply speaking, if some user and counterparty want to conduct an un-monitored transaction, they can. The transaction escrow scheme can help in monitoring only transactions in which at least one of the participants, the user or the counterparty, enables this monitoring. Similarly, the user's privacy against the escrow agency can only be protected for transactions with honest counteparties. A dishonest counterparty can always forward the transaction plaintext to the agency.

We call a transaction escrow scheme $(\alpha_{T,t}, \beta_{T,t})$**-probabilistic cumulative threshold escrow** if it satisfies the following properties, where $\alpha_{T,t}$ and $\beta_{T,t}$ are both functions from real values to the $[0, 1]$ interval, T is the global pre-specified cumulative privacy threshold, and t is the minimum allowed transaction size.

$\alpha_{T,t}$-probabilistic cumulative threshold disclosure. Independently for every user, regardless of his transaction pattern, if the user escrows transactions whose total cumulative value equals $A = T + \delta$, then with probability at least $1 - \alpha_{T,t}(\delta)$ (minus a negligible amount), all transactions of *this* user can be efficiently identified and de-escrowed by the agency.

$\beta_{T,t}$-probabilistic amount-revealing privacy. For any two escrows e, e' of two transactions conducted with some honest counterparties, the only thing that a (potentially malicious) escrow agency learns about these transactions is (1) whether or not they originate with the same user, and (2) the numerical amounts $val(e)$, $val(e')$ transacted in each case. Moreover, regardless of the user's transaction pattern and of the actions of the escrow agency, if the escrows correspond to transactions whose total cumulative value equals $A = T - \delta$, then all transactions of *this* user are revealed to the agency protection with probability at most $\beta_{T,t}(\delta)$ (plus a negligible amount).

Unlike in [JS04], disclosure depends probabilistically on the cumulative transacted amount. With probability of at least $1 - \alpha(\delta)$, which approaches 1 as $\delta = A - T$ increases, the escrow agency can open all escrowed transactions of a user whose transactions add up to A. Therefore, α represents the risk of not being able to open some user's escrows even though their cumulative transacted amount is higher than the threshold. Also, there is an additional privacy relaxation: β represents the risk of privacy violation for users whose cumulative transaction amount does not yet reach the pre-specified threshold.

Our escrow scheme is actually a family of schemes, each of which is an $(\alpha_{T,t}, \beta_{T,t})$-probabilistic cumulative threshold escrow scheme for some functions $\alpha_{T,t}$, and $\beta_{T,t}$. As the number of shares increases (and the scheme becomes less efficient), the "accuracy" of probabilistic disclosure gets better in the sense that for any value t, the two functions decrease more sharply, which reduces the risk of both erroneous disclosure and erroneous non-disclosure. Both functions decrease slower (and hence get worse) when the minimum transaction size t decreases. However, the impact of t on both these functions seem very small, and we conjecture that α and β will stay approximately the same even for very small values of t, thus eliminating the need for the minimum transaction size restriction.

4 Basic Threshold Escrow

Before summarizing the transaction escrow scheme of [JS04], we'd like to emphasize the difference between that scheme and the scheme proposed in this paper. In [JS04], the disclosure condition is, roughly, as follows: "If the number of transactions, *each* of which originates from the same user and satisfies a particular condition, is greater than some threshold d, then open the corresponding escrows." In this paper, the disclosure condition is as follows: "If the transactions *jointly* satisfy a particular condition (namely, the total transacted amount is above some threshold T), then open the corresponding escrows." One of the contributions of this paper is to build on the techniques of [JS04] to support a disclosure condition that spans *multiple* transactions of the same user.

4.1 Cryptographic Toolkit

Our constructions rely on the hardness of the Decisional Diffie-Hellman (DDH) problem in subgroup QR_p of quadratic residues in $\mathbb{Z}_p^*$, where p, q are large primes such that $p = 2q + 1$, and g is a generator of $\mathbb{Z}_p^*$. Our basic cryptographic tool is a *verifiable random function* (VRF) family, implemented in the Random Oracle Model and based on DDH. Let $H : 0,1^* \rightarrow \mathbb{Z}_p^*$ be an ideal hash function. The VRF family is defined by

(i) the key generation algorithm that picks a secret key $k \in \mathbb{Z}_q^*$ and the corresponding public key $pk = g^{2k} \bmod p$,
(ii) the evaluation algorithm $\mathsf{Eval}_k(x)$ which outputs $y = H(x)^{2k} \bmod p$ and a non-interactive zero-knowledge proof π of equality of discrete logarithms $x = \mathsf{DL}_h(y) = \mathsf{DL}_h(pk)$, which proves that the function was computed correctly (such proof can be accomplished in ROM with a few exponentiations using standard techniques, *e.g.*, [CP92]), and
(iii) the verification algorithm for verifying proof π of discrete-log equality.

4.2 Basic Transaction Escrow with Simple Threshold Disclosure

We assume that every user U is initialized with a secret key k_U, chosen at random in $\mathbb{Z}_q^*$, and that the corresponding public key $pk_U = g^{2k_U}$ is signed by the Certification Authority. We assume that the escrow agency has been initialized with the public/private key pair of an *unlinkable* CMA-secure signature scheme of Camenisch-Lysyanskaya [CL01], and that the disclosure threshold d is a global constant. We say that two transactions m and m' belong to the same *category* if and only if they are originate with the same user.[2]

Suppose user U wishes to perform a transaction described by plaintext m with some counterparty C. Before carrying out the transaction, C demands that the user present a *receipt* from the escrow agency, proving that the latter has received a correctly formed escrow of the transaction. U starts by picking a unique $(d-1)$-degree secret-sharing polynomial f. The coefficients are computed as $k_i = H(k, i)$ where $H : 0,1^* \rightarrow \mathbb{Z}_q$ is a pseudorandom function, and the polynomial is defined as $f(x) = k_0 + k_1 x + \ldots + k_{d-1} x^{d-1}$ *mod* q. Values $\{C_0, \ldots, C_d\}$ where $C_i = g^{2k_i}$ *mod* p serve as commitments to the coefficients.

The user sends to the escrow agency (via an anonymizing channel):

(i) *Tag* $t = \mathsf{Eval}_k(1)$, which allows the escrow agency to separate escrows into categories (note that t is constant for all transactions of the same category);
(ii) *Ciphertext* $c' = (c, \{C_i\}_{i=0,\ldots,d-1}, f(x))$. Here $c = (\mathsf{pad}^H(m|r))^{2k_0}$ *mod* p is the Pohlig-Hellman encryption of (padded) m under the key k_0, $\{C_i\}$ are the commitments to the polynomial coefficients, $x = H(c)$ is point in $\mathbb{Z}_q^*$ assigned for c, and $f(x)$ is the value of the polynomial on x;
(iii) *Anonymous signature* s on (t, c') computed as $s = \mathsf{Eval}_k(t, c')$.

[2] We simplify the scheme of [JS04] by assuming that all transactions are of the same type, and so only the user's identity determines the transaction category.

The escrow agency verifies that $(x, f(x))$, for $x = H(c)$, is a true data point on the polynomial committed to in $\{C_i\}$ by checking that $g^{2f(x)} = C_0 * C_1^x * \ldots * C_{d-1}^{(x^{d-1})}$ *mod* p. If there already exist escrows in the same category (*i.e.*, the escrow agency has previously received escrows with the same tag t), the agency checks that the commitments $\{C_i\}$ are the same as those supplied with previous escrows in the category. If the checks are successful, the escrow agency signs tuple (t, s, c, C_0) using the unlinkable signature scheme of [CL01], and returns the signature to the user as the escrow *receipt*. We omit the details of the protocol from [JS04] by which user U proves to the counterparty (who knows transaction plaintext m and the U's public key pk, but not U's secret k) that U possesses the receipt on a correctly formed escrow for this transaction, which implies that U must have given the correctly formed escrow to the escrow agency.

Provided that the escrow receipts are verified by honest counterparties, the above scheme provides automatic and unavoidable threshold disclosure. With each escrow, the user must submit a new *share* $f(x)$ of the $(d-1)$-degree polynomial f, and each escrow contains an encryption of the transaction plaintext under the same key $k_0 = f(0)$. Once d escrows have been collected in the same category, the escrow agency can interpolate the polynomial f from the d shares, compute $f(0)$ and decrypt all these escrows. Otherwise, this user's escrows remain secret to the agency.

5 Escrow with Cumulative Threshold Disclosure

To replace simple threshold disclosure with *cumulative* disclosure, we need to change the basic protocol of section 4.2 in which the user supplies a single secret-share $s = f(x)$ of the key k_0 that encrypts all of his transactions. As explained in section 2, s must be transferred to the agency with probability equal to $\theta = \frac{d}{T} * t$ where t is the value associated with this transaction, a.k.a. *transaction size*. We achieve this using a novel tool we call *robust probabilistic information transfer*.

5.1 Probabilistic Information Transfer: Definition

A probabilistic information transfer protocol is a protocol between two parties, user U and agency A. The public input is the probability $\theta \in [0, 1]$ with which information transfer should take place, the user's private input is the information that might be transfered, which in our case is the share $s = f(x)$, and the public input is a commitment to this information, which in our case is $C_s = g^s \bmod p$. Because we are interested in protocols that assume a trusted third party, we allow for this protocol to involve the *third* party, TTP. Even though a probabilistic information transfer will thus be a protocol between three parties U, A, and TTP, our secure implementation of that notion will involve the TTP party only in case one of the parties is faulty, and thus the protocol we propose works in the "optimistic off-line trusted third party" model, similarly to, *e.g.*, the fair-exchange protocol of [ASW00] or the general fair 2-party computation protocol of [CC00]. As in [CC00], we will assume that T has as the secret input its secret key

sk_{TTP}, while pk_{TTP} is publicly known. Finally, we assume that A, the agency, has a private/public key pair (k_A, pk_A), too, where k_A is its private key for a VRF function and pk_A is its verification counterpart. Additionally, we allow for an auxiliary public input aux, which represents the reference to some transaction to which this transfer refers. In our probabilistic escrow application, we will use aux to represent the escrow (t, c', s) (see section 4.2 above) on account of which this instance of the probabilistic information transfer protocol is executed.

Ideal Functionality for Probabilistic Information Transfer. The simplest way to describe our desired security property is to specify the *ideal functionality* I for the protocol, following the secure function evaluation paradigm (*e.g.*, [Can00]). We define a secure probabilistic information transfer protocol as a protocol that securely realizes this ideal functionality I in the *static* adversarial model where the adversary can corrupt (statically) either the user U or the agency A, but the trusted third party TTP is never corrupted.[3]

As mentioned above, we assume that the public input in this protocol consists of commitment $C_s = g^s \bmod p$ on U's information s, A's public VRF verification key pk_A, TTP's public encryption key pk_{TTP}, the probability θ, and the auxiliary public input aux. Given these public inputs, the ideal functionality I for probabilistic information transfer proceeds as follows. U can contribute some value s or a special symbol $\perp$, which designates U's refusal to participate. A contributes his private key k_A and either of the two special symbols: $\diamond$, which designates A's acquiescence, or $\perp$, his refusal to participate. The TTP party contributes his private key sk_{TTP}. The ideal functionality responds as follows. If either party contributes symbol $\perp$, or if $C_s \neq g^s \bmod p$, or if k_A does not correspond to pk_A, the ideal functionality I outputs $\perp$ to both U and A. Otherwise, I casts a coin r uniformly distributed in $[0, 1]$ and hands r to both U and A. Moreover, if $r < \theta$ then I also gives s to A. The outputs of TTP are null in every case.

This ideal functionality implements a secure probabilistic information transfer of s from U to A with probability θ, with the following caveats:

(1) The commitment C_s to the information s is known beforehand to A, and this commitment could contain some information about s.
(2) Whenever both parties start the protocol, but one of them decides to withdraw (by contributing the $\perp$ input), the other party learns about this;
(3) If U decided to proceed, then A learns if the odds came to his disadvantage and the message s has not been transferred to him; and
(4) U, too, learns whether the information has been transferred to A or not, and thus this probabilistic transfer protocol is *non-oblivious*.

Definition 1. *We call a protocol between U, A, and TTP, a* (statically) secure probabilistic information transfer protocol in the trusted third party model *if it*

[3] We note that Cachin and Camenish [CC00] who define a general notion of *fair* 2-party computation in the same optimistic third party model as we have here, allow for TTP to be corrupted as well, but this extra corruption ability seems unnecessary.

securely implements the above ideal functionality in the adversarial model where the adversary (statically) corrupts either the U or the A party, but never the TTP. We call such protocol optimistic *if the TTP party is contacted only in case either U or A is corrupted.*

Contributory Protocols. In the sequel we will consider only a special class of protocols that realize such functionality securely, namely a "contributory coin-toss" protocols, where the two players U and A make contributions r_U and r_A which are uniquely defined by the messages sent by each player, where the resulting coin toss is then computed as a deterministic function of these contributions, *e.g.*, $r = r_U \oplus r_A$, and where the information s is transferred if and only if $r < \theta$.

No Strategic Advantage for U. If a probabilistic information transfer protocol is a secure implementation of the above ideal functionality, then such protocol offers no strategic advantage to U in the following sense. If U ever decides to withdraw from the protocol, he may only do so *before* he learns A's contribution to the joint coin toss, and thus the likely outcome of the protocol. Clearly this is the case for the ideal functionality, and thus U's withdrawal at a midpoint of the protocol must be equivalent to a refusal to engage in the protocol in the first place, and thus can only happen before the coin toss r is decided. Consequently, U cannot gain any advantage by stopping and re-running the protocol on new inputs, since he can stop only when he is still oblivious to the outcome.

Technically, this suggests that there should be a communication round in the protocol which we can call "U's commitment point," such that: (1) If U does not execute this round correctly we say that "U stops before the commitment point", and this is equivalent to U contributing the $\perp$ surrender sign in the ideal world. As discussed above, before this commitment point U has only a negligible advantage in predicting the coin toss that determines whether s is going to be transferred to A or not. (2) If U does send this message correctly, this is equivalent to U actually contributing the correct s input in the ideal world. Therefore, if U stops or diverts from the protocol *after* the commitment point, then an honest A must still get the correct result: a fair coin toss r and the s value if the $(r < \theta)$ condition is satisfied. Most likely, A will have to rely on the trusted third party to retrieve the fairly generated r and, depending on the outcome, the correct s, using the messages U sent before (and including) the commitment point.

5.2 Probabilistic Information Transfer: Additional Properties

Observable Accountability. In any escrow scheme, but especially in our case where the agency learns the monetary values of all escrowed transactions, a corrupt agency may stage a directed denial of service attack against some user by refusing to issue receipts on his escrows. (While the agency does not know the user's identity, all escrows of that user are linkable.) While such a DoS attack cannot be prevented, it should at least be made detectable by an independent observer, say, a journalist. Then a user who believes that he is being denied service can

ask the journalist to observe a (re)run of the escrow protocol. If the agency does not reply with a valid receipt, the journalist can observe that the agency is at fault. This "observable accountability" should be satisfied not just by the probabilistic information transfer subprotocol, but also by the entire escrow protocol. (We note, however, that observability in the larger escrow protocol requires some slight modifications to the protocol presented in section 4.2.)

Observable accountability: All actions performed by both parties in the execution of the probabilistic information transfer protocol can be verified without revealing any long-term private information.

Verifiably Deterministic Coin Contribution for A. While giving any outside observer the ability to verify whether the parties follow the protocol correctly can work as a hedge against the denial of service attacks by a malicious agency, it is not sufficient. Suppose that a malicious agency refuses to serve some user if the coin comes out to the agency's disadvantage, but when the user re-runs the protocol, possibly accompanied by an outside observer, the agency performs correctly. This simple cheating strategy for the agent effectively transfers the information s to the agent with probability $1-(1-\theta)^2$, which is greater than θ. To prevent this attack, we will require the following property:

Verifiable deterministic coin contribution for A: In the algorithm specified by the probabilistic information transfer protocol, A's contribution to the coin toss is a *deterministic function* of (1) U's message which commits U's contribution to the coin toss, and (2) the auxiliary input aux (which in our escrow application will be instantiated with an escrow instance on account of which the probabilistic transfer is taking place). Moreover, if a malicious A attempts to compute its contribution differently, this deviation will be detected by U with an overwhelming probability.

If A's contribution to the coin toss is a deterministic function of U's contribution, and if the protocol is observably accountable, then A gains no advantage by first abandoning the protocol when the coin comes out to its disadvantage, and then agreeing to re-run it. However, A's contribution should be the same only when applied to the same instance aux in the context of which this protocol instance was invoked, thus facilitating only genuine re-runs. Otherwise, a malicious U, once discovering a winning combination between his contribution r_U and A's contribution r_A could try to use the same r_U (and hence induce the same lucky r_A response) for many different instances of the protocol.

Note that determinism of A's contribution does *not* imply that U is able to efficiently predict A's contribution to the joint coin toss. In our construction described in section 5.3, A's coin is computed using a verifiable random function (VRF) applied to U's inputs to the protocol. Because U does not know A's private VRF key, the output of the function appears random to U, yet the function is deterministic, and A is able to prove that it was computed correctly.

5.3 Probabilistic Information Transfer: Implementation

Even though any ideal functionality can be securely realized using secure 2-party computation [Yao82], such general techniques do not seem to yield a practical protocol in our case. Instead, we design an efficient (4-round, small constant number of exponentiations for both parties) protocol which securely achieves our ideal functionality assuming the presence of an *offline* Trusted Third Party (TTP). Thus, following the "optimistic" paradigm in two-party securecomputation, the TTP is only involved in case of some active faults in the protocol. In our application the role of the TTP can be naturally played by the Key Certification Authority, because a trusted KCA is required in our escrow scheme anyway.

Our protocol is observably accountable and "verifiably deterministic" for A. Note that any probabilistic protocol for A can be transformed into a deterministic one by simply giving A a private key and asking that all its random choices are computed via a pseudorandom generator or pseudorandom function based on that key. To achieve observable accountability, A's randomness will be generated by a *verifiable* random function (VRF) keyed with A's private key. In our protocol, the other party (U) can verify that the pseudorandomness involved in A's crucial moves is computed correctly using this VRF.

Cryptographic Setup. Recall that the user U has a private/public VRF keys (k_U, pk_U) (see section 4.1), and message $s \in \mathbb{Z}_q$ to (probabilistically) send to A. We assume that commitment $C_s = g^s \bmod p$ to s was made public before the protocol starts. We amend the key generation procedure of the escrow scheme so that A generates a private/public key pair (k_A, pk_A) for the same VRF function. We assume that U knows A's public key pk_A. (However, recall that A does *not* know U's public key pk_U.) The Key Certification Authority which plays the role of the TTP picks a private/public key pair (sk_{TTP}, pk_{TTP}) of a verifiable encryption scheme of Camenish-Shoup [CS03], with the plaintext space including elements of $\mathbb{Z}_q$. We will use the CS encryption to allow U to prove to A that the plaintext s corresponding to an encrypted value $c_s = Enc_{PK_{TTP}}(s)$ satisfies an equation $g^s = C_s \bmod p$. Such proof takes only a few exponentiations and is non-interactive in the random oracle model (see [CS03] for more details). We assume that the required probability θ can be rounded up as $\theta = i/2^l$ for some integers l and $i \in [0, 2^l]$. We will assume a second hash function $H' : \{0,1\}^* \to \{0,1\}^l$, which we will also model as a random oracle.

Robust Probabilistic Information Transfer Protocol with Off-Line TTP:

1. U picks a random $r'_U \in \mathbb{Z}_p^*$, computes $c_U = Enc_{PK_{TTP}}(r'_U)$ and $c_s = Enc_{PK_{TTP}}(s)$, and sends (c_U, c_s) to A. U also sends a non-interactive zero-knowledge proof [CS03] that the plaintext s encrypted in ciphertext c_s satisfies relation $g^s = C_s \bmod p$.
2. After verifying the proof, A computes $r'_A = \mathsf{Eval}_{k_A}(c_U, aux)$ and sends $c_A = Enc_{PK_{TTP}}(r'_A)$ to U.
3. U sends back to A a MAC value $h = \mathsf{Eval}_s(aux, \theta, C_c, c_s, c_U, c_A)$ on the transcript so far using s as the MAC key, together with a zero-knowledge

proof that h is computed correctly under the key s committed to in $C_s = g^s \bmod p$. Note that if s is treated as a VRF key, then C_s is its corresponding verification key, and thus this is the same VRF verification as discussed in section 4.1. This communication round is the "commitment point" for U in the protocol.

4. If everything verifies, A opens c_A as an encryption of r'_A by sending r'_A to U together with the random coins used in this encryption. A also proves that r'_A is correctly computed as $r'_A = \mathsf{Eval}_{k_A}(c_U, aux)$.
5. If A's de-commitment and the proof are correct, U similarly opens to A his ciphertext c_U as an encryption of r'_U. U also computes $r = (r_U \oplus r_A)/2^l$ where $r_U = H'(r'_U)$ and $r_A = H'(r'_A)$. If $r < \theta$ then U also sends s to A.
6. If U's de-commitment is correct, A computes r the same way as $r = (r_U \oplus r_A)/2^l$ where $r_U = H'(r'_U)$ and $r_A = H'(r'_A)$. If $r < \theta$ and A doesn't get s from U, or $g^s \neq C_s \bmod p$, then A hands $(aux, \theta, C_s, c_s, c_U, c_A, h)$ to TTP, together with the proof that $r'_A = \mathsf{Eval}_{k_A}(c_U, aux)$.
7. TTP decrypts $s = Dec_{sk_{TTP}}(c_s)$, $r'_U = Dec_{sk_{TTP}}(c_U)$, $r'_A = Dec_{sk_{TTP}}(c_A)$, verifies A's proof that r'_A is computed as A's VRF on input (c_U, aux), checks if $h = \mathsf{Eval}_s(aux, \theta, C_s, c_s, c_U, c_A)$. If any verification fails, TTP sends $\perp$ back to A and stops. Otherwise, TTP recomputes $r_U = H'(r'_U)$, $r_A = H'(r'_A)$, $r = (r_A \oplus r_U)/2^l$. If $r < \theta$, then TTP sends (r, s) to A, else sends r.

Theorem 1. *The above protocol is a robust probabilistic information transfer protocol in the optimistic trusted third party model. This is a a contributory protocol which is also observably accountable, and has a verifiably deterministic coin contribution for A.*

We postpone the proof to the post-proceedings version of the paper.

Performance. We estimate our scheme's performance by counting the number of cryptographic operations that the user and the agency must execute in each session. Let C_e be the cost of a single full exponentiation modulo 1KBit modulus. In our setting, the cost of Camenisch-Shoup encryption is approximately $10.5C_e$, and the cost of the associated proof is approximately $13.5C_e$. Assuming that each multi-exponentiation costs between $1.15C_e$ and $1.3C_e$, we estimate that the user has to perform the equivalent of $52.3C_e$ in each protocol session, while the escrow agency's cost is $29C_e$ ($30C_e$ if a share is transferred).

6 Accuracy of Probabilistic Threshold Escrow

To estimate accuracy, we are interested in the probability $\alpha_{T,t}$ of *erroneous non-disclosure*, *i.e.*, that the total transacted amount exceeds threshold T, but the escrow agency has not accumulated enough shares to reconstruct the decryption key, and the probability $\beta_{T,t}$ of *erroneous disclosure*, *i.e.*, that the escrow agency accumulates enough shares to reconstruct the decryption key even though the transacted amount is still under the threshold.

Suppose the decryption key is split into d shares (d is a parameter of the system). We'll call $s = \frac{T}{d}$ *share size*. This is the amount "corresponding" to one share of the key. Suppose that the user transacts some total amount A, and, for simplicity, assume that all transactions are of equal size t. If $t < s$, then for each instance of the probabilistic escrow protocol, the probability of revealing a share is simply $\frac{t}{s}$. If $t = is + x$ where $i > 0$ and $x < \frac{T}{n}$, the escrow agency demands i shares straight away, and then engages in the probabilistic escrow protocol in which the probability of revealing an additional share is $\frac{x}{s}$.

W.l.o.g., assume that $t < s$. Let $n = \frac{A}{t}$ be the number of transactions performed. Since for each transaction the probability of obtaining a share $\frac{t}{s} = d\frac{t}{T}$, the probability of obtaining exactly d shares after n transactions is the binomial probability $\binom{n}{d}(d\frac{t}{T})^d(1 - d\frac{t}{T})^{n-d}$, where $\binom{n}{d}$ is the binomial coefficient $\frac{n!}{(n-d)!d!}$. The probability that the escrow agency obtains fewer than d shares is the "tail" of the binomial probability distribution $p_{nd} = \sum_{i=0}^{d-1} \binom{n}{i}(d\frac{t}{T})^i(1 - d\frac{t}{T})^{n-i}$. The probability of disclosure is $p_d = 1 - p_{nd}$. Unfortunately, for realistic applications the number of trials n is insufficiently large to approximate the binomial distribution with a normal or Poisson distribution. Therefore, we do not attempt to derive a closed formula approximating p_{nd}.

Probability of Error. Probability of error is equal to p_{nd} if the total transacted amount is greater than or equal to the threshold, and to p_d if the total amount is less than the threshold. In fig. 1, we set the disclosure threshold $T = \$10,000$,

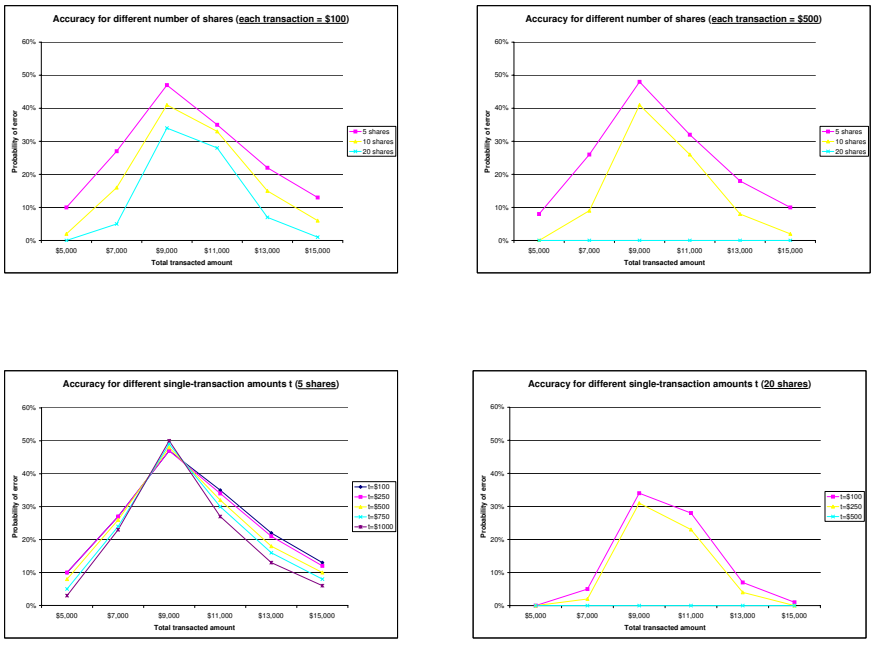

Fig. 1. Accuracy of probabilistic threshold disclosure

and calculate the probability of error as a function of the total transacted amount for different transaction sizes t and different number of shares d.

Figure 1 illustrates the basic efficiency-accuracy tradeoff of our probabilistic escrow scheme. For a larger number of shares, accuracy is better because (a) for any given transaction size t, both α and β functions (respectively, left and right sides of the "bell curve") become *steeper*, *i.e.*, the likelihood of erroneous disclosure or non-disclosure decreases sharply with the difference between the total transacted amount and the threshold, and (b) absolute probability of error decreases with the increase in the number of shares. The larger the number of shares, the less efficient the scheme is from the user's viewpoint, due to the difficulty of maintaining a large number of shares.

For a fixed number of shares and total transacted amount, lower single-transaction amounts are associated with higher probabilities of error, as demonstrated by fig. 1. Therefore, the best strategy for a malicious user who would like to transact an over-the-threshold amount without disclosure is to split the amount into lots of small transactions. Note, however, that the curve flattens as transaction sizes decrease. We conjecture that the marginal benefit to the cheating user from choosing ever smaller transactions is negligible. We also argue that for any minimum transaction size t, the spending pattern modeled in the tables (*i.e.*, total amount A is split into equal transactions, each of the minimum permissible size) is the worst-case scenario, and that for different transactions patterns probabilities of erroneous disclosure or non-disclosure will be better than those shown in the figures.

Future Directions. We are currently investigating an extension of the scheme which is *oblivious* to the user, *i.e.*, he does not learn if information transfer has been successful. The user won't be able to "game" the system by adjusting his behavior depending on the number of shares already accumulated by the agency.

References

[ASW00] N. Asokan, V. Schoup, and M. Waidner. Optimistic fair exchange of digital signatures. *IEEE Journal on Selected Areas in Communications*, 18:593–610, 2000.

[Blu82] M. Blum. Coin flipping by phone. *In Proc. 24th IEEE Computer Conference (CompCon)*, 15(1):93–118, 1982.

[Can00] Ran Canetti. Security and composition of multiparty cryptographic protocols. *Journal of Cryptology*, 13(1):143–202, 2000.

[CC00] C. Cachin and J. Camenisch. Optimistic fair secure computation. In *Proc. CRYPTO '00*, pages 93–111, 2000.

[CL01] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Proc. EUROCRYPT '01*, pages 93–118, 2001.

[CP92] D. Chaum and T. Pedersen. Wallet databases with observers. In *Proc. CRYPTO '92*, pages 89–105, 1992.

[CS03] J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In *Proc. CRYPTO '03*, pages 126–144, 2003.

[Fel87] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proc. FOCS '87*, pages 427–438, 1987.

[JO97] S. Jarecki and A. Odlyzko. An efficient micropayment scheme based on probabilistic polling. In *Proc. Financial Cryptography '97*, pages 173–192, 1997.

[JS04] S. Jarecki and V. Shmatikov. Handcuffing Big Brother: an abuse-resilient transaction escrow scheme. In *Proc. EUROCRYPT '04*, pages 590–608, 2004.

[NP98] M. Naor and B. Pinkas. Secure and efficient metering. In *Proc. EUROCRYPT '98*, 1998.

[Yao82] A. Yao. Protocols for secure computations. In *Proc. FOCS '82*, 1982.