Handcuffing Big Brother: An Abuse-Resilient Transaction Escrow Scheme

(extended abstract)

Stanislaw Jarecki[1] and Vitaly Shmatikov[2] *

[1] University of California, Irvine, CA**
stasio@ics.uci.edu
[2] SRI International, Menlo Park, CA
shmat@csl.sri.com

Abstract. We propose a practical abuse-resilient transaction escrow scheme with applications to privacy-preserving audit and monitoring of electronic transactions. Our scheme ensures correctness of escrows as long as at least one of the participating parties is honest, and it ensures privacy and anonymity of transactions even if the escrow agent is corrupt or malicious. The escrowed information is secret and anonymous, but the escrow agent can efficiently find transactions involving some user in response to a subpoena or a search warrant. Moreover, for applications such as abuse-resilient monitoring of unusually high levels of certain transactions, the escrow agent can identify escrows with particular common characteristics and automatically (*i.e.*, without a subpoena) open them once their number has reached a pre-specified threshold.

Our solution for transaction escrow is based on the use of Verifiable Random Functions. We show that by *tagging* the entries in the escrow database using VRFs indexed by users' private keys, we can protect users' anonymity while enabling efficient and, optionally, automatic de-escrow of these entries. We give a practical instantiation of a transaction escrow scheme utilizing a simple and efficient VRF family secure under the DDH assumption in the Random Oracle Model.

1 Introduction

Massive collection of personal and business data is increasingly seen as a necessary measure to detect and thwart crime, fraud, and terrorism. For example, all U.S. banks must report transactions over \$10,000. Regulations of the U.S. Securities and Exchange Commission effectively require financial firms to store all emails in case they are subpoenaed in some future investigation. Government authorities often demand that financial transactions, internal corporate communications, and so on be escrowed with law enforcement or regulatory agencies in

* Supported in part by ONR grants N00014-01-1-0837 and N00014-03-1-0961.

** Part of this work was done while visiting the Applied Cryptography Group at Stanford University.

such a way that the escrow agency can open the data pertaining to some user within the time period for which a subpoena or search warrant has been issued, or mine the collected data without a warrant for evidence of suspicious activity.

Existing techniques. Information stored in the escrow agency's database must be protected both from abuse by the escrow agency's employees and from external attacks. Unfortunately, existing escrow schemes sacrifice either user privacy, or efficiency of the escrow operation. Moreover, existing techniques allow mining of the escrowed data for evidence of suspicious activity only by letting the escrow agency de-escrow any entry at will.

Key escrow techniques [Mic92,KL95] implicitly assume that escrowed data are tagged by the key owner's identity or address. This enables efficient de-escrow of a subset of records pertaining to some user (*e.g.*, in response to a subpoena), but fails to protect anonymity of records against malicious employees of the escrow agency who can learn the number and timing of transactions performed by a given person, find correlations between transactions of different people, and so on. On the other hand, if escrows are *not* tagged, then there is no efficient procedure for opening the relevant escrows in response to a subpoena. Each entry in the escrow database must be decrypted to determine whether it involves the subpoenaed user. This is prohibitively inefficient, especially if the decryption key of the escrow agency is shared, as it should be, among a group of trustees.

Our contribution. We propose a *verifiable transaction escrow* (VTE) scheme which offers strong privacy protection *and* enables efficient operation of the escrow agent. Our scheme furnishes transaction participants with a provably secure privacy guarantee which we call *category-preserving anonymity*. We say that two transactions belong to the same category if and only if they were performed by the same user and are of the same *type* (*e.g.*, both are money transfers). An escrow scheme is *category-preserving anonymous* if the only information about any two transactions that the (malicious) escrow agent can learn from the corresponding escrow entries is whether the transactions fall into the same category or not. The agent cannot learn *which* category either transaction belongs to.

Of course, a malicious participant may reveal the transaction to the escrow agent. However, regardless of the user's transactions with dishonest parties who leak information to the escrow agent, all of his transactions with honest parties remain private in the sense of category-preserving anonymity — even if they belong to the same category as compromised transactions. While it does not provide perfect anonymity, category-preserving anonymity seems to give out no useful information, especially if transaction volume is high. (If volume is low, there may be undesirable information leaks, *e.g.*, the escrow agent may observe that only one category is ever used, and deduce that only one user is active.)

We present a VTE scheme with two variants. The first variant has an inexpensive escrow protocol, but does not achieve full category-preserving anonymity. The privacy guarantees it does offer might be acceptable in practice, however. The second variant achieves category-preserving anonymity at the cost of adding an expensive cut-and-choose zero-knowledge proof to the escrow protocol.

Our VTE scheme supports both (1) efficient identification and opening of escrows in response to a subpoena, and (2) efficient *automatic* opening of escrows that fall into the same category once their number reaches some pre-specified threshold. The scheme is also *tamper-resistant* in the sense that a malicious escrow agent cannot add any valid-looking escrows to the database. Finally, our scheme ensures correctness of the escrow entry as long as at least one participant in the escrowed transaction is honest. Note that there is no way to ensure escrow of transactions between parties who cooperate in concealing the transaction.

Our scheme employs Verifiable Random Functions. We show that by *tagging* entries in the escrow database using VRFs indexed by users' private keys, we enable efficient and, if necessary, automatic *de-escrow* (*disclosure*) of these entries, while providing category-preserving anonymity for the users. We instantiate our scheme with a practical construction based on a simple and efficient (shareable) VRF family secure under the DDH assumption in the Random Oracle Model.

Applications. A VTE scheme can be used in any scenario where transaction data must be escrowed but should remain private and anonymous. For example, a financial regulatory agency may collect escrows of all money transfers to ensure availability of evidence for future investigations of money laundering. Unless a court warrant is obtained, the agency should not be able to extract any useful information from the escrows, not even participants' identities. At the same time, the automatic opening capability of our VTE scheme can also support a scenario where the agency needs to identify all transfers which are made from the same account and share the same type, *e.g.*, all involve a certain organization or country, or more than a certain amount. These transactions should be secret and anonymous until their number reaches a pre-specified threshold, in which case the authority gains the ability to extract all corresponding plaintexts.

Related work. The problem of efficient classification and opening of escrows is related to the problem of search on encrypted data [SWP00,BCOP03]. In the latter problem, however, there is no notion of a malicious user who submits incorrect ciphertexts or interferes with record retrieval. Moreover, their techniques require the user to generate search-specific trapdoors, while we are also interested in scenarios where the escrow agent is able to open all escrows in a given category not because he received some category-specific trapdoor but because the *number* of escrows within a category reached a pre-specific threshold.

Paper organization. In section 2, we define verifiable transaction escrow and describe its security properties. In section 3, we present the simpler variant of our VTE construction, which is practical but does not achieve full category-preserving anonymity. In section 4, we present another variant which does achieve category-preserving anonymity, but employs an expensive cut-and-choose zero-knowledge protocol. In section 5, we show how to extend either construction to support automatic de-escrow capability. For lack of space, we omit all proofs from these proceedings. The full version of the paper, including all proofs, will be made available on `eprint` [JS04].

2 Definition of a Verifiable Transaction Escrow Scheme

A *Verifiable Transaction Escrow* (VTE) system involves an escrow *Agent* and any number of users. We assume that each transaction occurs between a *User* and a *Counterparty*. The two roles are naturally symmetric (users may act as counterparties for each other), but in some applications the escrow agent may only be interested in monitoring users (*e.g.*, bank clients), but not the counterparties (banks).

We assume that each transaction is adequately described by some bitstring m, and that there is a public and easily computable function $Type$, where $Type(m)$ of transaction m is application-specific, *e.g.*, "this transaction is a money transfer," or "this transaction is a money transfer between \$1,000 and \$10,000." The *category* of a transaction is the $\langle$user identity,type$\rangle$ pair.

2.1 Basic properties of a Verifiable Transaction Escrow scheme

A VTE scheme is a tuple $(AKG, UKG, U_1, A, U_2, C, U_3, J)$ of the following probabilistic polynomial-time (PPT) algorithms:

• AKG and UKG are *key generation* algorithms, which on input of a security parameter τ generate, respectively, Agent's key pair (k_A, pk_A) and, for each User, key pair (k_U, pk_U).

• (U_1, A) are interactive algorithms which define an *escrow protocol*. Its aim is to add an escrow of a transaction to the Agent's database in exchange for a *receipt* which will be later verified by the transaction Counterparty. The protocol runs between User (U_1) and Agent (A), on public input of Agent's public key pk_A. User's private input is (k_U, m), where m is the transaction description. Agent's private input is (k_A, D) where D is the state of Agent's escrow database. User's output is a receipt *rcpt*, and Agent's output is an *escrow* item e, which defines a new state of Agent's database as $D' = D \cup \{e\}$.

• (U_2, C) are interactive algorithms which define a *verification protocol*. Its aim is for the Counterparty to verify the receipt certifying that the transaction was properly escrowed with the Agent. The protocol runs between User (U_2) and Counterparty (C), on public input (pk_U, m, pk_A). User's private input is $k_U, rcpt$. Counterparty outputs decision $d = \mathsf{accept}/\mathsf{reject}$.

• (U_3, J) is a pair of interactive algorithms which defines a *subpoena protocol*. Its aim is to identify all transactions of a given type in which the user participated, and *only* those transactions. The protocol runs between User (U_3) and a public *Judge* (J), on public inputs (pk_U, T, D), where pk_U, T identify the $\langle$user,type$\rangle$ category to be subpoenaed, and D is Agent's database. User's private input is k_U. Judge has no private inputs. Algorithm J outputs M, which is either a symbol $\mathsf{contempt}$ if the User refuses to cooperate, or a (possibly empty) list $(m_1, m_2, ...)$ of transactions of type T involving user pk_U.

Completeness. If parties follow the protocol, then every escrowed transaction can be de-escrowed in the subpoena. In other words, for all keys (k_A, pk_A) and

(k_U, pk_U) generated by AKG and UKG, and for every m, D, D', if $\langle U_1(k_U, m), A(k_A, D)\rangle(pk_A)$ outputs $(rcpt, e)$ then $\langle U_2(k_U, rcpt), C\rangle(pk_U, m, pk_A)$ outputs $d = \mathsf{accept}$ and $\langle U_3(k_U), J\rangle(pk_U, Type(m), D' \cup \{e\})$ outputs M s.t. $m \in M$.

For notational convenience, we define predicate $Prop(e, m, pk_U)$ to be *true* if and only if $\langle U_3(k_U), J\rangle(pk_U, Type(m), D' \cup \{e\})$ outputs M s.t. $m \in M$.

Verifiability. The escrow agent receives a correct escrow of the transaction as long as at least one party in the transaction is honest. In particular, a malicious User has only negligible probability[3] of getting an honest Counterparty to *accept* in an escrow protocol unless the User gives to the Agent a proper escrow. Formally, for every PPT algorithms U_1^*, U_2^*, for every D, m,

$$\begin{aligned}\Pr[\ Prop(e, m, pk_U) \ \mid\ (k_A, pk_A) \leftarrow AKG(1^\tau);\ (k_U, pk_U) \leftarrow UKG(1^\tau);\\ (rcpt^*, e) \leftarrow \langle U_1^*(k_U, m), A(k_A, D)\rangle(pk_A);\\ \mathsf{accept} \leftarrow \langle U_2^*(rcpt^*), C\rangle(pk_U, m, pk_A)\ ] \ \geq\ 1 - \mathsf{negl}(\tau)\end{aligned}$$

Efficient and unavoidable subpoena. The subpoena procedure is *unavoidable* in the sense that the user is either publicly identified as refusing to cooperate, or all entries in the escrow database which involve the user and the specified type are publicly revealed. Namely, for every PPT algorithm U_3^*, for every D', m, e, for $T = Type(m)$,

$$\begin{aligned}\Pr[M = \mathsf{contempt} \vee m \in M \mid (k_A, pk_A) \leftarrow AKG(1^\tau);\ (k_U, pk_U) \leftarrow UKG(1^\tau);\\ M \leftarrow \langle U_3^*(k_U), J\rangle(pk_U, T, D' \cup \{e\});\ Prop(e, m, pk_U)] \geq 1 - \mathsf{negl}(\tau)\end{aligned}$$

Moreover, the subpoena protocol is *efficient* in the sense that its running time is linear in the number of escrows of the subpoenaed $\langle$user,type$\rangle$ category in the database D, rather than in the size of the whole escrow database D.

Tamper resistance. A malicious Agent can't add entries to the escrow database which would be identified as transactions involving some user during the public subpoena process, unless that user created these escrows himself. Namely, for every PPT algorithm A^*, for random keys k_U, pk_U generated by UKG, if A^* has access to user oracles $O_{U_1}(\cdot,\cdot)$, $O_{U_2}(\cdot,\cdot,\cdot)$, and $O_{U_3}(\cdot,\cdot)$, where $O_{U_1}(m, pk_A)$ follows the U_1 protocol on (k_U, m) and pk_A, $O_{U_2}(m, rcpt, pk_A)$ follows the U_2 protocol on $(k_U, m, rcpt)$ and pk_A, and $O_{U_3}(T, D)$ follows the U_3 protocol on k_U and (pk_U, T, D), then there is only negligible probability that $A^{*O_{U_1,U_2,U_3}(\cdot,\cdot,\cdot)}(pk_U)$ produces T^*, D^* s.t. $M \leftarrow \langle U_3(k_U), J\rangle(pk_U, T^*, D^*)$ where M contains some message m^* s.t. A^* did *not* run oracle $O_{U_1}(\cdot,\cdot)$ on m^* and some pk_A.

Category-preserving anonymity. By default, the only information learned by a malicious Agent about any two instances of the escrow protocol is whether the two transactions fall into the same *category*, *i.e.*, correspond to the same

[3] We say that a function $f(\tau)$ is *negligible* if for any polynomial $p(\cdot)$, there exists τ_0 s.t. for every $\tau \geq \tau_0$, $f(\tau) < 1/p(\tau)$. We denote a negligible function by $\mathsf{negl}(\cdot)$.

⟨user,type⟩ pair or not. Moreover, neither the transactions opened in the subpoena protocol, nor transactions reported to the Agent by some malicious Counterparties, should help the malicious Agent to crack the privacy of transactions done with honest Counterparties and which were not subpoenaed.

Formally, consider the following game between any PPT algorithms A^*, C^* and the VTE system. First, polynomially many user keys $\{(k_i, pk_i)\}$ are generated by the UKG algorithm. Then, if A^* has access to *flexible* user oracles $O_{U_1}(\cdot,\cdot,\cdot)$, $O_{U_2}(\cdot,\cdot,\cdot,\cdot)$, and $O_{U_3}(\cdot,\cdot,\cdot)$, where $O_{U_1}(i,m,pk_A)$ follows the U_1 protocol on (k_i,m) and pk_A, $O_{U_2}(i,m,rcpt,pk_A)$ follows the U_2 protocol on $(k_i,rcpt)$ and (pk_i,m,pk_A), and $O_{U_3}(i,T,D)$ follows the U_3 protocol on k_i and (pk_i,T,D), the following holds:

$$\begin{aligned}
\Pr[\ b = b' \mid (i_0,i_1,m_0,m_1,st,pk_A) &\leftarrow A^{*O_{U_1,U_2,U_3}(\cdot,\cdot,\cdot,\cdot)}(pk_1,...,pk_{p(\tau)});\\
b \leftarrow \{0,1\};\ (rcpt_b, st') &\leftarrow \langle U_1(k_{i_b}, m_b), A^*(st)\rangle(pk_A);\\
\overline{b} = \neg b;\ (rcpt_{\overline{b}}, st'') &\leftarrow \langle U_1(k_{i_{\overline{b}}}, m_{\overline{b}}), A^*(st')\rangle(pk_A);\\
(st''') &\leftarrow \langle U_2(k_{i_0}, rcpt_0), C^*(st'')\rangle(pk_{i_0}, m_0, pk_A);\\
(st'''') &\leftarrow \langle U_2(k_{i_1}, rcpt_1), C^*(st''')\rangle(pk_{i_1}, m_1, pk_A);\\
b' &\leftarrow A^{*O_{U_1,U_2,U_3}(\cdot,\cdot,\cdot,\cdot)}(st'''');\ ] \ \le\ \frac{1}{2} + \mathsf{negl}(\tau)
\end{aligned}$$

where the *test transactions* (i_0,m_0) and (i_1,m_1) and the queries of A^* to O_{U_1} and O_{U_3} oracles are restricted as follows:
(1) The test transactions are not subpoenaed, *i.e.*, O_{U_3} is not queried on either $(i_0, Type(m_0))$ or $(i_1, Type(m_1))$.
(2) If any of the ⟨user,type⟩ pairs involved in the test transactions are seen by the Agent in some query to O_{U_1} or O_{U_3}, then the two test transactions must have the same ⟨user,type⟩ pairs, *i.e.*, if for any $\beta = 0,1$, either O_{U_3} was queried on $(i_\beta, Type(m_\beta))$ or O_{U_1} was queried on (i_β, m'_β) s.t. $Type(m'_\beta) = Type(m_\beta)$, then $i_0 = i_1$ and $Type(m_0) = Type(m_1)$.

2.2 Additional desirable properties of a VTE scheme

Automatic threshold disclosure. A VTE scheme may support automatic opening of escrows involving transactions with the same ⟨user,type⟩ once their number reaches some threshold value, pre-set for transactions of this type. We show an example of such extension in Section 5.

Key management. In practice, a VTE scheme requires a *Key Certification Authority* serving as strong PKI. If a user's key is lost or compromised, the CA must not only revoke that key and certify a new one, but also reconstruct the old key to facilitate the subpoena of transactions which were escrowed under it. To avoid a single point of failure, the CA should implement this *key escrow* functionality via a group of trustees using standard threshold techniques. We stress that although majority of the CA trustees must be trusted, this is not a severe limitation of the proposed scheme because CA is invoked only when a new user enrolls in the system, or when the key of some user is subpoenaed and he refuses to cooperate. Moreover, the *secret keys* of the CA trustees need only be used during reconstruction of some user's key in the case of key loss and/or user's refusal to cooperate with a subpoena, both of which should be relatively infrequent events. Interestingly, while PKI is often viewed as a threat to privacy, in our scheme it actually *helps* privacy. Without PKI, escrow can only be implemented via a public-key scheme that cannot guarantee both user anonymity *and* efficient operation of the escrow scheme.

3 Basic Construction of a VTE Scheme

We present the simpler variant of our VTE scheme. As we explain in section 3.1, this scheme does not achieve full category-preserving anonymity, but its privacy protection can be good enough in practice. In section 4, we show a variant of the same VTE scheme which does achieve full category-preserving anonymity. Both variants use cryptographic primitives of *verifiable anonymous encryption*, *verifiable anonymous tagging*, and *anonymous signatures*, which we define and implement in section 3.2. In section 3.3, we discuss key management issues.

VTE construction overview. In our VTE construction, an escrow consists of (1) an encryption of the transaction plaintext, (2) a signature, and (3) a deterministically computed *tag* which is an output of a pseudorandom function indexed by the user's private key and applied to the *type* of the transaction. The tags enable the Agent to group entries in the escrow database into "bins" corresponding to tag values. Because a pseudorandom function assigns outputs to inputs deterministically, escrows corresponding to the same $\langle$user,type$\rangle$ category are always placed in the same bin, enabling efficient identification of the escrowed entries of a given category during the subpoena. However, the pseudorandomness helps to ensure that the tags reveal no more information than permitted by *category-preserving anonymity*, *i.e.*, the only information learned by the escrow agent about any two escrows is whether they belong to the same category.

The signature is included to disable Agent's tampering with the escrowed entries. The encryption and the tag must preserve *secrecy* of the transaction plaintext against chosen-plaintext attack, because a malicious Agent can cause a user to participate in transactions of Agent's choice and see the corresponding escrow entries (see the definition of category-preserving anonymity). The whole escrow must also protect user's *key privacy* against the same chosen-plaintext attack. To enable verification that an escrow is correctly formed, both the tag, the ciphertext, and the signature must be *verifiable* by the transaction counterparty, *i.e.*, given the transaction plaintext and the user's public key.

Initialization: Every user is initialized with a public/private key pair implemented as in section 3.2. The escrow agent is initialized with a key pair of any CMA-secure signature scheme.

Escrow protocol: We assume that before the escrow protocol starts, the user and the counterparty agree on transaction description m of type $T = Type(m)$.

1. The user sends to the escrow agent an *escrow* $e = (c, t, s)$ s.t.:
   (a) $c = \mathsf{Enc}_k\{m\}$ is a *verifiable anonymous symmetric encryption* of m.
   (b) $t = \mathsf{Tag}_k\{T\}$ is an output of a *verifiable anonymous tagging function*.
   (c) $s = \mathsf{sig}_k\{c, t\}$ is an *anonymous signature* on the (ciphertext,tag) pair.
2. The agent places escrow e in the escrow database in the bin indexed by the tag t, and sends his signature *rcpt* on e to the user.

Verification protocol:

1. The user forwards the escrow e and the agent's signature *rcpt* to the counterparty, together with a proof that:
   (a) c is a ciphertext of m under a key k corresponding to the public key pk.
   (b) t is a tag computed on type T under key k corresponding to pk.
   (c) s is an anonymous signature computed on (c, t) under the public key pk.
2. The transaction counterparty accepts if he verifies the agent's signature on e and the correctness of the above three proofs.

Subpoena protocol: The protocol proceeds on a public input of any subset D of the escrow database, the type T of the subpoenaed transactions, and the identity pk of the subpoenaed user:

1. The user computes tag $t = \mathsf{Tag}_k\{T\}$ and proves its correctness under pk.
2. Entries $(e_1, e_2, ...)$ in D which are indexed by tag t are publicly identified, and for each $e_i = (c_i, t, s_i)$, the user verifies the signature s_i on (c_i, t).
   (a) If the signature does not match, the user *provably denies* that the signature is valid under pk, and if the proof is correct the entry is skipped.
   (b) If the signature matches, the user publishes the transaction plaintext m_i by decrypting the ciphertext c_i under k, and proving correctness of the decryption under key k corresponding to pk.
3. If the user cooperates, the output includes all (and only) transactions of the subpoenaed type for that user. If any of the above proofs fails, the public output is the special symbol $\mathsf{contempt}$.

From the properties of the cryptographic primitives used in this VTE construction, the following theorem follows:

Theorem 1. *The basic VTE scheme satisfies (1) verifiability, (2) efficient and unavoidable subpoena, and (3) tamper resistance.*

3.1 Privacy leakage of the basic VTE scheme

In the above scheme, the user presents the (ciphertext, tag, signature) tuple to both the agent and the counterparty. This allows a malicious counterparty and a malicious agent to *link* their views of the escrow and verification protocols, and since the counterparty knows the user identity and the message plaintext, a malicious agent can learn an association between a tag and a $\langle$user,type$\rangle$ pair. This would violate category-preserving anonymity, because with this knowledge the escrow agent can learn the type and user identity of *all* transactions with the same tag, even those conducted with other, *honest* counterparties.

In practice, privacy protection can be increased by allowing the *type* of the transaction to range over some small set, for example of a hundred constants. If the index of the constant used for a given transaction is chosen by hashing the counterparty's identity, then there is only 1% chance that a dishonest counterparty can endanger the anonymity of transactions of the same type with any other honest counterparty. On the other hand, when a user is subpoenaed on a given type, he has to identify a hundred categories instead of one. Such privacy/efficiency trade-off may be acceptable in some applications.

3.2 Definitions and constructions for cryptographic primitives

Let p, q be large primes s.t. $p = 2q + 1$, and let g be a generator of $\mathbb{Z}_p^*$. The security of our constructions relies on the hardness of the Decisional Diffie-Hellman (DDH) problem in subgroup QR_p of quadratic residues in $\mathbb{Z}_p^*$, which says that tuples (h, h^a, h^b, h^{ab}) are indistinguishable from tuples (h, h^a, h^b, h^c) for $h \in QR_p$ and random a, b, c in $\mathbb{Z}_q$ (see, *e.g.*, [Bon98]). Our security arguments follow the so-called "Random Oracle Model" methodology of [BR93]. Namely, we assume an "ideal hash function" $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ which can be treated as a random function in the context of our constructions.

Verifiable random functions. A VRF family [MRV99] is defined by three algorithms: a key generation algorithm KGen outputing private key k and public key pk, an evaluation algorithm $\mathsf{Eval}(k, x) = (y, \pi)$ which on input x outputs the value of the function $y = f_k(x)$ and a proof π that the value is computed correctly, and a verification algorithm Ver which can verify π on inputs (pk, x, y, π). The VRF is secure if it is infeasible to distinguish an interaction with function f_k, for a randomly chosen key k, from an interaction with a purely random function which outputs uniformly distributed values in the same range. Moreover, the VRF needs to be verifiable, in the sense that any proof will be rejected unless the returned value y is indeed $f_k(x)$. The VRF concept and constructions were originally proposed for the standard model [MRV99,Lys02,Dod03], *i.e.*, without assuming ideal hash functions, but evaluation/verification cost for these constructions involves $\Omega(\tau)$ cryptographic operations. In contrast, in the Random Oracle Model, a simple VRF family can be constructed based on the DDH assumption, with evaluation and verification cost of 1-3 exponentiations. Similar or identical constructions were used before [CP92,NPR99,CKS00], without explicitly noting that the result is a VRF family.

We relax (slightly) the standard definition of VRF [MRV99] by replacing the uniqueness requirement with a computational soundness requirement.

Definition 1. *A VRF family (for a group family $\{G_i\}_{i=1,2,\ldots}$) is given by a tuple of polynomial-time algorithms (*KGen, Eval, Ver*) where* $\mathsf{KGen}(1^\tau)$ *outputs a pair of keys* (k, pk)*,* Eval *is a deterministic algorithm which, on any x, outputs* $(y, \pi) \leftarrow \mathsf{Eval}(k, x)$ *s.t.* $y \in G_n$*, and* $\mathsf{Ver}(pk, x, y, \pi)$ *outputs* 0 *or* 1*, which satisfy the following requirements:*

1. Completeness: *For every τ and x, if* $(k, pk) \leftarrow \mathsf{KGen}(1^\tau)$ *and* $(y, \pi) = \mathsf{Eval}(k, x)$ *then* $\mathsf{Ver}(pk, x, y, \pi) = 1$.

2. Soundness: *For any probabilistic polynomial-time algorithm A, for any values pk and x, the following probability is negligible:*

$$\Pr[\mathsf{Ver}(pk,x,y,\pi) = \mathsf{Ver}(pk,x,y',\pi') = 1 \wedge\ y \neq y' \mid (y,y',\pi,\pi') \leftarrow A(pk,x)]$$

3. Pseudorandomness: *For all probabilistic polynomial-time algorithms A_1, A_2,*

$$\begin{aligned}\Pr[\ \ b = b' \mid (k,pk) \leftarrow \mathsf{KGen}(1^\tau); (x,st) \leftarrow A_1^{O_{\mathsf{Eval}}(k,\cdot)}(pk); y_0 \leftarrow \mathsf{Eval}(k,x);\\ y_1 \leftarrow G_n; b \leftarrow \{0,1\}; b' \leftarrow A_2^{O_{\mathsf{Eval}}(k,\cdot)}(st,y_b)\ \ ] \ \ \leq \ \ \frac{1}{2} \ + \ \mathsf{negl}(\tau)\end{aligned}$$

where A_1 and A_2 are restricted from querying oracle $O_{\mathsf{Eval}}(k,\cdot)$ on the challenge input x chosen by A_1.

Construction: Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$ be an ideal hash function (modeled as a random oracle). Formally, the key generation picks a triple (p,q,g) as above s.t. the hardness of the DDH problem in QR_p is good enough for the security parameter. For ease of discussion, we treat (p,q,g) as chosen once and for all. We will construct a VRF function family indexed by such triples, whose range is the group of quadratic residues QR_p. The key generation algorithm picks a secret key $k \in \mathbb{Z}_q^*$ and the public key $pk = g^{2k} \bmod p$. The evaluation algorithm $\mathsf{Eval}(k,x)$ returns $y = h^{2k} \bmod p$ where $h = H(x)$, and a non-interactive zero-knowledge proof π of equality of discrete logarithm $x = \mathsf{DL}_h(y) = \mathsf{DL}_g(pk)$. This is a standard ZKPK proof of discrete-log equality which can be made non-interactive in the ROM model, *e.g.*, [CS97].

Theorem 2. *Algorithms* $(\mathsf{KGen}, \mathsf{Eval}, \mathsf{Ver})$ *define a Verifiable Random Function family, under the DDH assumption in the Random Oracle Model.*

Verifiable anonymous tagging function. We define a verifiable anonymous *tagging* function simply as a VRF, and we implement it as $\mathsf{Tag}_k\{x\} = f_k(x)$. It is easy to see that tags $\mathsf{Tag}_k\{T\}$ give no information about the category they represent, *i.e.*, user's identity pk and the transaction type T, except that, whatever category this is, it is identified with tag $\mathsf{Tag}_k\{T\}$. It is also easy to see that a VRF has good enough collision-resistance so that escrows of two categories go to different bins. In fact, a much stronger property holds:

Theorem 3. *Under the discrete log assumption, in the Random Oracle Model, the VRF family* $(\mathsf{KGen}, \mathsf{Eval}, \mathsf{Ver})$ *has a strong collision resistance property in the sense that it is infeasible to find pair $(k,x) \neq (k',x')$ s.t.* $\mathsf{Eval}(k,x) = \mathsf{Eval}(k',x')$.

Verifiable anonymous symmetric encryption. For escrows to be anonymous, the symmetric encryption Enc used by the user must be not only chosen-plaintext secure, but also *key-hiding*. Following [Fis99,BBDP01], we combine these in one definition that implies several natural anonymity properties. Even an adversary who decides who encrypts what, cannot tell, for ciphertexts created outside of his control, whether the messages and keys satisfy any non-trivial relation this adversary is interested in. For example, the adversary cannot tell if a ciphertext is an encryption under any given key, if two ciphertexts are encryptions under the same key, if two ciphertexts encrypt related messages, *etc.*

Let ($\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}$) be a symmetric encryption scheme. In our experiment, first the key generation algorithm is executed $p(\tau)$ times where $p(\cdot)$ is some polynomial and τ is the security parameter. Denote the keys as k_i, for $i \in \{1, p(\tau)\}$. Adversary can query the following *flexible* encryption oracle $O_{\mathsf{Enc}}(\cdot,\cdot)$: on input (i, m), $i \in \{1, p(\tau)\}$ and $m \in \{0,1\}^*$, $O_{\mathsf{Enc}}(i, m)$ outputs $\mathsf{Enc}(k_i, m)$.

Definition 2. *We say that a symmetric encryption scheme* ($\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}$) *is (chosen-plaintext-secure)* anonymous *if, for any polynomial* $p(\cdot)$ *and probabilistic polynomial-time adversary* A_1, A_2,

$$\Pr[\ b = b' \mid (k_1, ..., k_{p(\tau)}) \leftarrow (\mathsf{KGen}(1^\tau))^{p(\tau)}; (i_0, i_1, m_0, m_1, st) \leftarrow A_1^{O_{\mathsf{Enc}}(\cdot,\cdot)}(1^\tau);$$
$$b \leftarrow \{0,1\}; c \leftarrow \mathsf{Enc}(k_b, m_b); b' \leftarrow A_2^{O_{\mathsf{Enc}}(\cdot,\cdot)}(st, c)] \quad \leq \quad \frac{1}{2} \ + \ \mathsf{negl}(\tau)$$

We also extend the notion of (CPA-secure and anonymous) symmetric encryption by a *verifiability* property. We stress that this property is different from what is referred to as verifiable encryption in the context of *asymmetric* encryption schemes [ASW98,CD00]. We require that the secret key k of an anonymous encryption be generated together with a *commitment* to this secret key, which we will call a *public key* pk. This public key, however, is used not to encrypt but to enable efficient verification that a given ciphertext is a correct encryption of a given plaintext. In fact, our verifiability property for symmetric encryption is very similar to the verifiability property of VRFs. Namely, we require that the encryption procedure Enc is augmented so that along with output $c = \mathsf{Enc}_k\{m\}$ it produces a *proof* π of correct encryption evaluation. We also require an efficient procedure Ver which takes as inputs message m, ciphertext c, and a proof π. The algorithms ($\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Ver}$) must then satisfy an obvious *completeness* property, *i.e.*, that a correctly computed proof always verifies, and a *soundness* property, which says that it is intractable, for any (k, pk), to find a tuple (m, m', c, π, π') s.t. $m \neq m'$ but $\mathsf{Ver}(pk, m, c, \pi) = \mathsf{Ver}(pk, m', c, \pi') = 1$.

Construction: Instead of using our VRF family to encrypt directly, we replace the hash function in our VRF construction with a Feistel-like padding scheme $pad^H(m|r)$ similar to the OAEP padding [BR94,Sho01]. Assume message length is $|m| = \tau_1 = |p| - 2\tau - 2$ where τ is the security parameter. We define our padding scheme as $\mathsf{pad}^H(m|r) = (h_1|h_2)$ for $h_1 = H_1(r) \oplus m$ and $h2 = H_2(h_1) \oplus r$, where hash functions H_1, H_2 output bit strings of length τ_1 and 2τ, respectively, and r is a random string of length 2τ. Note that $(m|r)$ can be recovered from $(h_1|h_2)$. This padding is simpler than the OAEP padding and its variants because our (symmetric, anonymous) encryption needs only chosen plaintext security rather than chosen ciphertext security.

Using such padding we can encrypt as follows. KGen is the same as in the VRF scheme. $\mathsf{Enc}_k(m) = o^{2k} \bmod p$ where $o = \mathsf{pad}^H(m|r)$ is treated as an element in $\mathbb{Z}_p^*$. The decryption $\mathsf{Dec}_k(c)$ computes candidates o' and $-o' \bmod p$ for o, where $o' = c^{k'} \bmod p$, and $k' = \alpha * k^{-1} \bmod q$ where $\alpha = (q+1)/2$ (in integers). To decrypt we take as o either o' or $-o' \bmod p$, depending on which one is smaller than $2^{|p|-2}$. We then recover $m|r$ by inverting the padding scheme pad^H on o. The proof of correct encryption consists of the randomness r and a proof π of discrete-log equality $\mathsf{DL}_o(c) = \mathsf{DL}_g(pk)$.

Theorem 4. *The above scheme is a verifiable anonymous symmetric encryption scheme secure under the DDH assumption in ROM.*

Anonymous signatures. An *anonymous signature* is an undeniable signature scheme [CP92] with an additional property of *key-privacy*. Recall that an undeniable signature scheme requires that the recipient of a signature s produced under public key pk on message m cannot prove to a third party that this is a valid signature under pk. Instead, the third party must ask U to verify the signature validity or invalidity via an interactive proof protocol. Here we additionally require *key privacy* in the sense corresponding to the CPA-security of the anonymous symmetric encryption, *i.e.*, that it is infeasible to tell from a (message,signature) pair what public key was used in computing it.
Construction: Any VRF family immediately yields an anonymous signature scheme. In fact, the undeniable signature construction of [CP92] already has the required properties, because it is implicitly constructed from the same VRF construction as here. For better concrete security, we slightly modify the [CP92] construction. The signature on m is a pair $s = (r, \tilde{s})$ where r is a random string of length 2τ, and $\tilde{s} = f_k(m|r) = H(m|r)^{2k} \bmod p$. The proof of (in)correctness of a signature under public key pk is a zero-knowledge proof of (in)equality of discrete logarithm (*e.g.*, [CS03]) between tuples (g, pk) and $(H(m|r), \tilde{s})$.

3.3 Key management for discrete-log based VTE schemes

The discrete-log based keys used in our scheme can be efficiently secret-shared by the user with the CA trustees using Feldman's verifiable secret sharing (see, *e.g.*, [GJKR99] for an exposition). Using recent techniques of [CS03], the user can deliver a secret-share to each trustee encrypted under the trustee's public key, and the trustee can verify the share's correctness without the use of the trustee's private key. The resulting shares can then be efficiently used by the trustees in the subpoena process. For example, if the user refuses to cooperate, the CA trustees can efficiently compute the tag $t = (H(Type))^{2k} \bmod p$ for the subpoenaed user and type via threshold exponentiation protocol such as [GJKR99]. The trustees can also use the same protocol to verify signatures on and decrypt the escrows.

4 VTE Scheme with Unlinkable Receipts

As explained in section 3.1, category-preserving anonymity is hard to achieve unless the escrow agent and the transaction counterparty are somehow prevented from linking their views of the escrow and the verification protocols. We show how to achieve such separation of agent's and counterparty's views by replacing the standard signature scheme used by our basic VTE scheme with the *CL signature* scheme of [CL01,CL02], which enables the user to prove his possession of the agent's receipt to the counterparty in zero-knowledge. To integrate CL signatures into our VTE scheme, in section 4.1 we introduce a novel zero-knowledge proof of knowledge of *committed key and plaintext* (CKP-ZKPK).

Diophantine commitments. To use the CL signature scheme, we need a commitment scheme of [FO98,DF01] which allows a commitment to *integers* rather than to elements in a finite field. Consider a special RSA modulus $n = p'q'$, where $p', q', (p'-1)/2, (q'-1)/2$ are all prime and $|p'|, |q'|$ are polynomial in the security parameter τ. Consider also a random element b of group QR_n of quadratic residues modulo n, and a random element a of the subgroup generated by b in $\mathbb{Z}_n^*$. The commitment to an *integer* value m is $C = a^m b^{m'} \bmod n$ where m' is chosen uniformly in $\mathbb{Z}_n$. This commitment scheme is statistically hiding, and it is binding if strong RSA assumption holds for n [FO98,DF01].

CL signatures. The public key in CL signature consists of a special RSA modulus n as above, and three uniformly chosen elements a, b, d in QR_n. Let l_m be a parameter upper-bounding the length of messages that need to be signed. The public key is (n, a, b, d). The signature on m is a triple (v, e, s) where $v^e = a^m b^s d \bmod n$ and $2^{l_e} > e > 2^{l_e+1}$ where $l_e \geq l_m + 2$. This signature scheme is CMA-secure under the strong RSA assumption [CL02].

The CL signature comes with two protocols: (1) the *CL signing protocol*, in which the signer can issue signature (v, e, s) on $m \in \{0,1\}^{l_m}$ given only a commitment C_m to m; and (2) the *CL verification protocol* which is a zero-knowledge proof in which the prover can prove the knowledge of a signature on m to the verifier who knows only a commitment to m.

The commitments to m used in protocols (1) and (2) can be independent of the CL signature public key. However, for simplicity, in our application the instance of the Diophantine commitment scheme used in the CL signing protocol will be formed by values (n, a, b) which are parts of the CL signature public key.

Before we show how to use them, we need to make two modifications to the CL signatures as shown above. First, we use the [CL02] extension of the above scheme to signing a *block* of three messages (m_1, m_2, m_3). This is done simply by including three random elements a_1, a_2, a_3 in QR_n instead of one a in the public key of the CL signature scheme. The signature is a triple (v, e, s) where $v^e = a_1^{m_1} a_2^{m_2} a_3^{m_3} b^s d \bmod n$. In the CL signing and verification protocols adapted to a block of three messages, both the signer and the verifier know three separate commitments on these messages.

Second, we note that if in the CL signature verification protocol the verifier knows the message m itself instead of a commitment to it, the protocol still works and even gets easier. Similarly, if the verifier knows not the above Diophantine commitment to m, but $g^m \bmod p$ (also a commitment to m), the protocol still works, but the prover only shows knowledge of a signature on *some* integer m' s.t. $m' = m \bmod 2q$ (recall that $p = 2q+1, p, q$ are primes, and g is a generator of $\mathbb{Z}_p^*$). The same holds for the CL verification protocol extended to a block of messages (m_1, m_2, m_3). In our case, the verifier will know messages m_1 and m_2, and a commitment $g^{m_3} \bmod p$ to message m_3, and the prover will show possession of CL signature on block of messages (m_1, m_2, m'_3) s.t. $m'_3 = m_3 \bmod q$.

VTE scheme with unlinkable receipts. We recall the VTE construction of section 3, where k is the user's secret key, $pk = g^{2k} \bmod p$ is the public key, and m is the transaction plaintext. The escrow is a triple $e = (c, t, s)$ where $c = o^{2k} \bmod p$, $t = h^{2k} \bmod p$, $s = (r, \tilde{s})$, $\tilde{s} = H((c,t)|r)^{2k} \bmod p$, $h = H(Type(m))$, $o = \mathsf{pad}^H(m|r')$, and r, r' are random strings of length 2τ.

Let l_m, the maximum message length, be $|p|$, enough to represent elements in either $\mathbb{Z}_p^*$ or $\mathbb{Z}_q^*$. The public key of the escrow agent is the public key (n, a, b, c) of the CL signature scheme, except that a is chosen at random from the subgroup generated by b in $\mathbb{Z}_n^*$. If the escrow agent generates his key himself, he must prove knowledge of i s.t. $a = b^i \bmod n$.

The user sends $e = (c, t, s) = (c, t, (r, \tilde{s}))$ to the escrow agent as in the basic VTE scheme, but here he also includes three diophantine commitments C_o, C_h, C_k on *integer* values o, h, k using (n, a, b) as the instance of the commitment scheme. Using the zero-knowledge proof CKP-ZKPK of *committed key and plaintext* (see section 4.1), the user then proves his knowledge of integer values (o', h', k') s.t. o', h', k' are committed to in C_o, C_h, C_k, and $c = (o')^{2k'} \bmod p$, $t = (h')^{2k'} \bmod p$, and $\tilde{s} = H((c,t)|r)^{2k'} \bmod p$. If the proof succeeds, the user and the escrow agent run the CL signing protocol on the commitments C_o, C_h, C_k at the end of which the user holds a CL signature on the block (o', h', k') of the committed messages.

In the verification phase, the user sends to the transaction counterparty values (o, r'), together with the transaction plaintext m and his public key $pk = g^{2k} \bmod p$. The counterparty computes $h = H(Type(m))$ and verifies if $o = \mathsf{pad}^H(m|r')$. The user and the counterparty then run the CL verification protocol in which the user proves possession of a CL signature on integer values o, h, k' where the verifier knows o and h and $pk = g^{2k'} \bmod p$.

If the user passes both proofs, the first with the escrow agent as the verifier and the second with the transaction counterparty as the verifier, then under the strong RSA assumption needed for the diophantine commitment to be binding, $o' = o$, $h' = h$, and $k' = k \bmod q$, thus the escrow entry $e = (c, t, s)$ is computed correctly. Furthermore, the escrow agent learns only the (ciphertext,tag) pair $(c, t) = (o^{2k} \bmod p, h^{2k} \bmod p)$ and the signature s, while the counterparty learns only the values o, h associated with the plaintext m and the public key $pk = g^{2k} \bmod p$.

From the properties of the basic VTE scheme and the CKP-ZKPK proof system (see section 4.1), the following theorem follows:

Theorem 5. *The VTE scheme with unlinkable receipts satisfies (1) verifiability, (2) efficient and unavoidable subpoena, (3) tamper resistance, and (4) category-preserving anonymity, under the DDH and strong RSA assumptions in ROM.*

4.1 Zero-knowledge proof of committed key and plaintext

We present the ZK proof protocol required by the unlinkable-receipt VTE construction of the previous section. Recall that the user needs to prove in zero-knowledge to the escrow agent his knowledge of integer values o, h, k s.t. o, h, k are committed to in C_o, C_h, C_k, and $c = o^{2k} \bmod p$, $t = h^{2k} \bmod p$, and $\tilde{s} = H((c,t)|r)^{2k} \bmod p$. The public inputs in this proof are values (p, q, g), (n, a, b), (C_o, C_h, C_k), and $(c, t, r, \tilde{s})$. The prover's inputs are $o, h \in \mathbb{Z}_p^*$, $k \in \mathbb{Z}_q^*$, and the decommitment values o', h', k' in $\mathbb{Z}_n$.

ZKPK of Committed Key and Plaintext

Prover's Input: $k \in \mathbb{Z}_q^*$, the secret key
$o \in \mathbb{Z}_p^*$, the "plaintext"
$o', k' \in \mathbb{Z}_n$, the decommitment values

Common Input: (p, q, g), the discrete-log group setting
(n, a, b), the instance of a diophantine commitment scheme
$C_k = a^k b^{k'} \bmod n$, commitment to k
$C_o = a^o b^{o'} \bmod n$, commitment to o
$c = o^{2k} \bmod p$, the ciphertext

1. Prover P picks $\tilde{o} \leftarrow \mathbb{Z}_p^*$ and $\tilde{o}' \leftarrow \mathbb{Z}_n$, and sends $C_{\tilde{o}} = a^{\tilde{o}} b^{\tilde{o}'} \bmod n$, and $\tilde{c} = (\tilde{o})^{2k} \bmod p$ to the Verifier V
2. Verifier V sends to P a random binary challenge $b = 0$ or 1
3. P responds as follows:

$b = 0$: (a) P sends $(s, s') = (\tilde{o}, \tilde{o}')$ to V
(b) P performs a standard ZKPK proof of knowledge (*e.g.*, [CM99]) of (k, k') s.t. $a^k b^{k'} = C_k \bmod n$ and $s^{2k} = \tilde{c} \bmod p$

$b = 1$: (a) P sends $s = o * \tilde{o} \bmod p$ to V
(b) P performs a standard ZKPK proof of knowledge (*e.g.*, [CM99]) of (k, k') s.t. $a^k b^{k'} = C_k \bmod n$ and $s^{2k} = c * \tilde{c} \bmod p$
(c) P performs a ZKPK given by [CM99], of knowledge of values $(o, o', \tilde{o}, \tilde{o}')$ s.t. $a^o b^{o'} = C_o \bmod n$, $a^{\tilde{o}} b^{\tilde{o}'} = C_{\tilde{o}} \bmod n$, and $o * \tilde{o} = s \bmod p$

4. In both cases V accepts only if the appropriate ZKPK proofs verify. Additionally, if $b = 0$, V checks also if $a^s b^{s'} = C_{\tilde{o}} \bmod n$.

Fig. 1. Binary challenge proof system CKP-ZKPK

To simplify the presentation, we will show a ZKPK system for a slightly simpler problem, namely the ZK proof of knowledge *of committed key and plaintext* (CKP-ZKPK). Namely, the public values are $(p, q, g), (n, a, b), (C_o, C_k, c)$ and the prover proves knowledge of *integer* values o, k s.t. (1) they are committed to in C_o, C_k under commitment instance (n, a, b), and (2) $c = o^{2k} \bmod p$. One can see that the required ZKPK system is created by running three proofs in parallel: (i) one CKP-ZKPK proof for secrets o, k and public (C_o, C_k, c), (ii) another CKP-ZKPK proof for secrets h, k and public (C_h, C_k, t), and (iii) a standard ZKPK proof of knowledge (*e.g.* [CM99]) of k s.t. $H((c,t)|r)^{2k} = \tilde{s} \bmod p$ and k is committed to in C_k, where the public inputs are $(C_k, c, t, r, \tilde{s})$.

We present the CKP-ZKPK proof protocol in Figure 1. We note that this is a binary challenge protocol with $1/2$ soundness error, so to get security parameter τ this proof should be repeated τ times, or in the Random Oracle Model it can be made non-interactive by preparing the τ instances of it in independently in parallel, except that the challenge bits are computed by hashing together the first prover's messages of all these τ instances. The resulting protocol involves $O(\tau)$ exponentiations for both the prover and the verifier, which unfortunately makes this protocol quite expensive in practice.

Note that both ZKPK proofs referred to in the CKP-ZKPK protocol can be non-interactive in the Random Oracle Model considered here, and that they involve a small constant amount of exponentiations. We remark that the protocol proof system of [CM99] used in step (c) of case $b = 1$ for proving modular multiplication on committed values, can be simplified in our case, because here the multiplicative factor $s = o * \tilde{o}$ and the modulus p are publicly known, in contrast to the general case considered by [CM99], where the verifier knows s and p only in a committed form.

Theorem 6. *CKP-ZKPK proof system is computational zero-knowledge if the DDH problem for group QR_p is hard.*

Theorem 7. *CKP-ZKPK proof system is a proof of knowledge with soundness error $1/2$ if the strong RSA problem in group $\mathbb{Z}_n$ is hard.*

5 VTE Scheme with Automatic Threshold Disclosure

We describe an extension of the VTE scheme which enables the escrow agent to automatically open escrows that (1) fall into the same *bin*, *i.e.*, share the same $\langle$user,type$\rangle$ category, and (2) their number is no less than some fixed threshold, pre-specified for transactions of this type. This can be used, for example, to implement oversight of financial transactions which the following disclosure condition: if some user requests more than 10 transfers, via any set of banks, to some pre-specified "offshore haven," the plaintexts of the corresponding escrows must be automatically disclosed to the overseeing authority.

Using Feldman's non-interactive verifiable secret sharing scheme [Fel87], we modify the VTE scheme of section 3 as follows. To create an escrow of plaintext m under key k, the user computes the tag $t = \mathsf{Tag}_k\{T\}$ where $T = Type(m)$ as in section 3, but the ciphertext is computed differently. Let d be the publicly pre-specified threshold disclosure value that corresponds to this T. The user picks a unique d-degree *secret-sharing polynomial* $f(\cdot)$ by applying $d+1$ times a pseudorandom function indexed by the secret k, *i.e.*, $k_i = H(k, T, i)$ for $i = 0, \ldots, d$, where $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$, and setting $f(x) = k_0 + k_1 x + \ldots k_d x^d \bmod q$. A set of values $\{C_0, \ldots, C_d\}$ where $C_i = g^{2k_i} \bmod p$ serves as *public verification information* for this secret-sharing polynomial, The ciphertext is now $c' = (c, \{C_i\}_{i=0..d}, x, f(x), d)$, where $c = \mathsf{Enc}_{k_0}\{m\}$, π is the proof that c is a correct encryption of m under the "quasi-one-time" private key k_0 (and its public counterpart $C_0 = g^{2k_0} \bmod p$), and x is some unique value corresponding to this transaction, *e.g.*, $x = H(c)$. The user computes the private signature $s = (r, \tilde{s})$ on (c', t), and hands the escrow $e = (c', t, s)$ to the escrow agent.

The escrow agent checks that $(x, f(x))$ is a true data point on the polynomial committed to in set $\{C_i\}_{i=0..d}$ by verifying that $g^{2f(x)} = (C_0) * (C_1)^x * \ldots * (C_d)^{x^d} \bmod p$. Moreover, if the bin tagged with tag t in the escrow database has other entries, the agent checks that the argument x has not been used before with the tag t, and the values $\{C_0, \ldots, C_d\}$ are the same for this t as before. The agent then releases his signature on the escrow e to the user. The user presents it to the counterparty, who verifies it as before, except that correctness of the ciphertext $c = \mathsf{Enc}_{k_0}\{m\}$ is verified on (C_0, m, c, π) instead of (pk, m, c, π), and it is checked that d is the threshold value corresponding to type T.

To prevent the counterparty and the escrow agent from linking their views, the same mechanism as in section 4 may be deployed. The user sends commitments C_o, C_h, C_k on values o, h, k to the escrow agent (note the difference between C_o and C_0), proving his knowledge of o, h, k, k_0 s.t. $c = o^{2k_0} \bmod p$, $C_0 = g^{2k_0} \bmod p$, $t = h^{2k} \bmod p$, and $\tilde{s} = H((c', t)|r)^{2k} \bmod p$. The same zero-knowledge protocol as in section 4 may be used, and is even slightly simpler since C_0 is a simpler commitment to k_0 than the Diophantine commitment. After checking the proofs, the user and the escrow agent perform the CL signing protocol to give the user a CL signature on the block of messages (o, h, k, d). The user then sends to the counterparty values (o, r') as in section 4, together with d. The counterparty checks that o is properly formed and d is the proper threshold value for the given transaction type, and they run the CL verification protocol to prove the user's knowledge of a CL signature on values (o, h, k, d) where the verifier knows o, h, d and $pk = g^{2k} \bmod p$.

Acknowledgments. We want to thank Dan Boneh, Pat Lincoln, and Anna Lysyanskaya for helpful discussions and for proposing extensions and improvements. We also thank the anonymous referees for their suggestions.

References

[ASW98] N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. In *Proc. IEEE Symposium on Security and Privacy*, pages 86–99, 1998.

[BBDP01] M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In *Proc. Asiacrypt '01*, pages 566–582, 2001.

[BCOP03] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Searchable public key encryption. IACR 2003/195, 2003.

[Bon98] D. Boneh. The decisional Diffie-Hellman problem. In *Proc. 3rd Algorithmic Number Theory Symposium*, pages 48–63, 1998.

[BR93] M. Bellare and P. Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proc. ACM Conference on Computer and Communications Security*, pages 62–73, 1993.

[BR94] M. Bellare and P. Rogaway. Optimal asymmetric encryption. In *Proc. Eurocrypt '94*, pages 92–111, 1994.

[CD00] J. Camenisch and I. Damgard. Verifiable encryption, group encryption, and their applications to group signatures and signature sharing schemes. In *Proc. of Asiacrypt '00*, pages 331–345, 2000.

[CKS00] C. Cachin, K. Kursawe, and V. Shoup. Random oracles in Constantinople. In *Proc. of PODC '00*, pages 123–132, 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.

[CL02] J. Camenish and A. Lysyanskaya. A signature scheme with efficient protocols. In *Proc. SCN '02*, pages 268–289, 2002.

[CM99] J. Camenisch and M. Michels. Proving in zero-knowledge that a number is the product of two safe primes. In *Proc. Eurocrypt '99*, pages 107–122, 1999.

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

[CS97] J. Camenisch and M. Stadler. Proof systems for general statements about discrete logarithms. Technical Report TR 260, ETH Zürich, March 1997.

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

[DF01] I. Damgard and E. Fujisaki. An integer commitment scheme based on groups with hidden order. IACR 2001/064, 2001.

[Dod03] Y. Dodis. Efficient construction of (distributed) verifiable random functions. In *Proc. Public Key Cryptography '03*, pages 1–17, 2003.

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

[Fis99] M. Fischlin. Pseudorandom function tribe ensembles based on one-way permutations: improvements and applications. In *Proc. Eurocrypt '99*, pages 432–445, 1999.

[FO98] E. Fujisaki and T. Okamoto. A practical and porvably secure scheme for publicly verifiable secret sharing and its applications. In *Proc. Eurocrypt '98*, pages 32–46, 1998.

[GJKR99] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Secure distributed key generation for discrete-log based cryptosystems. In *Proc. of Eurocrypt '99*, pages 295–310, 1999.

[JS04] S. Jarecki and V. Shmatikov. Handcuffing Big Brother: An abuse-resilient transaction escrow scheme. IACR `eprint`, 2004.

[KL95] J. Kilian and F.T. Leighton. Fair cryptosystems revisited. In *Proc. Crypto '95*, pages 208–221, 1995.

[Lys02] A. Lysyanskaya. Unique signatures and verifiable random functions from the DH-DDH separation. In *Proc. Crypto '02*, pages 597–612, 2002.

[Mic92] S. Micali. Fair public-key cryptosystems. In *Proc. Crypto '92*, pages 113–138, 1992.

[MRV99] S. Micali, M. Rabin, and S. Vadhan. Verifiable random functions. In *Proc. FOCS '99*, pages 120–130, 1999.

[NPR99] M. Naor, B. Pinkas, and O. Reingold. Distributed pseudo-random functions and KDCs. In *Proc. of Eurocrypt '99*, pages 327–346, 1999.

[Sho01] V. Shoup. OAEP reconsidered. In *Proc. of Crypto '01*, pages 239–259, 2001.

[SWP00] D.X. Song, D. Wagner, and A. Perrig. Practical techniques for searches on encrypted data. In *Proc. IEEE Symposium on Security and Privacy*, pages 44–55, 2000.