How to Squeeze a Crowd: Reducing Bandwidth in Mixing Cryptocurrencies

Alishah Chator Matthew Green

Johns Hopkins University

***Abstract*—Several popular cryptocurrencies incorporate privacy features that "mix" real transactions with cover traffic in order to obfuscate the public transaction graph. The underlying protocols, which include CryptoNote and Monero's RingCT, work by first identifying a real transaction output (TXO), sampling a number of cover outputs, and transmitting the entire resulting set to verifiers, along with a zero knowledge (or WI) proof that hides the identity of the real transaction. Unfortunately, many of these schemes suffer from a practical limitation: the description of the combined input set grows linearly with size of the anonymity set.**

In this work we propose a simple technique for efficiently sampling cover traffic from a finite (and public) set of known values, while deriving a *compact* description of the resulting transaction set. This technique, which is based on programmable hash functions, allows us to dramatically reduce transaction bandwidth when large cover sets are used. We refer to our construction as a *recoverable sampling scheme*, and note that it may be of independent interest for other privacy applications. We present formal security definitions; prove our constructions secure; and show how these constructions can be integrated with various currencies and different cover sampling distributions.

***Index Terms*—Privacy, Anonymity, Mixing, Ring Signatures.**

I. Introduction

Cryptocurrencies such as Bitcoin suffer from well-known privacy limitations. These stem from the fact that each transaction on the currency's public ledger is explicitly linked to one or more preceding transaction outputs from which funds originate. A number of academic works [1], [2], [3], [4], [5] and for-profit companies [6], [7] have demonstrated that sensitive payment information can be extracted from the resulting public transaction graph.

Several recent currencies address this problem by directly incorporating cryptographic mixing into the consensus protocol. The underlying protocols, which include Zerocash [8], CryptoNote [9], Zerocoin [10] and RingCT [11] obscure the identity of the previous transaction(s) being consumed by hiding them within a larger set of cover transactions. In practice this is frequently accomplished using non-interactive witness-indistinguishable (WI)[1] or zero knowledge (ZK) proofs, such as zkSNARKs [13]. Regardless of the exact technology employed for the proof system, transactions in these systems can be viewed as making the following statement:

> *This transaction references M "real" previous transaction outputs $(I_1, \ldots I_M)$ embedded within a public "transaction output list" $\mathcal{T}$ of previous outputs drawn from the ledger.*

While the protocols above use different techniques, for the purposes of this paper we will consider only two aspects: (1) the size of the transaction list $\mathcal{T}$ in each transaction (which we denote by $|\mathcal{T}|$), and (2) the size of the resulting transaction as a function of $|\mathcal{T}|$. The former directly affects the privacy provided by the protocol: larger values of $|\mathcal{T}|$ may permit a larger anonymity set for each transaction. At the same time, limited space on the ledger can make larger transactions unworkable.

Protocols such as Zerocash and Zerocoin [8], [10] set $\mathcal{T}$ to be the set of of *all* previous transactions, using cryptographic accumulators and succinct proofs. This maximizes $|\mathcal{T}|$ and minimizes transaction size, though at the cost of using strong cryptographic assumptions. By contrast, "mixing" protocols such as CryptoNote and RingCT (used by Monero [14]) use a smaller cover set. In these protocols, the spender randomly samples a cover set for *each* transaction and transmits the description of $\mathcal{T}$ as part of the transaction data. The need to encode a new subset $\mathcal{T}$ within each transaction creates tension

[1]These WI proofs are often used as part of a larger primitive, such as a ring signature [12].

between the size of the anonymity set and the transaction size.

In this work, we focus exclusively on currencies that follow the mixing approach. Specifically, we address the following problem: using current approaches, the transaction bandwidth needed to describe $\mathcal{T}$ grows as $O(|\mathcal{T}|)$. While this is acceptable for small cover sets, it makes these protocols unworkable with larger amounts of cover traffic.[2] This is problematic, given that there is impetus within the development community to greatly increase the amount of cover traffic used in currencies such as Monero — to values as large as $|\mathcal{T}| = 100,000$ [15] — by incorporating asymptotically more efficient proof systems [16], [17]. As this work is deployed, the description of $\mathcal{T}$ will increasingly dominate as the source of transaction bandwidth, and may become the effective limit on the size of the anonymity set.

Our contribution. In this work we describe an alternative approach to sampling the transaction list $\mathcal{T}$ such that the resulting description of $\mathcal{T}$ is *compact*. Our approach is relatively simple, although the details and security proofs require considerable attention. Rather than sampling $\mathcal{T}$ using the current approach, we propose to sample and encode this set using a programmable keyed hash function (see *e.g.,* [18], [19]) with a relatively short key K. We show that using this approach, the description of $\mathcal{T}$ can be structured so that it grows with the number of real transactions M, rather than with the number of total transactions N. For large cover sets, our approach should represent a significant improvement in space efficiency over the naïve approach currently used by real currency systems [20], [14].

Of course, this approach cannot be achieved using any hash function. In this work we show how to construct such a function, as part of a protocol that we call a *recoverable sampling scheme*. We provide security definitions and security proofs for our constructions in the random oracle model.

We discuss our approach in two settings. First, we consider an approach that samples the cover traffic *uniformly* from the set of all previous transaction outputs, which closely matches the approach used by CryptoNote (as implemented in the ByteCoin currency) [9], [20]. We then move to a more complex (and realistic) setting where $\mathcal{T}$ is sampled according to a specific and *non-uniform* distribution. Finally, we discuss concrete scheme parameters and give bandwidth cost estimates for integrating our approach into various deployed currencies, including ByteCoin [20] and Monero [14]. As a final matter, we describe applications of our technique to other application areas, including anonymous communications and Client-Server Puzzles.

A. Intuition

At a high level, our approach is straightforward: rather than sampling and transmitting the full set $\mathcal{T}$ with each transaction, we define a specialized hash function $H_K(\cdot)$ with some useful properties. To construct a transaction and transaction list $\mathcal{T}$, the spender first constructs and transmits a key K within the transaction, along with an integer N. The nature of the function is such that, given this key, any party can now efficiently compute $\mathcal{T}$ as $(H_K(0), H_K(1), \ldots, H_K(N-1))$.

The challenge is to construct the function H such that the resulting set $\mathcal{T}$ will embed up to M "real" transactions chosen by the spender, without revealing *which* transactions are the real ones, and which are cover traffic.[3] To do this, we must solve several problems. First, we require a function H with a compact key K that can be programmed to incorporate the real transactions at random points. Second, we must ensure that the remaining "cover" transactions are all sampled from the appropriate distribution (which may not be uniform). Finally, we need to prove that this technique is as secure as the naïve sampling approach; *i.e.,* that it does not leak new information to an attacker who wishes to identify the real transactions.

As a warm up, we begin by describing a simple technique for sampling the cover set uniformly from the list of all transaction outputs on a ledger consisting of ℓ such outputs. For purposes of exposition, we will consider the simple case where there is only one real transaction ($M = 1$), and we will assume that all parties know the list of previous transactions.

As an ingredient we require an underlying keyed hash function $f : \{0,1\}^\kappa \times \mathbb{Z}_N \to \mathbb{Z}_\ell$, which we model as a random oracle.[4] We now define a simple programmable hash function $H : \{0,1\}^\kappa \times \mathbb{Z}_\ell \times \mathbb{Z}_N \to \mathbb{Z}_\ell$ as follows:

$$H_{k,C}(i) = f_k(i) + C \ mod \ \ell$$

Notice that a spender, who wishes to embed some real transaction index $I \in [0, \ell)$ in $\mathcal{T}$, can easily program

[2] In practice this set is generally encoded using $|\mathcal{T}|$ differentially-encoded transaction indices. Accumulator-based currencies work around this issue, since the set $\mathcal{T}$ can be represented by a single value indicating the current transaction's position in the ledger.

[3] In practice, this guarantee depends largely on the distribution of the real transactions. We define an approach that guarantees our technique is "no worse" than the naïve sampling approach.

[4] Such functions can easily be constructed from any standard cryptographic hash function.

this function as follows: sample a random $k \in \{0,1\}^\kappa$ and a random index $j \in [0, N)$. Now compute a value C such that $C \equiv I - f_k(j) \ mod \ \ell$. This implicitly defines $H_k(j) \equiv I \ mod \ \ell$ and sets each remaining $H(i), i \neq j$ to be uniform in $\mathbb{Z}_\ell$.

Of course, this simple scheme works only for cases where there is a single real transaction. In practice, real protocols may require the spender to embed *several* real transactions into $\mathcal{T}$. In our main construction, we generalize the above construction by replacing the single value C with a polynomial $P(\cdot)$ evaluated over a field F_p for some large prime p. This approach allows us to ensure that *for any* subset of M distinct indices j, $f_k(j) + P(j) \equiv I_j \ mod \ \ell$. It remains only to prove that the resulting scheme is "as secure" as the naïve sampling approach. We provide a security definition for this claim, and offer a proof in the random oracle model.

Of course, this proposal is not complete. Given a solution such as the one above, we must still adapt the details so that they work within a real cryptocurrency. In later sections of the paper we discuss altering the function H so that the cover traffic is sampled from a more realistic distribution. Additionally, we address the problem that our approach may produce *duplicate* transaction outputs within the transaction list $\mathcal{T}$, and discuss how to remove these. We provide a detailed discussion of these aspects of the problem in §V.

B. Outline of this work

The remainder of this work proceeds as follows. In the next two sections we provide definitions for our schemes. In §IV we provide a construction that assumes a *uniform* sampling distribution for cover transactions. In §V we discuss sampling for alternative distributions. In §VI we show how our constructions can be integrated with specific cryptocurrency mixing protocols, and give concrete estimates of transaction size. In §VII we describe some different potential applications for these techniques outside of cryptocurrency privacy. Finally, §VIII discusses related work.

II. Preliminaries

a) Notation: We will define $\nu(\cdot)$ to be a negligible function. Let $A \overset{c}{\approx} B$ indicate that the distributions A, B are computationally indistinguishable. We will use $\mathcal{D}$ to refer to a specific distribution for sampling transaction outputs.

b) Transaction sets: While thus far we have referred to transaction *sets*, in our main constructions we will relax the requirement that each element is unique, and we will add an implicit ordering. Henceforth we will consider $\mathcal{T}$ to be an ordered *multiset*, or merely a list of transactions.

c) Transaction ledger: We assume the existence of an append-only public ledger of transaction outputs $\mathbf{L} = (T_0, T_1, \ldots, T_{\ell-1})$ that is available to all parties in the system. New transaction outputs are appended to $\mathbf{L}$ after they have been verified by the network. By $\mathbf{L}_\ell$ we denote the first ℓ transactions on $\mathbf{L}$.

We will assume that each transaction output $T_i \in \mathbf{L}_\ell$ can be referenced uniquely by its index $I \in \{0, \ldots, \ell - 1\}$; henceforth we will use these indices exclusively to represent transaction outputs on the ledger.

d) Keyed hash functions with integer domain and range: Let m, n be positive integers. We define $f_{m,n} : \{0,1\}^\kappa \times \mathbb{Z}_m \to \mathbb{Z}_n$ as a keyed function that, on input a key $k \in \{0,1\}^\kappa$ and an integer $a \in \{0, \ldots, m-1\}$, outputs an integer $b \in \{0, \ldots, n-1\}$. We will assume that each pair (m, n) uniquely defines the function family, and that these functions can be re-constructed efficiently by any party.[5] In our security proofs we will model this function as a random oracle.

III. Definitions

The purpose of this work is to define an approach to sampling the transaction list that produces a compact description. The primitive we introduce for this purpose is. We now define this scheme.

Definition 3.1: A recoverable sampling scheme (RSS) is parameterized by a sampling distribution $\mathcal{D}$. It consists of two possibly probabilistic algorithms $(\mathsf{Sample}, \mathsf{Recover})$ with the following definition:

> $\mathsf{Sample}_\mathcal{D}(1^\lambda, \ell, \mathcal{I}, N) \ \rightarrow \ (\mathcal{T}, \mathsf{W})$. On input a security parameter λ, a ledger size ℓ, a set of M legitimate transaction indices $\mathcal{I} = \{I_0, \ldots, I_{M-1}\} \in \mathbb{Z}_\ell^M$, and a desired number of total transactions $N > M$, this algorithm outputs a tuple (ordered multiset) $\mathcal{T}$ containing N (possibly non-unique) elements, and a compact description W.
>
> $\mathsf{Recover}_\mathcal{D}(\mathsf{W}, \ell) \ \rightarrow \ \mathcal{T}$. On input a compact description W and the ledger size ℓ, outputs the transaction multiset $\mathcal{T}$ or the distinguished failure symbol $\perp$.

An RSS must satisfy three properties, which we refer to as *correctness*, *compactness*, and *security*. We define these below.

[5]Note that functions of this form can be constructed efficiently from any standard underlying hash function using a variety of techniques.

Correctness. Let $(\mathcal{T}, \mathsf{W}) \leftarrow \mathsf{Sample}_{\mathcal{D}}(1^\lambda, \ell, \mathcal{I}, N)$. For an RSS to be *correct*, several requirements must be met for all valid $(\lambda, \ell, \mathcal{I}, N)$. First, it must hold that $\mathcal{T}$ contains N elements and $\mathcal{I} \subset \mathcal{T}$. Finally, the following equality must be satisfied:

$$\mathsf{Recover}_{\mathcal{D}}(\mathsf{W}, \ell) = \mathcal{T}$$

Compactness. For an RSS to be useful, it must hold that given a fixed M the size of the compact description W should grow sublinearly as N increases. We note that the compactness requirement rules out a class of trivial schemes, including any scheme that simply outputs $\mathsf{W} = \mathcal{T}$.

Security. Intuitively, security for a recoverable sampling scheme requires that the scheme reveals "no more" information to an attacker than would be revealed by an ideal implementation that simply samples $N - M$ cover transactions and combines these with $\mathcal{I}$ in a random ordering. To define this form of security, we must first describe two experiments.

> **Real experiment**. Let $(\mathcal{A}, \lambda, \ell, \mathcal{I}, N, \mathcal{D})$ be the input to the experiment. Compute $(\mathcal{T}, \mathsf{W}) \leftarrow \mathsf{Sample}_{\mathcal{D}}(1^\lambda, \ell, \mathcal{I}, N)$ and run $\mathcal{A}(\ell, |\mathcal{I}|, N, \mathsf{W})$. The output of the experiment is $\mathcal{A}$'s output.
>
> **Ideal experiment**. Let $(\mathcal{B}, \lambda, \ell, \mathcal{I}, N, \mathcal{D})$ be the input to the experiment. Construct a multiset $\mathcal{C}$ consisting of $N - |\mathcal{I}|$ (possibly duplicate) transaction indices (sampled according to $\mathcal{D}$) from the set $\mathbb{Z}_\ell$; and shuffle these values randomly with the values in $\mathcal{I}$ to obtain the ordered list $\mathcal{T}$. Next run $\mathcal{B}(\ell, |\mathcal{I}|, N, \mathcal{T})$. The output of the experiment is $\mathcal{B}$'s output.

We are now prepared to define security for an RSS.

Definition 3.2 (Security for RSS): An RSS $\Pi = (\mathsf{Sample}, \mathsf{Recover})$ is secure if for every *p.p.t.* adversary $\mathcal{A}$, sufficiently large λ, and all valid $(\ell, \mathcal{I}, N, \mathcal{D})$, there exists a p.p.t. $\mathcal{B}$ such that the following holds:

$$\mathbf{Real}(\mathcal{A}, \lambda, \ell, \mathcal{I}, N, \mathcal{D}) \stackrel{c}{\approx} \mathbf{Ideal}(\mathcal{B}, \lambda, \ell, \mathcal{I}, N, \mathcal{D})$$

Discussion. We note that this is a purely comparative definition. That is, our definition does *not* imply that the "ideal" sampling scheme is itself secure for any distribution or set of input transactions. Indeed, for many values of $\mathcal{I}$ the real transactions may be easily distinguishable from the cover transactions! Addressing that problem is not the purpose of this work: instead, our goal merely to show that the RSS scheme performs "no worse" than the ideal approach, while offering an improvement in bandwidth efficiency.

As an additional note, our ideal experiment implements sampling with replacement, resulting in the potential for duplicates in the transaction list $\mathcal{T}$. This is slightly different than some implemented schemes that remove duplicates. We make this relaxation order to simplify the presentation of the rest of this paper, although in later sections we address the problem of removing duplicates from this set. Finally, to simplify our definitions, we do not explicitly include auxiliary inputs to the parties. This must be done for sequential composition of the primitive. Our constructions achieve this notion as well.

IV. A Uniform Sampling Technique

We begin by describing a technique for sampling the cover set *uniformly* from the set of all transactions on a ledger $\mathbf{L}_\ell$. This technique mirrors the approach of certain protocols such as CryptoNote [9] as implemented in the ByteCoin currency [20]. The algorithms we describe below allow us to embed M real input transactions into a transaction multiset $\mathcal{T}$ of size N, where $0 < M < N$. We will assume that both spender and verifier have access to the ledger $\mathbf{L}_\ell$.

The sampling algorithm. Let $\mathbf{L}_\ell$ be the ledger. Let $I_0, ...I_{M-1} \in \mathbb{Z}_\ell$ represent the *indices* (into $\mathbf{L}$) of the legitimate input transactions. Let N represent the desired size of $\mathcal{T}$ and $M > 1$ represent the number of legitimate input transactions (where $M < N$). We will make use of a field $\mathbb{F}_p$ of prime order $p \gg \ell$ (such that $\frac{1}{p/\ell}$ is negligible). To sample this set, the spender performs the steps in Figure 1.

The recovery algorithm. The above algorithm produces a set $\mathcal{T}$ that can be used to construct the transaction. However, the full description of $\mathcal{T}$ need not be included in the transaction. Instead, the spender may include the tuple $W = (k, P, \ell, N)$. This can be used by the verifier to recover $\mathcal{T}$ as described in Figure 1.

Correctness and compactness. Assuming a fixed M, the size of the compact description W clearly grows at most logarithmically as N increases.[6] It is easy to see that the algorithms above are *correct*, in the sense that $I_0, ..., I_{K-1}$ is contained within $\mathcal{T}$. Specifically, there

[6] And this is only because W contains a representation of N in order to simplify the description of the algorithm.

$\mathsf{Sample}_{\mathcal{U}}(1^\lambda, \ell, \mathcal{I} = \{I_0, \ldots, I_{M-1}\}, N)$

Choose a prime p such that $1/(p/\ell) \leq \nu(\lambda)$.
Construct the keyed hash function $f : \{0,1\}^\kappa \times \mathbb{Z}_N \to \mathbb{Z}_p$.
Sample a random key $k \in \{0,1\}^\lambda$.
Sample random $\{j_0, ..., j_{M-1}\} \subset \{0, \ldots, N-1\}$.
For $i = 0$ to $M-1$: Choose y_i such that $f_k(i) - y_i \equiv I_i \ mod \ \ell$.
Compute $a_0, ..., a_{M-1}$ that define an order-$(M-1)$ polynomial $P(\cdot)$ over $\mathbb{F}_p$ s.t. $\forall i \in [0, M)$, $P(j_i) = y_{j_i}$.
For $i = 0$ to $N-1$: compute $T_i = f_k(i) - P(i) \ mod \ \ell$.
return $\mathcal{T} = \{T_0, \ldots, T_{N-1}\}$ and $W = (p, k, P, \ell, N)$.

$\mathsf{Recover}_{\mathcal{U}}(\mathsf{W}, \ell)$

Construct the keyed hash function $f : \{0,1\}^\kappa \times \mathbb{Z}_N \to \mathbb{Z}_p$.
For $i = 0$ to $N-1$: compute $T_i = f_k(i) - P(i) \ mod \ \ell$.
return $\mathcal{T} = \{T_0, \ldots, T_{N-1}\}$.

Fig. 1. The RSS uniform sampling and recovery algorithms

exists $\{j_0, ..., j_{K-1}\} \subset \{0, \ldots, N-1\}$ such that for each j_i:

$$\begin{aligned} T_{j_i} &\equiv f_{N,\ell}(j) - P(j_i) \quad mod\ \ell. \\ &\equiv f_{N,\ell}(k,j) - y_i \quad mod\ \ell \\ &\equiv f_{N,\ell}(k,j) - f_{N,\ell}(k,j) + I_i \quad mod\ \ell \\ &\equiv I_i \quad mod\ \ell. \end{aligned}$$

A. Security

We now prove that the above scheme is a secure RSS in the sense of Definition 3.2.

Theorem 4.1: The protocol $\Pi = (\mathsf{Sample}, \mathsf{Recover})$ described above is a secure RSS if the function $f : \{0,1\}^\kappa \times \mathbb{Z}_N \to \mathbb{Z}_p$ is modeled as a random oracle.

Proof. To succeed in our proof, we must show that for every p.p.t. $\mathcal{A}$ there exists a p.p.t. adversary $\mathcal{B}$ such that for all p.p.t. distinguishers $\mathcal{Z}$ we have that $|\mathbf{Pr}[\mathcal{Z}(\mathbf{Real}(\mathcal{A}, \lambda, \ell, \mathcal{I}, N, \mathcal{U})) = 1] - \mathbf{Pr}[\mathcal{Z}(\mathbf{Ideal}(\mathcal{B}, \lambda, \ell, \mathcal{I}, N, \mathcal{U})) = 1]| \leq \nu(\lambda)$ over all valid $\ell, \mathcal{I}, N$. Let $\mathcal{A}$ be the adversary that interacts in the **Real** experiment. Given $\mathcal{A}$ we show how to construct $\mathcal{B}$, which satisfies the above requirement.

$\mathcal{B}$ conducts the **Ideal** experiment, and runs $\mathcal{A}$ (and answers its random oracle queries) as follows. When $\mathcal{B}$ receives $(\ell, |\mathcal{I}|, N, \mathcal{T})$ from the **Ideal** challenger, it parses $\mathcal{T} = (I_0, \ldots, I_{N-1}) \in \mathbb{F}_p^N$. It then selects p as in the real protocol, samples a random key $k \in \{0,1\}^\kappa$ and a random set of coefficients $(a_0, ..., a_{M-1}) \in \mathbb{Z}_p^M$ that define the polynomial $P(\cdot)$. For $i = 0$ to $N-1$, it programs the random oracle such that:[7]

$$f_k(i) \equiv P(i) - I_i \ mod\ \ell$$

Finally, it sets $\mathsf{W} = (p, k, P = (a_0, ..., a_{M-1}), \ell, N)$ and sends $(\ell, |\mathcal{I}|, N, \mathsf{W})$ to $\mathcal{A}$. When $\mathcal{A}$ produces an output, $\mathcal{B}$ uses this as its own output. This completes the simulation.

To complete the proof, we must show that if $f.(\cdot)$ is modeled as a random oracle, the simulated distribution provided to $\mathcal{A}$ is computationally indistinguishable from the **Real** experiment on the same inputs. This implies that the distribution of $\mathcal{B}$'s output must in turn be computationally indistinguishable from that of $\mathcal{A}$ in the **Real** experiment.

Our proof proceeds via a series of hybrids. The first hybrid represents the **Real** experiment, while the final hybrid is distributed as in the **Ideal** experiment. We will define $\mathsf{Adv}[i]$ to be the quantity $|\mathbf{Pr}[\mathcal{Z}(\mathbf{Hybrid\ i}(\mathcal{A}, \ell, \mathcal{I}, N, \mathcal{U})) = 1] - \mathbf{Pr}[\mathcal{Z}(\mathbf{Real}(\mathcal{A}, \ell, \mathcal{I}, N, \mathcal{U})) = 1]|$.

Hybrid 0. This hybrid implements the experiment $\mathbf{Real}(\mathcal{A}, \lambda, \ell, \mathcal{I}, N, \mathcal{U})$. Clearly $\mathsf{Adv}[0] = 0$.

Hybrid 1. This hybrid modifies the above hybrid as follows: for all $f(\cdot, \cdot)$ oracle queries $\mathcal{A}$ makes *prior* to receiving W, if the oracle query

[7] If the oracle has already been queried by $\mathcal{A}$ (and thus implicitly defined at this point) prior to this stage of $\mathcal{B}$'s operation, $\mathcal{B}$ aborts and outputs $\perp$.

has the form $(k, \cdot)$, then abort the experiment and output $\perp$.

We observe that k is random and not in $\mathcal{A}$'s view prior to receiving W. Thus if $\mathcal{A}$ makes q queries, the probability of abort is at most $q \cdot 2^{-\lambda}$. This bounds $\mathsf{Adv}[1] - \mathsf{Adv}[0] \leq q \cdot 2^{-\lambda}$.

Hybrid 2. This hybrid modifies the above hybrid as follows: it randomly samples the coefficients $a_0, ..., a_{M-1}$ that define $P(\cdot)$ and programs the random oracle such that $\forall i \in [0, N)$, $f_k(i) \equiv P(i) - I_i \mod \ell$.

Let us implicitly define $\mathcal{I} = (I'_0, \ldots, I'_{M-1})$, and $\mathcal{T} \setminus \mathcal{I} = (I'_M, \ldots, I'_{N-1})$, and define the corresponding locations of these transactions in $\mathcal{T}$ as $(j_0, \ldots, j_{N-1})$. We make the following observations:

1) The distribution of $(a_0, ..., a_{M-1})$ is identical to that of the previous hybrid. This is because in **Hybrid 1** it holds that (1) each $f_k(\cdot)$ is uniformly distributed in $\mathbb{F}_p$, and (2) for a given $\mathcal{I}$ and $\forall i \in [0, m)$ the value $P(i)$ is uniquely determined by $f_k(\cdot)$, thus (3) each pair $(f_k(i), P(i))$ is equally probable.This implies that the distribution of these pairs is identical in this hybrid and the previous hybrid. Because there is exactly one unique polynomial for each set of M such points, and the points are uniformly distributed in $\mathbb{F}_p^M$, then the coefficients in **Hybrid 1** must also be distributed uniformly in $\mathbb{F}_p^M$. Thus the coefficients are distributed identically in the two hybrids.
2) Similarly, each distinct tuple of coefficients $(a_0, ..., a_{M-1})$ uniquely defines a distinct tuple $\hat{P} = (P(j_0), \ldots, P(j_{M-1}))$. Because the coefficients are sampled uniformly in this hybrid, then the tuple $\hat{P}$ is distributed uniformly in $\mathbb{F}_p^M$. Thus $\forall i \in [0, M)$ it holds that $f_k(j_i) \equiv P(j_i) - I'_{j_i} \; mod \; \ell$ is uniform. This is identical to **Hybrid 1**.
3) For every index $i \in (j_M, \ldots, j_{N-1})$ (*i.e.,* the indices not in $\mathcal{I}$, every index I'_{j_i} in **Hybrid 1** is uniformly sampled from $\mathbb{Z}_\ell$. Provided that p is substantially larger than ℓ (by a large factor) then each point $f_k(j_i)$is also distributed (nearly) uniformly and indistinguishably from **Hybrid 1**.

   As a consequence, all values provided to $\mathcal{A}$ and all oracle queries are distributed identically to **Hybrid 1**. This bounds bounds $\mathsf{Adv}[2] - \mathsf{Adv}[1] \leq \frac{1}{p/\ell}$.

By summation over the hybrids we have that $\mathsf{Adv}[2] - \mathsf{Adv}[0] \leq q \cdot 2^{-\lambda} + \frac{1}{p/\ell}$, and therefore if $\frac{1}{p/\ell}$ is negligible then $\mathcal{B}$'s success probability is at most negligibly different from that of $\mathcal{A}$. □

V. Duplicates and Alternative Distributions

The scheme we presented above serves as a useful first step. However, two major problems remain. First, the transaction list $\mathcal{T}$ contains duplicates, which may reduce the effective size of the anonymity set. Second, while our above technique works well for uniform distributions, many desired applications may use alternate distributions. We discuss both issues below.

A. Duplicates

It is important to note that the scheme above offers only a probabilistic guarantee for the actual number of unique transactions in $\mathcal{T}$. There is a chance that duplicate transactions will occur in this list, resulting in a (unique) set of size less than N. We discuss several ways to handle this.

a) Expected number of unique transactions: The simplest strategy is to accept the existence of duplicate transactions and instead focus on the expected number of unique transactions for a transaction list of size N. We can compute this with the following formula:

$$\ell(1 - (\frac{\ell - 1}{\ell})^N)$$

If we use concrete numbers based on Monero with $\ell = 4,000,000, N = 1,000$, the expected number of unique transactions is 999.88. Based on this, it seems the risk of duplicate transactions is fairly low. However, we can do better if we truly want to minimize this risk.

b) Resampling: To eliminate duplicates entirely, we can resample $\mathcal{T}$ until we obtain N unique transactions. We can bound the number of attempts needed to minimize the chance of obtaining duplicates to 2^{-50} with the following formula, where r is the number of resamples.

$$r = \frac{-50 \log(2)}{\log(1 - \frac{\ell! \ell^{-N}}{(\ell - N)!})}$$

Using concrete numbers ($\ell = 4,000,000, N = 1,000$), then we find after 17 resamples the probability of

duplicates is within this bound. This may have a modest impact on the security reduction.

c) Oversampling: A final strategy is to actually sample N' values so that we obtain N unique transactions with overwhelming probability. We do this by computing the required value of N' such that the probability of obtaining fewer than N unique transactions is bounded by 2^{-50} using the following formula:

$$N' = (N+1) - \frac{\log(\frac{2^{50}}{\sqrt{2*\pi}})}{(\log(1/\ell))}$$

Using concrete numbers ($\ell = 4,000,000, N = 1,000$), then we find that if we set $N' = 1,004$ we will obtain N unique transactions with overwhelming probability.

B. Alternative Distributions

Some currencies, including Monero, use sample cover transaction outputs according to a non-uniform distribution. This strategy is designed to produce cover traffic that more closely resembles the distribution of real transaction outputs chosen by users. At a basic level, Monero achieves this goal by using a distribution that is biased against older transactions. Nominally this distribution is referred to as a triangular distribution, however it has differences from a triangular distribution so that it has further bias against older transactions. At a low level, Monero samples random numbers uniformly, and then transforms them to the appropriate distribution. We provide a outline of the Monero sampling technique in Figure 2.

Given this distribution, we will demonstrate how to apply the technique from Figure 1 to this distribution. In the sampling method we modify the y_i values to be computed in the following way:

$$y_i = f_{N,\ell}(k, j_i) - \frac{I_i^2}{\ell^2} \cdot 2^{53} \ mod \ \ell$$

And compute the T_i values as follows:

$$T_i = \sqrt{\frac{f_{N,\ell}(k,i) - P(i)}{2^{53}}} \cdot \ell \ mod \ \ell$$

Similarly we modify the recover algorithm to compute T_i as above.

For distributions such as the one above which utilize uniform sampling in their method, the conversion to our scheme is fairly straightforward. We simply replace the uniform sampling step with our programmable hash function. However, this is a special case and a more general strategy may be needed. In those situations we can simply make use of Inverse Transform Sampling. The general idea is that for a distribution X, with CDF F_X, we simply do the following: If u is the result of our uniform sampling technique, we solve for x s.t. $F_X(x) = u$. This implies that using our uniform sampling technique. we can construct an RSS for any distribution.

VI. Integration with Specific Cryptocurrency Protocols

In this section we provide concrete examples of how this technique can be implemented within specific cryptocurrencies. Here we focus on Monero (RingCT) and ByteCoin (CryptoNote), due to the fact that these are currently the most widely-used mixing cryptocurrencies and protocols. For each system we provide bandwidth cost estimates for using our approach in that system, and compare with the cost of the approach currently used by the system.

Remark: In the following analysis we focus only on the description of $\mathcal{T}$ as it appears within a transaction. The analysis below omits all additional data that real currencies embed in their transactions, including (notably) cryptographic proofs.

A. Overview of Protocols

We now describe the cryptocurrency protocols, and specifically the approach each uses to encode $\mathcal{T}$.

ByteCoin. Bytecoin [20] is one of the earliest mixing protocols, and is based on the CryptoNote protocol [9]. ByteCoin uses a 1-out-of-N approach for constructing $\mathcal{T}$. If a transaction references multiple real transaction outputs, ByteCoin will sample multiple $\mathcal{T}$. Each $\mathcal{T}$ is differentially encoded within the transaction, *i.e.,* for each $i \in [1, N)$ the transaction encodes $I_i - I_{i-1}$.[8]

Because CryptoNote does not hide payment values, cover traffic outputs are sampled uniformly from the subset of previous transation outputs *that have the same currency value* as the real transaction output being obscured. At the time of writing, the largest such subset (for outputs of value 100 BCN) gives us $\ell = 2,000,000$, and we use this for our analysis. In our simulations below, we use an estimate for the size of each ByteCoin description of $\mathcal{T}$ by computing the expected size of N uniformly-sampled indices when differential encoding is applied. Note that in ByteCoin, values of $M > 1$ are encoded using M *distinct* transaction lists $\mathcal{T}$, one for each real

[8]These are encoded using a standard VLQ integer encoding. The low-order 7 bits of each byte encode data, and the MSB acts as a flag indicating whether there are additional bytes to come.

$F_{MONERO}(\mathcal{I} = \{I_0, \ldots, I_{M-1}\}, N)$

1 : For $i = 0$ to $N - M + 1$:

2 : Sample a random $r \in \mathbb{Z}_{2^{53}}$.

3 : Compute $\Gamma_i = \sqrt{\frac{r}{2^{53}}} \cdot \ell$.

4 : Construct sorted set $\{T_0, \ldots, T_{N-1}\}$ of $\Gamma_0, \ldots, \Gamma_{N-M+1}, I_0, \ldots, I_M$.

5 : **return** $\mathcal{T} = \{T_0, \ldots, T_{N-1}\}$

Fig. 2. Monero sampling technique

transaction. For simplicity, we compute our ByteCoin simulations only for $M = 1$.[9]

Monero. The Monero currency [14] is based on the newer RingCT protocol [11]. Like ByteCoin, Monero uses a 1-out-of-N approach for constructing $\mathcal{T}$ (for values of $M > 1$ this results in M distinct lists $\mathcal{T}$ being sampled and encoded within the transaction). Unlike CryptoNote, RingCT hides transaction values. Because transaction outputs do not need to be bucketed by like value, this gives a larger available set of previous transactions for a spender to use as cover traffic: at the time of writing, approximately $\ell = 4,000,000$. Monero encodes the transaction indices of $\mathcal{T}$ using the same differential encoding as ByteCoin.

As we discussed in §V, Monero samples cover traffic using an (approximate) triangular distribution.[10] For our simulations, we estimated the size of a Monero encoding of $\mathcal{T}$ by running Monte Carlo simulations with $1,000$ trials and computing the average size. Monero uses the same encoding process as ByteCoin for values of $M > 1$, and so again we consider our results only for the case of $M = 1$.

B. Simulation Results

For this simulation we assume our RSS scheme uses a 128-bit key k and a 64-bit prime p.[11] Larger values for these parameters are possible, and will have a predictable impact on our efficiency.

Figure 3 provides an clear visualization of how our scheme compares to the existing approach used in Monero and ByteCoin. For both currencies, we plot how varying N impacts the size in bits of the transaction list. Because the size of our RSS compact description depends on M, we offer plots for several values of M.

Monero. As expected, the "current approach" Monero encoding grows linearly with the number of transactions included in the transaction list. When only one legitimate transaction is included ($M = 1$), we see that for $N \geq 9$ our scheme produces a smaller description than the current Monero approach. Even as we increase M to larger values $(2, 5)$, our scheme still outperforms Monero's current approach at reasonably small values of N.

ByteCoin. The "current approach" of ByteCoin is similar in appearance to Monero, although it differs subtly in the slope of the line. Again, our scheme proves superior to ByteCoin's current approach, though at somewhat smaller values of N in comparison with Monero. Here, our scheme offers better cost for $N \geq 4$. Additionally, increasing M only has a small impact on the comparative efficiency of our scheme.

The reason for the steeper slope of the "current approach" in the case of ByteCoin is due to the fact that the currency samples transactions from a uniform distribution. This causes the transaction indices to be more evenly spaced, and thus all require roughly the same number of bits. Since Monero has a preference for recent transactions, many of the included transactions will be numerically close and will therefore produce more compact differential encodings.

In both cases, our scheme outperforms current approaches at even relatively small values of N. Additionally, innovations such as new proof techniques [16], [17] promise dramatic increases to what is considered a prac-

[9]To evaluate the case where $M > 1$, one can simply take the measurements for $M = 1$ and multiply the resulting description size by M. Of course this is somewhat unfair to Monero, given that a more efficient version of the protocol could simply encode multiple real transactions within a single set $\mathcal{T}$.

[10]In practice, Monero's sampling is more complicated. Monero uses two sets, one of "recent" transaction outputs, and another of "all previous" transaction outputs, and ensures that at least 50% of the resulting transactions are in the recent set. For simplicity of exposition, we simulate only the simple process of sampling from the set of all previous transactions, although our results can easily be adapted to Monero's more complicated sampling process.

[11]Our results do not include the description of p or ℓ, as we assume that ℓ is indicated elsewhere in the transaction (as it is in Monero and ByteCoin) and that p can be identified deterministically given ℓ.

tical value for N. To illustrate the potential cost savings of our approach at larger values of N, we provide the following concrete numbers (for an exemplary $M = 5$):

N	RSS ($M = 5$)	Monero	ByteCoin
1,000	.06 kB	1.97 kB	5.94 kB
10,000	.06 kB	16.59 kB	55.4 kB
100,000	.06 kB	103.17 kB	497.86 kB

These bandwidth comparisons illustrate the potential for our scheme to offer significant cost savings in transaction bandwidth. For comparison, the current average *total* transaction size in Monero (including all proofs and metadata) as of this writing is approximately 13kB [21].

VII. Other Applications

Beyond the significant improvements offered in the area of mixing cryptocurrencies, RSS has the potential for extensions to a more diverse set of applications. Generally speaking, the idea behind our scheme is to enable the insertion of desired values into a distribution, without revealing where these insertions took place. In this section we offer two examples of how to leverage RSS in other areas.

Anonymous Communication. The general concept of "mixing" is applicable to many anonymous communications systems. In particular, Rivest *et al.* originally proposed using ring signatures to add plausible deniability to email communications [12]. Just as the true input to a cryptocurrency transaction can be hidden by adding cover inputs, the true sender of a message can be hidden by mixing with other identities. If all user public keys are posted on some publicly accessible bulletin board, then we can use RSS to efficiently sample this data according to some distribution. Thus, senders would be able to construct large, yet compact, anonymity sets for themselves. Of course, just as in the cryptocurrency case, auxiliary data such a cryptographic proof will determine the total bandwidth overhead.

Client-Server Puzzles. A somewhat different application of RSS comes from Client-Server Puzzles [22], which generally involves one party requiring another party to complete some computationally intensive task. A common example is searching the domain of some hash function for an output that satisfies a particular requirement. RSS allows the puzzle creator more control over the search space. They might choose to place the solution at a specific position or modify the difficulty of the puzzle by changing the number of solutions present. Alternatively they could introduce trapdoors into the puzzle, by placing a solution at an index that an honest client can compute using their shared secrets.

This is far from a comprehensive list of the applications of RSS, and a future extension of this work is to explore and enumerate the possibilities.

VIII. Related Work

a) Anonymity for cryptocurrencies: A number of works have proposed additional privacy protections for cryptocurrencies. Zerocoin, Zerocash, CryptoNote and similar works [10], [8], [23], [11] provide strong anonymity through the use of complex zero knowledge proofs. A separate line of works seek to increase anonymity by Bitcoin by allowing users to interactively mix transactions (e.g. CoinJoin [24], CoinShuffle, CoinSwap). Maxwell and Poelstra proposed "Borromean" ring signatures [25], in which the statement proven is a monotone boolean function of the signing keys. A separate line of work [26], [27] examines *off-chain* private payments.

b) Improved WI and ZK proof techniques: A different line of work examines the efficiency of privacy-preserving protocols such as ring signatures, which are widely used in cryptocurrencies. For example, Dodis *et al.* [28] proposed a constant-sized[12] RSA-based ring signature (using an accumulator due to Camenisch and Lysyanskaya [29]) in the random oracle model. Using a new proof system in the discrete log setting, Groth and Kohlweiss recently proposed a concretely efficient technique with $log(n)$-sized signatures [30] (though also in the random oracle model). Most recently, Malavolta and Schröder proposed an efficient *constant-sized* group signature in the CRS model based on zkSNARKs [31].

c) Programmable hash functions: A large number of works use programmable hash functions, frequently as an element of the security proof. Hofheinz and Kiltz [18] offer a canonical paper on the technique, although their proposals are not precisely suited to our application. A vast number of works (including for example [32], [33], [19]) employ these techniques. A noteworthy feature of some schemes similar to ours, (*e.g.,* the Hohenberger-Waters signature [19]) is that these schemes require *statistical* properties from underlying hash function f, and thus achieve security in the standard model even when the hash function is a PRF with a non-secret seed. Developing an RSS in the standard model is an interesting open problem.

[12]For convenience we ignore the security parameter. In practice, such "constant-sized" schemes have bandwidth $O(\lambda)$.

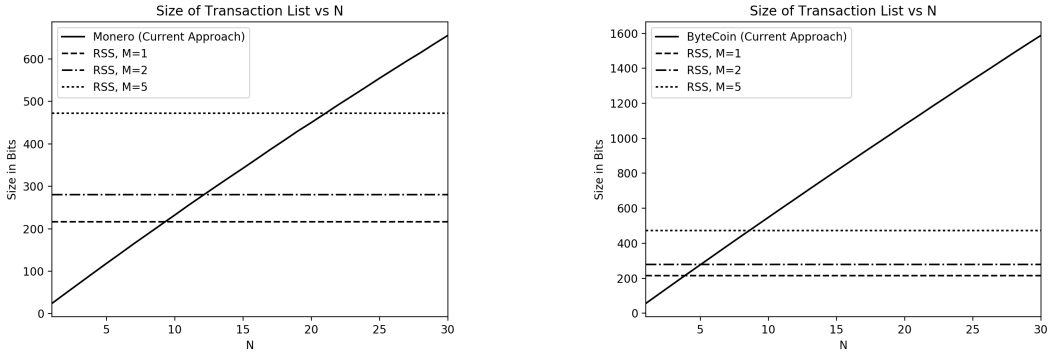

Fig. 3. Size of the transaction list $\mathcal{T}$ in bits for two popular cryptocurrencies, Monero (left) and ByteCoin (right). "Current approach" indicates the representation of the transaction list as a function of N (the number of transactions on the list). Note that this value is (mosty) independent of M, so we do not illustrate different values of M. "RSS" indicates the cost of the compact description W for different values of M, N. For ByteCoin we use $\ell = 2,000,000$ and for Monero we use $\ell = 4,000,000$.

IX. Conclusion

In this work we described a new approach to describing transaction sets in deployed mixing cryptocurrencies such as Monero and ByteCoin. We believe that our technique is promising and can offer substantial bandwidth improvements when the total size of the transaction set is large. Our work leaves several open questions. In particular, we believe that there may be many other applications for this technique. Additionally, we desire a scheme that offers security without relying on random oracles.

Acknowledgements. This work was supported by: The National Science Foundation under awards EFRI-1441209 and CNS-1414023; Google ATAP; The Mozilla Foundation; and the Office of Naval Research under contract N00014-14-1-0333.

References

[1] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: Characterizing payments among men with no names," in *Proceedings of the 2013 Conference on Internet Measurement Conference*, ser. IMC '13, 2013.

[2] D. Ron and A. Shamir, "Quantitative Analysis of the Full Bitcoin Transaction Graph," in *Financial Cryptography '13*, 2013.

[3] Block Chain Analysis, "Block chain analysis," http://www.block-chain-analysis.com/, 2014.

[4] H. A. Kalodner, S. Goldfeder, A. Chator, M. Möser, and A. Narayanan, "BlockSci: Design and applications of a blockchain analysis platform," Arxiv.org, 2017. [Online]. Available: http://arxiv.org/abs/1709.02489

[5] A. Miller, M. Moeser, K. Lee, and A. Narayanan, "An empirical analysis of linkability in the Monero blockchain," Available at https://arxiv.org/abs/1704.04299, 2017.

[6] Chainalysis, "Chainalysis inc," https://chainalysis.com/, 2017.

[7] Elliptic, "Elliptic enterprises limited," https://www.elliptic.co/, 2017.

[8] E. B. Sasson, A. Chiesa, C. Garman, M. Green, I. Miers, E. Tromer, and M. Virza, "Zerocash: Decentralized anonymous payments from bitcoin," in *IEEE Security and Privacy*, 2014.

[9] N. van Saberhagen, "Cryptonote v2.0," Available at https://cryptonote.org/whitepaper.pdf, October 2013.

[10] I. Miers, C. Garman, M. Green, and A. D. Rubin, "Zerocoin: Anonymous distributed e-cash from Bitcoin," in *Proceedings of the 2013 IEEE Symposium on Security and Privacy*, ser. SP '13, 2013.

[11] S. Noether, "Ring signature confidential transactions for monero," *IACR Cryptology ePrint Archive*, vol. 2015, p. 1098, 2015. [Online]. Available: http://eprint.iacr.org/2015/1098

[12] R. L. Rivest, A. Shamir, and Y. Tauman, "How to leak a secret," in *ASIACRYPT '01*, C. Boyd, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 552–565.

[13] B. Parno, C. Gentry, J. Howell, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *Proceedings of the 34th IEEE Symposium on Security and Privacy*, ser. Oakland '13, 2013, pp. 238–252.

[14] "The Monero Currency," Available at https://getmonero.org/, 2017.

[15] J. Buntinx, "What is RuffCT and How Will It Affect Monero?" *The Merkle*, 2017.

[16] S.-F. Sun, M. H. Au, J. K. Liu, and T. H. Yuen, "RingCT 2.0: A compact accumulator-based (linkable ring signature) protocol for blockchain cryptocurrency Monero," in *ESORICS 2017*, S. N. Foley, D. Gollmann, and E. Snekkenes, Eds. Cham: Springer International Publishing, 2017, pp. 456–474.

[17] "What is StringCT/RuffCT?" Available at https://monero.stackexchange.com/questions/5997/what-is-stringct/5999#5999, 2017.

[18] D. Hofheinz and E. Kiltz, "Programmable hash functions and their applications," in *CRYPTO 2008*, D. Wagner, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 21–38.

[19] S. Hohenberger and B. Waters, "Realizing hash-and-sign signatures under standard assumptions," in *Advances in Cryptology - EUROCRYPT 2009*, A. Joux, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 333–350.

[20] "Bytecoin," At https://bytecoin.org/.

[21] "Xmrchain," Available at https://xmrchain.net/, 2017.

[22] A. Juels and J. G. Brainard, "Client puzzles: A cryptographic countermeasure against connection depletion attacks." in *NDSS*, vol. 99, 1999, pp. 151–165.

[23] N. van Saberhagen, "CryptoNote v 2.0," October 2013. [Online]. Available: https://cryptonote.org/whitepaper.pdf
[24] G. Maxwell, "CoinJoin: Bitcoin privacy for the real world," Available at https://bitcointalk.org/index.php?topic=279249.0, August 2013.
[25] G. Maxwell and A. Poelstra, "Borromean ring signatures," Available at https://github.com/Blockstream/borromean_paper, 2015.
[26] E. Heilman, F. Baldimtsi, and S. Goldberg, "Blindly Signed Contracts: Anonymous On-Blockchain and Off-Blockchain Bitcoin Transactions," in *BITCOIN '16*, 2016.
[27] M. D. Green and I. Miers, "Bolt: Anonymous payment channels for decentralized currencies," in *CCS '16*, vol. 2016, 2016. [Online]. Available: http://eprint.iacr.org/2016/701
[28] Y. Dodis, A. Kiayias, A. Nicolosi, and V. Shoup, "Anonymous identification in ad hoc groups," in *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings*, 2004, pp. 609–626. [Online]. Available: https://doi.org/10.1007/978-3-540-24676-3_36
[29] J. Camenisch and A. Lysyanskaya, "Dynamic accumulators and application to efficient revocation of anonymous credentials," in *CRYPTO '02*, 2002, extended Abstract. [Online]. Available: http://cs.brown.edu/~anna/papers/camlys02.pdf
[30] J. Groth and M. Kohlweiss, "One-out-of-many proofs: Or how to leak a secret and spend a coin," in *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part II*, 2015, pp. 253–280. [Online]. Available: https://doi.org/10.1007/978-3-662-46803-6_9
[31] G. Malavolta and D. Schröder, "Efficient ring signatures in the standard model," In ASIACRYPT '17, 2017.
[32] D. Boneh and X. Boyen, "Efficient selective-ID secure Identity-Based Encryption without random oracles." in *EUROCRYPT '04*, vol. 3027 of LNCS, 2004, pp. 223–238.
[33] B. Waters, "Efficient Identity-Based Encryption without random oracles," in *EUROCRYPT '05*, vol. 3494 of LNCS, 2005, pp. 114–127.